Stop and Go Cruise Control

Final Report

Submitted to:
Dr. ir. Paul J. Th. Venhovens
BMW AG, Abteilung EW-10
80788 MüFcnchen
Tel: +49-(0)89 382-46888
Fax: +49-(0)89 382-44988

Submitted by:
CALIFORNIA PATH
University of California
Berkeley, CA 94720. USA
Dr. J.K. Hedrick, P.I.
Dr. Datta Godbole
Dr. Rajesh Rajamani
Pete Seiler

January 18, 1999

Contents

List of Figures

Abstract

A simulation model of the BMW experimental vehicle was constructed using the Simulink simulation package. This half car vehicle model was validated using data from a series of experiments completed at Crow's Landing. Only minor modification of the vehicle parameters was necessary to obtain good correspondence between the actual and simulated data. Next, a stop and go control law was designed using nonlinear control techniques. However this control law cannot be applied in all traffic situations, so a supervisor layer was designed to determine the traffic flow condition and adjust the controller appropriately. The unified control law is able perform the tasks of an Adaptive Cruise Controller in a variety of traffic flow conditions. This unified controller was tested with several traffic flow scenarios using the previously constructed simulation model. Finally, the developed controller was constructed in C-code. This C-code was debugged via comparisons with the simulink controller.

1 Vehicle Model

A simulation model of the BMW experimental vehicle was constructed using the Simulink simulation package. The utility of the simulation model lies in the ability to rapidly test Stop and Go ACC algorithms in a variety of situations. Simulation testing is not a replacement for the final tuning of the controller since unmodeled dynamics may affect the performance upon actual implementation. However, simulation-based design is useful because the qualitative performance of the controller as well as the general effect of controller gains can be determined.

The Simulink model is a half-car representation of the experimental vehicle. The lateral vehicle dynamics have been neglected since ACC algorithms can be adequately simulated under strictly longitudinal conditions. In the following sections, the static relations and dynamic equations used to create the vehicle model will be described.

1.1 Longitudinal Vehicle Model

The longitudinal dynamic equations are derived from force and torque balances on the vehicle chassis. The freebody diagram of the vehicle, Figure 1, displays the forces under consideration:

1. $F_d = c_D \cdot v^2$ = Wind drag force
2. F_{rr} = Rolling resistance force
3. F_f, F_r = Front, rear tractive forces
4. $W = m \cdot g$ = Vehicle weight
5. N_f, N_r = Front, rear normal forces

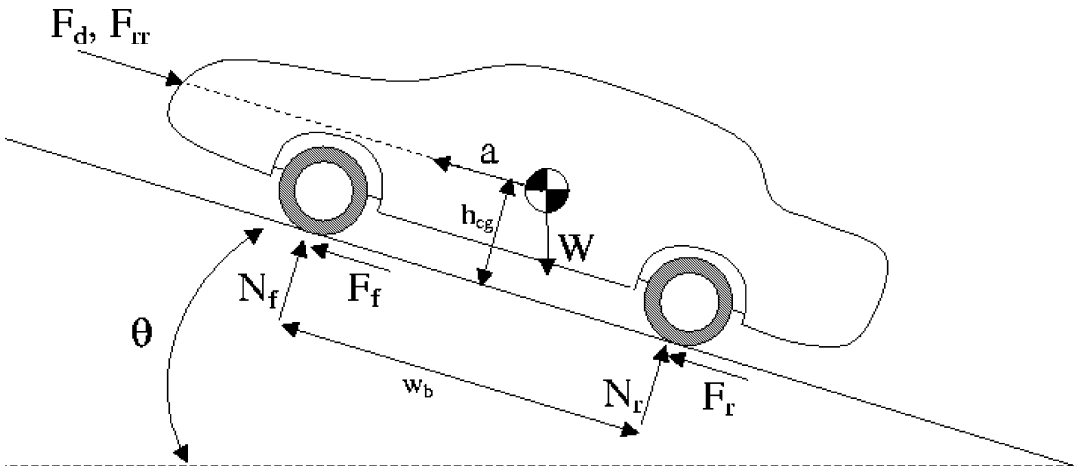

Figure 1: Vehicle free body diagram

Application of Newton's 2^{nd} law in the longitudinal direction results in:

$$m \cdot a = F_f + F_r - F_{rr} - F_d - W \cdot sin\Theta \tag{1}$$

The drag force is proportional to the square of the vehicle speed, while the rolling resistance is a constant (with respect to the vehicle dynamics) which is proportional to vehicle mass. The generation of tractive forces will be discussed in the next section.

Next, we can determine the normal forces on each tire by summing torques separately about the front and rear tire contact points. The drag and rolling resistances forces in Figure 1 are depicted acting through the

center of gravity, but it is assumed that they do not cause vehicle pitch. Thus, the static load distribution modified by acceleration and grade induced vehicle pitch is given by:

$$N_f = N_{s,f} \cdot cos\Theta - m \cdot g\frac{h_{cg}}{w_b} \cdot sin\Theta - m \cdot a \cdot \frac{h_{cg}}{w_b} \tag{2}$$

$$N_r = N_{s,r} \cdot cos\Theta + m \cdot g\frac{h_{cg}}{w_b} \cdot sin\Theta + m \cdot a \cdot \frac{h_{cg}}{w_b} \tag{3}$$

where $N_{s,f}$ and $N_{s,r}$ are the front and rear static normal forces on level ground, Θ is the grade angle, h_{cg} is the height of the vehicle center of gravity, and w_b is the wheelbase. The second term on the right hand side is the pitch due to the grade and the last term is the pitch due to vehicle acceleration.

1.2 Wheel Dynamics

The torques/forces acting on the rear wheel are shown in Figure 2:

1. $\tau_{b,r}$ = Rear brake torque
2. τ_d = Engine drive torque seen at the rear axle
3. F_r = Rear tractive force
4. N_r = Rear normal forces

It should be noted that the reaction force from the axle to the wheel has been omitted since it does not apply a torque about the center of the wheel.

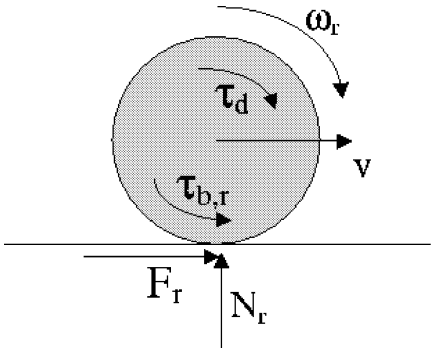

Figure 2: Rear wheel free body diagram

A torque balance about the rear wheel center yields the following equation:

$$\bar{J}_r \cdot \dot{\omega}_r = \tau_d - \tau_{b,r} - F_r \cdot r \tag{4}$$

where r is the tire radius and $\bar{J}_r$ is the sum of the rear axle inertia and the engine/transmission inertia reflected to the rear wheels. The front wheel dynamic equation will be similar to Equation 4 with three adjustments. First, the experimental vehicle is rear wheel driven, hence the τ_d term will not be present. Second, the inertia in the front wheel dynamic equation, J_f, will solely be due to the front axle inertia. Finally, $\tau_{b,f} \neq \tau_{b,r}$ since we have modeled the brake proportioning valve. The proportioning valve model simply splits brake pressure between the wheels as follows:

$$\tau_{b,r} = k_{pv} \cdot k_b \cdot P_{mc} \tag{5}$$

$$\tau_{b,f} = (1 - k_{pv}) \cdot k_b \cdot P_{mc} \tag{6}$$

where P_{mc} is the pressure at the master cylinder, k_{pv} is the percentage of the brake pressure seen at the rear wheels, and k_b is the brake gain (from brake pressure to brake torque). The brake gain, whether for a drum or disc brake, is highly uncertain and varies greatly depending on the operating conditions. In spite of this,

it is assumed to be the same for the front and rear brakes in the model. The brake gain was estimated using the vehicle test data taken at Crow's Landing.

The tractive forces on each wheel are then computed using a dynamic tire model. Typically the Bakker-Pacejka "Magic" formula is used to compute longitudinal tractive force as an algebraic function of tire slip. In braking situations, longitudinal slip is defined to be:

$$\lambda = \frac{r \cdot \omega - v}{v} \tag{7}$$

This tire slip definition has a singularity as v approaches zero which results in numerical integration difficulties, especially when simulating the low velocity trajectories present in the Stop and Go scenario.

To avoid this problem, define dynamic slip [3, 4] as:

$$\dot{\kappa} = -\frac{v - r \cdot \omega + |v| \cdot \kappa}{\sigma} \tag{8}$$

where σ, the relaxation length, is a function of the tire normal force. This dynamic slip is defined independently for each wheel. Notice that this is a stable first order system with $-(v - r \cdot \omega)/\sigma$ as the forcing input. In steady state (with a constant input), $\kappa \to (r \cdot \omega - v)/|v|$. The dynamic tire slip converges to the slip definition given in Equation 7 but it does not have the singularity at $v = 0$. The Bakker-Pacejka "Magic" formula uses this dynamic tire slip to compute the tractive force.

1.3 Unlocked Engine Dynamics

When the torque converter is not locked, the engine speed is governed by:

$$J_e \cdot \dot{\omega}_e = T_e - T_{pump} \tag{9}$$

J_e is the engine inertia, ω_e is the engine speed, T_e is the torque produced by the engine and T_{pump} is the input torque to the torque converter. A steady state engine map supplied by BMW, Figure 3, is used to compute the engine torque, T_e, as a function of throttle angle, α, and engine speed, ω_e.

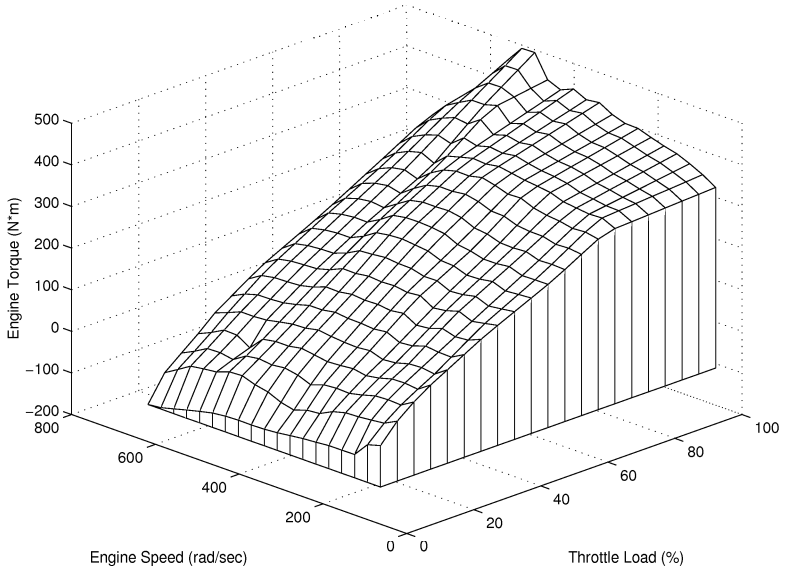

Figure 3: Engine map

1.4 Torque Converter Model

The engine torque, T_e, is transferred to the wheels through the transmission which consists of the gear box and a torque converter. The torque converter provides coupling between the engine and the gearbox using

fluid to transfer torque. The fluid dynamics are fast and the dynamics associated with torque production can be neglected. Thus the input and output of the torque converter can be obtained from steady state maps.

The torque converter data file obtained from BMW contains data columns for speed ratio, torque ratio, and pump (input) torque. The initial torque converter model computed the input torque as a function of speed ratio using this data. Then, the model used the torque ratio (computed as a function of speed ratio) and the input torque to compute the output torque. Consequently, the input torque depended only on speed ratio and not on input speed.

As discussed in Section 3.5, the model performance did not agree with the test data when this torque converter model was used. Hence, the following torque converter model, based on a description given in "Theory of Ground Vehicles" by J.Y. Wong [5], was eventually implemented in the Simulink model. The torque converter model uses the following relations:

1. $C_{sr} = \frac{\omega_{out}}{\omega_{in}}$ = speed ratio
2. $C_{tr} = \frac{T_{out}}{T_{in}}$ = torque ratio
3. $K_{tc} = \frac{\sqrt{T_{in}}}{\omega_{in}}$ = capacity factor

The torque converter model consists of the following calculations. First, the input and output speeds are used to compute the speed ratio. Then, the torque ratio and capacity factor are computed using maps, Figure 4, which are functions of the speed ratio. The capacity factor and input speed can then be used to compute the input torque. Finally, the input torque and torque ratio can be used to compute the output torque. The model performance was closer to the actual vehicle performance when this model was used (see Section 3.5).

This model shows a potential difficulty of Stop and Go cruise control design. In the vehicle following problem, it is frequently assumed that there is no tire slip and that the torque converter is locked. These assumptions result in engine speed and vehicle speed being proportional. However, the locked torque converter assumption does no hold in the Stop and Go application. Hence, it must be determined whether or not knowledge of the torque converter is needed by the controller. In the maps (Figure 4), the locked torque converter assumption is represented by a point on the Torque Ratio map with $C_{sr} = 1$, $C_{tr} = 1$. The maps display the highly nonlinear behavior of the torque converter when unlocked.

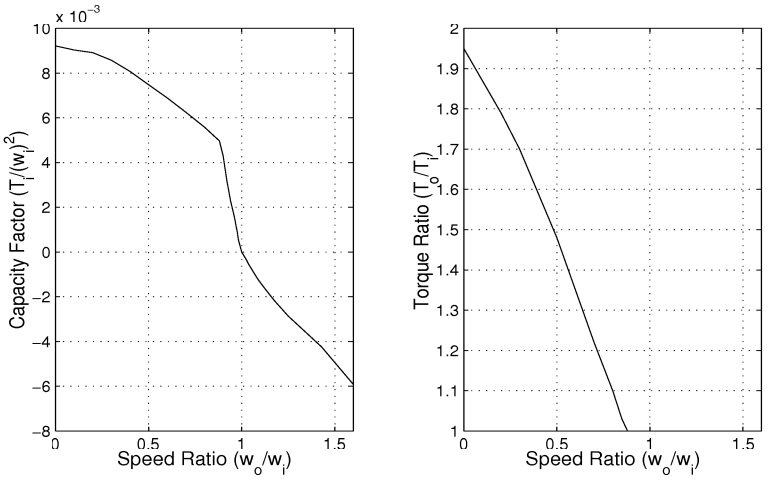

Figure 4: Torque converter maps

1.5 Lockup Logic

Under steady state conditions the torque converter is mechanically locked. In this mode the input and output torques and speeds of the torque converter are equal:

$$T_{pump} = T_{turb} = T_e \tag{10}$$

$$\omega_{pump} = \omega_{turb} = \omega_e \tag{11}$$

Therefore, when the torque converter is locked, the engine speed is algebraically related to the vehicle speed via the gear ratios. Thus the engine state is removed during the locked torque converter mode and the engine torque is applied to the rear wheels without dynamics. BMW supplied maps which give the torque converter lock/unlock status as a function of throttle load, engine speed and gear. These maps will be discussed in Section 3.6.

1.6 Gear Shift Logic

The purpose of the gear box is to match engine torque and speed with the torque and speed demanded by the driving conditions. There are two gear ratios which need to be considered: the final drive ratio, r_{drive}, and a changeable gear ratio, r_{gear}. BMW supplied gear shift maps were used to predict gear shifts as a function of throttle load and shaft (gearbox output) speed. For a given r_{gear}, the gear box relations are:

$$\tau_d = \frac{T_{turb}}{r_{gear} \cdot r_{drive}} \tag{12}$$

$$\omega_r = r_{gear} \cdot r_{drive} \cdot \omega_{turb} \tag{13}$$

In reality, the speeds of the left and right rear wheels are different when the vehicle is turning due to the use of a differential. However, this effect has be neglected since a half car model is being used. The validation of the gear shift logic is given in Section 3.3.

1.7 Actuator Models

Throttle and brake actuators have been implemented on the BMW experimental vehicle for ACC. However, no dynamic models were supplied for the actuators. Hence, the models implemented in the Simulink diagram initially were first order systems plus a pure time delay for each actuator. The throttle model also includes logic which prevents the throttle load from dropping below 7.6% when the engine is idling.

