Modeling In-Use Engine Conditions and the Resulting Emissions

by

Todd Haugsjaa

B.S., Mechanical Engineering
University of Massachusetts at Amherst, 1997

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

February 1999

Signature of Author [illegible]
Department of Mechanical Engineering
January 15, 1999

Certified by
Wai K. Cheng
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by [illegible]
Ain A. Sonin
Chairman, Department Committee on Graduate Studies

Modeling In-Use Engine Conditions and the Resulting Emissions

by

Todd Haugsjaa

Submitted to the Department of Mechanical Engineering
on December 28. 1998 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

A model for evaluating the engine-out emission levels under different driving situations has been developed. By adopting different driving characteristics (hard accelerations, timid driving, cruise-control speeds, or erratic driving) and by assigning different driving routes, the model produces second-by-second emission levels from the car. In this thesis, quasi-static emission maps were used for emission calculation. As such, a strong correlation between emission mass flowrates and brake power has been established except when enrichment is encountered. At low powers, where stoichiometric mixtures are seen, a linear brake power dependence is obtained for HC, CO, and NO emissions. At points where a high level of enrichment occurs, HC emissions increase, much higher CO emissions are seen and NO emissions decrease considerably.

Thesis Supervisor: Wai K. Cheng
Title: Professor of Mechanical Engineering

[illegible]

Contents

Chapter 1 - Introduction

1.1 Background

Recently, out of concern for their environmental impact, more and more attention has been paid to the emission levels exiting the engine and tailpipe of vehicles. In an effort to reduce these levels, much research has been done to model emission formation mechanisms inside an engine. See [1] for an account of the in-cylinder hydrocarbon formation mechanisms. As helpful as these models are in understanding basic mechanisms, laboratory studies seldom encompass what engines are subjected to under real world driving conditions; in particular, the effects of transient loads and speeds, open loop enrichment, and other unsteady events need to be addressed.

Understanding the relationship between transient engine conditions and emissions is essential. Such an understanding will highlight important unsteady engine emission mechanisms and will assist the engineering efforts to lower their levels. It will also provide a basis for assessing the impact that vehicle emissions will have on regional air quality.

There are several methods of obtaining vehicle emission data for analysis. The data may be collected via:

1) quasi-static operation where engine conditions are held constant with a dynamometer for each data point. From this data, a quasi-static emission map can be constructed. The map is typically represented by contours of an exhaust species' specific emission against engine variables such as torque and speed.

2) dynamic drive-cycles, where either:

a) a representative driving cycle is followed with the vehicle on a chassis dynamometer which imposes an appropriate load/speed relationship, or
b) an on-the-road test is conducted, where the vehicle is equipped with apparatus to measure emissions on-board.

1.1.1 Quasi-static Maps

Quasi-static mapping is readily available as a standard engine database and provides the basic emission characteristics for a vehicle. The map is vehicle specific because the emission levels are dependent on the engine controller setting for levels of EGR, spark timing and fuel metering. The emission levels of the vehicle can be mapped against load and speed under the particular engine management system.

While quasi-static maps are common, they do not reflect the transients that occur while driving (open-loop enrichment, prior engine operating trajectory effects, catalyst lags, etc.). To get closer to real-world emission evaluations, dynamic testing methods have been developed to analyze emission levels in a driving simulation.

1.1.2 Drive-cycles Conducted on a Dynamometer

The drive-cycle tests are more realistic than quasi-static maps since the engine operating conditions are continuously changing during the data collection process. The test produces second by second emissions data while the vehicle follows a predetermined velocity profile. The standard driving cycle used is the Federal Testing Procedure (FTP). While it provides a common ground for vehicle certifications, the procedure differs substantially from real-world driving. In particular, it ignores the effect of road grades and hard accelerations and thus a substantial part of the driving experience. This shortcoming is illustrated in Figure 1.1 where a Ford Taurus on a flat road was driven to its maximum accelerating and decelerating limits from steady speeds of 0, 10, 20, etc. mph [2]. Plotted with a solid line against the vehicle's speed are the achievable accelerations and decelerations of the car; thus these lines represent the operating speed/acceleration envelope. On the same plot, the dots inside this envelope indicate the accelerations and decelerations that occur during the FTP cycle for each second.

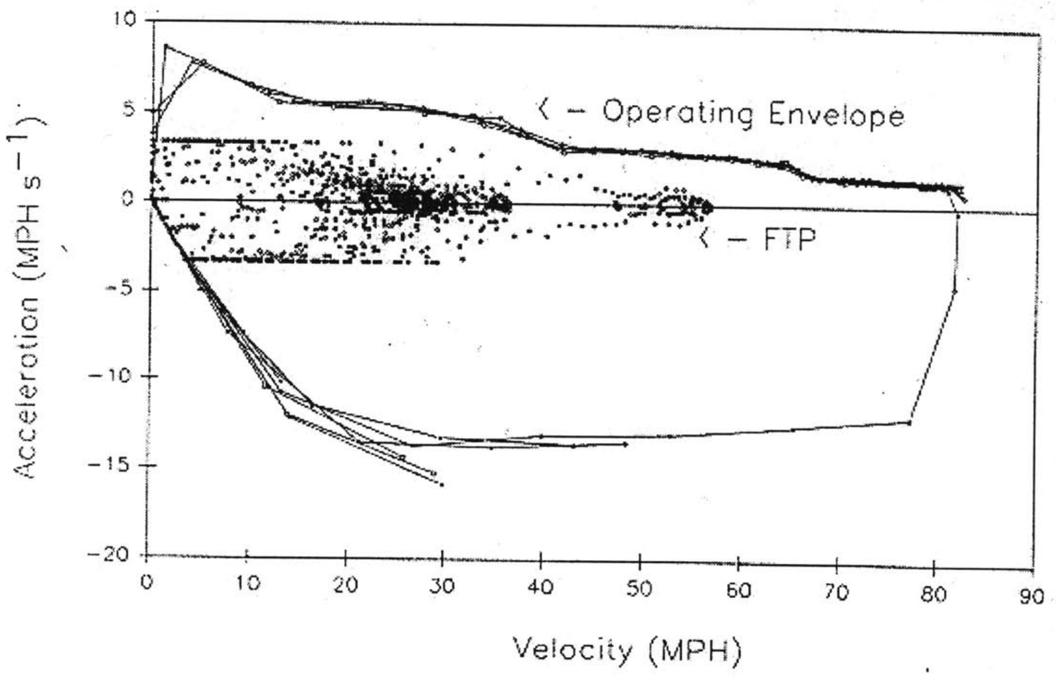


Figure 1.1: The extreme accelerations of a 1991 Ford Taurus with a 3.0L V-6 engine plotted against vehicle speed (solid line) and the accelerations reached during the FTP cycle (dots).

While the boundary of the operating envelope represents extreme driving conditions, Figure 1.1 illustrates that much of the achievable vehicle operating modes are not addressed by the FTP. Furthermore, the data from the FTP are aggregated into three phases. Thus, while suitable for comparing vehicles, it cannot easily be used to make a realistic emission assessment.

1.1.3 On-board Emissions Measurements

Since on-the-road driving conditions can differ substantially from the FTP conditions, use of the latter data could seriously under-assess real-world vehicle emissions. It has become evident that there are crucial vehicle operating conditions where emission levels can far exceed those under normal driving. To identify these conditions, it is necessary to evaluate real-world driving emissions by means of on-board emissions measurements.

On-board equipment allows the analysis of many different conditions encountered on the road such as hard acceleration up hills, steady or erratic driving characteristics, AC loading, etc. The down side to this is the major time and money required for outfitting a vehicle with the proper equipment and computers and then processing all the data into a comprehensive form.

1.2 Objective

Realizing the importance of understanding real-world driving emission levels and considering the large resources required to examine them in practice, it is attractive to develop a phenomenological model that can simulate a driving route and produce expected emission levels. The overall strategy is to have a vehicle driving model that relates the vehicle on-the-road behavior to engine operating conditions, and then relate the latter to emissions. The model is phenomenological in the sense that it is used as a structure to summarize data from an on-the-road test with different vehicles and with different route characteristics. This investigation is the first phase in an effort to model real world driving emissions. The scope is to make use of the basic information from quasi-static maps in reproducing the engine out emissions for a given road and driving profile.

The output of this phase 1 effort provides the basis for comparison with the data obtained from on-board emissions equipment in a road test with the same road and driving profile for phase 2. By comparing these results, the departure from quasi-static behavior can be quantified and appropriate transient models can be developed.

In this phase of the study only the engine out data are considered. The tailpipe emissions are very much catalyst specific, and the catalyst is responsible for much of the transient tailpipe emissions behavior. The study of these effects is postponed to phase 2 of the project.

Chapter 2 - Engine Quasi-Static Emissions Data

For the first stage of the project, a basic quasi-static map was needed to evaluate the model. A dynamometer mounted 2.0 liter Ztec engine from a Ford Contour was used to map the quasi-static engine out emission levels and the specific fuel consumption (sfc). Torque values range from 40 to 140 N-m; the engine speed ranges from 1500 to 4000 rpm. The engine was managed by the Ford EEC IV controller unit under all time.

When mapped against torque and engine speed, certain trends in the emission indexes (EI) and specific emissions for individual species can be seen. The EI value of an emission species is defined by the mass emission normalized by the fuel mass input to the engine. This gives a good indication of how much of the fuel becomes a particular emission species. The specific emission is the mass flowrate of the emission normalized by the brake power. This takes into account fuel efficiency of the engine (sfc) by comparing the emission species to the brake power produced. For example, two vehicles may have the same emission index maps, but if one has higher values of sfc (it is less fuel efficient), it will have higher emission flowrates than the other car.

The sfc predominantly decreases with increasing torque due to lower throttling losses, and at high torque sfc decreases as speed increases (Figure 2.1a). Values range from 240 to 360 g/kW-hr. As torque and speed increase, the air to fuel equivalence ratio (Lambda) decreases according to the engine management strategy (Figure 2.1b). The CO emission index (EICO) and specific CO emission (sCO) (Figures 2.1c-d) both increase accordingly. The phenomenon responsible here is the tendency of the enriched mixtures to burn incompletely. Thus, high CO emissions occur when lambda is below 1 (the mixture is fuel rich) as is demonstrated in Figure 2.2. From this, it can be seen that EICO is only a function of lambda, regardless of the speed and torque. Ranges seen are from 0.84 to 1 for lambda, 1% to 60% for EICO, and 20 to 140 g/kW-hr for sCO. Inside the boundary of ~100 N-m and ~3000 rpm, sCO stays at a relatively flat level of ~20 g/kW-hr. Beyond this boundary, the values increase rapidly with both speed and load to ~140 g/kW-hr. Since the operating points in the FTP stay within the mentioned boundary, radically high values of CO emissions are usually not seen in that test. Road driving, however, often reaches beyond these boundaries and, as will be seen later in this thesis, it results in heavy CO emissions.

