Report

Abstract:

The following documents a simple auto braking system using distance measurement to control a car model and also outlines a benchmark for it to meet. The system measures distance between it (the 'ego' vehicle) and the car in front of it (the 'lead' vehicle) and calculates both the time to collision and the amount of braking force to apply. There are three levels of braking force which are calculated using a Stateflow controller. It uses examples from MathWorks which can be found in the references at the bottom of this paper as well as the 'car_model' from the provided notes

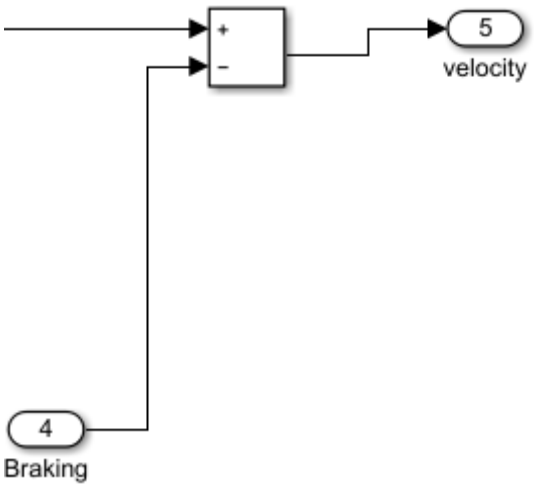
ADAS AEB Benchmark

The goals of this model are as follows:

- Use sensors to determine vehicle stopping times using either radar or lidar.
- Potentially use sensor fusion to improve performance
- Control an existing car model with simulated data and observe the AEB system performance

Overview

The system has two main pieces, the car model and the AEB system controller. The car model was taken from the Moodle page for this module and modified with braking support.

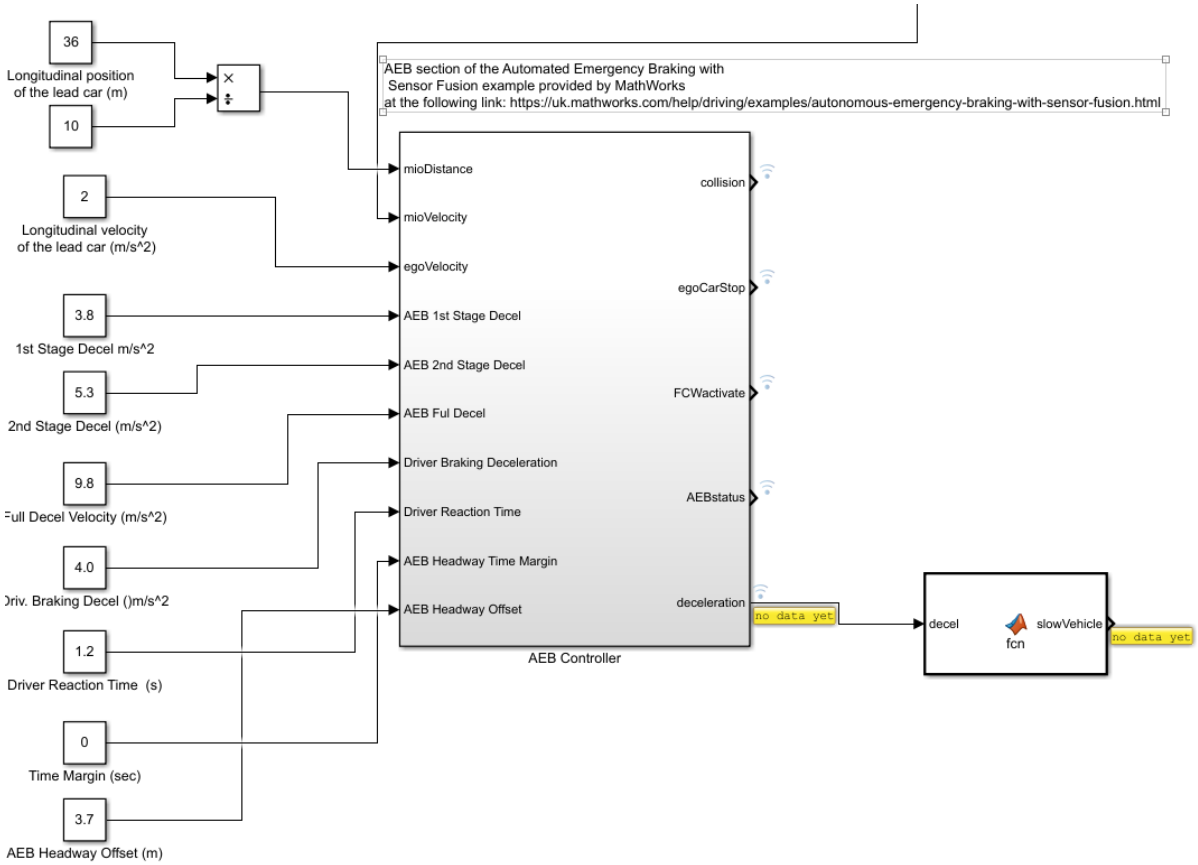


Braking is simply subtracted from the velocity of the car. This model is not ideal as without using a floor function, it can result in negative values. A floor function was defined in MATLAB and implemented to solve this problem.

```
function floorVelocity = fcn(velocity)
if velocity<0
    floorVelocity = 0;
else
    floorVelocity = velocity;
end
```