As will be discussed in Section 3.1, the black box model was sufficient to represent the brake dynamics. The throttle actuator dynamics were found to be so fast that they could essentially be neglected. However, very fast throttle dynamics were left in the model to prevent an algebraic loop in Simulink due to the throttle logic.

2 Simulink Model Description

In this section, the Simulink vehicle model and all files needed to run the model will be described. Simulink is a program for modeling and simulating systems. It uses a graphical interface to build models in block diagram form. Figure 5 shows the upper layer of the Simulink vehicle model. The upper portion of this picture contains the vehicle model subsystems and the lower portion contains the controller subsystems.

The vehicle model can best be understood by examining the upper layer and all subsystems and comparing them with the dynamics equations described in Section 1. However a brief overview will be given by following the flow of the model from left to right. In the upper left of Figure 5 are the brake and driveline subblocks. The two actuators inputs, brake and throttle load, enter these blocks. Specifically, the brake subsystem takes in the brake actuator input load and outputs the net brake torque. This net brake torque enters the wheel dynamics block, which implements the equations from Section 1.2 for each wheel. The driveline block implements the engine map, torque converter equations, lockup logic, and gear map. The inputs to this block are throttle load and rear wheel angular velocity and the outputs are engine speed, gear ratio and torque converter output torque. The previously mentioned wheel dynamics block uses the brake torque (from the brake subsystem) and the turbine torque and gear ratio (both from the driveline block). It also inputs the front and rear tractive forces (which are generated by the tire forces block, yet to be described). This block then outputs the front and rear wheel angular velocities which are used by the tire forces block. The tire forces block also uses the vehicle velocity and normal forces on each tire to generate the tractive forces using the equations given in Section 1.2. These normal forces are generated by the normal forces block, which implements the vehicle pitch due to acceleration and road grade as described in Section 1.1. Finally, the longitudinal vehicle dynamic equations given in Section 1.1 are implemented in the longitudinal dynamics

block. This block uses the tractive forces and gravitational force acting in the longitudinal direction to generate the vehicle velocity and position.

The controller is then implemented in the lower half of the Simulink diagram. This controller can be removed and/or replaced if other testing is desired. This section can be described by following the flow from right to left. The first block on the lower right simulates the dynamics of the preceding vehicle. This block integrates a given preceding vehicle velocity profile. The controller uses the range and range rate generated by the preceding vehicle block. The controller has 2 levels. The upper level computes a desired vehicle acceleration using a free flow and stop and go controller. A scaling function is used to switch between the controllers. The lower level then tracks this desired acceleration using the throttle and brake actuators. The lower level also includes switching logic to prevent the application of the throttle and brakes at the same time.

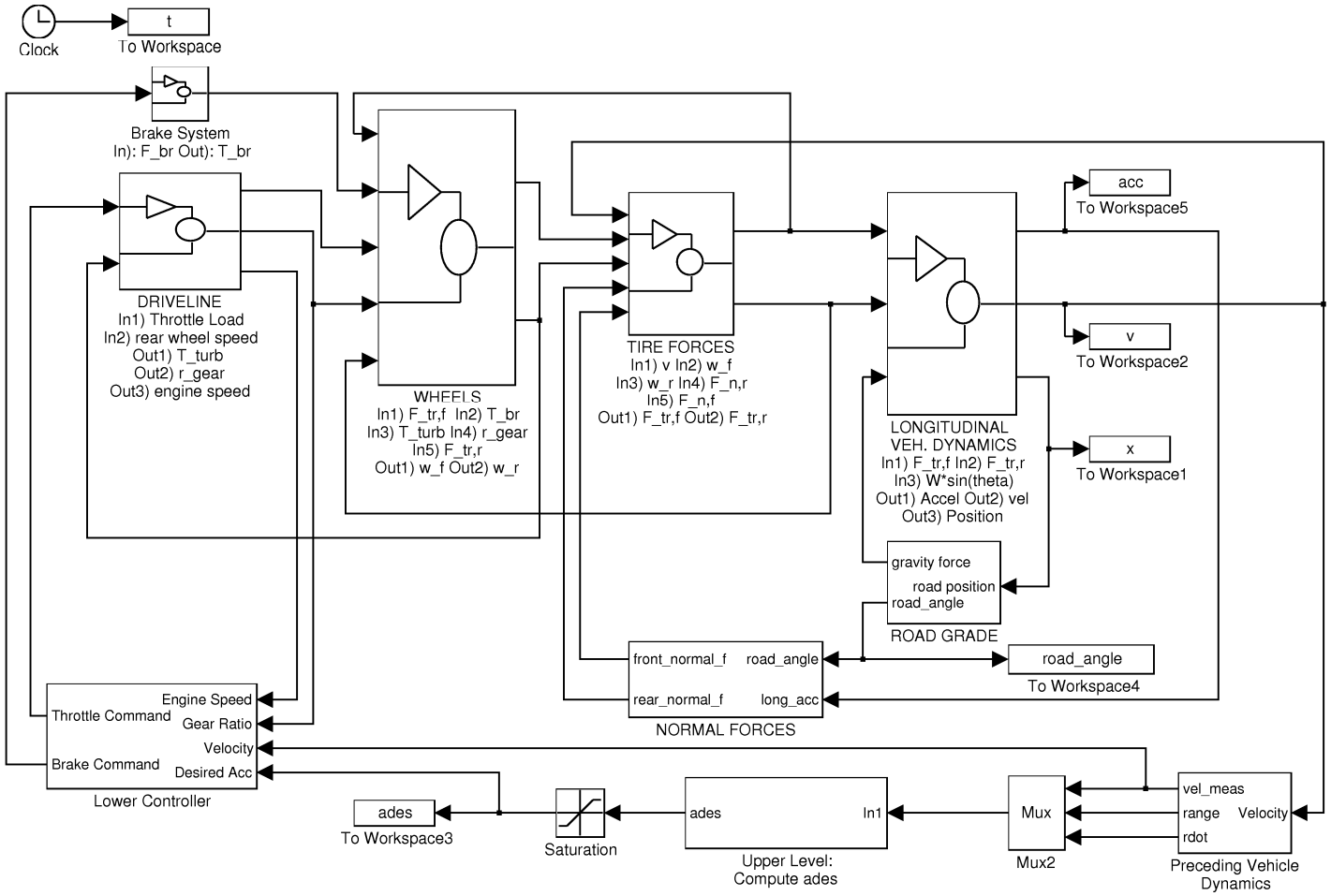

Figure 5: Simulink vehicle model

The following is a list of files needed to run the vehicle model:

- **vehicle4.mdl** This is the Simulink model which contains the vehicle dynamics and Stop and Go controller.
- **bmw_par2.m** This file contains all vehicle parameter information. This file calls many of the files listed below to set all these vehicle parameters.
- **veh_sim5.m** This file is used to run the model from the matlab command line. It can run bmw_par2.m to set the vehicle parameters. It can also set the initial conditions for a particular ACC scenario and sets the controller gains. Finally it can be used to run the simulation and plot the results. These options are given by a simple text menu when this file is run in Matlab.

- **torqcon.m** This file contains torque converter model data.
- **shift1.m** This file contains gear shift and lockup logic data.
- **gearmap.m** This file converts the gear shift data in shift1.m into a map which can be used in Simulink.
- **e38usa.m** This file contains miscellaneous vehicle data including gear ratios, drag coefficients, vehicle mass, and static weight distribution.
- **m73b54.m** This file contains engine data.
- **MF_tire.m** This file contains Magic Tire Formula parameters.
- **int.m** This function is used by m73b54.m to manipulate the engine data into a usable form.
- **supervis.m** This file contains scaling functions used by the supervisor controller. This file is not needed if the Stop and Go controller is removed from vehicle3.mdl

All of these files are needed to run the control simulations, but most files will be transparent to the user. To run the control simulations, the user really only needs to call veh_sim5.m. This file will produce a menu with four options. At startup, the user should always choose option 1, which calls bmw_par2.m. This will initialize all vehicle variables used by the model. Then the user should run option 2 and select one of the offered ACC scenarios. This option will set all the initial conditions. Finally, option 3 should be chosen to run the simulation and plot the results. If another scenario is desired, the user should return to option 2 and re-initialize. If the user only intends to run the vehicle model, they should strip out the control portion of the block diagram. They still need to run bmw_par2.m on startup to set the vehicle variables. They will have to strip out all the code in option 2 of veh_sim5.m which deals with initial conditions (or create a file of their own to set the initial conditions). Then, they can run the simulation from the Simulink model or try to copy option 3 from the veh_sim5.m file and run the simulation from the Matlab command line.

3 Model Validation

The Simulink model of the BMW test vehicle longitudinal dynamics was verified using data from a series of experiments completed at Crow's Landing. The model was broken into subsystems which were tested as much as possible before moving on to overall model validation. In the end, only minor modification of the vehicle model/parameters was necessary to obtain good correspondence between the simulation and the actual data. Further work on the lockup logic is still needed.

3.1 Actuator Models

As mentioned in Section 1.7, the brake and throttle actuators can be reasonably modeled with a first order system plus a pure time delay, as shown in Figure 6. Thus, the system satisfies the following differential equation:

$$\tau \cdot \dot{y}(t) + y(t) = k_a \cdot u(t - \tau_d) \tag{14}$$

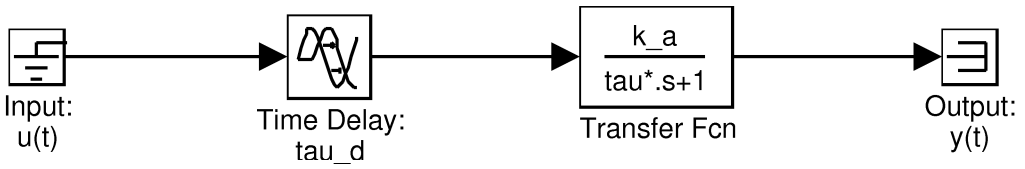

Figure 6: Model of first order system + time delay

Using the Euler approximation, this differential equation can be written in discrete form as:

$$y(t + \Delta t) = \left[1 - \frac{\Delta t}{\tau}\right] y(t) + k_a \cdot \frac{\Delta t}{\tau} \cdot u(t - \tau_d) \tag{15}$$

Rewriting the equation, we see that there are three unknown parameters: A, B, and τ_d.

$$y(k+1) = A \cdot y(k) + B \cdot u(k - \tau_d) \tag{16}$$

The method of least squares will be used to find suitable estimates for these unknown parameters. For a test with N samples, the relation given by Equation 16, can be put into the following matrix form:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(\tau_d) \\ y(\tau_d+1) \\ y(\tau_d+2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} y(0) & u(-\tau_d) \\ y(1) & u(-\tau_d+1) \\ \vdots & \vdots \\ y(\tau_d-1) & u(-1) \\ y(\tau_d) & u(0) \\ y(\tau_d+1) & u(1) \\ \vdots & \vdots \\ y(N-1) & u(N-1-\tau_d) \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \Rightarrow Y = \Phi\Theta \tag{17}$$

In the least squares problem, we are trying to find the parameters, $\hat{\Theta}$, to minimize the cost function: $J = ||Y - \Phi\Theta||$. The solution to the matrix equation minimizes the error in the sense of least squares. For the actuator modeling, we will first assume that we know the system delay and find the optimal values of A and B for the given delay. The value of the cost function will be computed for these parameters, i.e. $J_{opt} = ||Y - \Phi\hat{\Theta}||$. This procedure will be repeated for many values of τ_d and the time delay value that produces the minimal J_{opt} will be chosen.

3.1.1 Brake Actuator

We have step response data for 6%, 9% and 25% step inputs. The least squares model validation was done on the 9% step input data because there are some nonlinear effects which are more apparent at the low and high step input responses. Using the methodology outlined above, estimates for A, B, and τ_d were computed. Since we know $\Delta t = 20msec$, we can convert back to the time domain equivalent parameters: $k_a = 2.8$, $\tau = 0.13sec$, and $\tau_d = 40msec$. The least squares model matches the true data (Figure 7), which is expected since this data was used for the estimation. Furthermore, if we zoom in on the leading edge of the step response plot (Figure 8), we notice that the first non-zero input occurs at 0.54 seconds. If there was no time delay in the system, we would expect the output to be non-zero at the next sample time, 0.56 seconds. However, we can see that the effect is delayed by the predicted 40 msec, at which time (0.60 sec) both the simulated and actual output values are nonzero. Another interesting point is that the simulated response is noticeably slower than the actual output response (this effect is magnified by the zooming). We could lower the estimate of τ to improve the estimate on the leading edge, but then the simulated response would have a much larger spike due to the spike in the input waveform.

The true test of the least squares model is how it estimates the brake actuator for different values of step inputs. Figures 9 and 10 show the step responses for 6% and 25% inputs, respectively. For the 6% step input, the least squares model has a steady state gain which is larger than the actual gain. The experimental data showed that the steady state gain of the actuator was not a constant as modeled, but varied with the size of the step input. Furthermore, the least squares time constant is larger than the actual time constant. For the 25% input, we observe another nonlinear effect, since the output does not possess the spiked waveform present in the input. The first order system plus time delay cannot model these effects. However, the steady state gain of the least squares system corresponds reasonably well with the actual steady state gain. It is hoped that, in the context of the entire vehicle model, these differences will have a minor effect on the overall simulation accuracy.

3.1.2 Throttle Actuator

The typical throttle actuator step response, upper subplot of Figure 11, shows no dynamics between the input and output. Even zooming in on the response (lower subplot), shows essentially no dynamics. On the basis of these step responses, we determined that the throttle dynamics were negligible relative to the total vehicle dynamics. Therefore, these dynamics can be removed from the Simulink model. A very fast first order model for the throttle dynamics remains in the model to prevent an algebraic loop due to the throttle logic.