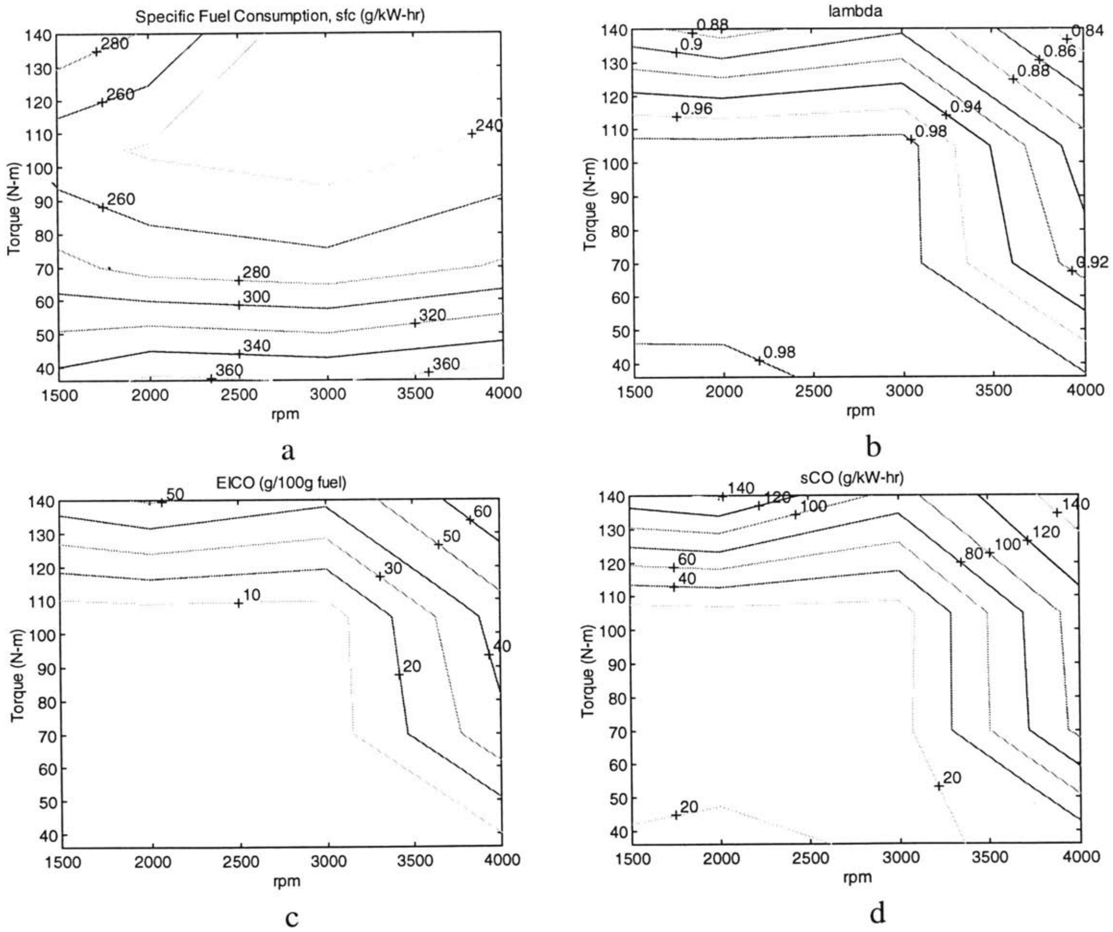


Figure 2.1: Contour maps of (a) specific fuel consumption (sfc), (b) lambda, (c) Emission Index CO levels (EICO), and (d) specific CO levels (sCO) all mapped against engine torque and speed.

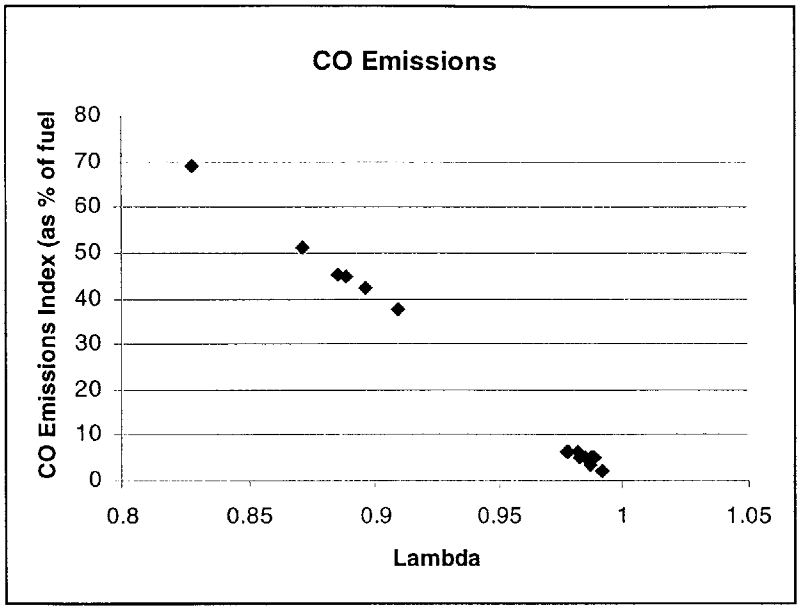
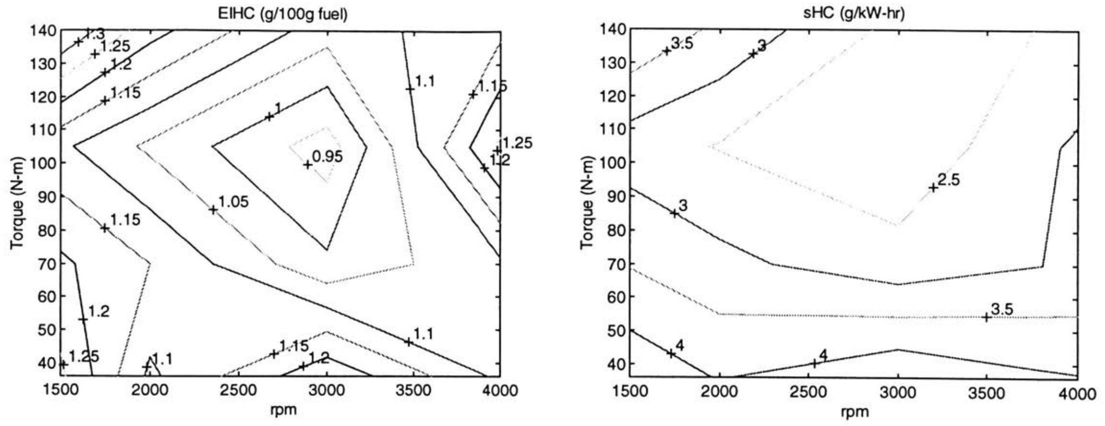


Figure 2.2: Linear relationship between EICO and lambda.

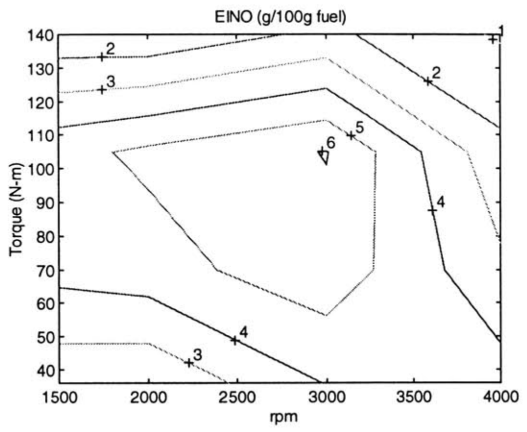
` The contour of EIHC (Figure 2.3a) has a valley in the middle ranges of torque (100 N-m) and speed (3000 rpm), but has a small spread of only 1% to 1.3% over the entire map. The sHC levels (Figure 2.3b) have a limited range of 2.5 to 4 g/kW-hr; since sHC is the product of the EIHC and sfc, and the former values are relatively flat over the map, the sHC values (Figure 2.3b) predominantly follow the trends of the sfc (Figure 2.1a).

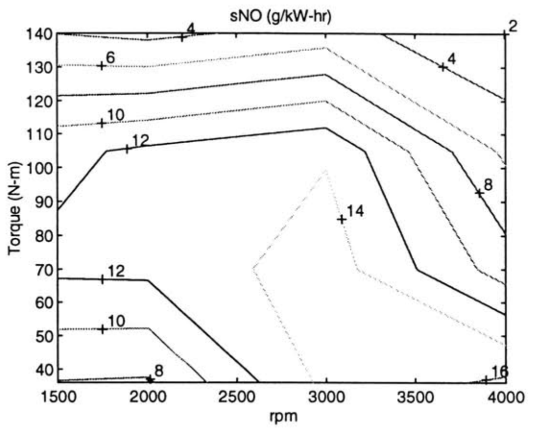


a b

Figure 2.3: Contour maps of (a) Emission Index HC levels (EIHC), and (b) specific HC levels (sHC) both mapped against engine torque and speed.

The values of EINO (Figure 2.4a) have a substantial spread of 1% to 6% over the map. The values peak at around 3000 rpm and 100 N-m. Production of NO is both sensitive to temperature and to the availability of oxygen. From the peak value at ~3000 rpm and ~100 N-m, the drop off of the EINO at higher torque and speed is due to enrichment (see Figure 2.1b); both the combustion temperature and the oxygen availability are reduced with enrichment. Also at high speeds, the combustion phasing is effectively retarded, leading to a lower combustion temperature. The drop off at low load and rpm is due to a higher heat loss relative to the charge energy, again resulting in lower combustion temperatures. Levels of sNO (Figure 2.4b) extend between 2 and 16 g/kW-hr, with the high values encountered at low torque, high speed levels (due to high sfc values at low torque).





a b

Figure 2.4: Contour maps of (a) Emission Index NO levels (EINO), and (b) specific NO levels (sNO) both mapped against engine torque and speed.

While these maps give a good indication of how emissions relate to the engine conditions, transient effects play a large role in determining the actual emissions. After setting the groundwork for the model in this stage, stage 2 will identify how these transients affect the emissions.

Chapter 3 - Computer Model

3.1 Overview

The model described in this thesis provides the second-by-second quasi-static engine out emissions for a vehicle. It can either create a velocity profile for a vehicle from a specified road profile or can follow a specified velocity profile (such as the FTP). Of primary interest is the case for a given road profile since it represents real-world driving scenarios. The case for a given velocity profile is also incorporated since it helps in evaluating the emission modeling by computing the emission levels for a standardized test profile that is available for every passenger car.

