A Study of Vehicle Dynamics Using ODEs

Mark Do, Mckale Chung, Carmelli Dao, Miekale Smith, Odette Beaudin

March 25, 2025

Vehicle dynamics is the study of how vehicles respond to driver inputs and external forces, focusing on aspects such as handling, stability, and maneuverability. A key model to simplify 4 wheel dynamics is the bicycle model, turning four-wheel vehicle into a two-wheel system while retaining the dynamics. Despite its simplicity, this model accurately captures the core characteristics of vehicle motion, making it a powerful tool for analyzing handling performance.

Understanding vehicle dynamics is critical for improving vehicle safety, optimizing performance, and developing autonomous driving systems. Accurate models help engineers predict a vehicle's response to steering, braking, and acceleration, which is essential for designing control systems that enhance stability and reduce the risk of accidents.

The bicycle model can be represented by a system of Ordinary Differential Equations (ODEs) that describe the relationship between the car's lateral velocity, yaw rate, and steering input. These equations capture the essential dynamics of the vehicle, such as how it responds to steering inputs and external disturbances. Simulating these ODEs allows for the analysis of key performance metrics, such as yaw stability, cornering behavior, and lateral acceleration, providing valuable insights into the car's handling characteristics. The bicycle model used is displayed in figure 1 below.

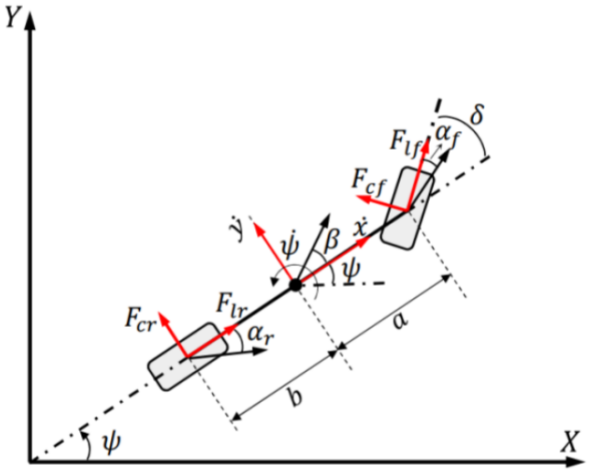

Figure 1: Bicycle model of two-axis (four wheel) car. [1]

Through this report, we analyze the ODEs governing the bicycle model, quantifying the vehicle dynamics with various external factors.

Part 1: Warm Up

This warm up showcases the accuracy of analytical and numeric approximations of a simple ODE.

$y'(x) = y(x)$, with initial condition $y(0) = 1$.

1.1 Analytical Solution to the ODE by Separation

The equation is a first order separable equation.

$$\begin{aligned}
\frac{dy}{dx} &= y \\
\frac{dy}{y} &= dx \\
\int \frac{dy}{y} &= \int dx \\
\ln|y| &= x + C \\
y &= e^{x+C} \\
y &= e^C e^x
\end{aligned}$$

Substituting the initial condition $y(0) = 1$:

$$\begin{aligned}
1 &= e^C e^0 \\
e^C &= 1
\end{aligned}$$

Therefore, the solution to the ODE is $y = e^x$

1.2 Approximating the ODE by Power Series

Solve the above ODE by the method of power series.

$$\begin{aligned}
y &= \sum_{n=0}^{\infty} c_n x^n \\
y' &= \sum_{n=1}^{\infty} n c_n x^{n-1} \\
&= \sum_{n=0}^{\infty} (n+1) c_{n+1} x^n
\end{aligned}$$

Subbing these into the original ODE:

$$\begin{aligned}
y' &= y \\
y' - y &= 0 \\
\sum_{n=0}^{\infty} (n+1) c_{n+1} x^n - \sum_{n=0}^{\infty} c_n x^n &= 0
\end{aligned}$$

This means that for all n, $(n+1)c_{n+1} = c_n$. For $k \in \mathbb{R}$:

$$\begin{aligned}
n = 0: &\quad c_1 = c_0 \\
n = 1: &\quad 2c_2 = c_1 \quad \Rightarrow \quad c_2 = \frac{1}{2}c_0 \\
n = 2: &\quad 3c_3 = c_2 \quad \Rightarrow \quad c_3 = \frac{1}{6}c_0 \\
n = 3: &\quad 4c_4 = c_3 \quad \Rightarrow \quad c_4 = \frac{1}{24}c_0 \\
n = 4: &\quad 5c_5 = c_4 \quad \Rightarrow \quad c_5 = \frac{1}{120}c_0 \\
&\vdots \\
n = k: &\quad (k+1)c_{k+1} = c_k \quad \Rightarrow \quad c_{k+1} = \frac{1}{k!}c_0
\end{aligned}$$

The function y may be written as follows:

$$y = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + ... + c_nx^n$$

$$y = c_0 + c_1x + \frac{1}{2!}c_0x^2 + \frac{1}{3!}c_0x^3 + \frac{1}{4!}c_0x^4 + ... + \frac{1}{n!}c_0x^n$$

Subbing in the initial condition:

$$y(0) = 1 = c_0 + 0$$

$$c_0 = 1$$

Therefore, y may be written as follows:

$$y = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + ... + \frac{1}{n!}x^n$$

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

This is the Maclaurin series representation of e^x. Therefore, by power series, we find the same answer as was found in question 1:

$$y = e^x$$

1.3 Power Series Accuracy

Plotted solutions from questions 1 and 2:

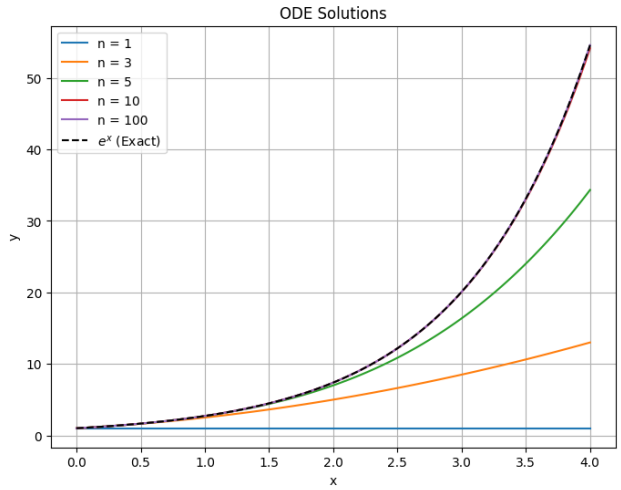

Figure 2: Power Series Calculations based on the Number of Approximation Terms

As n is increased, the accuracy gets closer to the exact solution of the function. For $n = 10$ and $n = 100$, the solution is so close to the exact solution that it overlaps on the plot. The figure below outlines the error seen for each value of n:

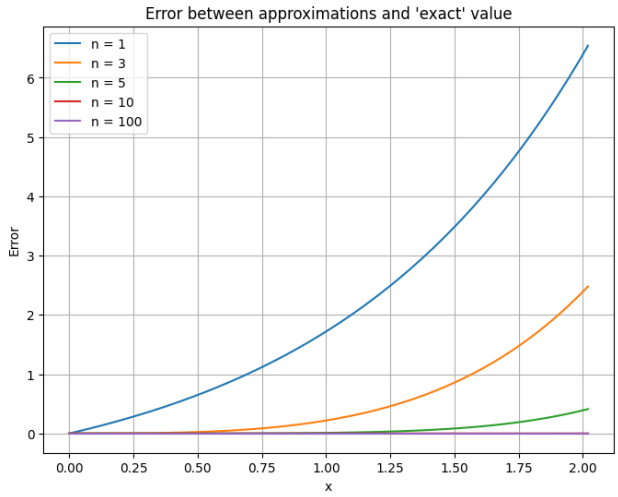

Figure 3: Power Series Errors based on the Number of Approximation Terms

1.4 Approximating the ODE with Forward Eulers Method

Using Forward Eulers method, the following plot was generated:

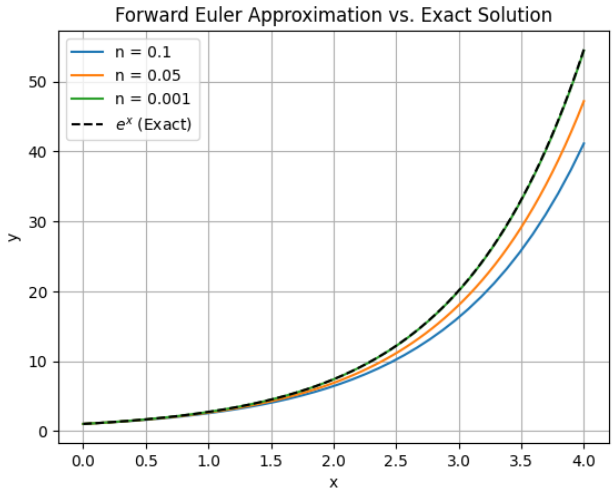

Figure 4: Forward Euler Approximations with Varying Step Sizes

As the step size is decreased, the the approximation gets closer to the actual solution. The $n = 0.001$ seemed to approximate the solution very accurately. Decreasing the spacing generally will increase accuracy, but for computers that cannot store as much data in their floats, a small grid spacing can accumulate round-off error.