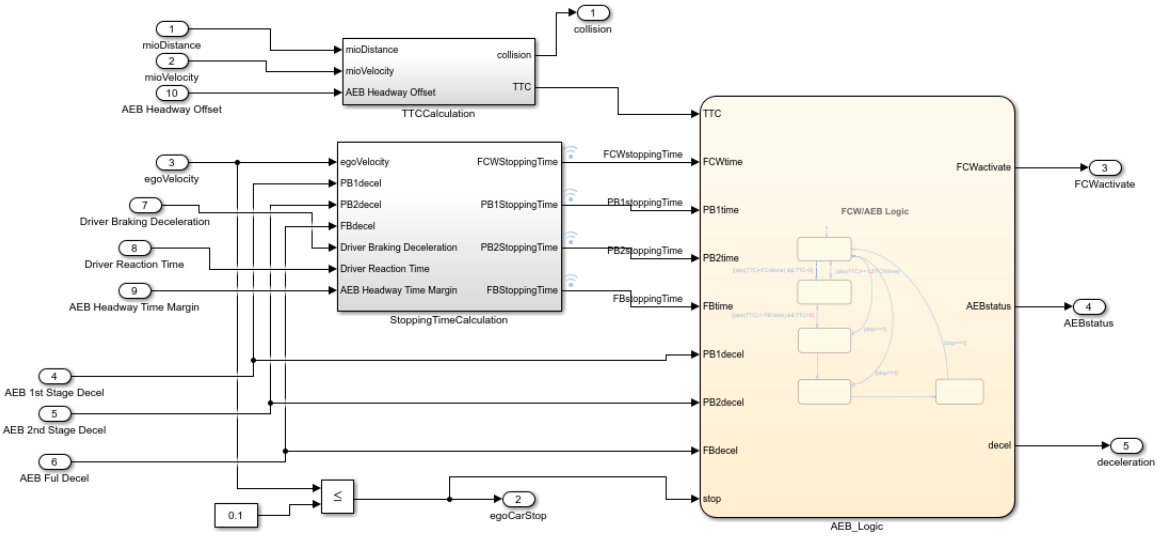
This will output 0 if the input ('velocity') reaches a negative value.

The velocity is fed into the second part of the system, the AEB controller. This is an overview



The algorithms were taken from the MATLAB demo 'AEBTestBenchExample'. The blocks were modified to have external constants to define parameters inside the equations and then inserted into this model.

This is an overview of the internal blocks of the AEB controller. It has three parts, two more blocks and a Stateflow controller to control the level of braking applied



The TTC Calculation block uses the following formula to determine the time to collision (TTC)

TTC = d-H/abs(v)

Where d = distance to the lead vehicle, H = headway offset and v = lead vehicle velocity.

This block also calculates the time to forward collision T_{FCW}

$T_{FCW} = D_{react} + (V_{ego} /$ Driver Braking Deceleration) where D_{react} is driver reaction time in seconds and V_{ego} is the velocity of the ego vehicle in meters per second squared

The times for each of the braking stages are the same and are calculated with this formula:

$T_{partial} = (V_{ego} / d_{PB})$ + headway time margin in seconds

Where $T_{partial}$ = stopping time for braking stage, V_{ego} = velocity of ego vehicle

The simulation uses a state machine to compare these times to the TTC to determine how much braking force (if any) to apply

The logic is as follows:

- Where (abs(TTC) < FCWtime) and TTC<0 – warn driver of impending collision
- Where (abs(TTC) < PB1time) and TTC<0 – enable first braking stage
- Where (abs(TTC) < PB2time) and TTC<0 – enable second braking stage
- Where (abs(TTC) < FBtime) and TTC<0 – apply brakes fully

Testing

By modifying the constants, we can trigger the auto braking. The system will trigger with a velocity value above 7. Below the model has been set up with the following parameters

- Gear : 5
- Throttle Gain : 0.089
- Gradient : 0
- Distance to ego vehicle : 100
- Lead car longitudinal velocity : 2
- 1st stage deceleration : 3.8
- 2nd stage deceleration: 5.3
- 3rd stage deceleration: 9.8
- Driver Braking deceleration: 4.0
- Reaction time : 1.2
- Time margin : 0
- Headway Offset : 3.7

The collision warning is activated when the distance goes below 37 meters. However it fails to slow the vehicle because the braking output is not routed back to the car. Doing so results in the following error:

```
'cfarrell_assign2OLD/AEB Controller/AEB_Logic' updates persistent or state
variables while computing outputs, therefore it cannot be used in an algebraic
loop.
```

I could not resolve the above issue and because of this I could not implement a proper closed-loop control system to perform the above tasks.

Therefore the system fails to meet any of it's goals.

Conclusion

My lack of experience using Simulink and inability to rectify problems with the model resulted in the system falling long short of it's goals. It does not read in external sensor data or accurately model vehicle braking dynamics. Also the choice if using the Mathworks examples was not a good choice as it's AEB controller implementation uses *relative* distances of the lead vehicle and not distances of the ego vehicle.

If I were to reimplement this, I would use the formulae presented in the Mathworks examples as a guide and design my own implementation of the system based on those.