When following a road profile, the model uses topology data, stopping points and speed limits for the route along with a pre-defined acceleration profile to create a time profile of vehicle conditions. Conditions such as speed, acceleration and the corresponding rpm and brake power are calculated and used to evaluate the emissions.

If a velocity profile is given as the input, the engine conditions are calculated according to the road load requirements. For the FTP test, the road is assumed to be level, so aerodynamic and road drag and acceleration are the only indicators of the power requirements.

In both cases, once the engine operating conditions are found, the emissions are then interpolated from quasi-static emissions maps for the engine. In this stage of research, the emissions are determined solely from the quasi-static map. For future work, the output from this will be compared to actual on-road tests. The vehicle used for this stage is a Ford Contour with a 2.0 liter Ztec engine; specifications for the vehicle used in this model are provided in Appendix A.

3.2 Description of Procedure

3.2.1 Using a Prescribed Road Profile

The procedure used to create the time profile of vehicle conditions from a road elevation profile will be described first. The procedure of following a predetermined velocity profile against time is a subset of the above and is discussed at the end.

The structure of the discussion below follows the sequence of information flow in the computer program. The procedure of obtaining the various data is discussed.

3.2.1.1 Input Files

The program begins by reading in data files containing the road profile (distance traveled, elevation, stop points, speed limits), physical car data (drag coefficients, frontal area, wheel diameter, displacement, number of gears, min/max torque and rpm, overall gear ratios), the shifting schedule (the engine speed at which upshifts and downshifts are requested for different torques). Also read are the quasi-static engine maps for engine out emission-indexes as well as the specific fuel consumption. The format of these files are explained in Appendix B.

3.2.1.2 Creating the Road Profile

For a given route, the road profile was generated with a topographical route selection program (TOPO!). By specifying a desired route on an electronic map, the program will create a profile of elevation against distance traveled (not horizontal distance). The image of this profile is saved as a compressed bitmap and is imported into an image analysis program, ImagePro Plus. A tracer follows the elevation of the route. The pixel coordinates of this trace (the origin being at the top-left corner of the image) are recorded by the tracer and are saved to a file. The scale of the x- and y-axes must also be noted in the number of pixels per the length in miles or feet.

A subroutine contained in the model converts the pixel information into miles traveled and elevations in feet. It also reads a file containing the points along the route where the speed limit changes and where stops occur. This data is combined with the topological data and saved in another file for use by the main model. Once this is done, the new file can be used for later runs of the program without converting the pixel information again.

An example of this procedure is the route from Central Square in Cambridge, MA to the Route 128 interchange on Route 2. The route is marked on the TOPO! map in Figure 3.1, and this route's elevation profile, produced by TOPO!, is in Figure 3.2. This is a typical route taken by many commuters each day from the city to a suburban town community.

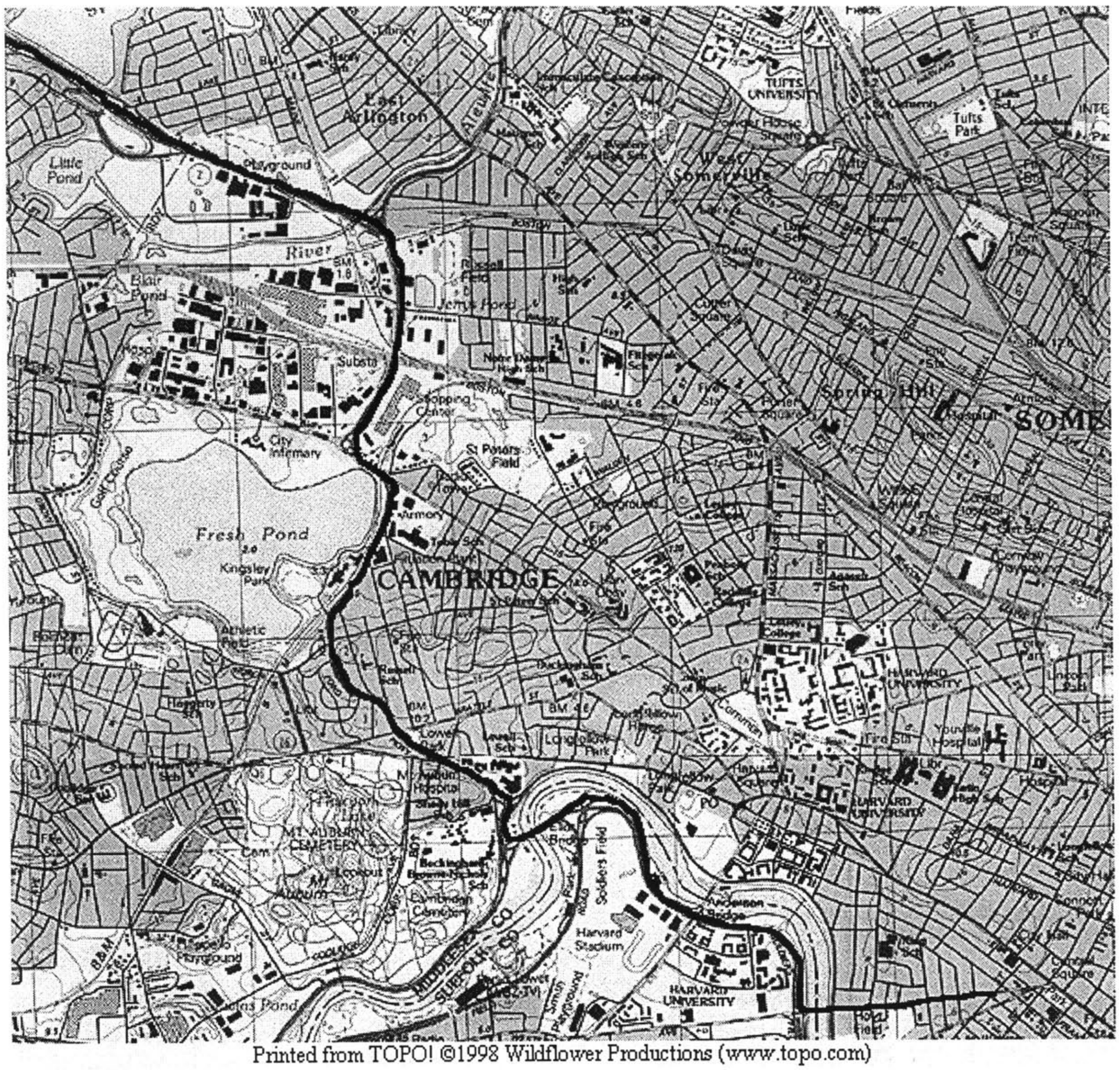
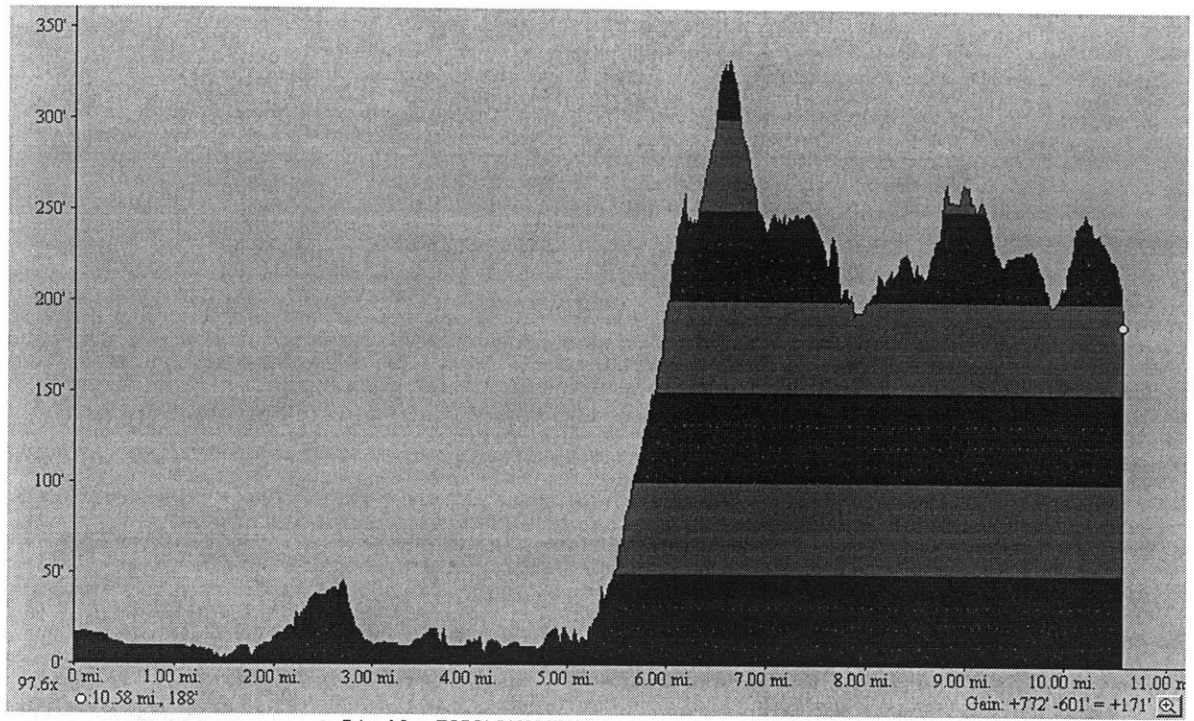


Figure 3.1: TOPO! map of Cambridge, MA area with the route from Central Square marked by a heavy line.



Printed from TOPO! ©1998 Wildflower Productions (www.topo.com)

Figure 3.2: TOPO! elevation profile of the route from Central Square in Cambridge, MA to the Route 128 interchange on Route 2. The x-axis is the distance traveled (not horizontal distance) in miles; the y-axis is the elevation in feet.

3.2.1.3 Shifting Schedule

The engine is connected to the wheels via the transmission. For a manual transmission, the shifting schedule is driver specific; for an automatic transmission, it depends on the shifting controller (either electronic or mechanical analog type) strategy. In order to make this model less vehicle and driver specific and more applicable to any vehicle, the shifting schedule was chosen to be a reasonable approximation of how most automatic transmissions will shift. The procedure is based on a normalized engine speed versus torque map on which the shifting boundaries are drawn; see Figure 3.3. An upshift occurs when the operating conditions are above the upshift boundary (in region A) and downshifts occur when they are below the downshift boundary (in region C).

