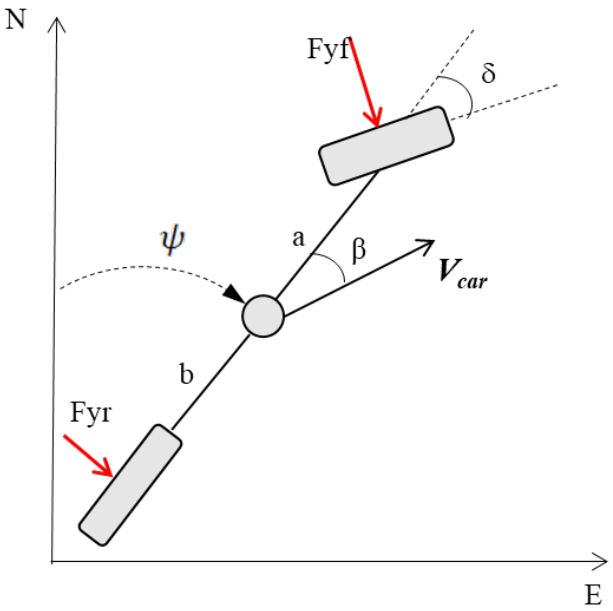
Simple Yaw Vehicle Model (DRAFT)

In order to derive the yaw dynamics model, you need to sum forces in the lateral direction and sum moments about the center of gravity of the vehicle to find equations of motion for yaw accerelation ($\ddot{\psi}$) and lateral acceleration ($\dot{V}_y$). Then combine the equations to develop a transfer function with input steer angle (δ) and output yaw (ψ), or alternatively you can place the equations into a state space format. Use the notes below to develop your model. Derive the system model for the autopilot system using the modeling steps and Newton's laws of motion.



The tire force is related to the lateral velocity at the tire through the following approximation:

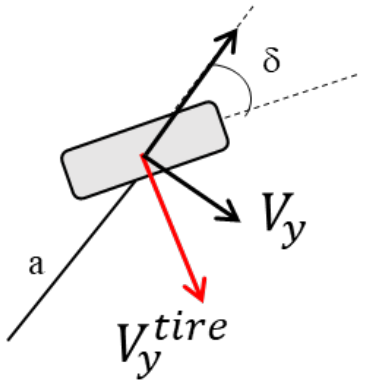
$$Fy = \frac{-C}{V_{car}} \times V_y^{tire}$$

You must include centripetal acceleration when applying Newton's lateral equations:

$$\Sigma F_y = m\ddot{y} = m\left(V_{car}\dot{\psi} + \dot{V}_y\right)$$

Note that V_y^{tire} is the lateral velocity in the tire frame (as shown in the figure below):

$$V_x \cong V_{car}$$



The velocity at the tire (V_y^{tire}) can be found by recalling that $\vec{V}_B = \vec{V}_A + \vec{\omega}$ x $\vec{r}_{A/B}$.

A new "run_car" script has been developed for testing your controllers in simulation for either the Lincoln MKZ or the Indianapolis Autonomous Challenge (IAC) Dallara Indy Lights vechicle. The new script will allow you to select predefine routes used for testing the controller. You can also test your control capability with constant and sinusoidal desired headings or lateral position references.

Note – *run_car.p* includes an inner loop steering feedback control system that you can decide to include or not. Should you decide to include it, you will need to identify the inner loop steering dynamics (running the steering system while stopped, i.e. Vel=0, is a good way to do this). You may also use the homogeneous response (i.e. $\delta = 0$) to isolate the lateral vehicle response from the steering response to validate your vehicle yaw dynamics.

The table below contains the approximate values for the parameters needed to model the yaw dynamics of the car.

Table 1: Parameters for the Dallara Indy Lights Model

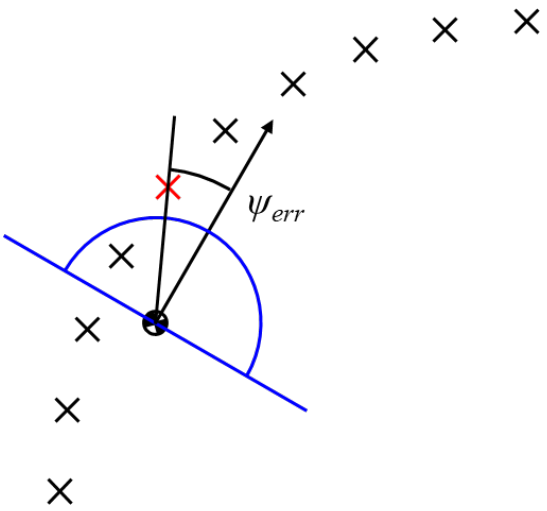
Parameter	Name	Units	Value
a	Front CG distance	M	1.72
b	Rear CG distance	M	1.25
C_f	Front Tire Cornering Stiffness	N/rad	143,000
C_r	Rear Tire Cornering Stiffness	N/rad	143,000
I_z	Yaw Mass Moment of Inertia	Kgm^2	1200
m	Vehicle Mass	Kg	720
N	Steering Column Gear Ratio		12

Table 2: Parameters for the Lincoln MKZ Model

Parameter	Name	Units	Value
a	Front CG distance	m	1.257
b	Rear CG distance	m	1.593
C_f	Front Tire Cornering Stiffness	N/rad	120,000
C_r	Rear Tire Cornering Stiffness	N/rad	184,600
I_z	Yaw Mass Moment of Inertia	Kgm^2	4292
m	Vehicle Mass	kg	1856
N	Steering Column Gear Ratio		14.8

Note that the combination of rack and pinion as well as the steering kinematics for the wheel results in an overall steering ratio of N for the car (i.e. the motors turns N degrees for 1 degree the tires turn). The steer angle at the tire is measured by using a quadrature optical encoder (similar to the data sheet provide) that has 500 pulses per revolution. Reading both the high and low transitions of the A and B channels of the encoder results in a resolution of 2000 counts/revolution and when combined with the steering ratio results in a measure resolution of the steering wheel of 24,000 counts/revolution (or 0.015 deg). The angle reported by run_IAC_car.p is the steer angle at the tire. Be sure to apply the correct scaling if analyzing the motor/steering wheel angle.

You may also use *run_car.p to* evaluate the performance of your control on real-world scenarios. You can choose a trajectory by entering 1, 2, or 3 as the last input to *run_IAC_car.p* (example code). The function will return a "waypoint" for you to steer the vehicle towards to track a path. The figure below shows a series of waypoints that the vehicle will attempt to drive to follow a path. The red X is the waypoint that the vehicle should drive towards now (this is the point returned by *run_car.p*). The heading error (ψ_{err}) is the difference between the current heading and the desired heading towards the red X. The heading error can be calculated as shown in the equation below.



$$\psi_{err} = \tan^{-1}\left(\frac{E_{des} - E}{N_{des} - N}\right) - \psi$$

Note: for the arctan, use atan2. Also, this heading error, ψ_{err}, needs to be wrapped (i.e. limited) to $\pm$ 180 degrees using *wrap_angle.m* function provided on canvas or the class website. Be sure to record the vehicle position returned by *run_car.p* to evaluate the effectiveness of the controller. Using the instructions provided you may want to plot the vehicle position on Google Earth.

Note that lateral position can be defined from North or from any arbitrary line by rotating the car into that frame by the desired heading (ψ_{des}) as shown below. The GPS heading provided will be measured from North in radians.

Alternatively you can measure the lateral error to the waypoints and control lateral vehicle position knowing that:

$$\dot{y}_{err} = V_x \sin(\psi) + V_y \cos(\psi)$$