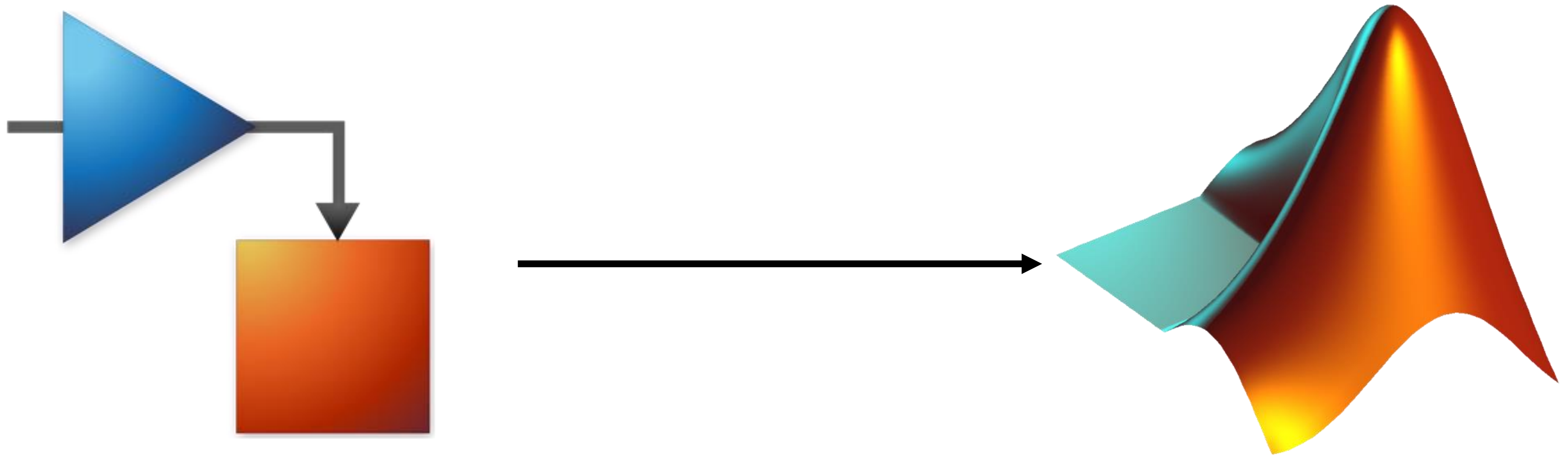
ME452 Vehicle dynamics project

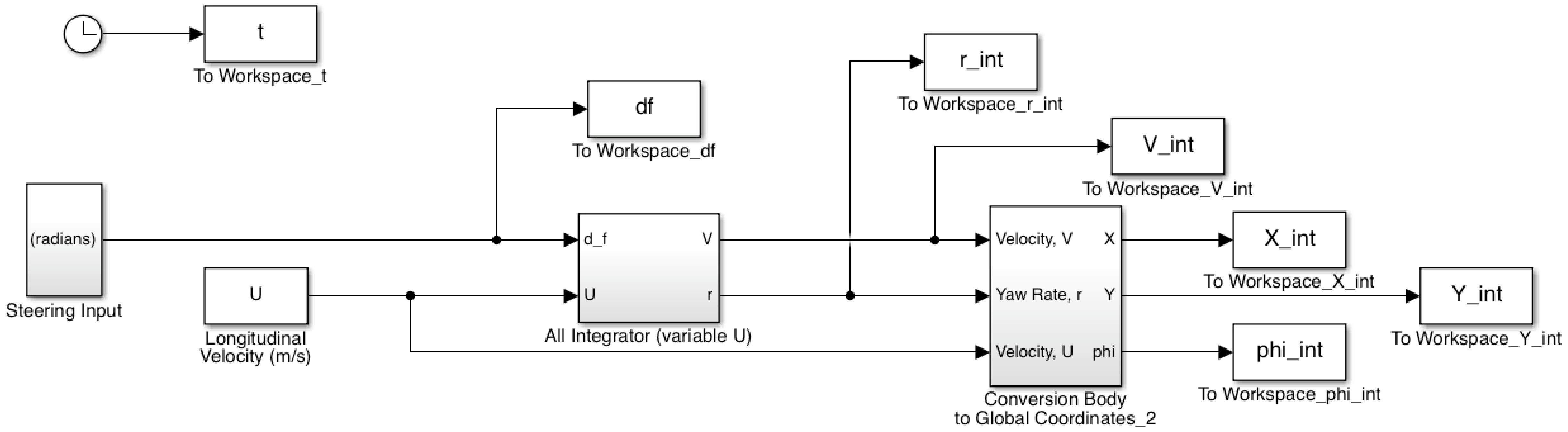
Satya Prasad Maddipatla and Wushuang Bai

2021-05-02

MATLAB Simulink is strong in simulation. However, in many engineering implementations a system is required to be implemented outside MATLAB environment considering factors such as cost and computational efficiency.
The objective of this document is to setup a simple example converting a Simulink model into an equivalent MATLAB code version.