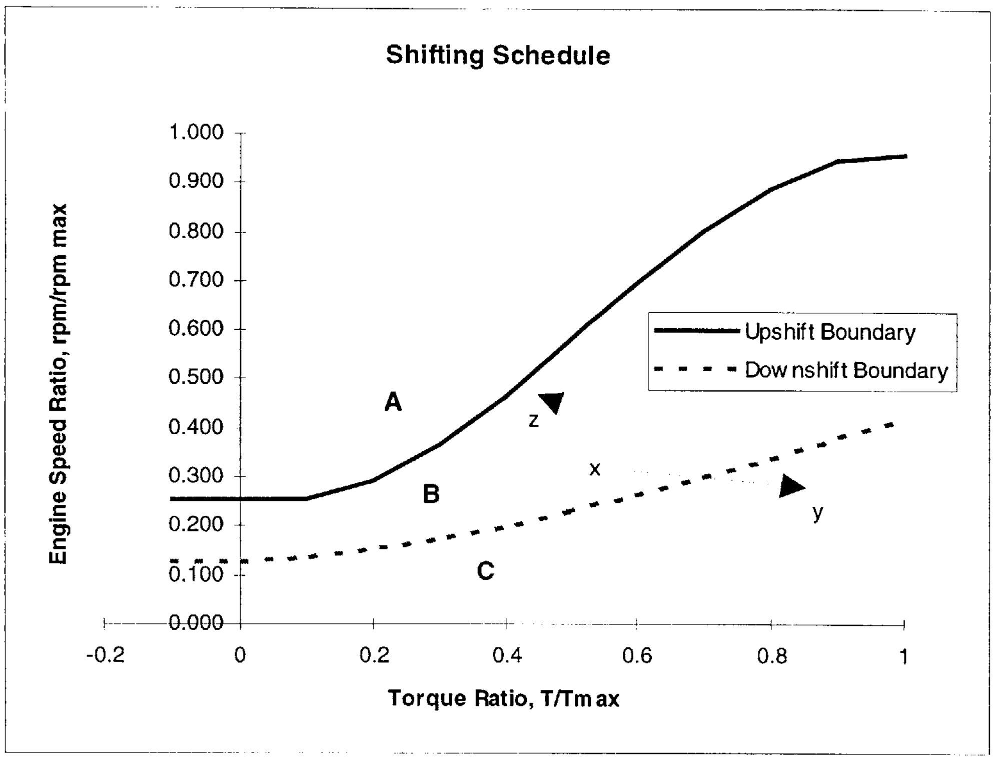


Figure 3.3: Shifting Schedule. Upshifts are requested above the upshift boundary (in region A) and downshifts are requested below the downshift boundary (in region C). Points x, y, and z demonstrate a possible engine trajectory, where a hill is encountered.

For example (see trajectory xyz in the figure for the following discussion), say initially the vehicle is traveling on a level road at a constant speed in fourth gear. Operating point x in the stable region B on the map (no shifting required) represents this. Then a steep hill is encountered. The driver will depress the gas pedal to maintain the speed (increase the torque). Eventually, the engine will have trouble maintaining the needed power due to its limited torque range and the operating point will drift across the shifting boundary to point y. Then the shifting controller requests a downshift. In the lower gear, the engine speed increases and the torque needed drops. Thus, the operating point returns to the stable region (at z). In general, more than one downshift may be needed for the engine to keep the operating point in the stable region. The upshift scenario (e.g. going downhill) is similar, but for this case, the operating point crosses from region B up to region A.

3.2.1.4 Calculating the Vehicle Motion

To figure out the car's velocity profile against time, the program first has to decide the car's acceleration rate: accelerating; cruising at a constant speed; decelerating for a traffic light (using brakes); or decelerating by coasting (i.e. for a slower speed limit). This decision is based on the vehicle's current conditions, speed limit, etc., and the location of up-coming stops.

Let us consider the simple case where the car accelerates from rest and the next stop is far away enough to provide ample room to reach a specified cruising speed, cruise for a while, and finally decelerate to the stop. Figure 3.4 shows a generic acceleration profile to reach the target speed and the corresponding speed in the acceleration part of this journey.

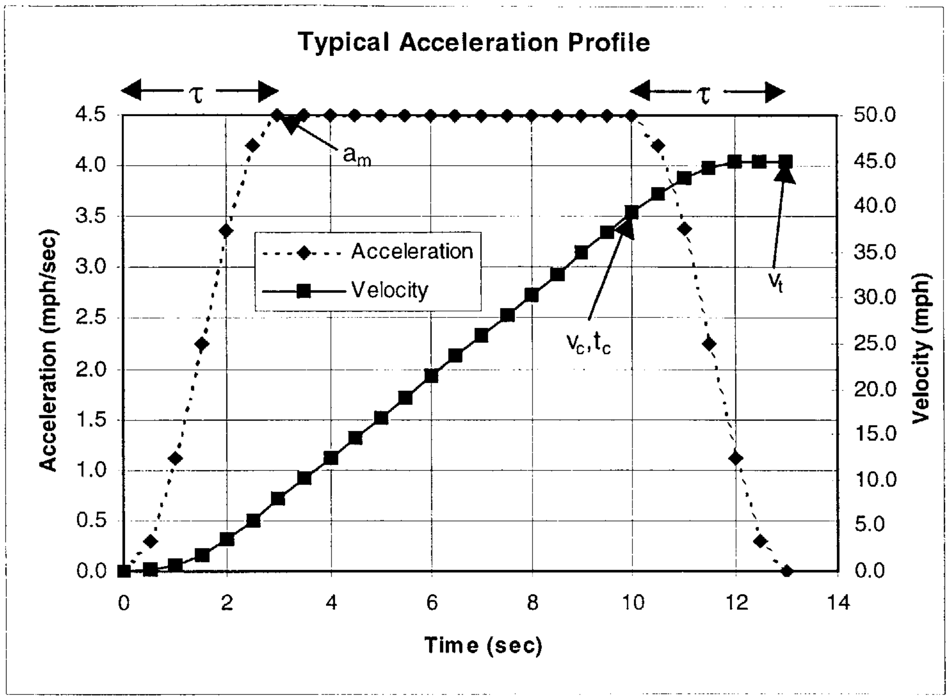


Figure 3.4: A representative acceleration profile. This figure shows the profile followed to reach the target velocity, v_t=45 mph. Here, the maximum acceleration, a_m, reached is 4.5 mph/sec. It takes τ=3 seconds to reach this level. The acceleration rate falls at time, t_c, when the velocity reaches v_c.

The first part of the acceleration profile follows an offset cosine curve:

$$a(t) = \frac{a_m}{2}[1 - \cos(\frac{t\pi}{\tau})] \quad \text{where} \quad 0 < t \le \tau \tag{1}$$

so that the maximum acceleration, a_m, is reached at a predetermined time, τ. For the second part of the acceleration, after time τ, the acceleration is fixed at a_m and the speed increases linearly with time.

At a certain time, t_c (to be defined later), the acceleration tapers off in the third part of the acceleration profile. This last part follows the second half of the cosine curve, which is designed to have the same duration, τ, as the initial acceleration build up and is now represented by:

$$a(t) = \frac{a_m}{2}[1 + \cos(\frac{t - t_c}{\tau} \cdot \pi)] \quad \text{where} \quad t_c < t \le t_c + \tau \tag{2}$$

The objective is that the vehicle speed should reach the target speed, v_t, at the end of the acceleration profile. Thus, the velocity, v_c, at the switching time, t_c, may be solved from:

$$v_t = v_c + \int_{t_c}^{t_c + \tau} \frac{a_m}{2}[1 + \cos(\frac{t - t_c}{\tau} \cdot \pi)] \cdot dt \tag{3}$$

$$v_c = v_t - \frac{a_m}{2} \cdot \tau \tag{4}$$

The procedure is to keep track of the velocity, v, until $v = v_c$; then the acceleration tapers off. Normal values of $a_m = 4.5$ mph/s and $\tau = 3$ seconds are used. These settings give realistic profiles that match practical driving.

This strategy is used unless the required torque reaches the maximum torque output of the engine; then the acceleration is recalculated according to the maximum available torque. The calculations for this scenario are covered at the end of section 3.1.2.5.

Consider the case where the vehicle approaches a stretch of road where speed limit increases. When this occurs, the same acceleration profile is followed, except now the starting velocity is non-zero.

When the speed limit decreases and there is ample road space ahead (no stopping point), the vehicle will coast to the lower speed. The value for a_m is now negative (-1 mph/s), but the above procedure still holds true.

Now, assume the vehicle has reached a cruising speed and a stopping point is approaching. The deceleration profile for braking to a stop is similar to the acceleration profile except that a_m is now negative, set at -6 mph/sec, and the target velocity is zero. In order to stop the car at (or slightly before) the stopping point, the distance required to stop, x, from the current velocity, v, must be approximated by:

$$x = -\frac{v^2}{2 \cdot a_m} \tag{5}$$

This follows from:

$$x = \iint a(t) \cdot dt \tag{6}$$

For these calculations, the acceleration rate, a(t), is assumed to be constant at a_m to avoid complicated double integrals of the piecewise acceleration profile. Thus:

$$x = \frac{1}{2} \cdot a_m \cdot t^2 + v \cdot t \tag{7}$$

The time to reach this distance, t, can be calculated by:

$$v = \int a(t) \cdot dt \tag{8}$$

$$t = \frac{-v}{a_m} \tag{9}$$

Substituting equation (9) into (7) results in the stopping distance, x, shown in (5). When the next stop location equals this stopping distance, a stop is requested.

Now consider the case where two stopping points are close together. It is possible that the cruising speed is never reached before a deceleration is required to stop the car. Since the model constantly calculates the distance required to stop the car from its current velocity, the above procedure still holds true. In the case that the car must decelerate for a stop, the acceleration profile is interrupted, and the deceleration profile is begun.

Each type of acceleration/deceleration has its own maximum value and time to reach that value, τ. In this model, all types of acceleration require τ=3 seconds to reach their respective maximum value. For normal acceleration, the maximum is 4.5 mph/sec; for braking to a stop, -6 mph/sec; and for gliding to a lower speed, -1 mph/sec. When braking to a stop, the car is assumed to be in neutral and all power for the stop comes from the brakes, thus, there is no load on the engine.

To make the velocity profile more realistic, a random number generator was used to vary the target speed that the car aims for. Every 1/5 mile the goal speed is reset within 15% of the speed limit. This way, a wider variety of accelerations are seen at different velocities. For example, if the given speed limit is 65 mph, the model will vary the car's target speed between 55.25 and 74.75 mph. So, at one point, it may be asked to reach 57 mph, and 1/5 mile later it may be aiming for 70 mph, and 1/5 mile later, 65mph.

Now, with the acceleration known, current velocity and distance can be determined from basic physics. The time increment used by the model can be dt = 0.25, 0.5, or 1 second. The '0' subscript indicates the value at the prior time increment.

$$v = a \cdot dt + v_0 \tag{10}$$

$$x = \frac{1}{2} a \cdot dt^2 + v \cdot dt + x_0 \tag{11}$$

If the car is stopped, it is told to wait for 60 seconds at each light.