For both the analytical and numeric solution, they approximate better as the number of steps/terms increases. While Eulers method did converge well with a small step size, the numeric solution has a much lower order of accuracy (order of 2). The analytical solution also provides a direct function to compare to the exact solution, whereas the numeric solution requires calculating each value at x_i.

Part 2: A Study of Vehicle Dynamics using ODEs

PART A

1 Solver Implementation

The RK4 and Euler solvers were implemented within a single Python class. Appendix A contains the code for the solvers. These solvers are made to solve the equation 3 in Appendix E which describe the lateral velocity and yaw rate respectively.

2 Error Analysis

The "ground truth" solution was obtained using the RK4 method with a step size of 10^{-7}. The global error was evaluated at $t = 1s$ by comparing the solutions from the RK4 and Euler solvers against the ground truth.

The examined grid sizes ranged from $\Delta t = 0.1$ to $\Delta t = 0.001$. The observed error slopes were approximately 1 for the Euler method and 4 for the RK4 method, aligning with their theoretical first-order and fourth-order error behaviors, respectively.

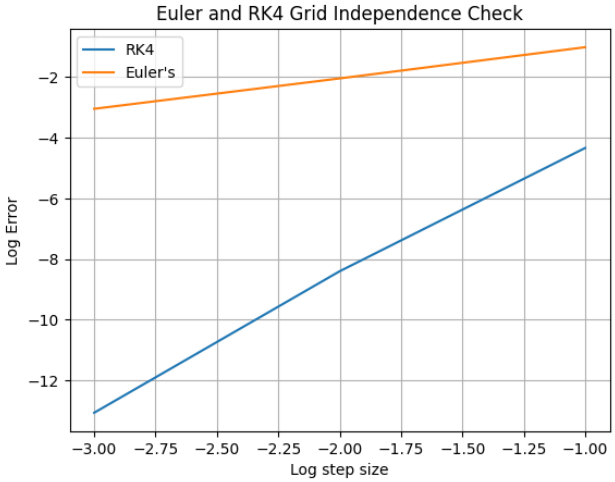

Figure 5: Log Error Graph for Euler and RK4 Solvers

PART B

B.1 Lateral Acceleration and Yaw Rate Based on Lateral Velocity

The figures below are the acceleration vs. time and yaw rate vs. time graphs for different velocity values. Qualitatively speaking, stability can be seen here as the lateral acceleration graphs converging to 0 or diverging to negative infinity. Similarly, the yaw rate for diverging graphs explode to infinity while converging graphs asymptote to some finite value. Appendix B contains code for the stability checker. The logic behind this stability will be explained in the upcoming section.

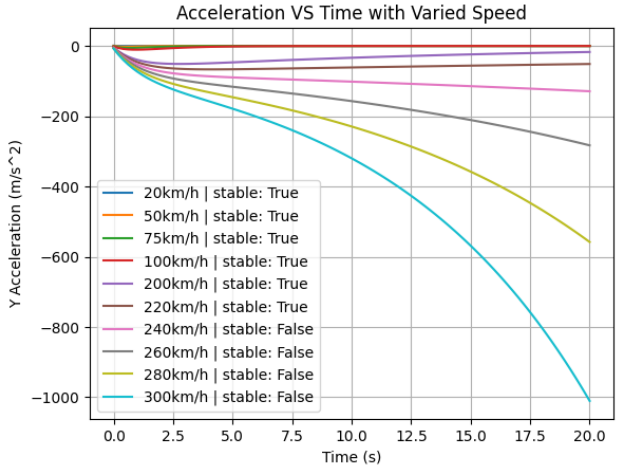

Figure 6: Lateral Acceleration vs. Time graph for different x velocities

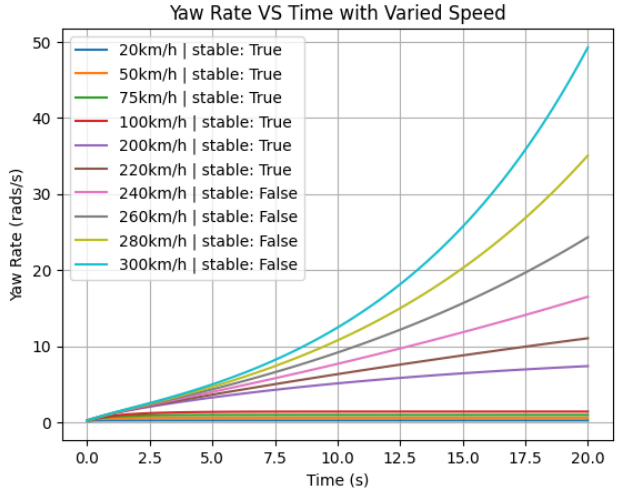

Figure 7: Yaw Rate vs. Time graph for different x velocities

B.2 Highest Stable Speed of the Vehicle

A 'stable' solution, one where the car is deemed 'stable', is mathematically defined as an ODE where the change of state vector A has non-positive real-component eigenvectors. We can break the ODE into the forcing term $B\delta$ and the homogeneous term Ay. Since the eigenvalues of A determine the

form of the solution, A giving non-negative eigenvalues mean that the solution will have positive exponent in the numerator (i.e. e^{λ} where λ is positive) and thus will diverge to infinity, causing both the yaw rate and lateral velocity to follow suit.For now, the control input delta has no effect on A thus no effect on the stability of the ODE.

Numerically searching for this stability point yields $228.06 km/h$ to be the last velocity (within 2 decimal places) at which stability occurs. The code for this numerical approximation is found in Appendix B

B.3 Car Handling

To gauge the car's handling, the ground velocity is calculated using equations 5 and 6 with inputs from RK4 solver. The ground velocity is then numerically intergated over time, using a left-point Riemann sum, and then graphed with different step steering inputs below in Figure 8:

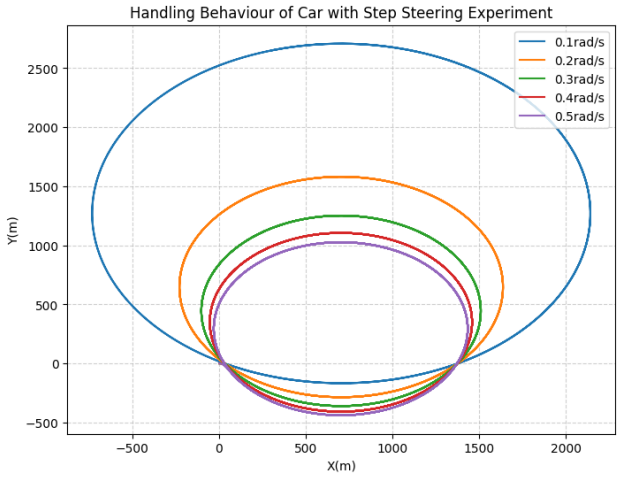

Figure 8: Step steering experiment for various step inputs

(a.): To test if the car is over-steering or under-steering at 100km/h, the steady state yaw rates(yaw rate at a point far away from the step impulse(t=0)) for various speeds were collected. These were then graphed in Figure 9 below:

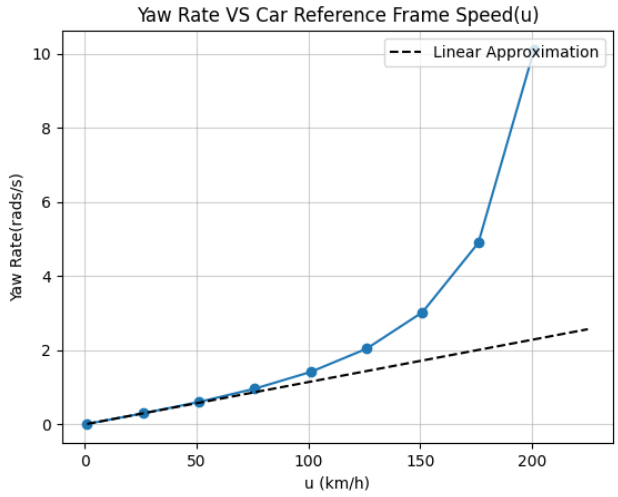

Figure 9: Determining under/over steering at 100km/h

Since the graph of the yaw rate vs car reference frame speed is concave up(point at x=100 is above the linear approximation), it can be concluded that the behavior of the car is over-steering. This occurs because, as speed increases while maintaining a constant steering angle, the yaw rate increases at an accelerated rate. However, it is worth noting that the degree of oversteer at this speed is minor, as the deviation from the linear approximation at x=100 is minimal.

C.1 Vehicle Stability with Variable Loads

Table 1 shows a comparison between the previous and current values for mass, a, and b. The new calculated values assume that a 50kg mass has been added to the rear weight of the car. Once a 50kg mass has been added in the rear, the a distance, which represents the distance of the CoM (centre of mass) from the front axle, will increase and the b distance, which represents the distance of the CoM from the rear axle will decrease. In the event where a front mass is added to the front weight of the car, the a value will still increase, and the b value will still decrease.

Values	Original	Rear Weighted	Front Weighted
Mass (kg)	1400	1450	1450
a (m)	1.14	1.192413793	1.1941172
b (m)	1.33	1.277586206	1.2758827

Table 1: Table for previous and current values

Generally, vehicles will have varying weight distributions based on their target industry. In the case for many FSAE vehicles, a weight distribution of 40:60, where the car is more concentrated in the rear, is preferred [2]. Traditionally, a car with a weight distribution of 40:60 will have the tendency to understeer, which is a phenomenon that occurs when the ratio of the front slip angle and the rear slip angle is less than 1 [3]. Although understeering is undesirable, rear weighted vehicles will have the tendency to reach a higher maximum lateral acceleration compared to other vehicles that do not have weights that are as concentrated in the rear [4].

Based on calculations from Part B, it can be determined that 227.3km/h is the highest stable speed of the car. The results of the generated graph 10 show that the car will diverge with a

greater Yaw Rate and Y Acceleration as time increases. Based on the varying masses added, it is evident that as a heavier mass is added in the rear axle, the car will reach a point where it no longer diverges and maintains stability.

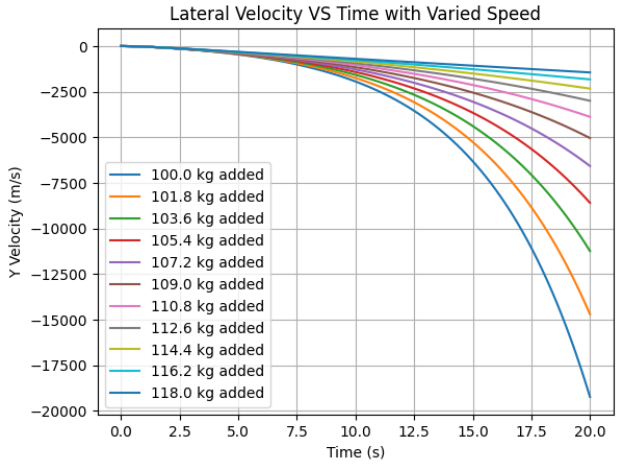

Figure 10: Acceleration vs Time with Varied Weight

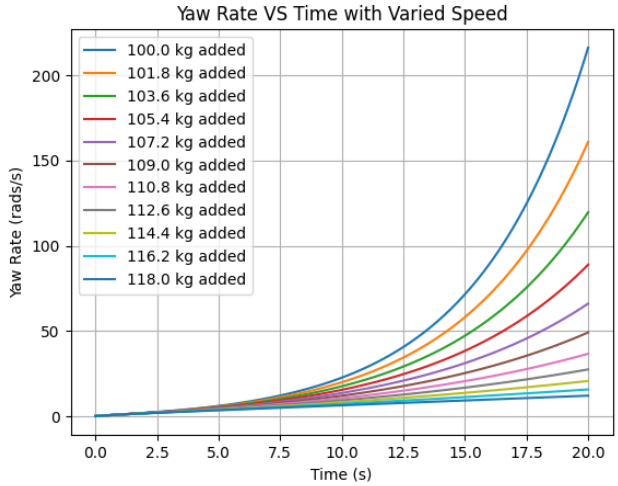

Figure 11: Acceleration vs Time with Varied Weight

The handling performance will be affected as the mass increases. From (reference figure) it is evident that as more mass is added, the rate at which the yaw rate increases will decrease. Given a vehicle driving at it's maximum stable speed, the yaw rate will eventually reach a point where it is no longer increasing once a mass of a certain weight is added. Once a mass of 50 kg has been added, the slope for Yaw Rate and Acceleration vs Time will be close to zero, meaning that the vehicle will not reach a point of instability no matter how long the vehicle continues to drive for.

To determine the mass added at which the vehicle becomes stable, a stability checker was implemented within the main body of code. Using the stability checker, it was determined that the vehicle will stabilize after a mass of 117 kg was added to the rear of the car. Once the vehicle stabilizes, the vehicle will no longer experience a rapid change in the acceleration and yaw rate.

C.1 Vehicle Stability in Winter Conditions

As demonstrated in the following section, driving faster and steering harder can be dangerous when operating vehicles in difficult conditions, such as ice or snow. When operating a vehicle in winter conditions, the stiffness of the tire will decrease since the stiffness is linearly proportional to the friction coefficient between the tires and the road. An increase in ice on the road will contribute to more slippery conditions, and hence the fiction coefficient along the surface of the road will decrease. Therefore, the stiffness of the tire will also decrease.

The following tire stiffness values were given in the assignment:

Front Tire Cornering Stiffness, *Caf*	25,000	N/rad
Back Tire Cornering Stiffness, *Car*	21,000	N/rad

Table 2: Given Tire Cornering Stiffness Values

A dry asphalt road typically has a friction coefficient ranging from 0.7 to 0.8 [**?**]. However, an icy road will typically have a friction coefficient less than 0.2. Since the tire corning stiffness values are linearly proportional to the coefficient of friction on the road, they can be scaled down by the same amount.

$$C_af : 25,000 * \frac{0.2}{0.75} = 6,667$$

$$C_ar : 21,000 * \frac{0.2}{0.75} = 5,600$$

To compare the impact of driving fast and steering hard, the simulation was run with a range of speeds and steering angles at both standard and icy tire stiffness. The following pairs of speeds and steering angles diverged during both the standard and icy simulations.

Speed, u (km/hr)	Steering Angle, δ (rad)
250	0.1
250	0.15
250	0.2
250	0.25

Table 3: Pairs of speed and steering angles where the simulation diverges.

The values in the table above demonstrate that the steering angle does not impact the stability of the simulation, as the simulation will always converge as long as the speed is lower than 227 kilometers per hour, even with high steering angles. Although this seems unlikely, this result makes sense as the steering angle, δ, does not appear in the given equations that model the path of the car and therefore should not impact convergence.

Despite the fact that the steering angle will not affect the convergence of the simulation, it will still impact the controllability of the car. By plotting the trajectory of the car, it can be determined quantitatively which values the car is no longer safe to drive. The following plot shows the trajectory of the car under standard conditions (i.e. when driving on asphalt without ice).