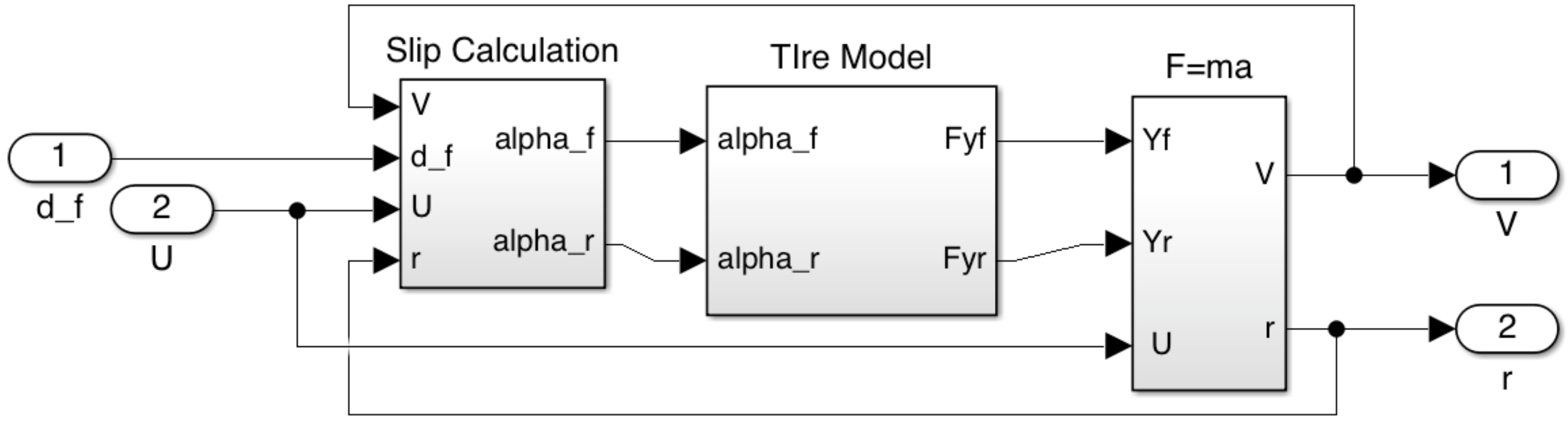


This example uses a vehicle bicycle model which takes steering angle and longitudinal velocity as inputs to generate vehicle position and yaw in global coordinates.

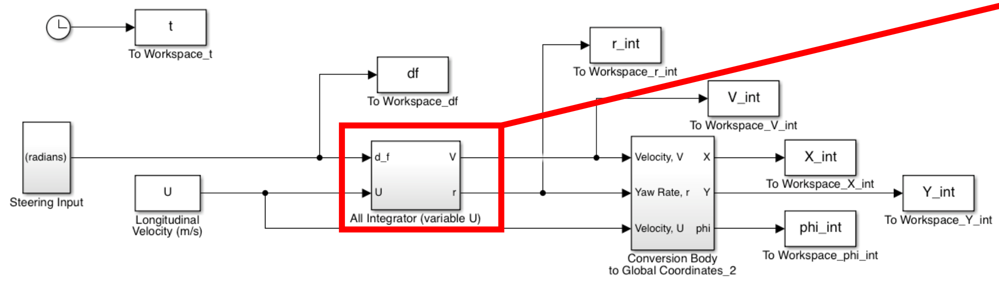


https://github.com/msvsprasad/ME452_BicycleModelProject.git

The all-integrator block takes steering angle and longitudinal velocity as input to generate lateral velocity and yaw rate. This includes 3 subsystems to model tire slip, tire lateral force and equation of motion.