3.2.1.5 Calculating the Engine Operating Conditions

The engine speed, rpm, brake power, P_b (W), and torque, T (N-m), can be calculated from the vehicle information and the road load requirements. The symbols are defined as follows: G.R. is the overall gear ratio, d is the wheel diameter (m), α is the road angle (degrees), M is the vehicle mass (kg), g is gravity (m/s^2), ρ_a is the air density (kg/m^3), a is acceleration (mph/sec), v is vehicle speed (mph), d is the tire diameter (m), A_v is the frontal area (m^2), and C_d and C_r are the aerodynamic and road drag coefficients. Note that α can be positive or negative depending on whether the road grade is uphill or downhill. The relationships between these parameters are as follows. (Terms in the equations are in the dimensions defined above.)

$$rpm = \frac{v \cdot G.R. \cdot 26.82}{d \cdot \pi} \tag{12}$$

$$P_b = (F_r + F_d + F_a + F_c) \cdot v \cdot 0.44704 \tag{13}$$

$$T = \frac{P_b \cdot 30}{rpm \cdot \pi} \tag{14}$$

The different F's are the forces (N) required to overcome road drag, aerodynamic drag, acceleration, and climbing an incline, respectively:

$$\begin{aligned} F_r &= C_r \cdot M \cdot g \cdot \cos(\alpha) \\ F_d &= 0.5 \cdot \rho_a \cdot v^2 \cdot C_d \cdot A_v \cdot 1609.3^2 \\ F_a &= M \cdot a \cdot 0.44704 \\ F_c &= M \cdot g \cdot \sin(\alpha) \end{aligned} \tag{15}$$

The engine speed is set to idle if the operating point is calculated to be below idle.

As mentioned before, if the model's torque exceeds the vehicle's maximum torque, then the motion data must be recalculated based on the maximum torque. In the following, the equations are written to be dimensionally consistent instead of in the dimension specific form of equations 12-15. First, find the brake power that can be produced by rearranging equation 14:

$$P_b = \frac{T \cdot rpm \cdot \pi}{30}$$

The acceleration is defined by manipulating the relation F_a=ma and equation 13 into:

$$a = \frac{F_a}{M}$$

$$a = \frac{\left(\frac{P_b}{v} - F_r - F_d - F_c\right)}{M}$$

Now, the new velocity of the vehicle and the total distance traveled must be redefined as:

$$v = a \cdot dt + v_0$$

$$x = \frac{1}{2} \cdot a \cdot dt^2 + v \cdot dt + x_0$$

3.2.1.6 Determining Emission Levels

When the engine operating conditions (speed, torque) are known, the quasi-static engine-out emissions at that instant can be determined from the corresponding map. The current torque and rpm are used to do a two-dimensional interpolation on the engine maps to find specific fuel consumption, sfc (g/kW-hr), and emission index numbers, EI (grams of emission per 100 grams fuel). This index can be converted into an emission rate (grams/sec) by using the brake power (kW) and sfc (g/kW-hr).

$$\dot{m}_e = EI \cdot sfc \cdot P_b \cdot \frac{100}{3600} \quad (16)$$

These values are determined second-by-second; all information is written to a file.

3.2.1.7 Gear Shift

The first time step begins in first gear. For the next time step the next gear requested (if any) must be determined. For the current torque, the speeds for the upshift and downshift boundaries are interpolated from the gear-shift map (Figure 3.3). If the current speed lies above the upshift boundary speed, the gear is increased. A downshift will occur if the engine speed falls below the downshift boundary. Whether further shifting is required will be determined at the end of the next time step. Cruising may require upshifts or downshifts since torque varies with the road grades.

3.2.1.8 Repeat

After determining the next gear, the above steps are repeated, starting with the next acceleration rate from the profile and the characteristics of the road at that location. If the distance reached by the recently calculated time step exceeds the next distance value, then the distance index will increase and the next distance step will be used. This continues until the calculated distance reaches the distance given by the last data point of the road profile.

3.2.2 Using a Prescribed Velocity Profile

When the velocity profile is already defined (such as with the FTP cycle), the procedure is simplified. The input files are the same as described above, except the road profile is replaced by a velocity profile, with each data point containing the time, duration, dt, and velocity (for the FTP, when there is a change in the velocity, dt=1 second). Acceleration and distance can easily be calculated:

$$a = \frac{(v - v_0)}{dt} \qquad (17)$$

$$x = \frac{1}{2} \cdot a \cdot dt^2 + v \cdot dt + x_0 \qquad (18)$$

From this, power, rpm and torque are calculated as shown in equations 12-15. No provisions are made if the vehicle's maximum torque exceeds the torque required to follow the velocity profile. The assumption with the FTP, though, is that all vehicles can reach the low accelerations requested by the test. Emissions levels and gear selections also follow the procedures described above. Note that in this case no elevation changes occur, so power calculations have zero angle and thus no climbing force.

Chapter 4 - Results and Discussion

In order to verify that the model properly calculates engine conditions, shifts at reasonable times and correctly reads the emission maps, the velocity profile from Bag 1 of the FTP was followed (Figure 4.1). The model was also tested with a road profile following a route from Central Square in Cambridge, MA onto Storrow Drive, then west on Route 2 until the interchange with Route 128. This required more checks to verify that it was properly locating stops, following the acceleration profile, etc.

Both procedures produced series of data which contain time, duration, acceleration, velocity, distance, road angle (in the first case, zero), gear, torque, rpm, power, emissions and fuel mass flowrate and cumulative masses. This data was loaded into MATLAB and several different graphs were created to analyze the output.

4.1 Following the Velocity Profile in the Federal Test Procedure (FTP)

The Bag 1 FTP cycles are shown in Figure 4.1. Each sequence of idle, acceleration, cruise and deceleration is referred to as one cycle; thus there are five cycles shown in the figure. The first cycle models suburban street driving. The second cycle models a cruise at highway speeds for several minutes. A level road is assumed.

The corresponding engine power, speeds, and torques required for the test, as calculated by the model, are shown in Figures 4.2-4.4. The gear shift schedule described in section 3.2.1.3 is used. Note that the maximum required brake torque is 120 N-m (Figure 4.3), which is lower than the maximum torque output of the engine (176 N-m). These graphs also highlight the limited range of operation in the FTP: the power required never exceeds 30 kW (Figure 4.2) and the engine speed barely tops 3000 rpm a few times (Figure 4.4). Negative brake power and torque indicate the use of brakes to decelerate: no load is transmitted to the engine.

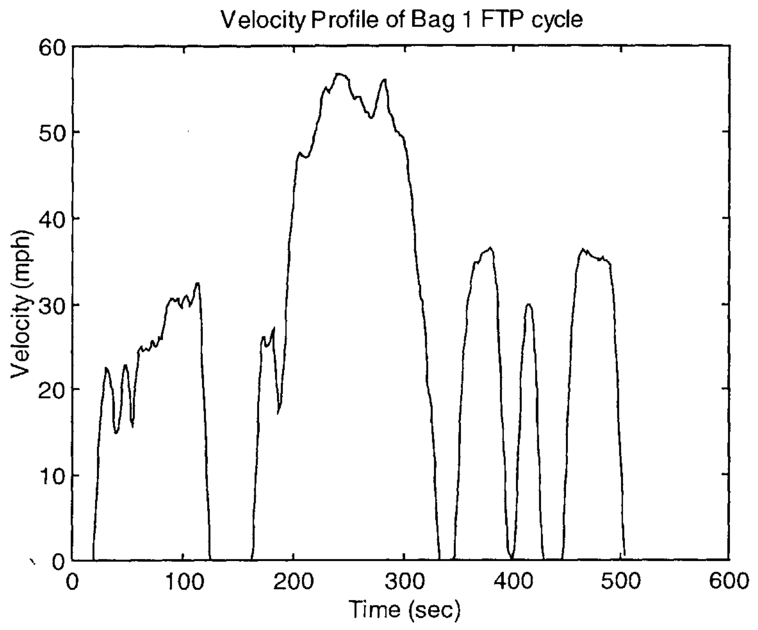


Figure 4.1: The velocity profile followed by Bag 1 of the FTP.

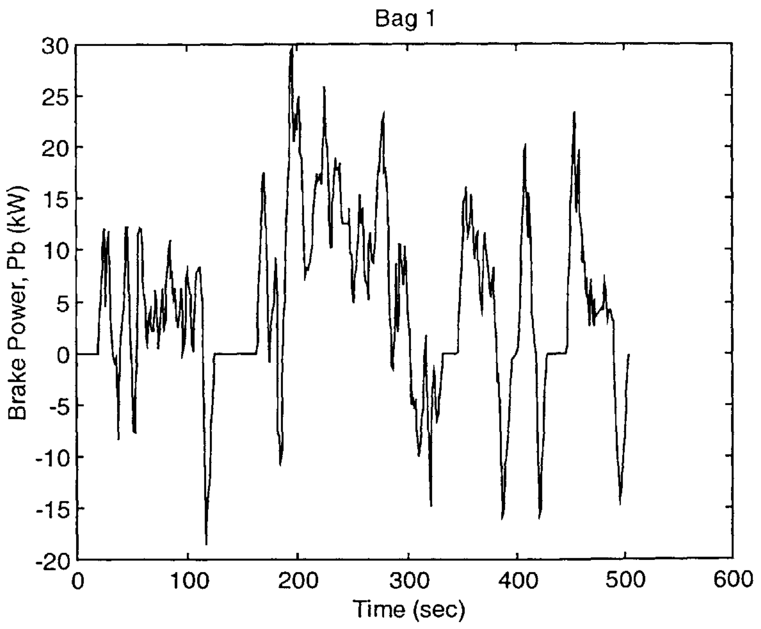


Figure 4.2: Brake power against time for Bag 1.

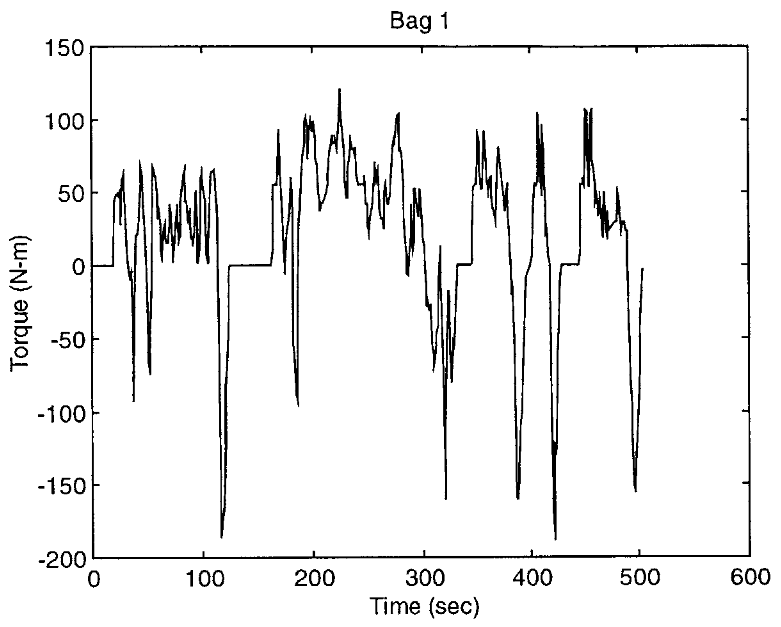


Figure 4.3: Engine torque against time for Bag1.