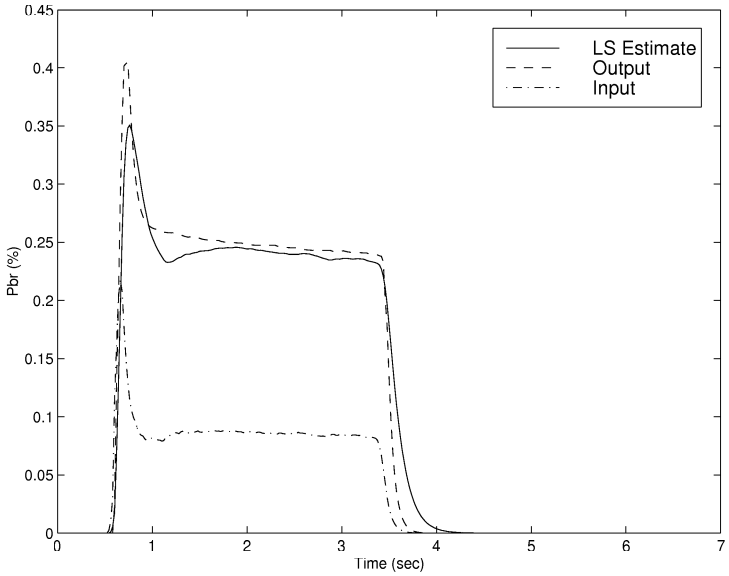

Figure 7: Comparison of LS estimate and actual data for 9% step input

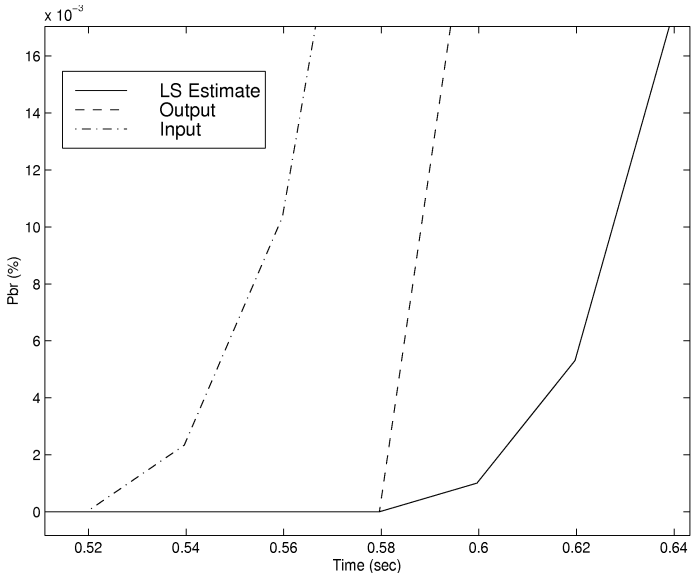

Figure 8: Zoom on the leading edge of 9% step response

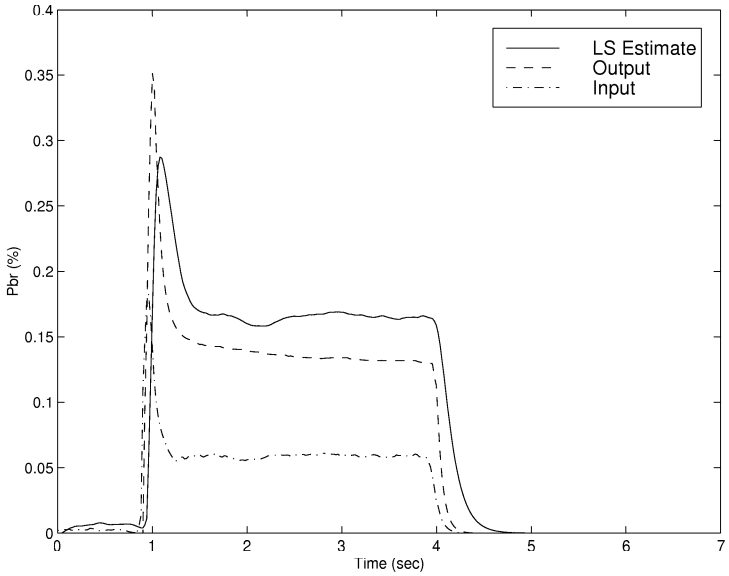

Figure 9: Comparison of LS estimate and actual data for 6% step input

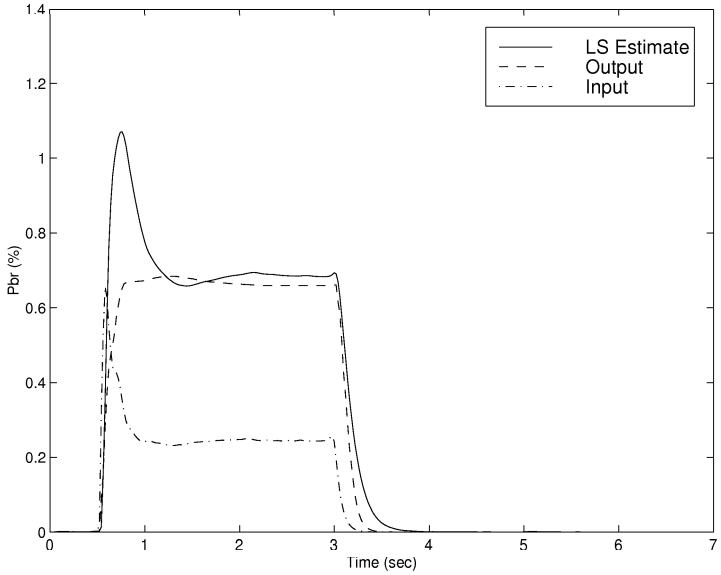

Figure 10: Comparison of LS estimate and actual data for 25% step input

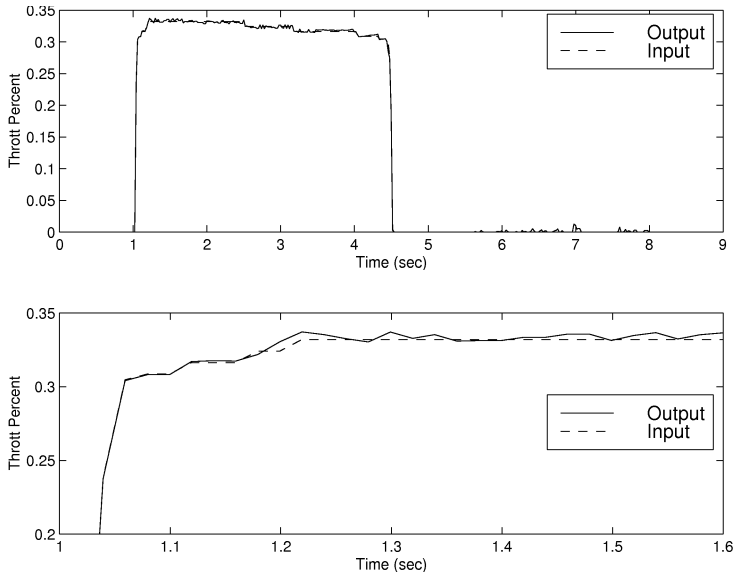

Figure 11: Throttle actuator step response

3.2 Drag Coefficient and Rolling Resistance Estimates

Next, we tried to identify suitable estimates for the drag coefficient and rolling resistance using engine-in-neutral coast down data. This was complicated by wind and grade effects which corrupt the data. However, these effects can be removed by examining coast down data in the north and south directions.

If we assume no slip and a locked torque converter, the vehicle dynamics are modeled by the following differential equation:

$$m \cdot \dot{v} = -c_D \cdot (v - v_{wind})^2 - F_{rr} - m \cdot g \cdot sin(\theta) + T_e \cdot r_{drive} \cdot r_{gear} \cdot r_{wheel} \tag{18}$$

where c_D is the drag coefficient (kg/m), v is the vehicle velocity (m/s), v_{wind} is the wind velocity which is positive when moving with the wind, F_{rr} is the rolling resistance (N), m is the vehicle mass (kg) which is corrected with the approximate passenger masses, θ is the grade which is positive when going uphill, r_{drive} is the differential gear ratio, r_{gear} is the transmission ratio, and r_{wheel} is the wheel radius (m).

It should be noted that before we used the coast down data, we tried to eliminate the effect of the grade using a closed throttle vehicle test. We ran this test in the north and south directions and noticed that the steady state vehicle speed was slightly different. For this test, we can assume the drag effects are neglible (since steady state speeds were under 3 m/s). Then, Equation 18 yields the following relation in steady state:

$$0 = -F_{rr} - m \cdot g \cdot sin(\theta) + T_e \cdot r_{drive} \cdot r_{gear} \cdot r_{wheel} \tag{19}$$

The difference in steady state vehicle speeds is transmitted back to the engine and hence there is also a steady state difference in engine speeds. Since T_e is only a function of engine speed for closed throttle, we can use the engine map to compute the engine torque in the north and south directions. We can theoretically compute the grade by examining the differences in these computed torques. In practice, the differences in engine speeds was very small and the resulting grade estimate was very small. Furthermore, this method relies on knowledge of the engine map, which is questionable itself, and neglects torque converter slip, which is significant at low velocities.

Another method for drag and rolling resistance estimation was needed. As mentioned above, we decided to use engine-in-neutral coast down tests, which removed the uncertain engine map from the estimation process. Equation 18 can then be simplified to:

$$\dot{v} = -A \cdot (v - v_{wind})^2 - B - g \cdot sin(\theta) \tag{20}$$

where $A = c_D/m$ and $B = F_{rr}/m$. If we discretize this equation and group terms in form of a least squares problem, we get:

$$[v(k+1) - v(k) + g \cdot sin(\theta) \cdot \Delta t] = \begin{bmatrix} -\Delta t \cdot (v(k) - v_{wind}(k))^2 & -\Delta t \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \tag{21}$$

The interesting thing is that the drag estimate, A, is independent of the grade, θ. This can be seen by lumping the grade term on the left hand side with the rolling resistance parameter, B. Call this lumped parameter C. Then, C can be computed using the least squares and can be considered a constant with respect to θ. Changing θ only changes the relation between B and $g \cdot sin(\theta) \cdot \Delta t$, but does not change C. Thus, the drag estimate, A, is independent of this choice of grade. Furthermore, the rolling resistance estimate, B, was found to be independent of v_{wind}.

The effect of grade and wind can be removed using this decoupling concept. First, estimate the drag coefficient in the north and south directions for various values of v_{wind} (with the signs obviously changing based on vehicle direction). Since the drag coefficient must be independent of direction, the value of v_{wind} that produces $\hat{c}_{D,North} = \hat{c}_{D,South}$ must be correct. Figure 12 shows the drag coefficient estimates for coast down tests in both directions as a function of v_{wind}. From this plot, we can deduce that $\hat{c}_D = 0.4$, $\hat{v}_{wind,north} = +2.2m/s$, and $\hat{v}_{wind,south} = -2.2m/s$, i.e. the wind was moving south to north.

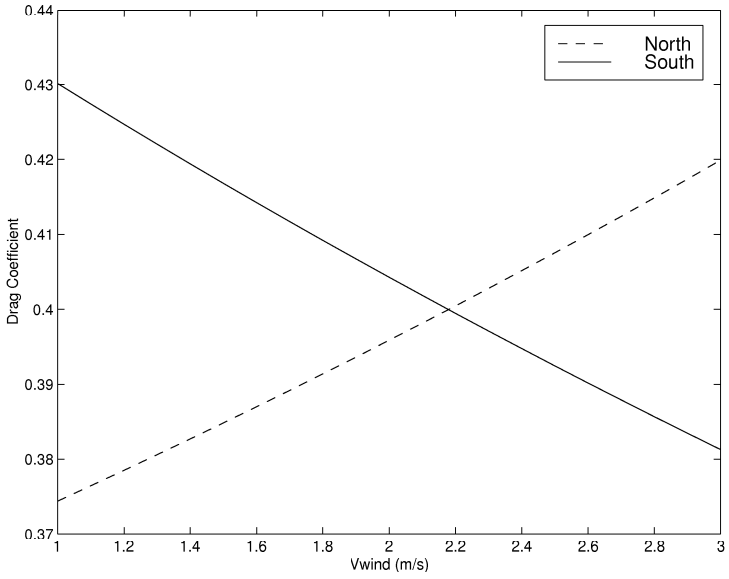

Figure 12: Drag coefficient as a function of v_{wind}

Next, we can use this estimate of c_D and estimate the rolling resistance in the north and south directions for various values of θ. Again, the value of θ that produces the same estimate of rolling resistance in both directions must be correct. Figure 13 shows that $\hat{F}_{roll} = 228N$ and $\hat{\theta}_{north} = 0.002rads$.

Finally, Figure 14 shows the predicted coast down responses in the north and south directions using the estimated drag and rolling resistance parameters. The correlation is good and appears to be independent of direction. Furthermore, the values currently used in the simulation model are $c_D = 0.4335$ and after correcting for passenger mass, $F_{roll} = 226N$. Thus, we will assume the current values in the Simulink model are acceptable.

3.3 Gear Shift Maps

The gear shift maps use information of the throttle load and shaft speed to determine when to shift up or down. This subsystem can be tested independently of all other dynamics in the vehicle since we have actual throttle load, engine speed, and gear data from our testing at Crow's Landing.

Figure 15 shows the response from one test run. The upper plot show quite a discrepancy between the simulated shift times and the actual shift times, especially in the 3→4 and 4→5 shift times. The shift times can be a bit deceiving however because if the engine speed or throttle angle is changing slowly, even two similar shift maps can produce very different results. The lower plot is an attempt to analyze the data independent of these effects. This plot shows the actual and simulated engine speeds when a shift occurred. Furthermore, it shows the shift engine speed predicted by the BMW supplied shift tables for the throttle angle at which the shift occurred. Since the throttle load was held approximately constant at 29% during this test, this line represents the shift engine speeds predicted by the shift maps for this throttle load. The

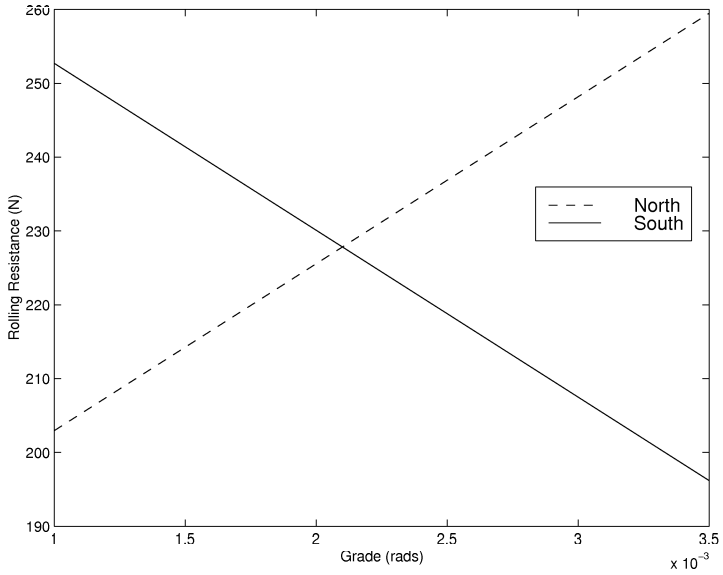

Figure 13: Rolling resistance as a function of θ

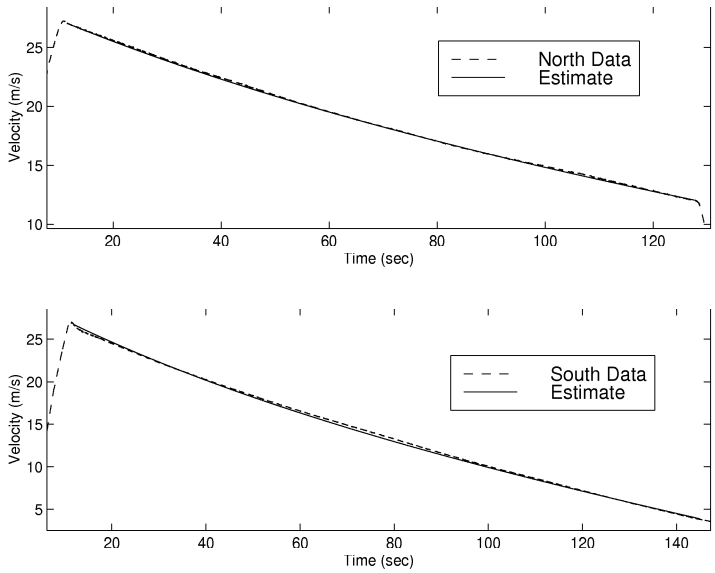

Figure 14: Coast down responses in north and south directions

shift numbers 1-8 correspond in order to: 1→2, 2→3, 3→4, 4→5, 5→4, 4→3, 3→2, and 2→1. In this lower plot, we can see that the shift engine speed in the simulation and actual data does indeed differ significantly for the 3→4 and 4→5 shifts. More disturbing, the simulated engine shift speed differs from the shift speed predicted by the shift tables.

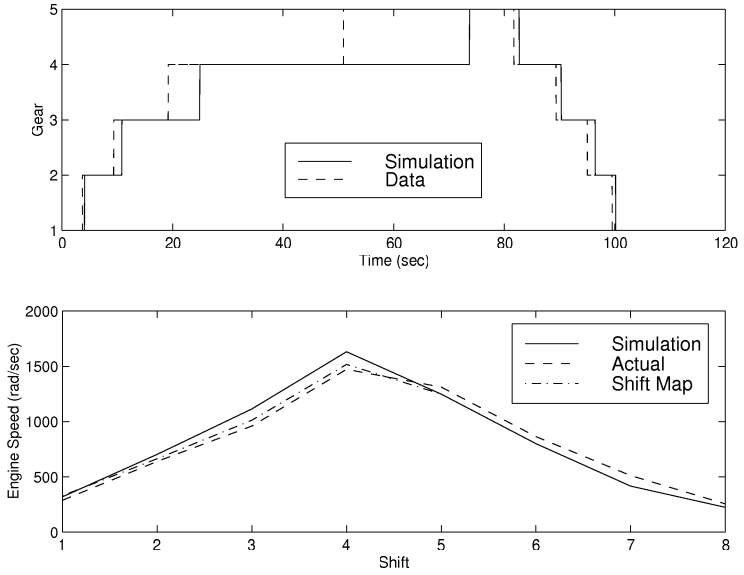

Figure 15: Time (upper) and engine speed (lower) at shifts using old maps

This implies that our gear shift model does not accurately represent the gear shift tables, let alone the true shift map on the test vehicle. Figure 16 plots the old up shift model as a function of throttle load and shaft speed. It is apparent in this plot that the problem is one of poor resolution. The solution to this problem is to refine the model by evaluating the shift table at more throttle load and engine speed values.