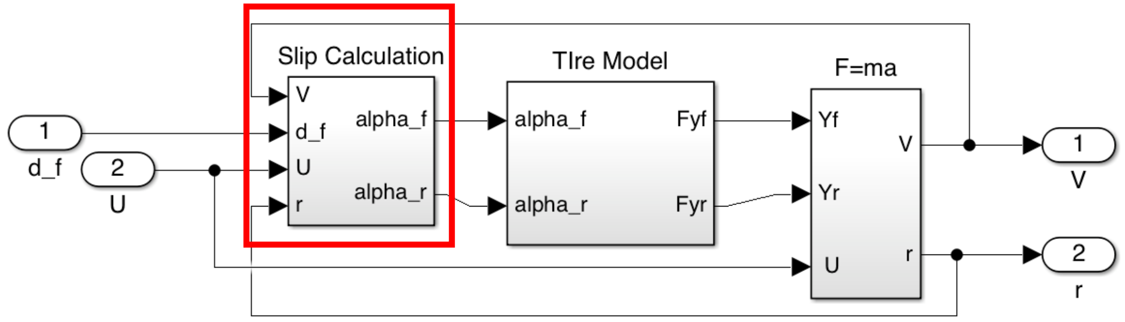


Subsystems in all integrator block



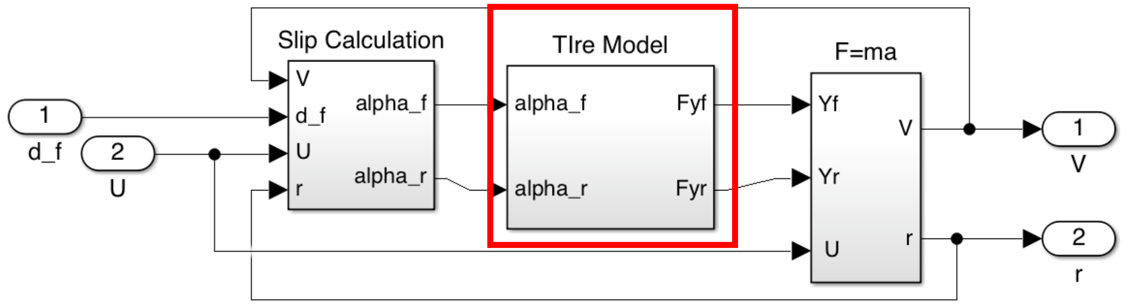
Bicycle model

fcn_slipAngles estimates slip angles.



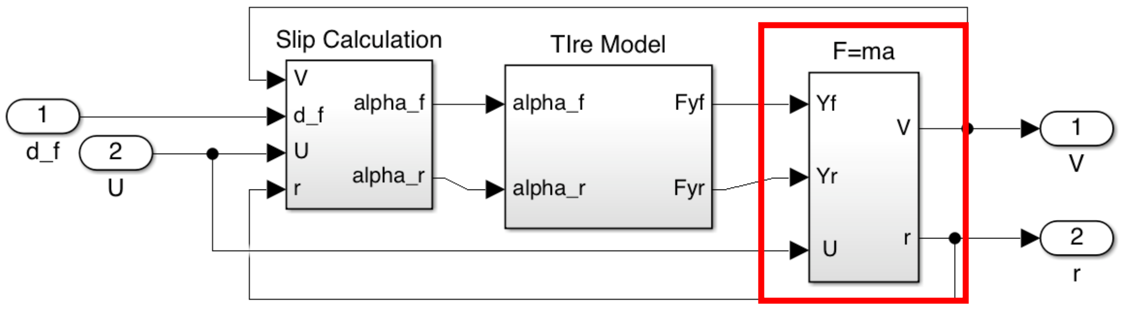
```
function alpha = fcn_slipAngles(U, V, r, delta_f, vehicle)
%% fcn_slipAngles
%   This function computes slip-angles for front and rear wheels. It
%   assumes a bicycle model.

%% Calculate Slip-Angles
alpha_f = (V+vehicle.a*r)/U - delta_f;
alpha_r = (V-vehicle.b*r)/U;
alpha   = [alpha_f; alpha_r];

end
```

fcn_lateralForces estimates lateral forces.