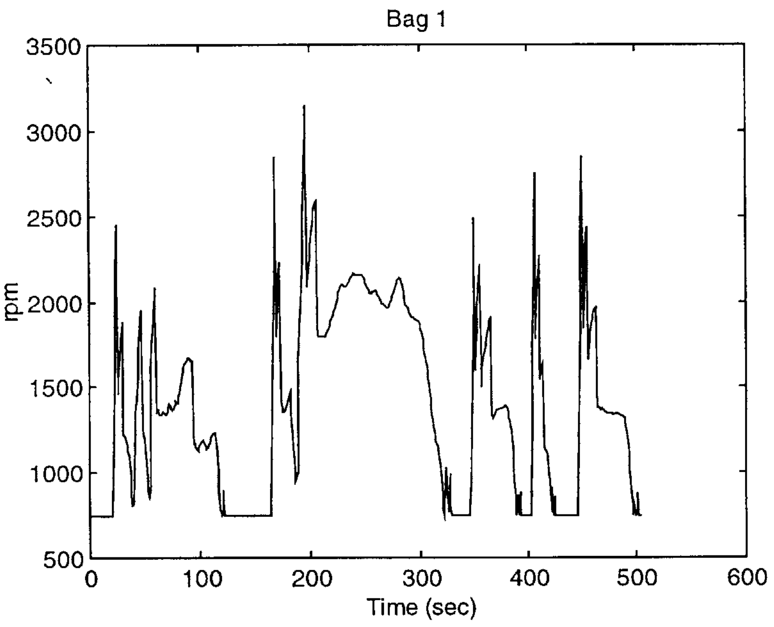


Figure 4.4: Engine speed against time for Bag1.

Figure 4.5 illustrates the vehicle's motion and gear selections for the first 300 seconds. The first (suburban) cycle has an acceleration with a couple minor decelerations, where downshifts occur. This is followed by the second (highway) cycle where higher speeds are reached. Downshifts are requested to meet the power requirements of accelerating after cruising.

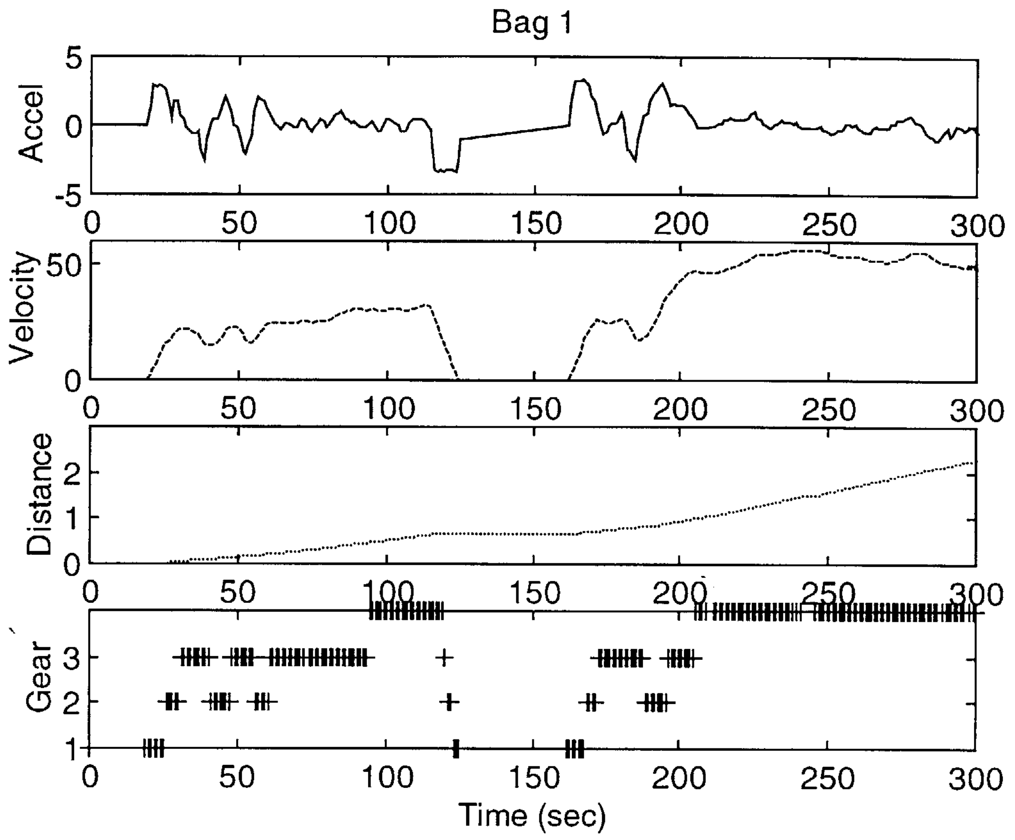


Figure 4.5: The first 300 seconds of Bag 1. This illustrates the basic physics of the vehicle motion (acceleration, velocity, distance traveled as well as gear requested).

After determining that the time data was reasonable (engine speed and torque within limits, acceptable shift points, etc.), the engine-out emissions were calculated according to the quasi-static emission maps. Figure 4.6 represents the cumulative emissions as time progresses. The slopes of each curve indicate the general overall flowrates of each species. Level spots correspond to no load imposed on the engine and thus essentially no emissions. In general, larger slopes mark heavy accelerations.

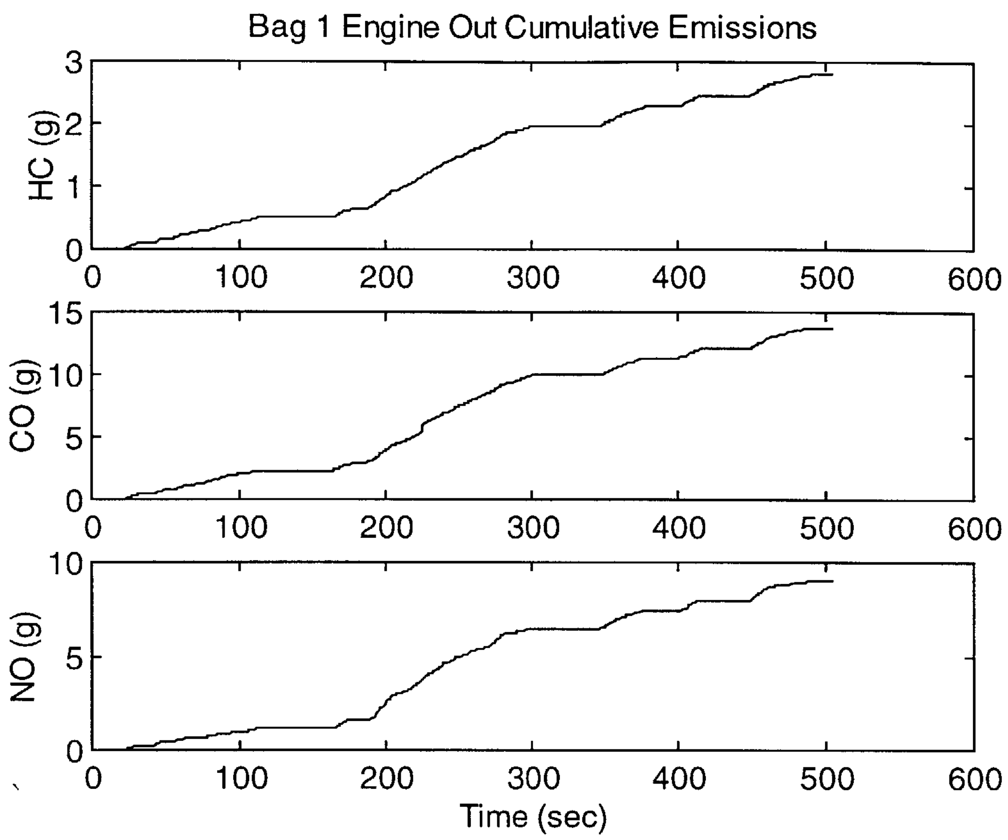


Figure 4.6: Cumulative emissions from Bag 1.

Trends were sought for the emissions data by graphing the emitted species flow rates against engine parameters like engine speed, torque, and power. As can be seen in Figures 4.7-4.10, the predominant relationship found is between the emission flow rates and the brake power.

The calculated mass flowrate of HC against brake power (Figure 4.7) shows a good correlation between the two variables. At lower power, the ratio of HC flowrate to power is greater than it is at higher powers (this can be seen as the quick rise and then steady incline in the graph). A similar effect is seen with the fuel flowrate in Figure 4.10. The relationship may be established as follows. If HC and f are the hydrocarbon and fuel flowrates, then:

$$\begin{aligned} HC &= 100 \cdot EIHC \cdot f \\ &= 100 \cdot EIHC \cdot P_b \cdot sfc \end{aligned} \tag{19}$$

Since EIHC is almost flat over the map (Figure 2.3), HC is proportional to P_b with a proportional "constant" equal to the sfc. At high power, sfc goes down (more efficient engine), thus the observed change in slope in Figure 4.7.

The CO emission rates are shown in Figure 4.8; the emission increases with the engine power. Because the operating points of the FTP are within the 100 N-m, 3000 rpm boundary of the engine map, where the EICO is essentially flat, the CO emission rate versus power behavior reflects mainly the change in sfc. This effect is similar to the HC behavior as discussed in the last paragraph. The data point scatter around this linear relationship (below 0.1 g/s CO) in Figure 4.8 is due to interpolation noise. Above this value there are five points, at ~0.1, 0.15, 0.27 and 0.47 g/s, where operating conditions reach outside the 100 N-m , 3000 rpm boundary and into the enriched zones. This enrichment results in high sCO values and thus, elevated CO flowrates.

In Figure 4.9, the flowrate of NO also increases with power; there is, however, no decrease in the slope of the relationship at the higher power range. This behavior is due to the fact that the EINO is higher at elevated power, thus offsetting the lower sfc there. The overall effect is that the NO flowrate increases almost linearly with brake power.