After changing the tire stiffness values to the icy road approximations, the trajectory of the car follows the path seen in the plot below.

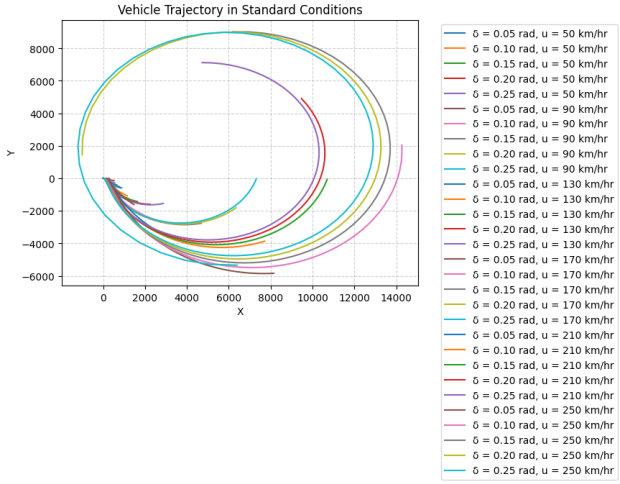

Figure 12: Car trajectory with different pairs of speeds and steering angles in standard conditions

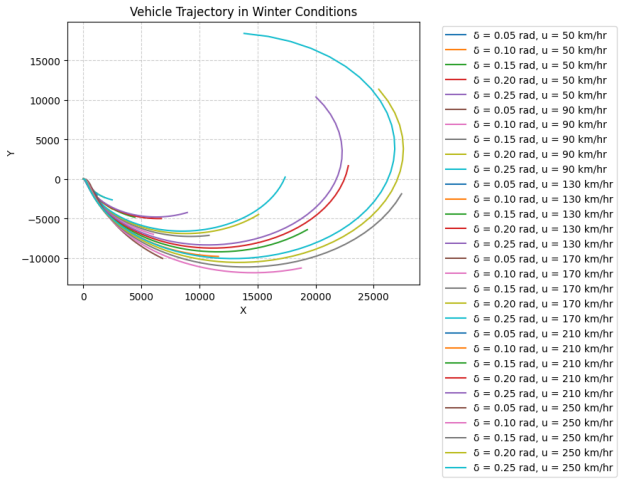

Figure 13: Car trajectory with different pairs of speeds and steering angles in icy conditions

When comparing Figures 1 and 2, one of the key differences is the turning radius. The vehicle trajectory under icy conditions follows a much wider turning radius, suggesting that the decrease made the vehicle less responsive to steering inputs. The plots also suggest that the under icy conditions the vehicle appears to under-steer more significantly. This is caused by the lowered friction coefficient lowering the vehicle's ability to make tight turns. At higher speeds both plots show outward deviations that suggest the vehicle is more prone to loosing control. This effect is even more pronounced in the icy conditions as the decreased stiffness leads to a decrease in lateral grip. Additionally, under standard conditions the impact of steering angle creates a noticeable difference in the vehicle path. However, in icy conditions, even high steering angles fail to produce tight turns, further demonstrating how the vehicle is less responsive while driving on an icy road. These factors conclude that the vehicle is less controllable under icy conditions than under standard ones, especially at higher speeds.

The following table outlines pairs of speeds and steering angles where the car is barely controllable under icy conditions. These pairs were determined qualitatively from Figures 1 and 2, and

are not the only conditions at which the car will be barely controllable.

Speed, u (km/hr)	Steering Angle, δ (rad)	Justification
130	0.1	Here the car is significantly under-steering, an
170	0.15	As the speed increases, even with a larger steering ang
210	0.1	A higher speed and low steering angle mak
250	0.1	Here the speed is higher than the maximu
250	0.25	Even at a large steering angle, the car will not be responsive at th

Table 4: Pairs of speeds and steering angles where the car is "barely controllable".

C.3 Vehicle Stability with Winter Tires

Another way to model wintry conditions is to have variable tire cornering stiffness based on the steering angle. This is done by using the following piecewise conditions given in the project information.

$$C_{\alpha f} = C_{\alpha r} = \begin{cases} 20000 & \delta \leq 0.06 \\ 100 & 0.06 < \delta \leq 0.2 \\ 0 & \delta > 0.2. \end{cases}$$

When standard tires are replaced by winter ones, the tire cornering stiffness will increase with lower steering angles to make up for the lowered friction on icy roads. This can also be represented by a piecewise function, as seen below.

$$C_{\alpha f} = C_{\alpha r} = \begin{cases} 20000 & \delta \leq 0.06 \\ 5000 & 0.06 < \delta \leq 0.3 \\ 0 & \delta > 0.3. \end{cases}$$

To examine the impact that adding winter tires will have on the controllability of a car, a plot of the trajectory that occurs with and without winter tires based on the piecewise conditions above was made. The following graphs show how the vehicle trajectory will vary based on the presence or absence of winter tires. To analyze the effectiveness of winter tires, the tire cornering stiffness will no longer be considered a constant.

Without winter tires, the graphs appear to show oscillations where the input of δ is between 0.1 rads/sec and 0.2 rads/sec. Oscillations imply that the vehicle is in a state of instability. However, after a while the oscillations in the graph experience a shorter amplitude before the slope becomes linear. This implies that the vehicle will respond better to driver input after the vehicle has experiences a period of instability. For values of δ between 0.02 rads/sec and 0.06 rads/sec, the driver will have better control over the vehicle. The curved path of the slopes imply that the vehicle will respond better to a the varied steering inputs.

With winter tires, the vehicle no longer experiences periods of oscillation for higher values of δ. For all of the given steering inputs, the graph shows tighter and more controlled curves. The presence of tighter and more controlled curves implies that the vehicle will have a better response to a given steering angle when winter tires are applied to the car.

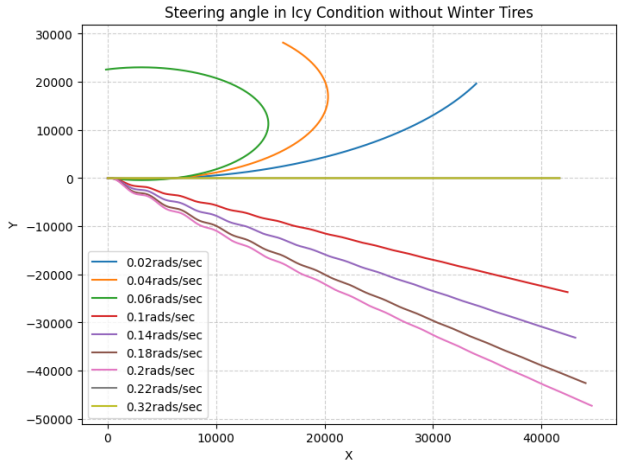

Figure 14: Car trajectory without winter tires.

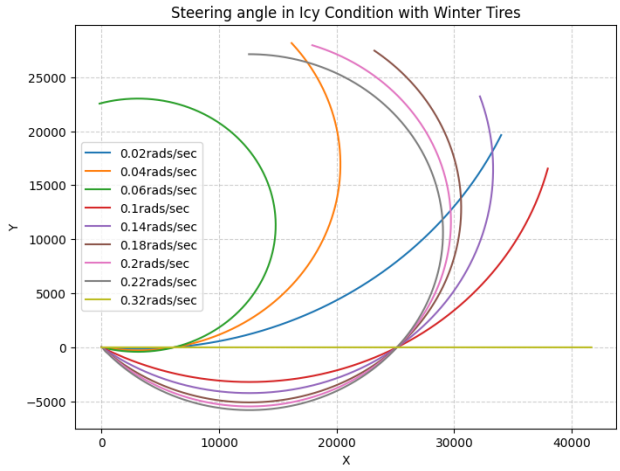

Figure 15: Car trajectory with winter tires.

PART D

The slip angle of a tire describes the angle between its path of travel, and the direction it is facing. This comes as a result of centrifugal forces during turns, and may lead to a lack of control for the driver [5]. The cornering stiffness is a measure of the force that the tire generates to mediate this deformation, and therefore reduce the slip angle. It describes the amount of lateral force the tire generates per degree of slip angle [7]. Therefore, increasing the cornering stiffness increases the tires response to mediate slip angle and deformation.