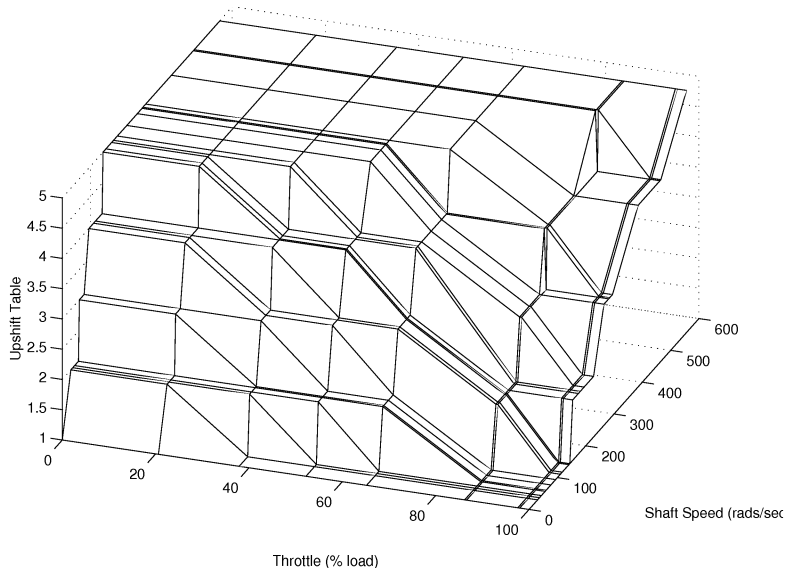

Figure 16: Old gear upshift model

The new up shift map, Figure 17, is more refined and should model the data contained in shift maps in a more accurate fashion. The down shift map was similarly modified. Figure 18 shows the results using the same data (as in Figure 15) on the new gear shift maps. The lower plot shows the expected result: the simulated shift engine speed agrees almost exactly with the shift engine speed predicted by the BMW

supplied shift tables. Furthermore, the agreement between the simulated shift times and actual shift times (upper plot) is improved. Overall, there are still some errors between the simulated shift times and actual shift times, but the refinement of the gear map has significantly reduced these errors.

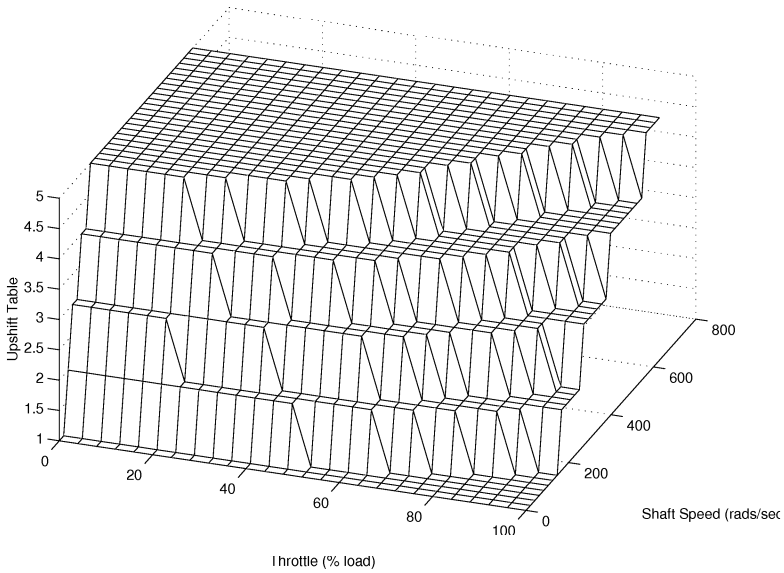

Figure 17: New gear upshift model

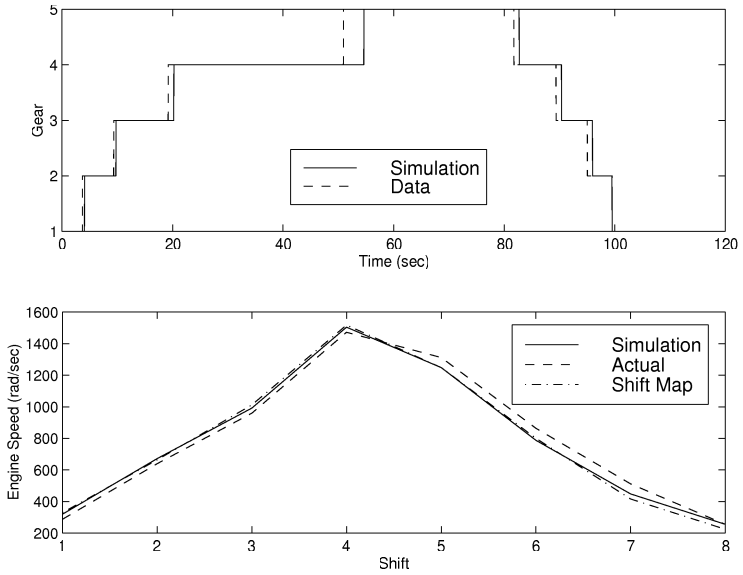

Figure 18: Time (upper) and engine speed at shifts using new maps

3.4 Final Reduction Value

Since we have data for the shaft speed and rear wheel speeds, we can compute the final drive gear reduction and compare it to the value used in the simulation. These measurements are related by:

$$\omega_{shaft} = \frac{(\omega_{L,Rear} + \omega_{R,Rear})}{2 * r_{drive}} \tag{22}$$

Thus a plot of the shaft speed measurement should lie on top of a plot of the wheel speed measurement reflected back through the final reduction. Figure 19 shows the plot of the shaft speed measurement and the

reflected wheel speed using the final reduction value of $r_{drive} = 0.3559$. It is apparent from this plot that the actual final gear reduction value is not equal to this estimate. If we correct the final reduction value by $r_{drive} = 0.3559 * 1.05$, then we can see that the shaft speed and reflected wheel speeds are almost identical. It was found that with this corrected value of r_{drive}, the shaft and reflected wheel speeds agreed for all step throttle tests.

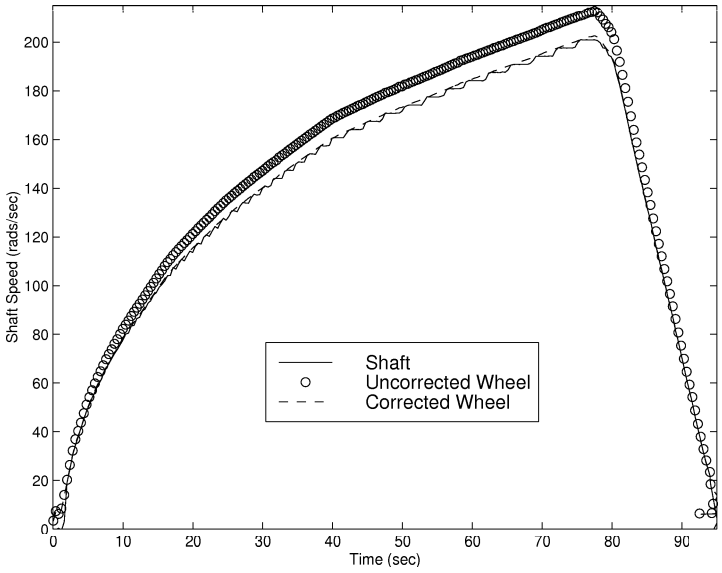

Figure 19: Shaft speed vs. reflected wheel speed (before and after correction)

3.5 Torque Converter Model

The engine subsystem was removed from the vehicle model to test the torque converter model. This was accomplished by using the known engine speed and current gear data as inputs to the model. Then, simulated and actual vehicle speeds were compared. Since the drag and rolling resistance effects were identified, this simulation tested the validity of the torque converter and wheel models. Specifically, we attempted to validate that the torque converter output torque was properly modeled (since this torque is transmitted to the wheels and governs the vehicle speed).

Figure 20 shows the actual vehicle velocity for one data file in comparison to the simulated vehicle speed with the old and new torque converter models. The simulated speed using the old model differs differs from the actual vehicle speed in two respects. The simulated speed profile contains large jumps in velocity at the up shift times, which occur approximately at 10, 20 and 55 seconds. This characteristic is not present in the actual velocity profile. Between gear shifts, the simulated wheel speed approaches a profile which differs significantly from the actual vehicle speed. Although these results could be due errors in the tire model, we decided to test the simulation with the new torque converter model described in Section 1.4.

In the torque converter data file, it states that the test was performed at a constant input speed of 2000 rpm. Thus, the capacity factor can be computed using this constant speed and the Pump (input) torque column of data ($K_{tc} = \frac{\sqrt{T_{in}}}{\omega_{in}}$). Using the new torque converter model, the correspondence between the simulated and actual vehicle speeds is improved. (Figure 20). The velocity offset between gear shifts is reduced. The velocity behavior has small jumps during gear shifts, but the large surges in velocity are no longer present. The differences between simulated and actual vehicle speed are small at this point and will be corrected after the engine model has been reinserted. Notice that during the braking portion of the test, the correspondence is quite good. However, this is due to a choice of brake gain (from brake pressure to brake torque) which results in good estimation.

3.6 Lockup Logic

It is possible to determine whether or not the torque converter locked during a particular experiment by examining engine and shaft speed test data. For example, Figure 21 shows the engine speed data, which is

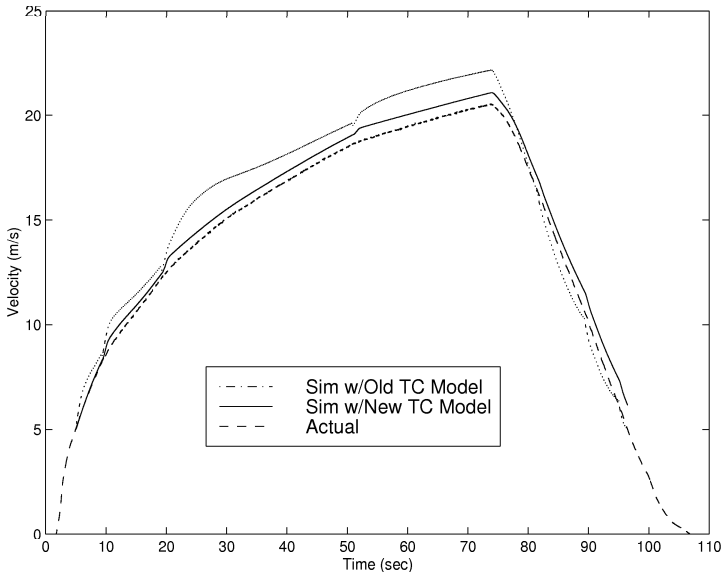

Figure 20: TC testing: simulated and actual vehicle speed

the TC input speed, and the shaft speed divided by the current gear ratio, which is the TC output speed. The up shifts occur at 3, 8, 16, and 40 seconds and the first down shift doesn't occur until 85 seconds. At approximately 55 seconds, the engine speed rapidly drops and approaches the TC output speed. Just before 70 seconds, the TC lockup is completed as the TC input and output speeds converge. Finally, the torque converter unlocks just before the first down shift.

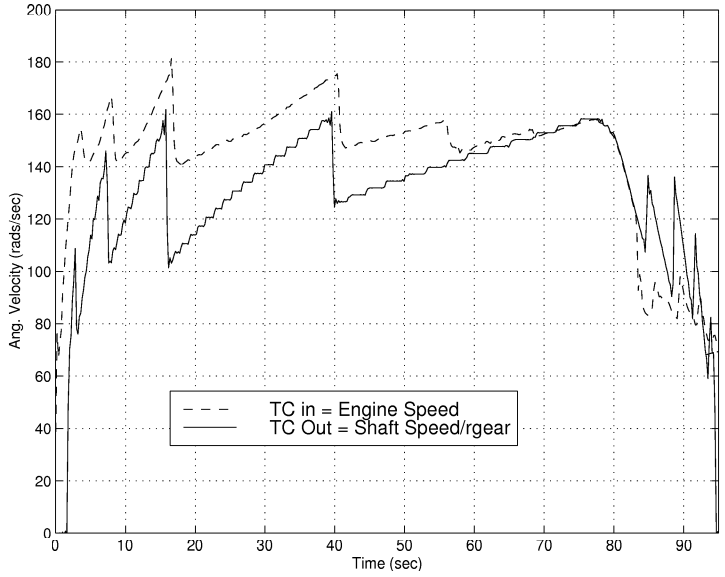

Figure 21: TC input and output speeds, showing TC lockup

We have several experiments in which torque converter lockup is observed. The lock/unlock throttle loads and engine speeds for each experiment were recorded and compared to the lock/unlock maps given by BMW. All of the lockups in this comparison occurred while the vehicle was in fifth gear. Figure 22 shows the lockup maps for fifth gear in comparison with the test points (the circles). Obviously the lockup map shows little correspondence with the observed lockups. Also it was noticed in that many unlock points occurred at zero throttle load, so we would expect only one unlock engine speed for this given throttle angle. However, the engine speeds at unlockup varied from 108 to 377 rads/sec.

We decided to use a simple piece-wise linear fit, shown in Figure 22, of the observed lockup points. The

lockup model thus consists of this experimentally determined lockup map. The lockup model only allows lockups in fifth gear even though fourth gear lockups are possible according to the BMW supplied lockup maps. An unlock map was similarly determined. One further problem was noticed upon implementation of this lockup model. When the lockup occurs in the experimental data, the engine speed slows down to match the output speed of the torque converter. In simulation, the opposite occurs, the output torque converter shaft speeds up to match the engine speed. Consequently, the vehicle velocity increases greatly since it tightly coupled to the torque converter output speed. It is believed that there is some unmodeled engine management logic which forces the observed drop in the engine speed. As a simple fix, a gain is used to drop the engine torque in the locked torque converter mode.

The lockup logic and the locked torque converter models obviously are highly uncertain at this point and in need of further development. But, the torque converter should not lockup in the Stop and Go scenario. Thus we can safely test these algorithms in spite of the uncertainties in the lockup map. In the future, the lockup model should be improved so that the effect of lock/unlock dynamics can be determined in the normal ACC scenario.

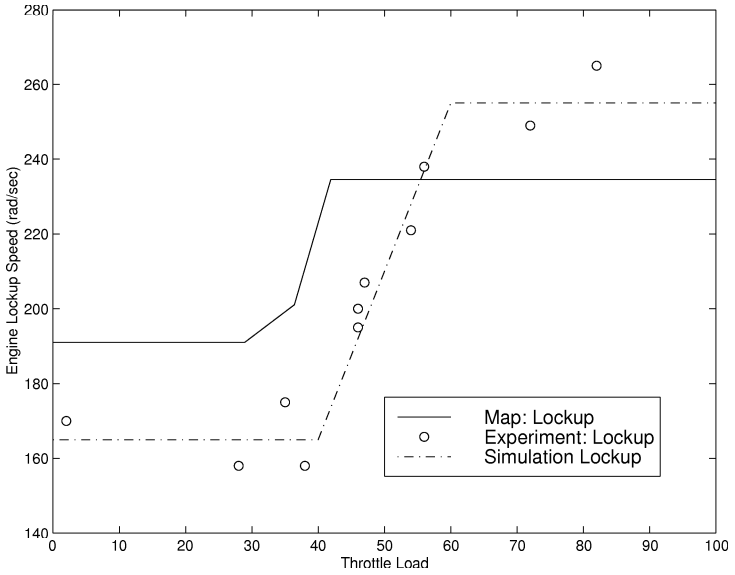

Figure 22: Comparison of lockup map with experimental lockup points

3.7 Overall Model Simulation

Finally, we decided to test the overall model fidelity. The engine model was included, but due to their questionable status, the lockup logic and locked drivetrain dynamics were not included. The results of the initial simulations showed that the vehicle speed was larger than the actual velocity for all test data. Thus, we reduced the output engine torque by a constant correction factor of 35 $N - m$. This constant factor accounts for engine losses, most notably the air conditioning pump which was running on the day that we performed these experiments. With this single correction factor we were able to obtain good matching between actual and simulated vehicle performance. Figures 23- 25 show the comparisons of most measured data for an experiment where the torque converter did not lock up. Previously we had tested each subsystem independently or as part of a reduced vehicle model. However, these plots show that the subsystems (e.g. gear shifting, torque converter, ect.) respond properly when placed in the complete vehicle model. The simulation agrees with the experimental data up to the torque converter lock up point for all test runs.