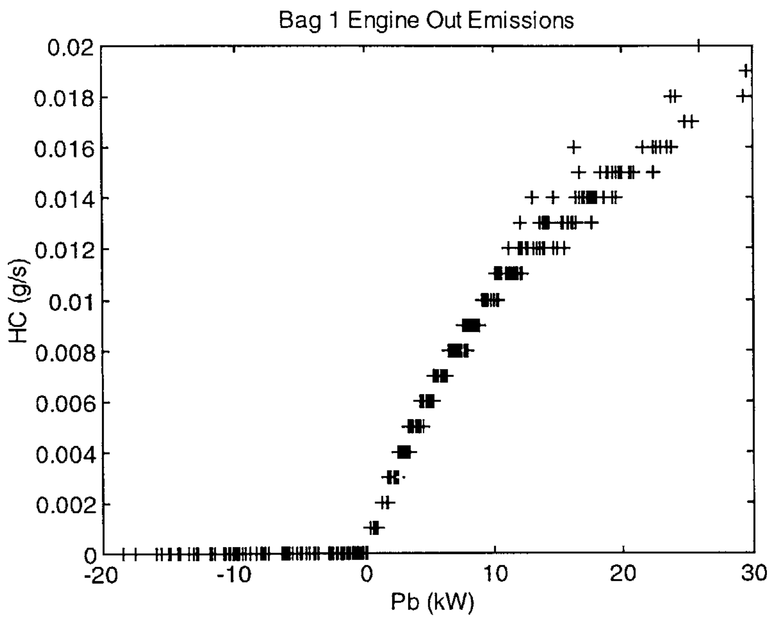


Figure 4.7: Flowrate of HC emissions against brake power for Bag 1.

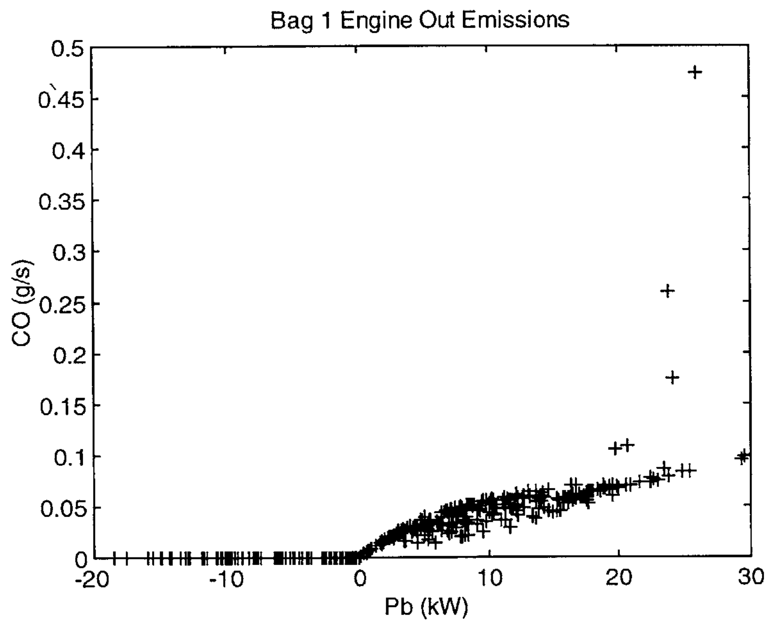


Figure 4.8: Flowrate of CO emissions against brake power for Bag 1.

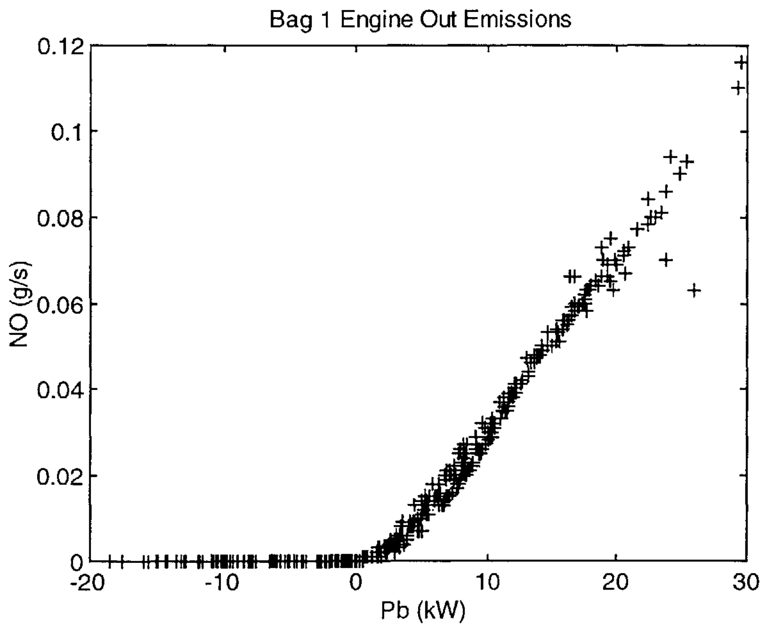


Figure 4.9: Flowrate of NO emissions against brake power for Bag 1.

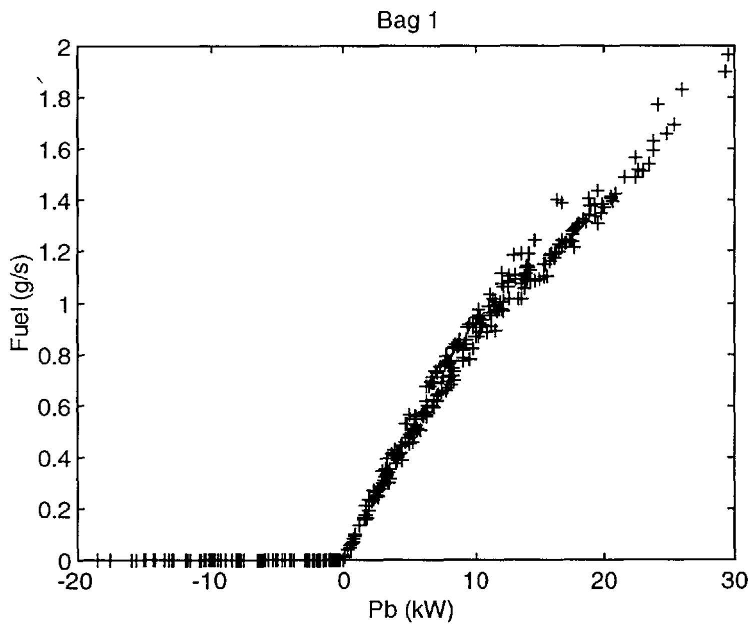


Figure 4.10: Fuel flowrate against brake power for Bag 1.

4.2 Following the Road Profile

The route used is as follows (See Figures 3.1 and 3.2 for the map and elevation profile from TOPO!). Starting in Central Square, the chosen road profile (see Figures 4.11-4.13 for the elevation and velocity profiles graphed in MATLAB) follows Western Avenue, over the Charles River and onto Storrow Drive (around 300 seconds or 0.5 miles). Up to this point, it is city driving through four stoplights with a speed limit of 35 mph. Storrow Drive has a limit of 50 mph until it meets up with the Fresh Pond Parkway. At ~900 seconds or 3.5 miles, the Fresh Pond Parkway slows down to 35 mph until the route follows Route 2 past Alewife T station. This first section has a fairly constant elevation: no drastic changes occur. But after the final stop light at Alewife (~1100 seconds or 5 miles), the main highway begins. The speed limit increases to 55 mph and at ~1200 seconds, a large hill is encountered in the Belmont area. Just over the top, there is a shorter descent, followed by smaller hills that neither gain nor lose major elevation. As seen in Figure 4.11, the vehicle speed varies around the given speed limit. This produces a wider range of accelerations for all velocities which is more realistic to driving than holding a constant speed.

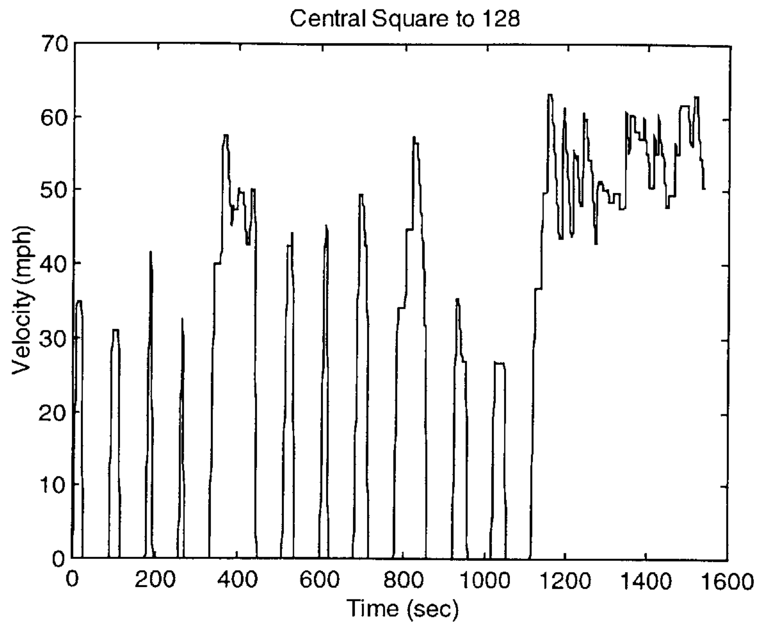


Figure 4.11: Velocity profile requested by the model for travel from Central Square in Cambridge to the Route 128 interchange on Route 2.

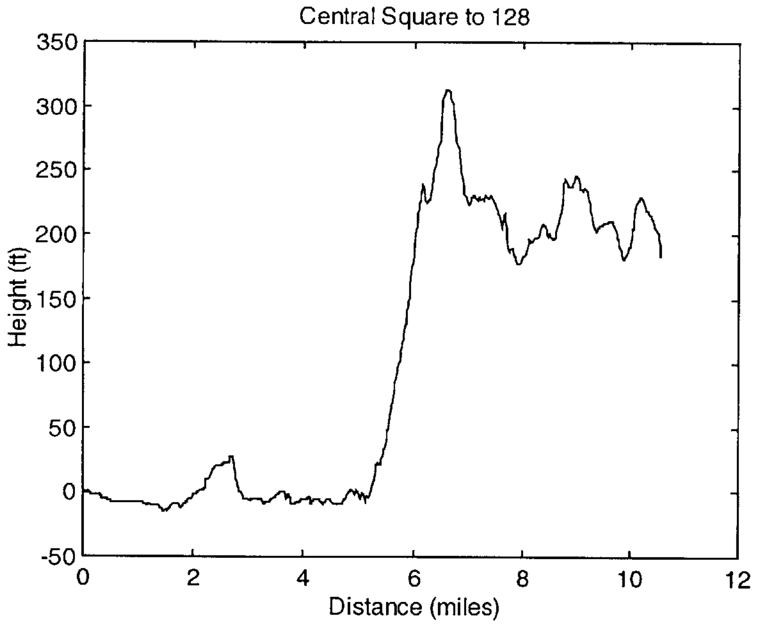


Figure 4.12: Elevation Profile against the distance traveled along the route from Cambridge, MA to Route 128

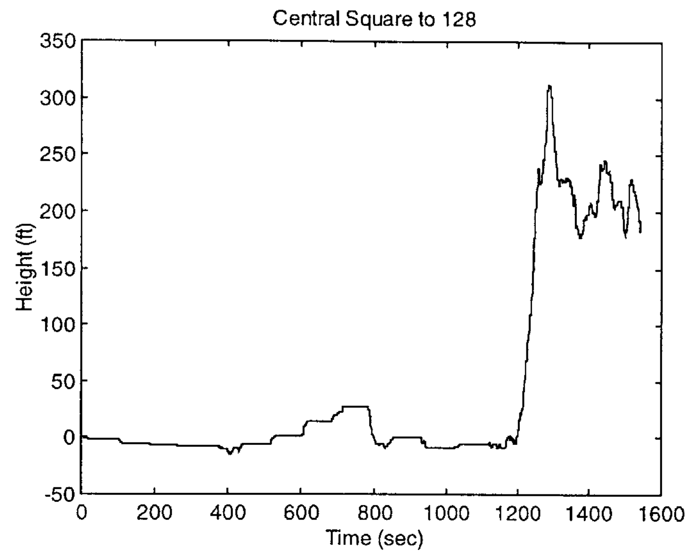


Figure 4.13: Elevation plotted against the time from Cambridge to Route128.