Equations 1. and 2. show that the magnitude of lateral velocity should decrease as cornering stiffness for either the front or rear tires is increased. The magnitude of the yaw rate is significantly based on the difference between the front and rear wheel. Increasing the front wheel cornering stiffness contributes to a larger yaw rate magnitude. Conversely, Increasing the rear cornering stiffness, in comparison to the front wheel should have the opposite effect; reducing the end magnitude of yaw rate.

D.1 Generalization of stability behaviour

The eigenvalues of the homogeneous system when $\delta(t) = 0$ describe the roots of the system. These are found as follows:

$$A = \begin{bmatrix} d & e \\ f & g \end{bmatrix}$$

Where d, e, f, and g respectively are the a_{11}, a_{12}, a_{21} and a_{22} function terms of the A matrix in the system in Equation (4). The particular solution is a scalar, and can shift the graph for $\delta(t) \neq 0$. However, it will not impact the time-dependent behaviour of the system. This generalization will focus using the homogeneous solution to the equation to observe time-dependent behaviour. The characteristic equation is found by solving:

$$\det(A - \lambda I) = 0$$

$$\begin{vmatrix} d - \lambda & e \\ f & g - \lambda \end{vmatrix} = 0$$

Expanding the determinant:

$$(d - \lambda)(g - \lambda) - ef = 0$$

$$\lambda^2 - (d + g)\lambda + (dg - ef) = 0$$

Solving for λ:

$$\lambda = \frac{(d + g) \pm \sqrt{(d + g)^2 - 4(dg - ef)}}{2}$$

When the discriminant is equal to zero, there is one root to this characteristic equation. When the roots are complex, the homogeneous solution to the matrix A in Equation (4) should have sinusoidal solutions, and a driver would experience oscillations in the lateral and yaw velocity.

When the discriminant is positive, and the characteristic equation has two real roots. If both roots are negative, then the behavior will exponentially decay to the particular solution. However, if at least one root is positive, then it will grow exponentially, causing instability in the system. The break-point between these two events may be found when one of the eigenvalues is equal to zero. This occurs when $dg - ef = 0$.

$$\lambda^2 - (d+g)\lambda + (dg - ef) = 0$$

$$\lambda^2 - (d+g)\lambda = 0$$

$$\lambda(\lambda - (d+g)) = 0$$

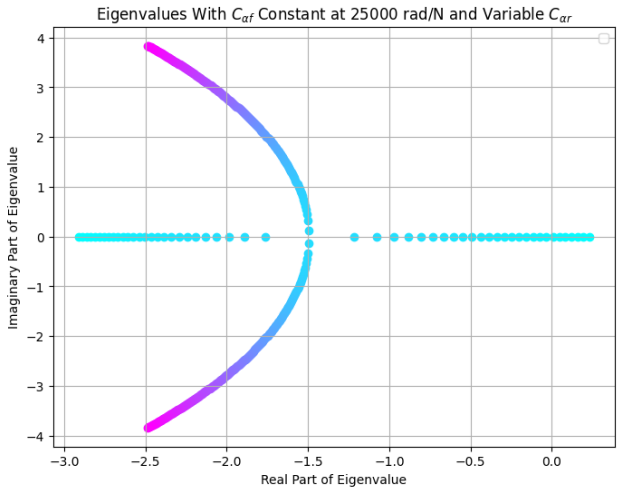

Figure 16: Eigenvalues of the System's Homogeneous Matrix A in Equation (4) as C_{ar} is Modified

The Figure above graphs the eigenvalues while $C_{af} = 25000$, and $17000 \leq C_{ar} \leq 50000$. Higher values of C_{ar} are drawn in purple, while lower ones are drawn with a gradient to turquoise. Each point will two points, except for the case with one real root, drawn as the red point in the center of the graph.

The black points represent the two eigenvalues for when there is one zero root. This was found at $C_{ar} \approx 18026$ using a Python code in Appendix C. For values of C_{ar} below this, there will be one positive root, and the system will experience positive exponential behavior.

The red point at approximately $(-1.5, 0)$ shows the singular point that has one real root. This break point was found at $C_{ar} \approx 21446$ by solving for when the discriminant is equal to zero. This was achieved using the Python code in Appendix D. When C_{ar} is increased above this point, the system should have oscillations.

Between $C_{ar} = 21446$ and $C_{ar} = 18026$, the system has 2 negative roots, and so the system purely exponentially decays without oscillations. These findings show that there are two conditions for instability: When there is a positive root and when there are complex roots. This gives a framework for balancing the relationship of C_{ar} and C_{af}. Should C_{ar} be increased too far, the vehicle will experience sinusoidal oscillations. The value of C_{ar} does not necessarily need to be greater than C_{af} for this case, but the larger it becomes, the more aggressive the oscillations.

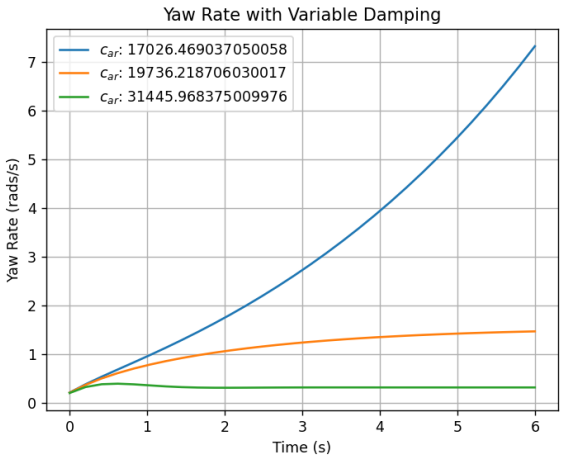

Figure 17: Enter Caption

Should C_{ar} be decreased too far, the vehicle will experience exponential velocity and yaw rate. This means that even small external forces could result in a loss of control, if applied for an extended period of time. The below figures visualize the system when C_{ar} is set above, between, and below the stated breakpoints:

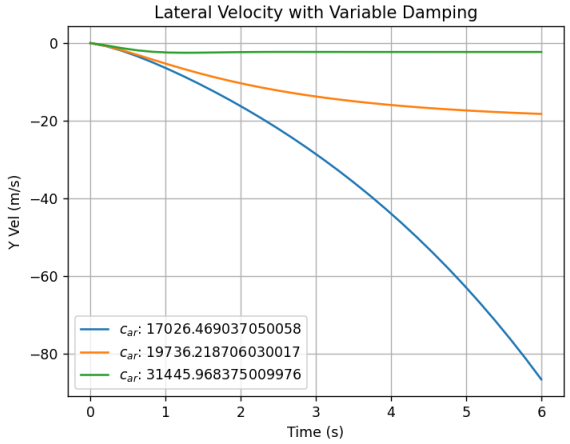

Figure 18: Lateral Velocity Underdamped, Overdamped, and Unbounded System Behaviour by Modifying The Rear Cornering Stiffness

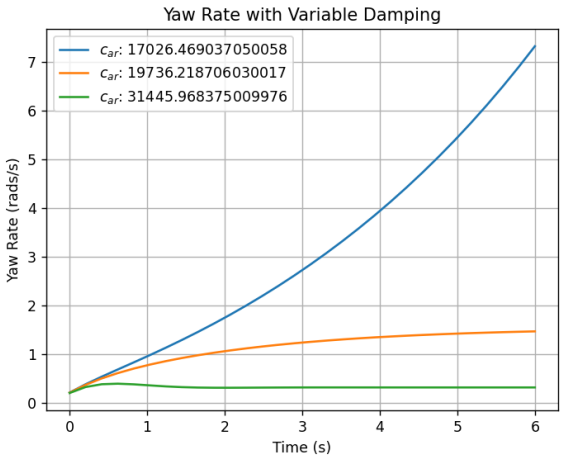

Figure 19: Yaw Rate Underdamped, Overdamped, and Unbounded System Behaviour by Modifying The Rear Cornering Stiffness