3.8 Validation Conclusions

The results of this analysis are quite encouraging. Only minor modification of the vehicle parameters was necessary to obtain good correspondence between simulation and the actual data. Specifically, we changed the final drive gear ratio and added a correction factor on the engine. Furthermore, we created a brake actuator

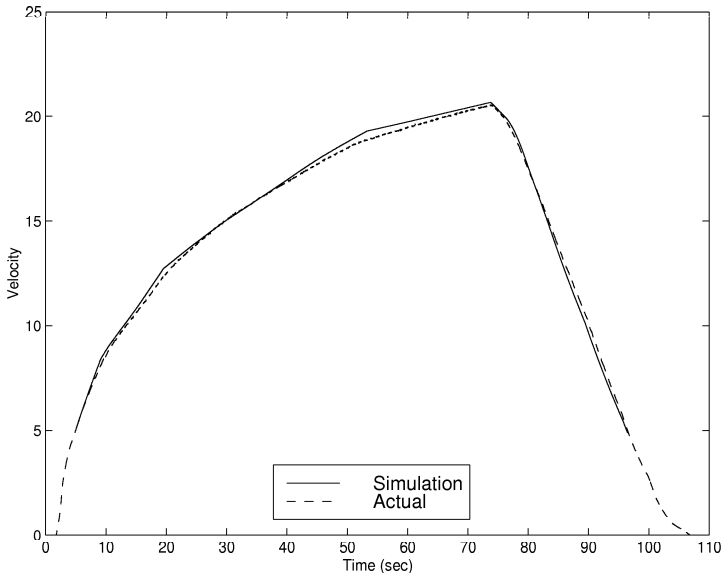

Figure 23: Comparison of simulated and actual vehicle speed

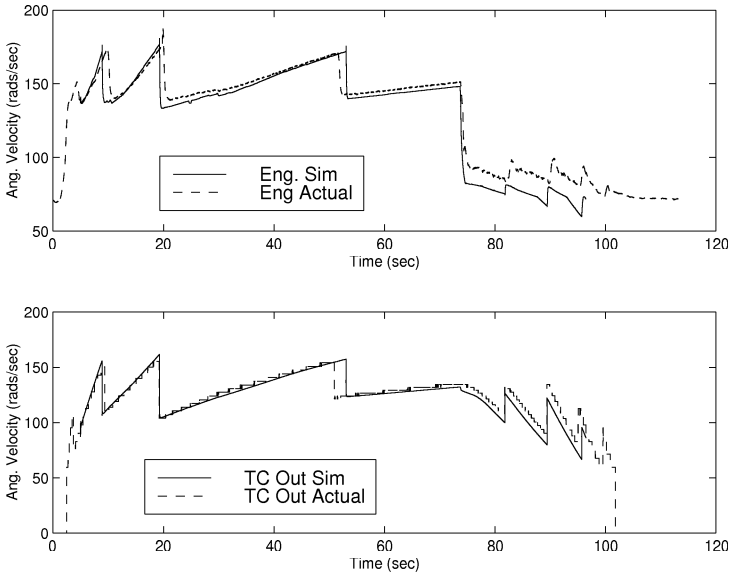

Figure 24: Comparison of simulated and actual TC input/output speeds

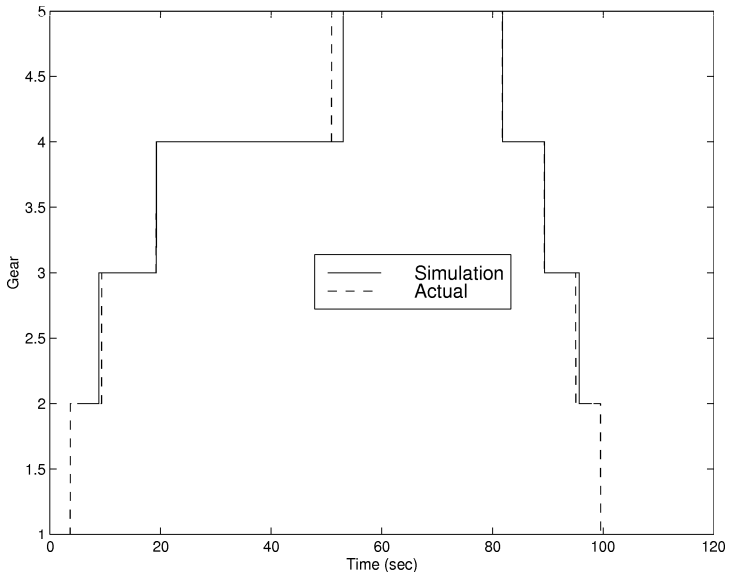

Figure 25: Comparison of simulated and actual gear shifts

model using least squares estimation and eliminated the throttle actuator dynamics. We also modified the gear shift maps, but that was only to bring them into agreement with the supplied gear shift tables.

At this point, only the lockup logic and locked torque converter model need further work. The lockup logic could be improved with many experimental data points. The real concern lies in the dynamics during lockup. The logic that forces the engine speed to drop down to the torque converter output speed must be determined. Similarly, the engine management logic during the locked mode must be modeled. As previously noted, these model flaws should not affect Stop and Go controller testing.

4 Stop and Go Controller Development

The following section will describe the design of a sliding control law for use in stop and go vehicle following.

4.1 Upper Level Controller

First, define the following function:

$$S = (\dot{r} - \dot{r}_{des}) + \Lambda \cdot (r - r_{des}) = \dot{e} + \Lambda \cdot e \tag{23}$$

where $r_{des} = t_h * v^{k_o} + d_o = 6.33 * v^{0.48} + 2$. The desired following distance, r_{des}, is a curve-fit of human driver data which was supplied by BMW. Note that if $\Lambda < 0$ and $S = 0$ then the spacing error will exponentially decay to 0. The surface given by the relation $S = 0$ is called the sliding surface. The goal is to use the control input or a synthetic input to force $\dot{S} = -K \cdot S$ and hence cause convergence to the sliding surface.

In this problem, the vehicle acceleration is assumed to be a synthetic input to the upper surface. The acceleration is obviously not an input which can be controlled directly. But the assumption is that we can use the throttle and brake actuators to track an acceleration profile fast enough that these dynamics are transparent to the upper surface.

To simplify the problem, assume that $\dot{r}_{des} \approx 0$. This assumption is needed because we will need to differentiate S, so the inclusion of this term requires the controller to have knowledge of vehicle jerk. The function is redefined as:

$$S = (v_p - v) + \Lambda \cdot (r - r_{des}) \tag{24}$$

S is defined such that the synthetic input, vehicle acceleration, appears in the first derivative of S as follows:

$$\dot{S} = \ddot{r} + \Lambda \cdot (\dot{r} - \dot{r}_{des}) = (a_p - a) + \Lambda \cdot (\dot{r} - k_o t_h v^{k_o - 1} a) \tag{25}$$

Next, we will solve for the vehicle acceleration needed to force S to converge to zero:

$$-K \cdot S = (a_p - a) + \Lambda \cdot (\dot{r} - k_o t_h v^{k_o - 1} a) = (a_p + \Lambda \cdot \dot{r}) - (1 + k_o t_h v^{k_o - 1}) \cdot a \tag{26}$$

If we assume no knowledge of a_p:

$$a_{des} = \frac{1}{1 + k_o t_h v^{k_o - 1}} \cdot (\Lambda \cdot \dot{r} + K \cdot S) \tag{27}$$

Roughly, the gain K can be tuned to change speed of convergence to the sliding surface. It can also be used to overcome any uncertainties in our system. The gain Λ represents the speed of convergence to $e = 0$ once on the sliding surface. Since we have not used a discontinuous control law, the system will technically not stick and slide on the surface $S = 0$. However, the system should converge within a small boundary layer of the sliding surface.

It is important to know how the gains K and Λ should be tuned to provide the desired performance. In the definition of a_{des} (Equation 27), the lead coefficient depends on the desired following distance parameters. Assuming that these are fixed, the expression for a_{des} depends on terms involving $r - r_d$ and $\dot{r}$. Substituting the definition of S given in Equation 23 into Equation 27 reveals that the coefficient for $r - r_d$ is $K \cdot \Lambda$. The coefficient for $\dot{r}$ is $K + \Lambda$. Thus the effect of K and Λ is symmetric and the values can be chosen to give a trade off between the effect of range error and range rate. This gain tuning idea will be used in Section 5 to tune the controller for various traffic flow situations.

It is now up to the lower level to use the throttle and brake actuators to track this desired acceleration trajectory. The lower level controller consists of throttle and brake controllers as well as logic to switch between the two.

4.2 Throttle Controller

The throttle controller analysis follows work by Gerdes [1] and Maciuca [2]. The dynamic equations for the vehicle will be used to solve for the necessary throttle angle to generate the acceleration a_{des}. First, use the wheel dynamics, Equation 4, to solve for the tractive forces and substitute in the longitudinal dynamic equations (Equation 1). The result is:

$$m \cdot a = (\tau_d - \tau_{b,r} - \tau_{b,f} - \bar{J}_r \cdot \dot{\omega}_r - J_f \cdot \dot{\omega}_f)/r - F_{rr} - F_d - W \cdot sin\Theta \tag{28}$$

If we assume that there is no tire slip, then $\dot{\omega}_r = \dot{\omega}_f = a/r$. Furthermore, if the torque converter is locked, then $v = r_{drive} \cdot r_{gear} \cdot r \cdot \omega_e$, $a = r_{drive} \cdot r_{gear} \cdot r \cdot \dot{\omega}_e$ and $T_e = r_{drive} \cdot r_{gear} \cdot \tau_d$. This assumption needs to be checked on the vehicle to determine whether torque converter slip significantly affects the controller performance. Finally, the rear wheel inertia can be written as $J_r = J_r + J_e/(r_{gear} \cdot r_{drive})^2$, where J_r is the inertia only of the rear axle and the second term is the engine/transmission inertia reflected to the rear axle.

Using these relations and Equation 28 results in:

$$T_e - r_{drive} \cdot r_{gear} \cdot (\tau_b + r \cdot F_{rr} + r \cdot F_d + r \cdot W \cdot sin\Theta) = \beta a \tag{29}$$

where $\tau_b = \tau_{b,r} + \tau_{b,f}$ and β is the corrected mass:

$$\beta = [J_e + (r_{drive} \cdot r_{gear})^2 \cdot (J_r + J_f + m \cdot r^2)]/(r_{drive} \cdot r_{gear} \cdot r) \tag{30}$$

The throttle brake switching logic described in Section 4.4 prevents the brakes and throttle from being actuated at the same time. Therefore we can assume that $\tau_b = 0$ in Equation 29. We can then solve for the desired engine torque as a function of the desired acceleration:

$$T_{e,des} = \beta a + r_{drive} \cdot r_{gear} \cdot r \cdot (F_{rr} + F_d + W \cdot sin\Theta) \tag{31}$$

As discussed in Section 1.3, the engine torque is a nonlinear function of throttle load and engine speed: $T_e = f(\alpha, \omega_e)$. We can then invert this engine map to compute the desired throttle load:

$$\alpha_{des} = f^{-1}(T_{e,des}, \omega_e) \tag{32}$$

In the simulation, the throttle controller is just an open loop command of α_{des}. This commanded signal is filtered by a first order filter with a pole at 10 Hz to remove high frequency components. The high frequency are undesirable because they wear out the actuator and the chattering can be uncomfortable to the driver. On the experimental vehicle a simple PID loop might be needed to ensure that the actual throttle load converges to the desired.

4.3 Brake Controller

The analysis above can also be used to find the desired brake torque in situations where the vehicle must decelerate. We can start from Equation 29. If we are using the brakes, then the switching logic should ensure that we are not also using the throttle. However, this does not imply that T_e is equal to zero. We must evaluate the engine map to find the engine torque that is being produced at closed throttle: $T_{e,ct} = f(0, \omega_e)$. We can use this relation and solve Equation 29 for the desired brake torque:

$$\tau_{b,des} = (T_{e,ct} - \beta a)/(r_{drive} \cdot r_{gear}) - r \cdot (F_{rr} + F_d + W \cdot sin\Theta) \tag{33}$$

Unfortunately, the cost of brake torque measurement makes it use in closed loop control prohibitive. However, an accurate measurement of the master cylinder pressure can be easily and relatively cheaply obtained. Therefore, the brake controller will track the desired master cylinder pressure: $P_{mc,des} = \tau_{b,des}/k_b$. This approach generates another problem. As discussed in Section 1.2, the brake gain is highly uncertain. Even if we perfectly track $P_{mc,des}$ perfectly, the tracking of $\tau_{b,des}$ may be flawed due to mismatch between the estimated k_b and the actual value. But the gain on the upper surface can be increased in order to compensate for errors generated the lower surface.

The simulation brake controller is just an open loop command to generate the desired brake torque. This open loop command uses the gain from brake pressure to brake torque, k_b and the actuator gain from the actuator input to the brake pressure, $k_actuator$. The actuator gain is the gain found via least squares analysis in the model validation section. The open loop command is given by $F_{br,des} = \tau_{b,des}/(k_b \cdot k_{actuator})$. A simple PID loop will be needed to ensure that the actual master cylinder pressure converges to the desired.

4.4 Throttle/Braking Switching Logic

Throttle/Brake switching logic is needed to decide which of the two lower level controllers described above should be acting. Moreover, it is needed to prevent chattering which may occur due to rapid switching between the two controllers. A simple switching logic would be to use the throttle controller for positive desired accelerations and the brake controller for negative accelerations. The flaw in this logic is that the vehicle will decelerate by a small amount at closed throttle due to wind drag and rolling resistance. Therefore it may not be necessary to apply the brakes when a_{des} is a small negative value.

Using this intuition, we can again use Equation 29 to find this residual acceleration which occurs at closed throttle (assuming a level road):

$$a_{resid} = [T_{e,ct} - r_{drive} \cdot r_{gear} \cdot r \cdot (F_{rr} + F_d)]/\beta \tag{34}$$

Since a_{resid} represents the deceleration of the vehicle for closed throttle, $a_{des} \geq a_{resid}$ implies that the throttle controller should be used. If $a_{des} \leq a_{resid}$ then a_{des} is commanding more deceleration than can be generated by the wind drag and rolling resistance. In this case the brake controller should be used to generate the additional deceleration.

This logic solves the problem of choosing the appropriate lower level controller. However the potential of rapid throttle/brake switching still remains when a_{des} is close to a_{resid}. Hysteresis is added to the logic to prevent such chattering.

4.5 Simulation Results

The next step in the controller design process is to implement and test it on the validated Simulink vehicle model. To make the scenario realistic, the velocity of the experimental vehicle was recorded during a real stop and go situation. This velocity trajectory is used in the model as the lead vehicle velocity profile. This particular trajectory was obtained while driving on Castro Street in downtown Moutainview. Figures 26- 28 summarize the results of the simulation testing.

Figure 26 shows the tracking of the lead vehicle velocity (upper subplot) and the desired following distance (lower subplot). The range tracking is good, but it could be improved. However, the improved performance comes at the cost of increased control effort, i.e larger accelerations. Improved performance also requires more high frequency control effort since the controller attempts to track more of the preceding vehicle oscillations. The controller should be tuned on the vehicle such that the range tracking is within the bounds of human comfort. Any improvements in performance beyond this are unnecessary.