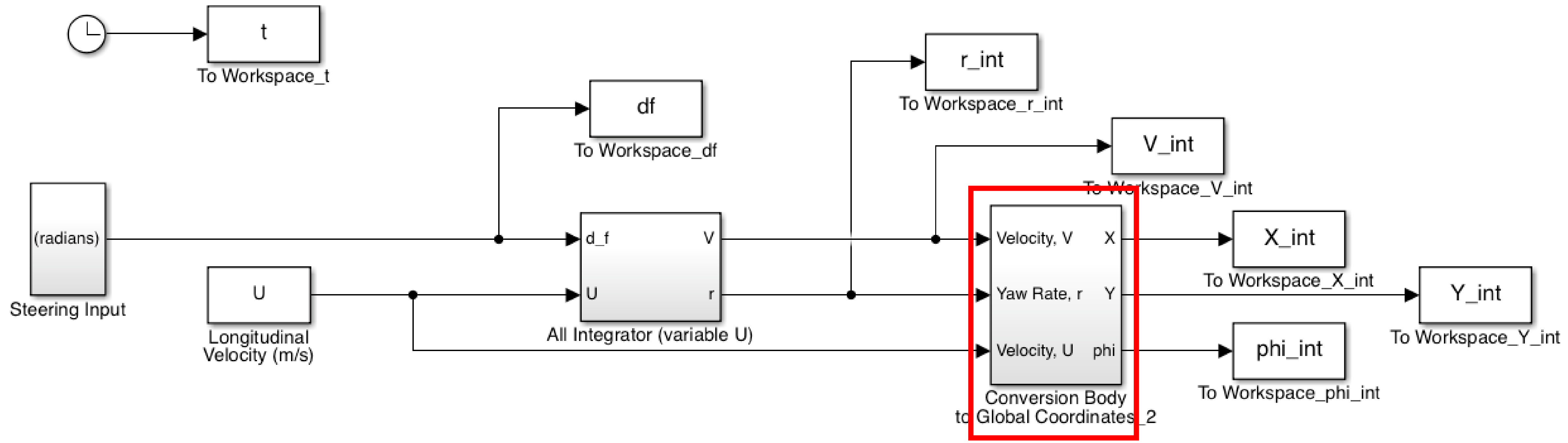
```
function Fy = fcn_lateralForces(alpha, vehicle)
%% fcn_lateralForces
%   This function computes lateral-forces at front and rear wheels. It
%   assumes a linear model.
%
%% Calculate Lateral Tire Forces
Fyf = vehicle.Caf*alpha(1);
Fyr = vehicle.Car*alpha(2);
Fy  = [Fyf; Fyr];

end
```

fcn_lateralDynamics implements Newton’s II law of motion.

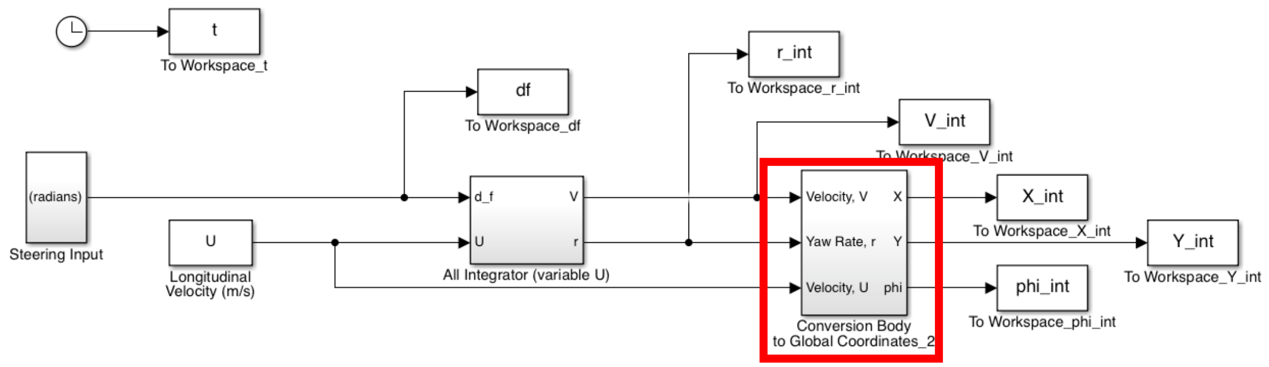


```
function dydt = fcn_lateralDynamics(~, y, U, Fy, vehicle)
%% fcn_lateralDynamics
%   This function calculates lateral acceleration and yaw acceleration.
%
%% Calculate Lateral acceleration and Yaw acceleration
r = y(2);
dVdt = (1/vehicle.m)*(Fy(1)+Fy(2))-r*U;
drdt = (1/vehicle.Iz)*(vehicle.a*Fy(1)-vehicle.b*Fy(2));
dydt = [dVdt; drdt];

end
```

The Conversion Body to Global Coordinates block takes longitudinal velocity, lateral velocity and yaw rate to generate X,Y position and yaw angle in global coordinate system.