D.2 Performance Cars: Why Wider Rear Tires may be Desired

As stated earlier, the larger the C_{ar} becomes, the more aggressive the oscillations. The below figures show this behaviour as C_{af} is held constant at 25000, and C_{ar} is increased above this value:

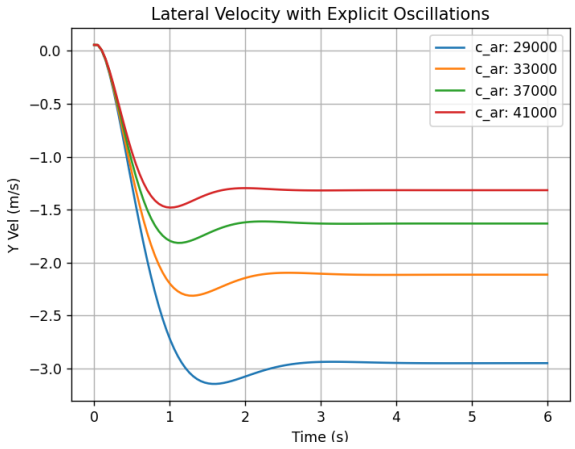

Figure 20: Lateral Velocity with Explicit Oscillations

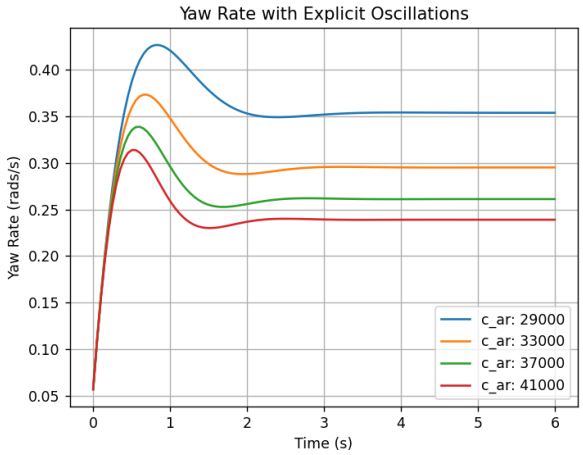

Figure 21: Yaw Rate With Explicit Oscillations

Performance cars may be designed with wider rear tires to achieve a larger C_{ar} in order to purposely induce under damping. Performance cars are designed to be agile to excel in entertainment or racing. Compared to Figure 18, the under-damped velocity seen in graph Figure 20 would enable faster aggressive turns. This would be vital for non-linear tracks, and races where vehicles need to weave between each other with quick directional changes. Additionally, wider tires leads to increased friction to prevent oversteer [6]

Race cars generally have wider tires than commercial cars, in both the front and rear wheels. This reduces slip angle, which in turn makes the vehicle more responsive to driver inputs [5]. This responsiveness is not as necessary for typical drivers, as small hand movements may be unintended and difficult to control.

While this system may be considered 'instable', it could be controllable for experienced drivers. If the relation between the width of tires was designed so the system has one real root, then the system would experience exponential growth and be fully uncontrollable over time.

Traditional cars will typically design C_{ar} and C_{af} to be under-damped to avoid instability and over responsiveness. If the velocity displays exponential decay, then the steering is predictable during turns, which is safe for the everyday driver who typically does not make sharp turns.

D.3 Observing System Behaviour as Mass is Reduced

In the case where the mass of the car is reduced, the magnitude of lateral velocity and Yaw rate increases:

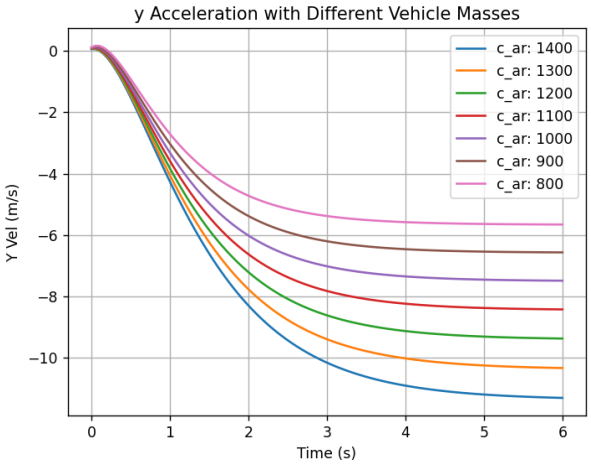

Figure 22: Lateral Velocity as Vehicle Mass is Reduced

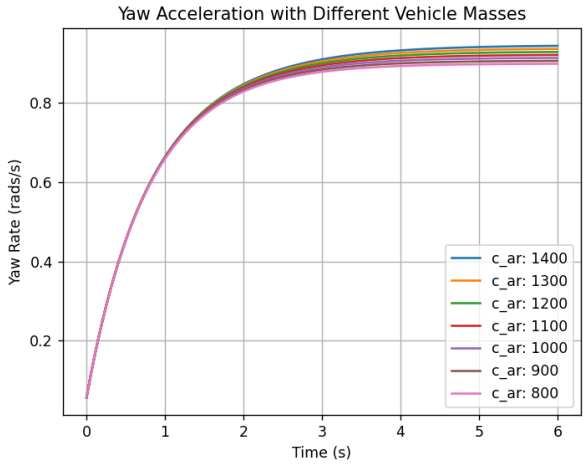

Figure 23: Yaw Rate as Vehicle Mass is Reduced

From the figures above, the magnitude of lateral velocity decreased as the mass decreased. The yaw rate for all tested masses ended around 0.9 rad/s.

By strictly decreasing the mass of the car, the lateral velocity could be considered more 'controllable', though at the same time could be considered less responsive.

PART E

E.1 Briefly summarize what you learned from carrying out this course project (skills, knowledge, ideas, philosophy).

We gained intuition of how iterative / forward solvers work in higher dimensions. It is as simple as changing the variable type we used in Python, which reflects how you could turn scalars into vectors.

We found that being able to simplify such a complicated system down into an ODE and just solve it quite easily (few lines of code) is very powerful. As long as we can derive / find a physics derivation for any ODE system in the real world (ex: a more complicated bicycle model, or a movement model for a drone), we can use RK4 or another iterative solver to find the solution. Having a base model makes adding additional complexity easier, allowing an engineer to quickly mock up a realistic simulation.

This also provides a base to fully explore how modifying one or multiple components can completely change the behaviour of the vehicle. By changing the speed, tires and weight positioning in the car you can get completely different results. This shows how thorough design must be, because changing a few factors on the car can greatly vary its performance.

E.2 What else can you do with this simple car model and its numerical simulations? Be creative.

One thing we can do is create a feedback controller for car movement. Given the car's current state, we can solve an optimization in terms of the steering output (delta) in order to keep the car following a desired trajectory (X and Y), minimizing the error (which is a function of the difference between target X and predicted X given by the bicycle equation).

We can make estimations on the maximum curve for a highway depending on the maximum safe turn rate, utilizing tire and weather conditions. The maximum turn rate would be at where the "worst" handling car would go unstable at +30 km/h above the speed limit, under poor traction conditions (ex: rain or snow). The number derived here can provide a good upper bound on the maximum turn rate of curves on a highway. This information could be integrated in the decision making process of a self-driving car, depending on the detected weather conditions.

Integrate it in video games as the model used for the car physics. The surface friction of different materials and weather conditions in the game would determine the tire friction coefficients allowing different driving conditions. This would require computing the matrix defining the y velocity and yaw rate and switching between the models given the car's current speed.

Part F

There are several key factors that would need to be taken into consideration when planning to build a vehicle performance testing center in the Kitchener-Waterloo region. The goal of the testing facility is to evaluate the performance of the various Toyota vehicles, especially the RAV4. Ideally the facility would be able to test cornering stiffness, tire performance, and emergency handling under both standard and winter conditions.

Based on the findings in this report, winter conditions have a significant impact on the controllability of vehicles. Therefore, ample testing under these conditions is required to ensure the the safety of the vehicle. There should be a variety of testing tracks with different friction coefficients ranging from 0.8 to 0.1.