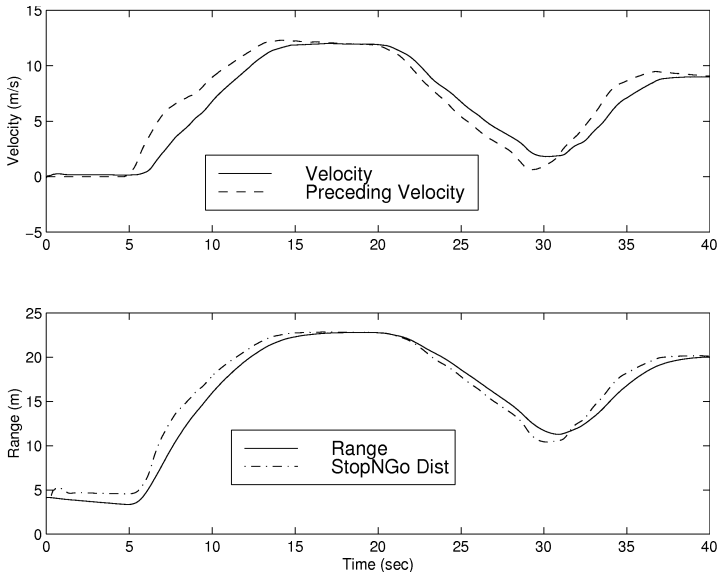

Figure 26: ACC range and velocity tracking

The upper subplot of Figure 27 simply shows that the vehicle acceleration is tracking the desired acceleration. Thus the lower level controller is doing a reasonable job tracking the a_{des} produced by the upper surface controller. There is a large spike and some oscillatory response of the controlled vehicle acceleration in the first second of the simulation. This characteristic is due to the use of the dynamic tire model and is not indicative of the controller performance. The lower subplot is a comparison of the controlled vehicle acceleration and the preceding vehicle acceleration, which was also recorded on the experimental vehicle. The acceleration levels are comparable, implying that the stop and go controller does not require unreasonable control effort.

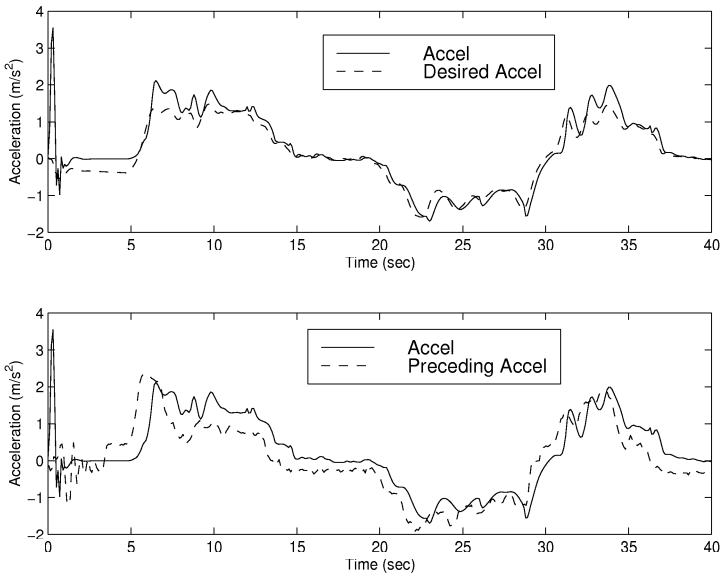

Figure 27: ACC acceleration tracking

Finally, Figure 28 is a comparison of controlled vehicle actuator commands and the actual actuator commands recorded on the experimental vehicle. There are several interesting aspects to this plot. First, the controlled vehicle uses the throttle to slow down as much as possible before switching to brakes. For example, between t=15-20 seconds, the human driver in the preceding vehicle releases the throttle and uses the brakes to slow down. However, the controller just lowers the throttle load in an attempt to slow down

without switching to brakes. Eventually the controller is forced to switch to the brakes. Notice that the controller exclusively uses either the brakes or throttle to control the vehicle. The human driver, on the other hand, applies both the throttle and brakes between t=30-38 seconds. Actually, there is a brief moment at t=20 seconds where the controller is applying both throttle and brakes, but this is only due to the filters which cause the commanded throttle load to decay to zero.

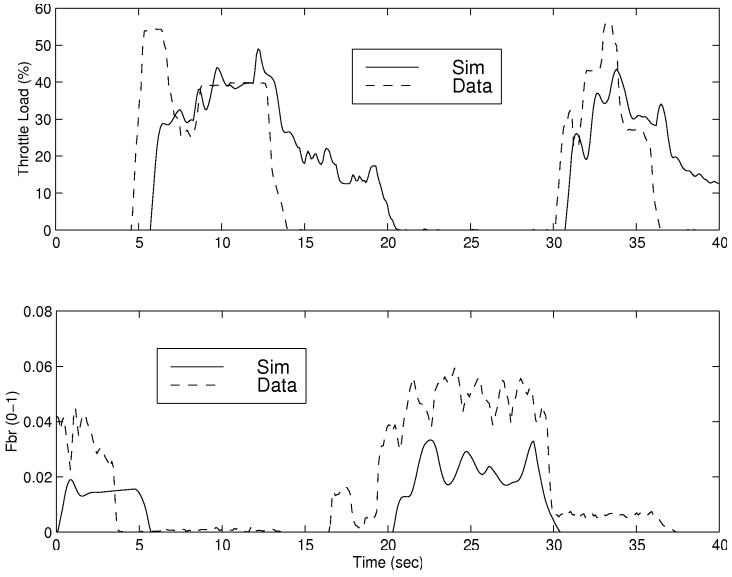

Figure 28: ACC actuation commands

5 Supervisor Logic

The controller in the previous section is suitable for use in the stop and go scenario. However, we want the controller to also perform well in the free flow situation and to smoothly switch between the two cases. In this section, we will describe a supervisor controller which characterizes the situation and chooses the appropriate control law.

5.1 Gain Scheduling

The traffic scenario can be described using the vehicle velocity, range and range rate. The gain scheduling makes use of range-range rate diagrams for a graphical description of traffic scenarios. The following regions, shown in Figure 29, are used by the supervisor:

1. High Speed Cut-In: Vehicles detected in this region are moving faster than the ACC vehicle. Given enough time, they will move safely away from the ACC vehicle. Hence the ACC vehicle should ignore the detected vehicle and track its driver-set velocity.

2. Normal: Vehicles in this region are traveling at about the same relative velocity of the ACC vehicle. This is the standard vehicle-following scenario and the controller should try to force the state to the origin of the range-range rate plot (i.e. range error and range rate converging to zero).

3. Low Speed Detection: Vehicles detected in this region are moving slower than the ACC vehicle, so the brakes must be applied to prevent a collision. Since the vehicle spacing is quite large, an aggressive controller is not needed. Range tracking is not important at this point, so the range error gain should be zero. The range rate should be reduced using a low $\dot{r}$ gain which results in low decelerations.

4. Low Speed Cut-In: This region represents vehicles which have cut-in front of the ACC vehicle and have a lower velocity. Obviously this is a critical situation and an aggressive controller is needed to prevent a collision.

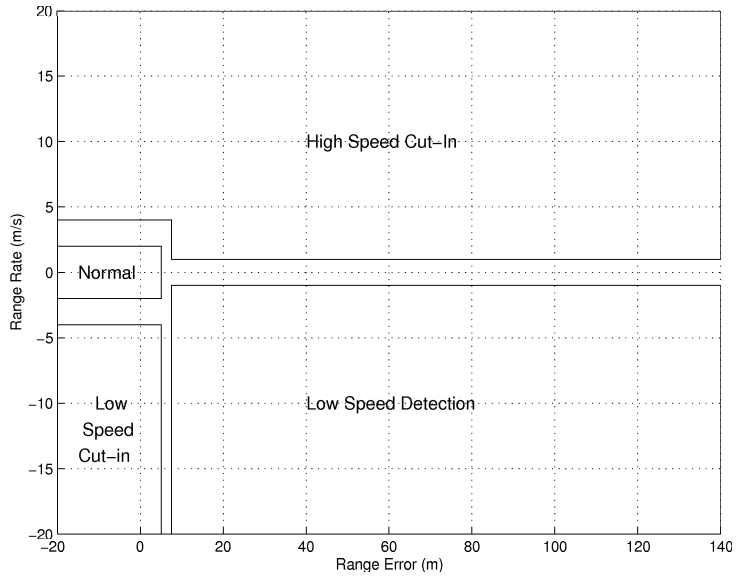

Figure 29: Range error vs. range rate regions

The Normal, Low Speed Detection, and Low Speed Cut-In regions all use the stop and go controller described above with gains tuned appropriately for the given region. The High Speed Cut-In region uses two PID controllers for desired velocity tracking: one for large velocity errors and one for small velocity errors. Notice that the range-range rate plot has several gaps between the defined regions. Linear scaling functions (scaling the a_{des} produced by the appropriate controller), defined on these gaps, are used to switch smoothly between each region. For example, the scaling function for the normal region is given in Figure 30. The controller for each region produces an a_{des} which is multiplied by the appropriate scaling function. The sum of these scaled desired accelerations is then passed to the lower controller as a composite desired acceleration.

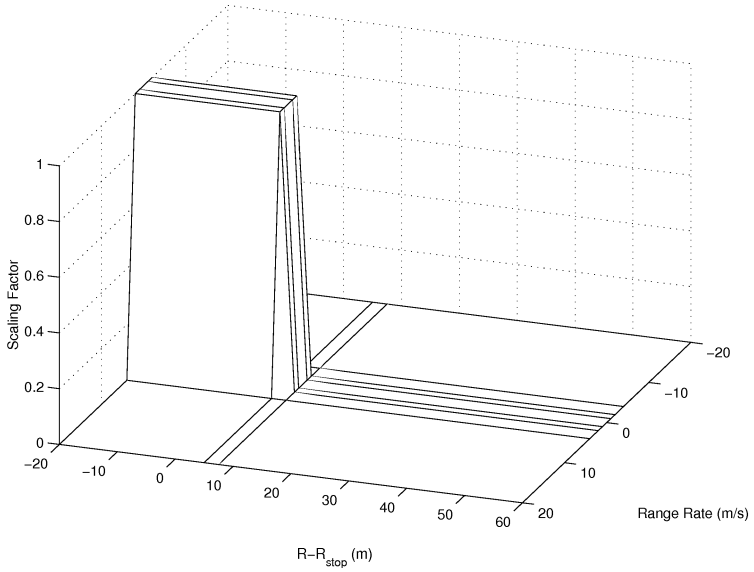

Figure 30: Scaling function for normal region

Thus far we have not described how the controller changes with respect to velocity. The controller should be more aggressive in the stop and go scenario (low velocities) due to the frequent accelerations/decelerations involved in this type of traffic. To build this velocity dependence into the controller, a stop and go controller was tuned with aggressive gains. Then another scaling function, show in Figure 31, was introduced to smoothly switch between the freeflow and stop and go scenarios. The unified controller works as follows:

if the detected vehicle is in the high speed cut-in, normal, low speed cut-in regions, or any combination, then the supervisor uses the stop and go controller at low speeds and the controllers described above at high speeds. In the crossover region between low and high speeds, the supervisor scales between the freeflow and stop and go controllers. If the detected vehicle is in the low speed detection region, the same controller is used regardless of vehicle velocity, so no scaling is needed.

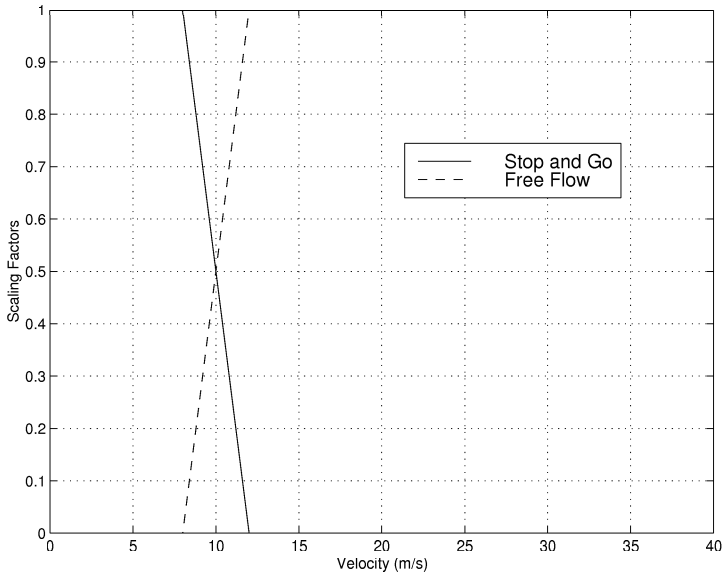

Figure 31: Velocity scaling factors

5.2 Low Velocity Logic

At very low vehicle velocities, control is quite difficult due to delays in wheel speed measurements, torque converter dynamics, and various unmodeled dynamics. It has been the experience at PATH that bring controlled vehicles to rest requires an open loop term to smooth out the trajectory. This correction is an additional desired brake torque term appended to Equation 33:

$$\tau_{b,des} = (T_{e,ct} - \beta a)/(r_{drive} \cdot r_{gear}) - r \cdot (F_{rr} + F_d + W \cdot sin\Theta) + \tau_{lowvelocity} \tag{35}$$

The form of the low velocity term used by PATH is:

$if(v > 2.5)$

$\tau_{lowvelocity} = 0$

$else$

$\tau_{lowvelocity} = min(400.0 * \frac{2.5-v}{2.5}, 200.0)$

end

There are three parameters in this term: the turn on velocity (2.5), the maximum torque correction (200.0) and the slope of torque vs. velocity (400.0). The values chosen here were tuned in simulation. However, all three of these terms need to be tuned on the vehicle to provide a comfortable stop.

5.3 Simulation Results

The unified controller is designed to perform well under a variety of traffic flow situations. A handful of situations were simulation tested to verify that the controller has a wide range of operation.

The first scenario is a high speed cut-in. The ACC vehicle, traveling at 22 m/s detects a vehicle in its lane at 35 m and traveling at 30 m/s. As shown in Figure 34, the detected vehicle is strictly in the high speed cut-in region. The unified controller tries to track the driver setpoint velocity of 25 m/s. Figure 32 shows a bit of overshoot in the velocity tracking, but this can be removed by adding more damping (increasing the derivative gain on the PID controllers). This plot also shows that the range continually increases as the faster

moving vehicle moves out of range. Finally, Figure 33 shows the acceleration needed by the ACC vehicle to achieve this profile. The initial noisiness in this plot is a numerical problem as the simulation attempts to settle out the initial conditions. After these initial transients, the vehicle acceleration is relatively small which is good for driver comfort. Also, the lower level controller is reasonably tracking the desired acceleration produced by the upper level controller.

Next a low speed detection scenario was simulated. In this scenario, the ACC vehicle, traveling at its setpoint of 25 m/s, detects a vehicle moving at 12.5 m/s. The detection is made at an initial range of 150m (approximately the range of the radar). This scenario is complicated by the eventual switch over from the low speed detection region to the normal region, as shown in Figure 37. In the low speed detection region, the range error gain is zero, i.e. the controller strictly tries to reduce the range rate. However, in the normal region, the controller range error gain is non-zero, which can cause some erratic behavior at the boundary crossing. In this case, the plot of acceleration (Figure 36) shows an increase in deceleration at t=20 sec (the beginning of the crossover). This is due to the higher gains used by the normal mode controller. Also notice that there is a spike in vehicle acceleration at t=15 sec. This is due to a downshift in gears by the ACC vehicle. Finally, the range and range rate responses (Figure 35) show no overshoot which is good from the standpoint of driver comfort.

In the low speed cut-in scenario, a vehicle moving at 20m/s is spotted 8m in front of the ACC vehicle, which is moving at its set velocity of 25m/s. Figures 38-40 show the response of this critical scenario. There is a large initial application of the brakes to reduce the relative velocity and increase the vehicle spacing to a safe distance. This braking maneuver brings the vehicle across the normal region. Once the vehicle spacing is at a safe distance, the vehicle accelerates to bring the relative velocity back to zero. The brakes are applied rather harshly, so the low speed detection controller gains could be reduced.

The results of the stop and go controller simulations were shown in Section 4.5. However the controller also needs to be tested in the framework of the unified controller because the velocity trajectory passes from the stop and go region to the free flow region. The following results use the same preceding vehicle trajectory used in the previous stop and go controller testing. The tracking and results (Figure 41) are essentially the same as the pure stop and go controller tracking (Figure 26). Also a comparison of Figures 27 and 42 show that the vehicle accelerations are comparable. Figure 43 shows the stop and go scaling function (1 represents stop and go region and 0 represents freeflow). Thus the unified controller performs similar to the pure stop and go controller even as it switches between the stop and go and freeflow controllers.