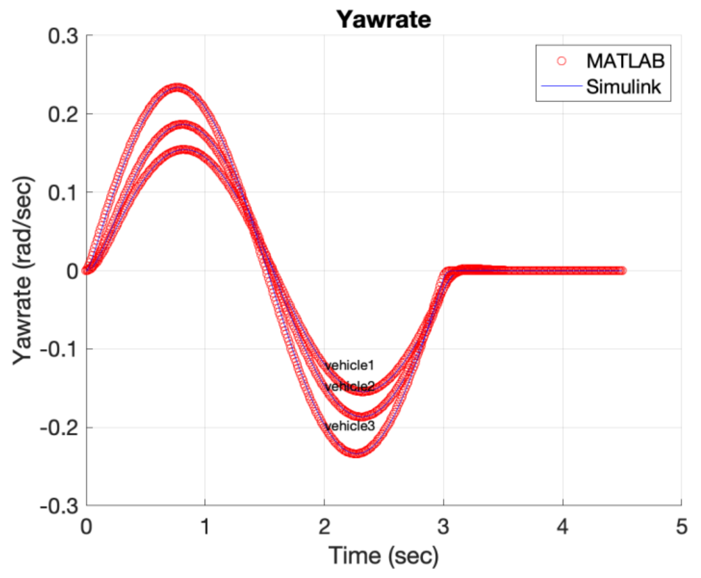


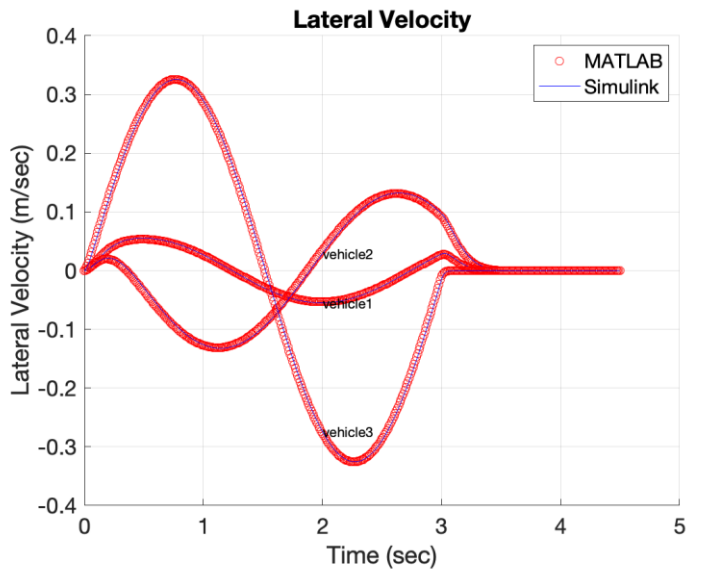
fcn_body2globalCoordinates converts from body to global Coordinates.

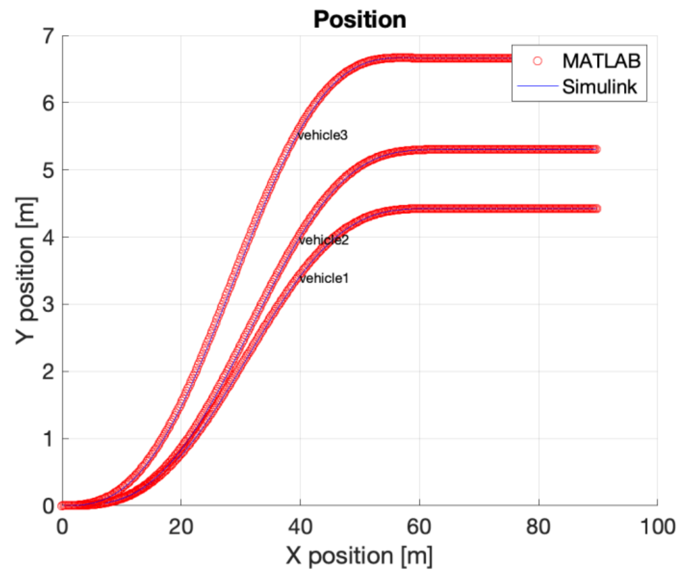


```
function dydt = fcn_body2globalCoordinates(~, y, U, V, r)
%% fcn_body2globalCoordinates
%   This function calculates velocites in global coordinates.
%
%% Calculate velocities in Global coordinates
Phi = y(3);

dXdt   = U*cos(Phi)-V*sin(Phi);
dYdt   = U*sin(Phi)+V*cos(Phi);
dPhidt = r;
dydt   = [dXdt; dYdt; dPhidt];

end
```

Finally, the results from MATLAB code are compared with the results from Simulink to ensure to be the same. This has been tested using three different vehicle parameters.







https://github.com/msvsprasad/ME452_BicycleModelProject.git