There should also be a high speed slalom track in order to test the vehicle performance at high speeds. As demonstrated in this report, the operation of the vehicle is highly dependent on the operating speed and must be rigorously tested to ensure safety. Depending on the build and design of the car, there will be a maximum operating speed before the stability of the car diverges and is no longer controllable. For example, the vehicle in this report had a maximum speed of 227.3 kilometers per hour.

Additionally, there should be enough standard tracks to test the car under various operating conditions, such as changing the amount and location of mass in the vehicle. Part C1 of this report outlines how changes in the amount of mass added to different locations in the vehicle can have a great impact on the performance of the car.

To meet all of the criteria mentioned above, a significant amount of land will be needed. The Motor Vehicle Testing Centre (MVTC), located in Blaine, Quebec, has over 25 kilometers of testing tracks and is located on 546 hectares of land [8]. Since this the proposed testing center would primarily focus on testing the RAV4 model, a smaller testing site would suffice. Somewhere between 100 to 150 acres should offer enough space to sufficiently test all the criteria mentioned.

Since Toyota Motor Manufacturing Canada (TMMC) is currently located in Cambridge, Ontario, it would be convenient to locate the testing facility somewhere in the Cambridge region. Some possibilities for locations include near the Breslau airport, or in Brant, Norfolk, or Halton County. For lower costs, developers of this facility might consider land in Ayr or New Hamburg where the land value is lower, even though these location will be farther from TMMC.

Buying a plot of land this large in the Kitchener-Waterloo-Cambridge regions will be costly. According to Century 21 Realty Inc, an acre of farmland in Ontario is being sold for 4,000 to 5,000 CAD. Since Cambridge is a relatively populated region, land value is higher, so it would be reasonable for an acre to sell for 5,000 CAD [9]. Therefore, a 150 acre site could sell for a minimum of 750,000 CAD. However, in southern Ontario, the a currently 100 acre sites for sale for upwards of 3.5 million CAD [10].

Additionally, a significant amount of money would be needed to develop and build on the land. Recently, Ford acquired the 163-acre Nanjing Test Center in China and budgeted 103 million USD to build the facility [11]. Similarly, the American Center for Mobility built a testing facility that cost 110 million USD in Willow Run, Michigan [12]. Therefore, the development of this testing facility in Cambridge would cost approximately 160 million CAD.

References

[1] Z. Pan, Project 1: ODEs in the Context of Vehicle Dynamics, ME 303: Advanced Engineering Mathematics, Dept. of Mech. and Mechatronics Eng., Univ. of Waterloo, Winter 2025.

[2] N. Hiromichi, "Preferable Front and Rear Weight Distributions of a Formula Car," *SAE International*, Feb 2006

[3] Science Direct, "Oversteer." `https://www.sciencedirect.com/topics/engineering/oversteer`, (accessed March 22, 2025).

[4] J. Chen, F. Yang, "Effect of Mass and Wheelbase on Car Dynamics," *RICAI*, April 2023

[5] H. Heisler, "8.4.5 Cornering stiffness (cornering power)," *Advanced Vehicle Technology*, 2002. doi:10.1016/b978-0-7506-5131-8.x5000-3.

[6] G. S. Vorotovic et al., *Determination of Cornering Stiffness Through Integration of A Mathematical Model and Real Vehicle Exploitation Parameters*. University of Belgrade.

[7] J. C.-B.- Contributor (2020), *Tyre Dynamics*, *Racecar Engineering*. Available at: `https://www.racecar-engineering.com/tech-explained/tyre-dynamics/` (Accessed: 16 March 2025).

[8] Canada Foundation for Innovation, "Motor Vehicle Test Centre (MVTC)." navigator.innovation.ca., `https://navigator.innovation.ca/en/facility/transport-canada/motor-vehicle-test-centre-mvtc`, (accessed March 22, 2025).

[9] Century 21 Realty Inc., "Ontario Land for Sale." ontariofarmsandland.com., `https://ontariofarmsandland.com/listing/ON/Simcoe/1861-Charlotteville-Road-5-N3Y-4K1/206321229`, (accessed March 22, 2025).

[10] Century 21 Realty Inc., "1861 Charlotteville Road 5 Simcoe, ON N3Y 4K1" ontariofarmsandland.com., `https://ontariofarmsandland.com/listing/ON/Simcoe/1861-Charlotteville-Road-5-N3Y-4K1/206321229`, (accessed March 22, 2025).

[11] A. Brzozowski, "Ford To Spend 103M On New Chinese Proving Ground" fordauthority.com., `https://fordauthority.com/2017/11/ford-to-spend-103m-on-new-chinese-proving-ground/`, (accessed March 22, 2025).

[12] K. Nagl., "American Center for Mobility awards 24 million contract for driverless car proving grounds" crainsdetroit.com., `https://www.crainsdetroit.com/article/20170615/news/631581/american-center-mobility-awards-24-million-contract-driverless-car?utm_source=chatgpt.com`, (accessed March 22, 2025).

[13] OpenAI., ChatGPT. `https://chatgpt.com/` Used for polishing report writing.

A Euler and RK4 Solvers

General Code for Euler's and RK4 ODE systems using the bicycle model. The full class definition contains more functions, but these are the parts relevant to Part A.1 and Part A.2.

```
class BicycleSolver:
    def __init__(self, m, a, b, C_alpha_f, C_alpha_r, I_z, u):
        self.a = a
        self.u = u
        self.A = np.array(
            [[-1 * (C_alpha_f + C_alpha_r) / (m * u), (-1 * (a * C_alpha_f - b * C_alpha_r) /
            [-1 * (a * C_alpha_f - b * C_alpha_r) / (I_z * u), -1 * (a**2 * C_alpha_f + b**2 *
        ,dtype=np.float64)

        self.B = np.array(
            [[C_alpha_f / m],
            [a*C_alpha_f / I_z]]
        ,dtype=np.float64)

    def solve(self, iterator, func, init_vector: np.ndarray, init_t, max_iter: int, step_size:
        """
        Generalized iterative solver that solves Y' = func using the specified iteration metho
        """
        if y_accel_hist != None:
            print("Storing accel values")

        y = init_vector.copy()
        res = []
        t = init_t
        for i in range(max_iter):
            y = iterator(func, y, t, step_size, y_accel_hist)
            t += step_size
            res.append(y)

        return y, np.array(res)

    def check_stability(self):
        eigenvalues = np.linalg.eig(self.A).eigenvalues
        for eigenvalue in eigenvalues:
            if eigenvalue > 0:
                return False

        return True

    @staticmethod
    def eulers_method(func, y_curr: np.ndarray, t, step_size: float, y_accel_hist=None) -> np.n
        """
        Euler's Method for n-dimensional systems.
```

```
        Args:
            func: F(Y, t). (n, 1) output shape
            y_curr: Y_i. (n, 1) shaped
            step_size: delta t. Scalar

        Returns:
            y_next: Y_i+1. (n, 1) shaped
        """
        y_next = y_curr + step_size * func(y_curr, t)

        if y_accel_hist:
            y_accel_hist.append(func(y_curr, t))

        return y_next

    @staticmethod
    def rk4(func, y_curr: np.ndarray, t, step_size: float, y_accel_hist=None) -> np.ndarray:
        k1: np.ndarray = func(y_curr, t)

        v2: np.ndarray = y_curr + step_size / 2 * k1
        k2: np.ndarray = func(v2, t + step_size / 2)

        v3: np.ndarray = y_curr + step_size / 2 * k2
        k3: np.ndarray = func(v3, t + step_size / 2)

        v4: np.ndarray = y_curr + step_size * k3
        k4: np.ndarray = func(v4, t + step_size)

        if y_accel_hist != None:
            y_accel_hist.append((k1 + 2*k2 + 2*k3 + k4) / 6)
            print("appended")

        return y_curr + step_size / 6 * (k1 + 2*k2 + 2*k3 + k4)

    @staticmethod
    def bicycle_model(A, B, y, delta) ->np.ndarray:
        assert((np.matmul(A, y) + B*delta).dtype == np.float64)
        return np.matmul(A, y) + B*delta

    @staticmethod
    def target(t):
        """
        Helper Function to test solver. Taken from the project manual, this computes the soluti
        using init [0,0] and table parameters.
        """
        return np.array(
            [[-13.0964*np.e**(-1 * 1.9745 * t) + 24.468*np.e**(-1*0.9839*t) - 11.3720],
```

```
            [-0.2496*np.e**(-1 * 1.9745 * t) - 0.69262*np.e**(-1*0.9839*t) + 0.9457]]
        )

    def visualize_results(self, histories, labels, init_t, t_final, max_iter, title="", titleb=
        plt.close('all')
        plt.figure()
        for i, hist in enumerate(histories):
            t_values = np.linspace(init_t, t_final, max_iter)
            if y_accel_hists:
                plt.plot(t_values, [val[0] for val in y_accel_hists[i]], label=labels[i])

            else:
                plt.plot(t_values, [val[0] for val in hist], label=labels[i])

        # Y Vel or Accel
        plt.xlabel('Time (s)')
        if y_accel_hists:
            plt.ylabel("Y Acceleration (m/s^2)")
        else:
            plt.ylabel("Y Vel (m/s)")

        plt.legend()
        plt.grid()
        plt.title(title)
        plt.show()

        plt.figure()

        # Yaw Rate
        for i, hist in enumerate(histories):
            t_values = np.linspace(init_t, t_final, max_iter)
            plt.plot(t_values, [val[1] for val in hist], label=labels[i])

        plt.xlabel('Time (s)')
        plt.ylabel("Yaw Rate (rads/s)")
        plt.legend()
        plt.grid()
        if titleb:
            plt.title(titleb)
        else:
            plt.title(title)
        plt.show()
```