The same stop and go scenario was tested with the inclusion of realistic sensor noise. Noise with a standard deviation of 0.5 m and 0.5 m/s was added on top of the range and range rate signals. Then the signals were discretize to represent a sensor sample time of 10 Hz. Furthermore, a variable time delay was added to the velocity signal. This delay was computed as the time it takes a wheel to revolve between hall effect sensors (assuming that there are 8 equally spaced hall effect sensors on the wheel). As the vehicle velocity approaches zero, the delay in the velocity signal becomes quite significant. A simple signal processing scheme consisting of first order filters with poles at 5 Hz was used to reduce the effect of this sensor noise. The tracking and acceleration plots of the stop and go scenario with noise are given in Figures 44 and 45. In comparison to Figures 41 and 42, these plots show that the controller performance is degraded by the noise but is still quite good.

The final scenario tested was the approach of a stopped vehicle. The ACC vehicle, moving at 10m/s, detects a stopped vehicle at an initial range of 150 m. All noise has been included into the sensor measurements, so this scenario should test the ability of the low velocity logic to bring the vehicle to a smooth stop. Figure 46 shows a fairly smooth response, but several interesting aspects are more apparent in Figure 47. First, the three spikes in acceleration before t=20sec are all due to down shifts in gear. At t=25sec, the vehicle acceleration diverges from the desired acceleration due to the low low velocity correction term. Then at t=30sec, the vehicle accelerates and then decelerates in a short time span. This is due to the crossover between the low speed detection and normal regions. Recall that the low speed detection region uses the same controller regardless of vehicle velocity. As the vehicle enters the normal region, the stop and go controller attempts to reduce the range error by accelerating and then must decelerate to bring the range rate back to zero. Running the test without the low speed correction term results in very little change in performance. These simulations are rather inconclusive due to the fact that we have not captured all the dynamics realized in low velocity vehicle dynamics. This confirms PATH's experience that the low velocity correction term needs to be tuned on the experimental vehicle.

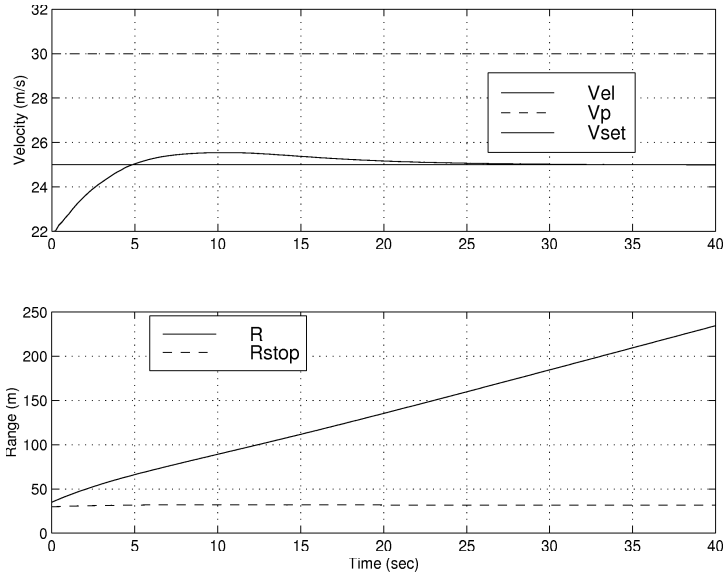

Figure 32: High speed cut-in: Tracking

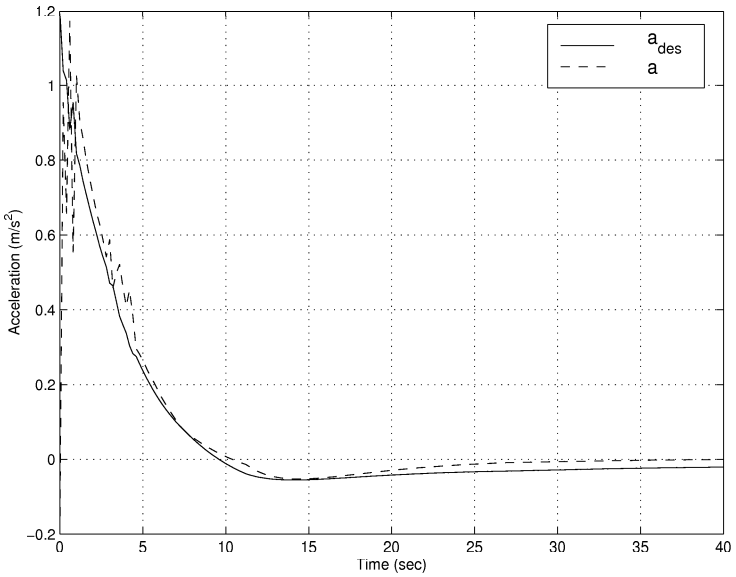

Figure 33: High speed cut-in: Acceleration

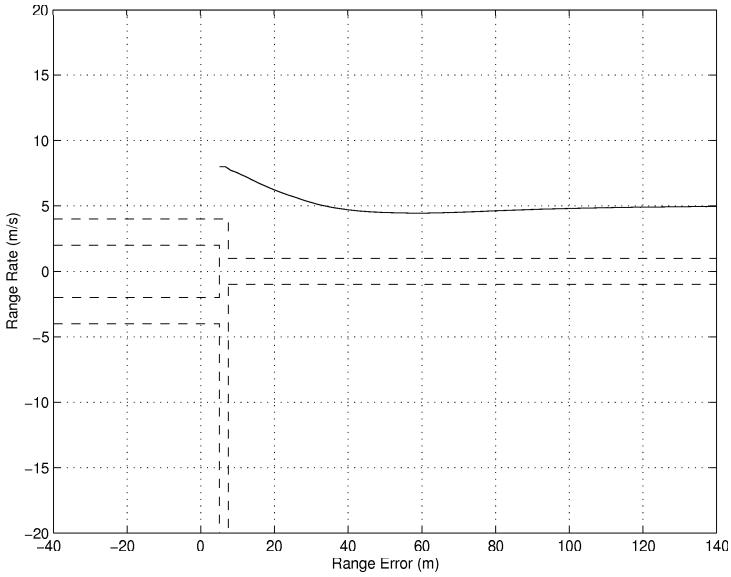

Figure 34: High speed cut-in: Regions

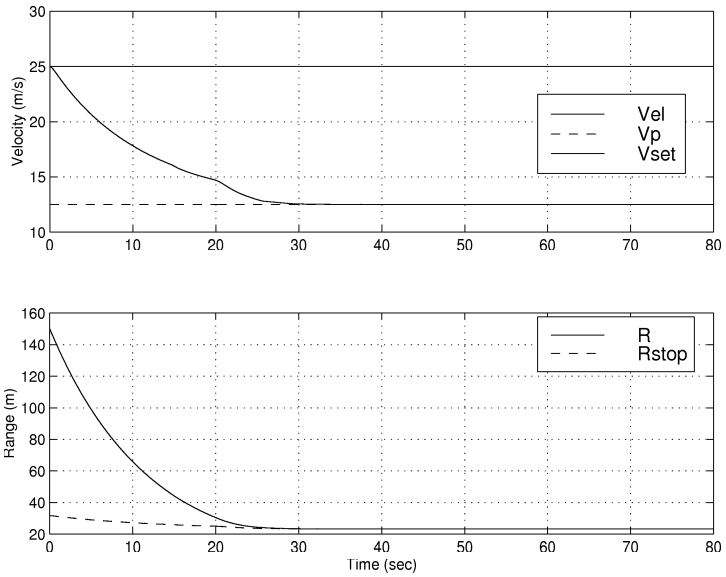

Figure 35: Low speed detect: Tracking

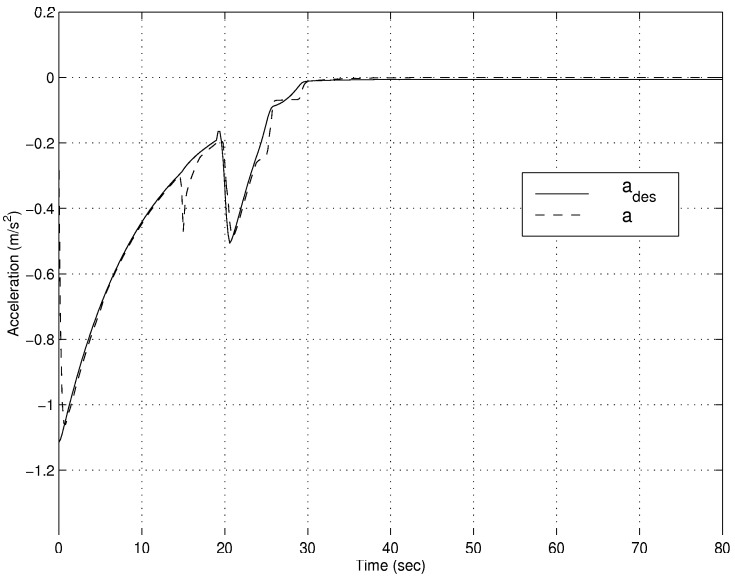

Figure 36: Low speed detect: Acceleration

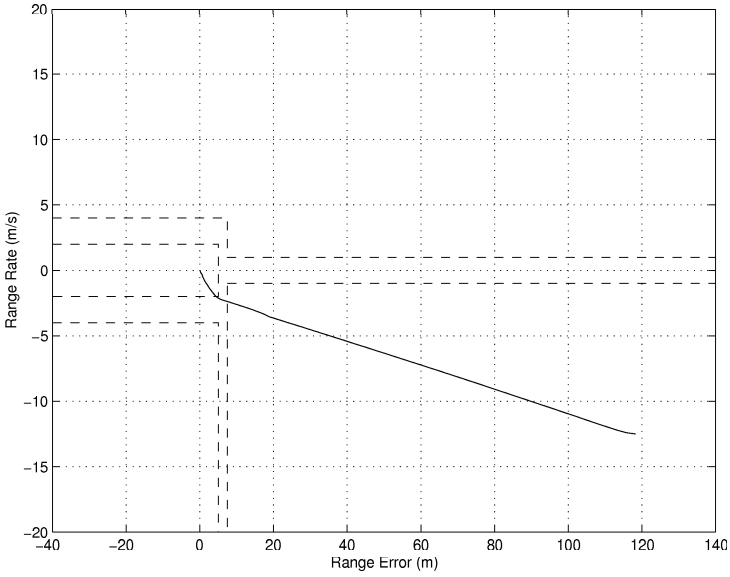

Figure 37: Low speed detect: Regions

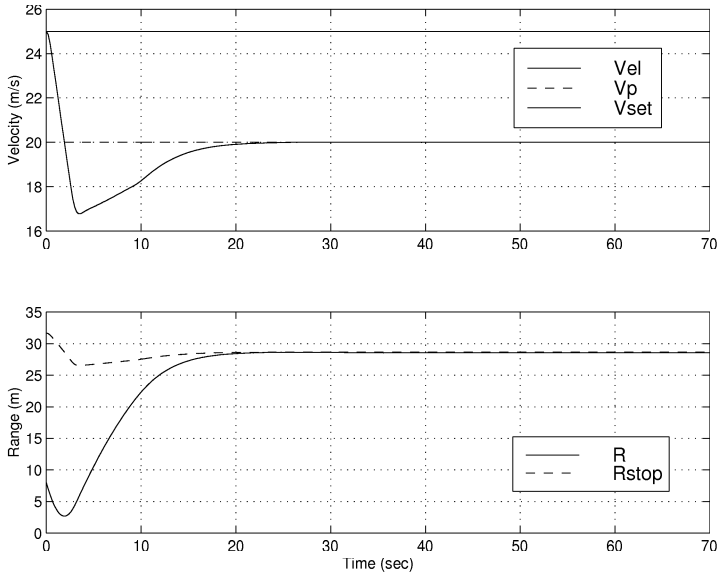

Figure 38: Low speed cut-in: Tracking

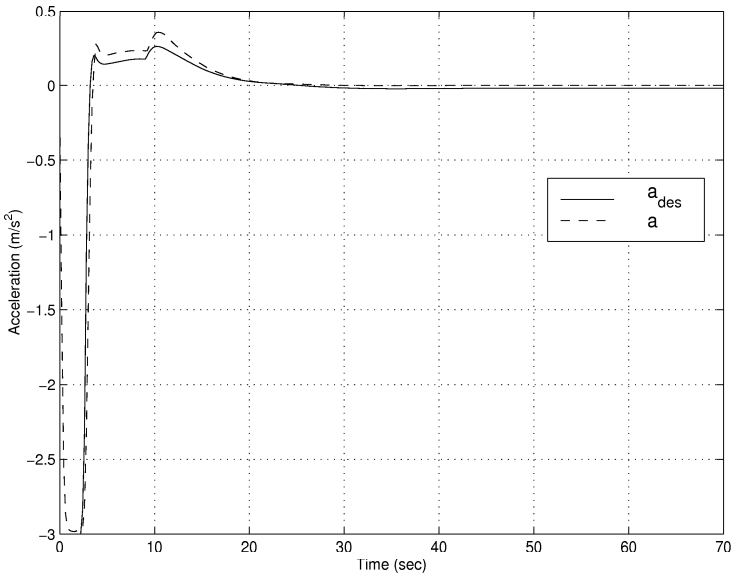

Figure 39: Low speed cut-in: Acceleration

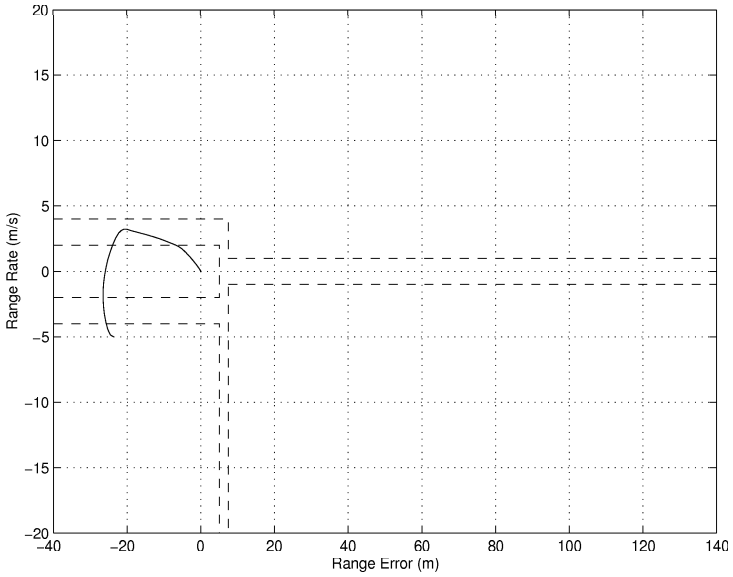

Figure 40: Low speed cut-in: Regions

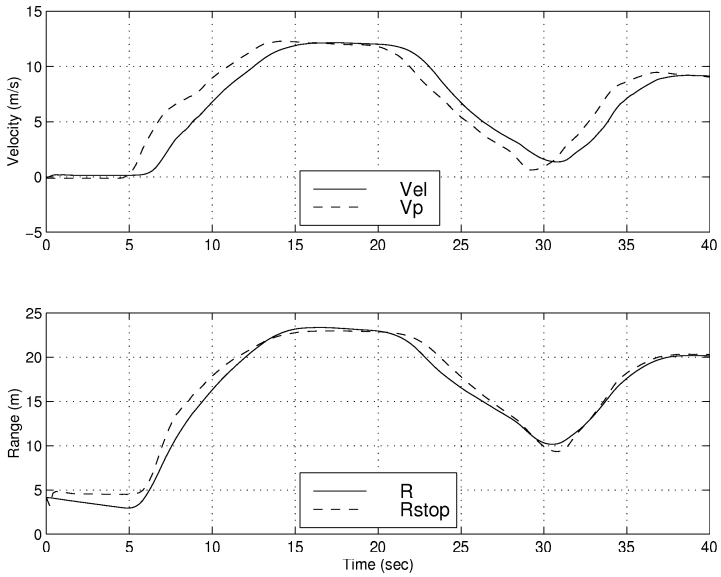

Figure 41: Stop and go: Tracking

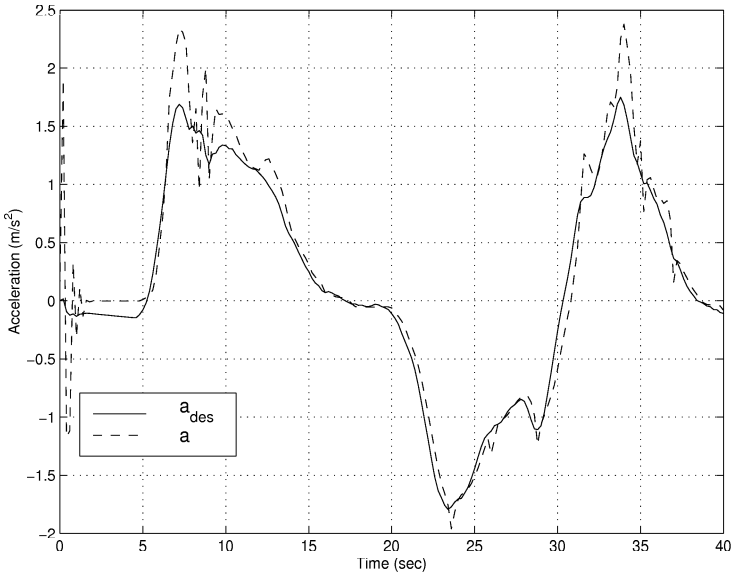

Figure 42: Stop and go: Acceleration

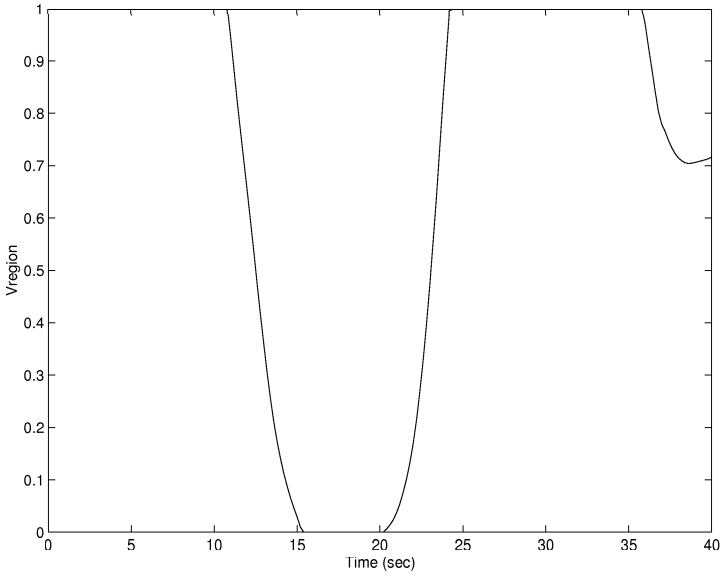

Figure 43: Stop and go: velocity scaling function

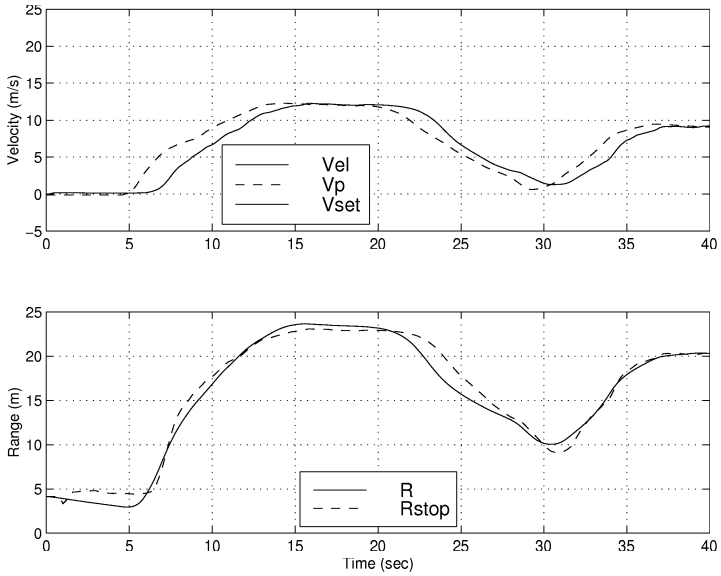

Figure 44: Stop and go with noise: Tracking

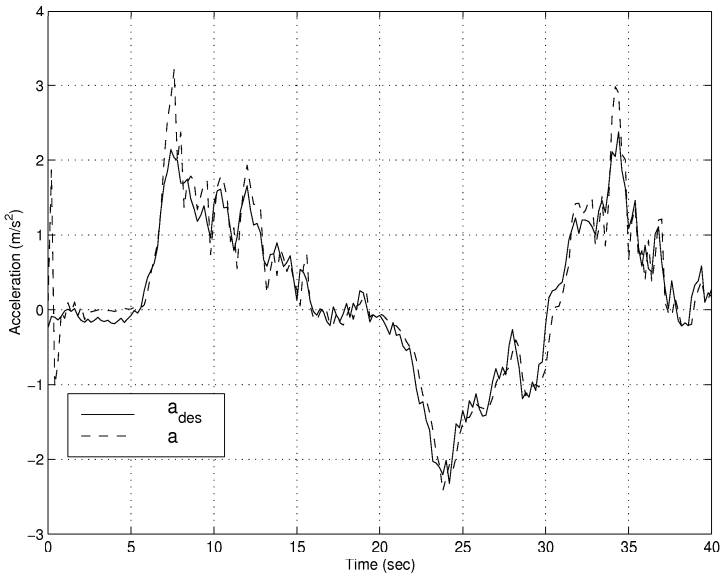

Figure 45: Stop and go with noise: Acceleration

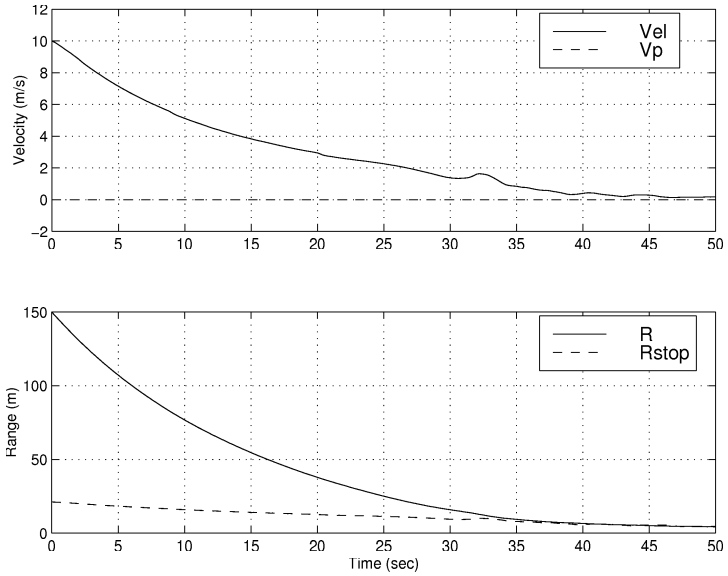

Figure 46: Approaching stopped vehicle: Tracking

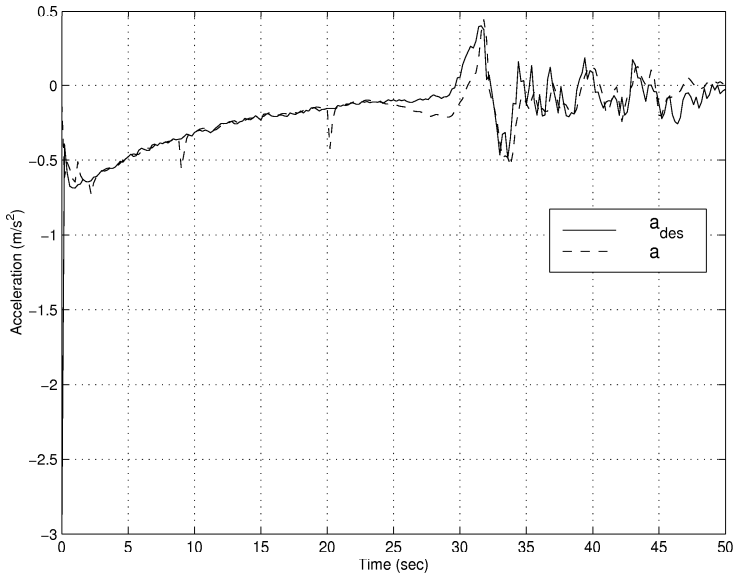

Figure 47: Approaching stopped vehicle: Acceleration

6 Controller Code

The unified controller was constructed in Simulink block diagram form and hence, it needed to be converted to C-code for implementation on the experimental vehicle. The following is a list of files needed to compile and run the C-code version of the controller:

- **upper_controller.c** This function takes in velocity, range, range rate, and the driver set velocity and returns the desired acceleration. The desired acceleration is computed using the supervisor controller described in Section 5.
- **lower_controller.c** This function takes in the gear engaged, velocity, desired acceleration, engine speed and pointers to the throttle and brake actuation commands. The function sets the actuation command variables to the desired values. The desired actuation commands are computed using the lower controller described in Section 4.2- 4.4.
- **sliding_controller.c** This function takes in the range error, range rate, and sliding controller gains. It returns the desired acceleration for a given sliding controller. This function performs the sliding control law computations given in Section 4.1. This function is called by upper_controller.c to compute the desired acceleration for each region (except the high velocity cut-in region).
- **pid_controller.c** This function takes in the velocity and driver set velocity. It is called by the upper_controller.c function to compute the desired acceleration for the high velocity cut-in region. This function uses two PID controllers (one for small and one for large errors) to compute the desired acceleration.
- **interpol2.c** This function performs a two dimensional table lookup using linear interpolation. It is used to compute the scaling functions for each region and to compute the desired throttle angle from the inverse engine map.
- **interpol.c** This function performs a one dimensional lookup using linear interpolation. It is used by interpol2.c and also to compute the velocity scaling function.
- **nrutil.c** This file contains functions which are used to allocate memory and create vectors/matrices using arrays.
- **control_par.h** This header file contains all the parameters used by the controller. This includes all vehicle model parameters and controller gains.
- **nrutil.h** This header file contains declarations for all functions contained in nrutil.c. This header file must be defined in all files in which a vector or matrix is initialized.
- **inv_speed.dat**, **inv_throttle.dat**, **inv_torque.dat** These files contain the data for the inverse engine map. The inverse engine map gives throttle angle (% throttle load) as a function of engine speed (rads/sec) and desired engine torque (N-m). Thus the inv_speed.c and inv_torque.dat files contain vectors while the inv_throttle.dat file contains the two dimensional map.
- **range_error.dat**, **range_rate.dat**, **lowvcutin_region.dat**, **lowvdetect_region.dat**, **normal_region.dat**, **highvcutin_region.dat** These files contain the scaling function data for each region described in Section 5. The scaling function maps are functions of range error (m) and range rate (m/s). The files range_error.dat and range_rate.dat contain vector data and the rest of the files contain the two dimensional maps for the appropriate region.
- **velocity_scaling.dat** This file contains the velocity scaling function data. The first column of this data file is the velocity (m/s) and the second column is the scaling function value.

The following piece of code will run the controller and place the desired throttle and brake actuator commands into the variables `u_throttle` and `u_brake`:

```
desired_accel = upper_controller(velocity,range,range_rate,set_velocity);
lower_controller(gear_engaged, velocity, desired_accel, engine_speed,&u_throttle,&u_brake);
```

The controller c code was tested in the following manner. The vehicle model was simulated for a given traffic scenario and the data for all controller inputs and outputs was recorded. Then, the data for the controller inputs was fed to the c-code version of the controller. The outputs of the c-code controller were then compared against the outputs of the simulink controller. Figure 48 shows the desired acceleration produced by the simulink and c-code controller for the stop and go scenario. This figure shows that the simulink and c-code upper controllers produce identical desired accelerations. Figure 49 compares the brake and throttle commands produced for this scenario. The throttle commands are almost identical, but there is some small lag in the c-code throttle command due to the implementation of discrete filters (as opposed to the continuous time filters used in the simulink controller). The brake controller shows some small differences between the c-code and simulink controllers. This is due to the computation of closed throttle engine torque. The c-code controller uses fewer points in its computation of closed throttle engine torque. These plots show that the c-code controller matches the simulink controller for the stop and go testing scenario. The c-code was also verified in all the scenarios given in Section 5.3.

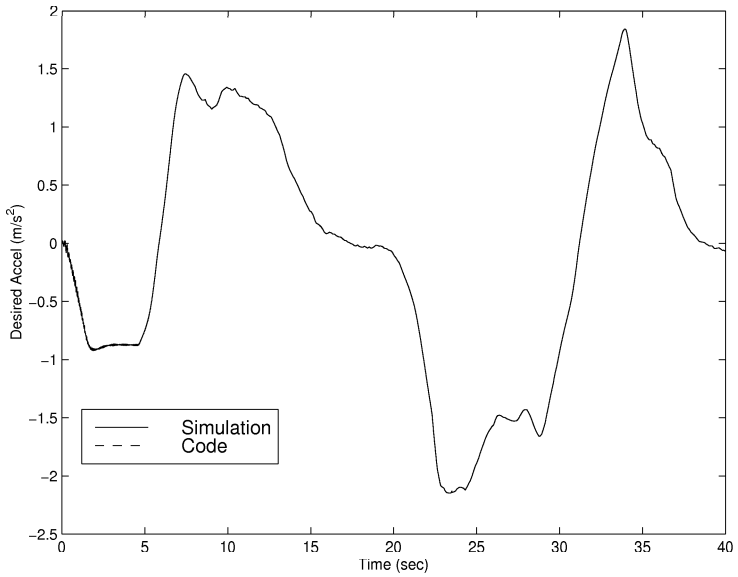

Figure 48: Code testing: desired acceleration

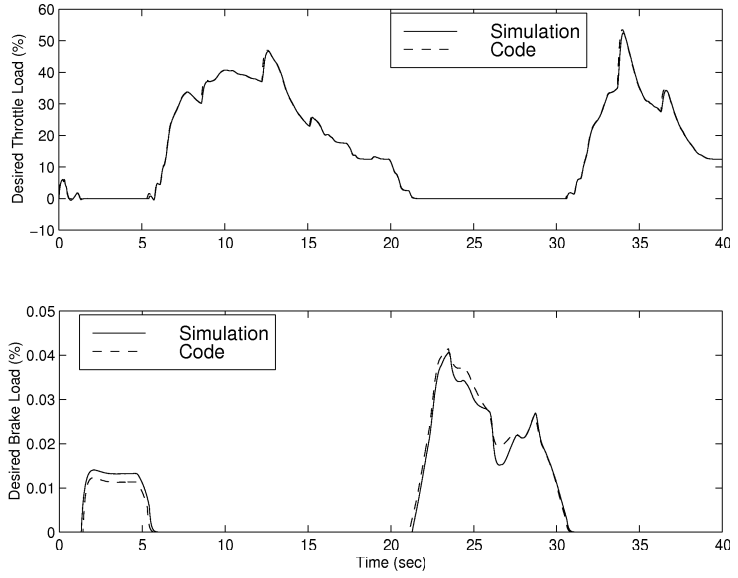

Figure 49: Code testing: actuator commands

References

[1] J. Christian Gerdes. *Decoupled Design of Robust Controllers for Nonlinear Systems: As Motivated by and Applied to Coordinated Throttle and Brake Control for Automated Highways.* PhD thesis, University of California at Berkeley, 1996.

[2] D. Maciuca. *Nonlinear Robust and Adaptive Control with Applications to Brake Control for Automated Highway Systems.* PhD thesis, University of California at Berkeley, 1997.

[3] H. B. Pacejka. The tyre as a vehicle component. In *XXVI FISITA Congress*, June 1996.

[4] H.B. Pacejka and I.J.M. Besselink. Magic formula tyre model with transient properties. *Vehicle System Dyanmics Supplement*, 27:234–249, 1997.

[5] J.Y. Wong. *Theory of Ground Vehicles.* John Wiley and Sons, Inc., second edition, 1993.