B Stability Checker

The stability checking code for the solver. This function assumes the class has already been initialized, and simply computes eigenvalues for the A matrix in Equation (4). If any of the eigenvalues are positive, the function returns False for stability. True otherwise.

```
class BicycleSolver:
    def check_stability(self):
        eigenvalues = np.linalg.eig(self.A).eigenvalues
        for eigenvalue in eigenvalues:
            if eigenvalue > 0:
                return False

        return True

u_values_km = np.linspace(228, 230, 100)
u_values = [u * 1000 / 3600 for u in u_values_km]

s
```

The code used to numerically solve for the switching point between stable and unstable behaviour is below.

```
init_vector = np.array([[0],[0]], dtype=np.float64)
t_final = 20
init_t = 0
step_size = 0.2
max_iter = int(t_final / step_size)

histories = []
labels = []

y_accel_hists = []

for u in u_values:
    y_accel_hist = []
    solver = BicycleSolver(
        m = 1400,
        a = 1.14,
        b = 1.33,
        C_alpha_f = 25000,
        C_alpha_r = 21000,
        I_z = 2420,
        u = u
    )

    if solver.check_stability():
        print(f'{u * 3600/1000:.4f}km/h is stable')
    else:
```

```
        print(f'{u * 3600/1000:.4f}km/h is unstable')
        break
```

C Solving For When the System's Determinant is Zero

General Code for solving for when the determinant of the vehicle's homogeneous characteristic polynomial is zero. Assume that d, e, f and g correspond to the a_{11}, a_{12}, a_{21}, and a_{22} items of the system's matrix A in Equation (4).

```
from scipy.optimize import fsolve

def find_determinant_0(var, u_var, k_vars):
    """
    Generalized function to solve for any unknown variable.
    :list var: List containing the unknown variable.
    :string unknown_var: The variable to solve for.
    :dict k_vars: Dictionary of known variables.
    """
    # Update the known variables dictionary with the unknown one
        k_vars[u_var] = var[0]

    d = compute_d(k_vars["C_alpha_f"], k_vars["C_alpha_r"],

        k_vars["m"], k_vars["u"])

    e = compute_e(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["m"], k_vars["u"])

    f = compute_f(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["k"], k_vars["u"])

    g = compute_g(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["k"], k_vars["u"])

    lhs = (d*g)
    rhs = (e*f)

    return lhs - rhs


def find_unknown(unknown_var, known_vars, initial_guess=25000):
    solution = fsolve(find_determinant_0, [initial_guess], args=(unknown_var, known_vars))
    return solution[0]

# Example usage:
```

```
known_values = {
    "C_alpha_f": 25000,
    # "C_alpha_r": unknown (we want to solve for this)
    "m": 1400,
    "u": 75/3.6,
    "a": 1.14,
    "b": 1.33,
    "k": 2420
}

unknown_variable = "C_alpha_r"
result = find_unknown(unknown_variable, known_values)
```

D Solving For When The Discriminant is Zero

A modified version of the code presented in Appendix C to find when the determinant of a matrix is zero.

```
from scipy.optimize import fsolve

def find_discriminant_0(var, u_var, k_vars):
    """
    Generalized function to solve for any unknown variable.
    :list var: List containing the unknown variable.
    :string unknown_var: The variable to solve for.
    :dict k_vars: Dictionary of known variables.
    """
    # Update the known variables dictionary with the unknown one
        k_vars[u_var] = var[0]

    d = compute_d(k_vars["C_alpha_f"], k_vars["C_alpha_r"],

        k_vars["m"], k_vars["u"])

    e = compute_e(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["m"], k_vars["u"])

    f = compute_f(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["k"], k_vars["u"])

    g = compute_g(k_vars["a"], k_vars["C_alpha_f"], k_vars["b"],

        k_vars["C_alpha_r"], k_vars["k"], k_vars["u"])

    lhs = (d + g) ** 2
    rhs = 4 * (d * g - e * f)
```

```
    return lhs - rhs


def find_unknown(unknown_var, known_vars, initial_guess=25000):
    solution = fsolve(find_discriminant_0, [initial_guess], args=(unknown_var, known_vars))
    return solution[0]
# Example usage:
known_values = {
    "C_alpha_f": 25000,
    # "C_alpha_r": unknown (we want to solve for this)
    "m": 1400,
    "u": 75/3.6,
    "a": 1.14,
    "b": 1.33,
    "k": 2420
}

unknown_variable = "C_alpha_r"
result = find_unknown(unknown_variable, known_values)
```

E Equations

$$\dot{y} = -u\dot{\psi} - \frac{C_{\alpha f}}{m}\left(\frac{\dot{y} + a\dot{\psi}}{u} - \delta\right) - \frac{C_{\alpha r}}{m}\left(\frac{\dot{y} - b\dot{\psi}}{u}\right) \tag{1}$$

$$\ddot{\psi} = -\frac{1}{I_z}\left(aC_{\alpha f}\left(\dot{y} + \frac{a\dot{\psi}}{u} - \delta\right) - bC_{\alpha r}\left(\dot{y} - \frac{b\dot{\psi}}{u}\right)\right) \tag{2}$$

$$\begin{bmatrix}\ddot{y}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}\frac{-C_{\alpha f}-C_{\alpha r}}{mu} & \frac{-aC_{\alpha f}+bC_{\alpha r}}{mu} - u\\ \frac{-aC_{\alpha f}+bC_{\alpha r}}{I_z u} & \frac{-a^2C_{\alpha f}-b^2C_{\alpha r}}{I_z u}\end{bmatrix}\begin{bmatrix}\dot{y}\\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}\frac{C_{\alpha f}}{m}\\ \frac{aC_{\alpha f}}{I_z}\end{bmatrix}\delta \tag{3}$$

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

$$\dot{X} = u\cos(\psi) - (\dot{y} + a\dot{\psi})\sin(\psi) \tag{5}$$

$$\dot{Y} = (\dot{y} + a\dot{\psi})\cos(\psi) + u\sin(\psi) \tag{6}$$

$$x(t) = \begin{bmatrix}\dot{y}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}-13.0964e^{-1.9745t} + 24.4684e^{-0.9839t} - 11.3720\\ -0.2496e^{-1.9745t} - 0.6962e^{-0.9839t} + 0.9457\end{bmatrix} \tag{7}$$

F Ownership:

1. Mark Do: A, E, eigen proof
2. Mckale Chung: D, E, eigen proof, Part 1
3. Carmelli Dao: C, F
4. Miekale Smith: B3, E, intro
5. Odette Beaudin: B, C, F