Most of the graphs of emission mass flowrates versus brake power resemble those for the FTP Bag 1 profile below ~20kW. Above this, though, variations are seen since the vehicle operating points in this case can exceed 60 kW (Figure 4.14), which is about double the maximum power in Bag 1, and can exceed speeds of 4000 rpm (Figure 4.16). The torque required (Figure 4.15) often reaches the car's maximum value of 176 N-m.

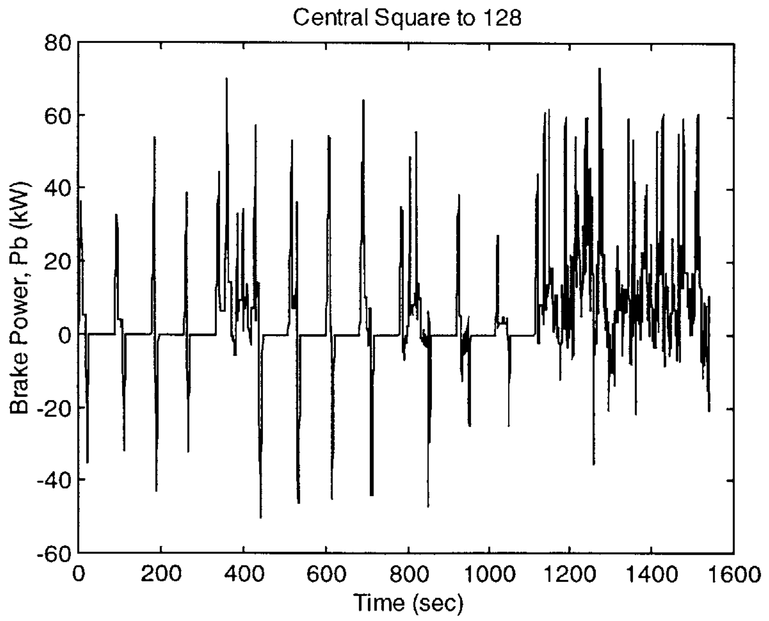


Figure 4.14: Brake power requested to follow the route from Cambridge to Route 128.

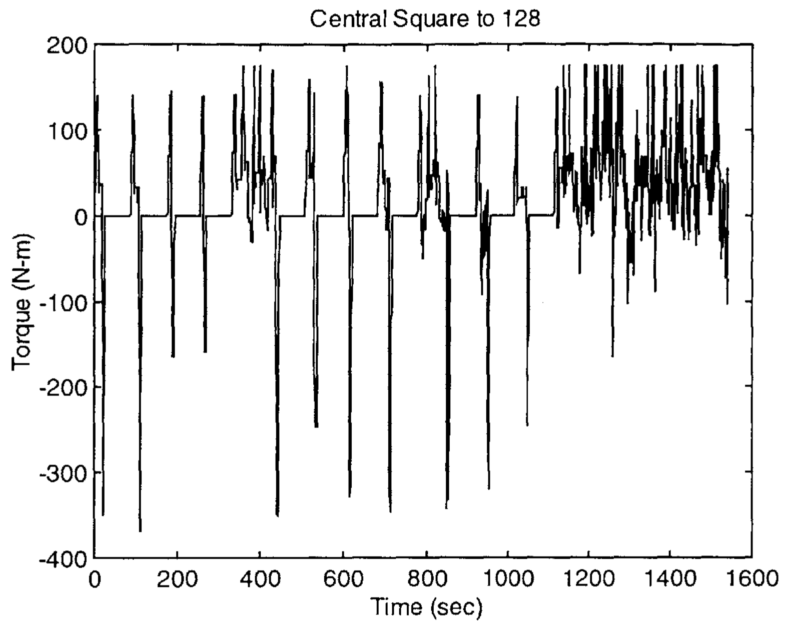


Figure 4.15: Engine torque requested to follow the route from Cambridge to Route 128.

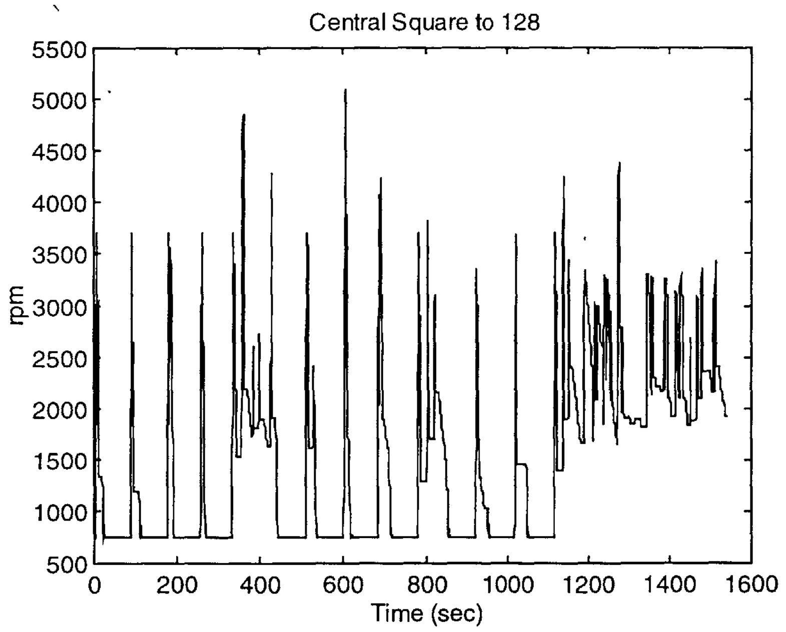


Figure 4.16: Engine speed (rpm) requested to follow the route from Cambridge to Route 128.

Figure 4.17 shows the vehicle behavior for five accelerations with four decelerations in between. The first two accelerations reach constant speeds that require fourth gear. In the third and fourth acceleration, the car reaches a higher speed, but immediately decelerates for a stop. Because of the acceleration, the torque required is much higher than it would be for cruising. Thus, the engine operating conditions never cross the upshift boundary in Figure 3.3 and a gear past second is never requested. The fifth acceleration brings the vehicle to a cruising speed but, just past 350 seconds, an acceleration requires two downshifts to provide enough power.

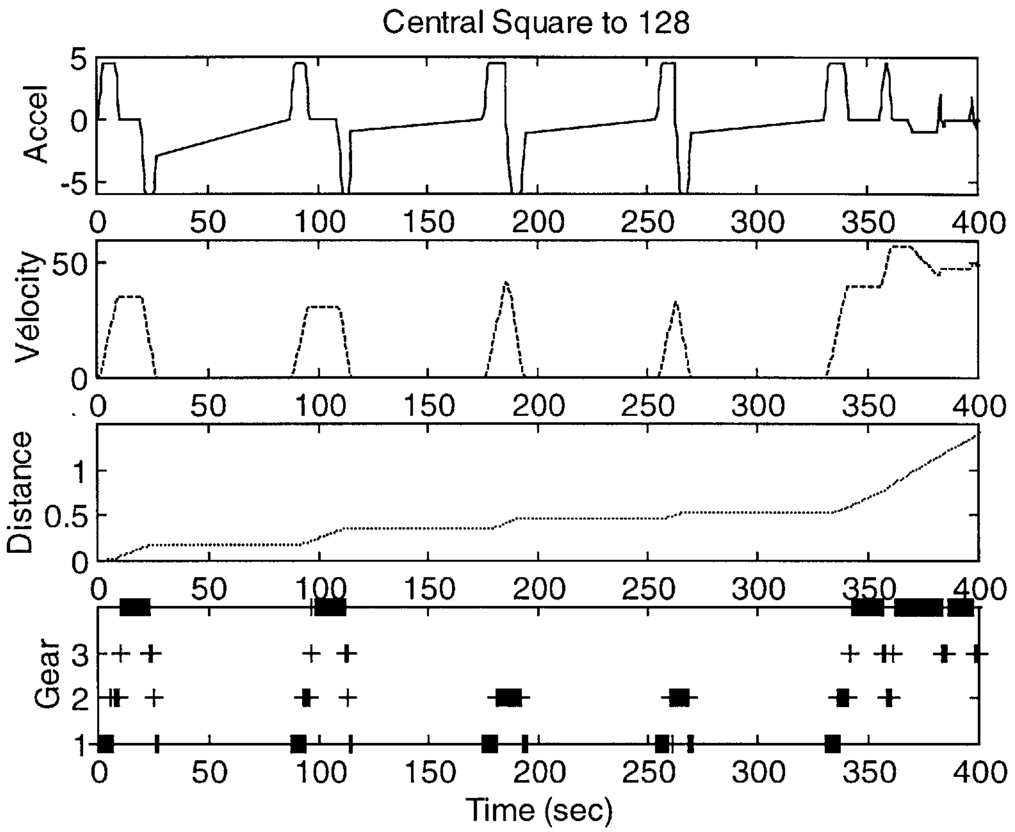


Figure 4.17: The first 400 seconds of the route from Cambridge to Route 128. This illustrates the basic physics of the vehicle motion (acceleration, velocity, distance traveled as well as gear requested).

Figure 4.18 illustrates the cumulative emissions. At 1200 seconds, when the vehicle is climbing the hill in Belmont, a large slope is seen in the graphs. After the top is reached, the slope decreases (~1300 seconds), indicating a lower emission flowrate due to the decrease in power requirements. The large jumps seen in the CO graph correlate to very high flowrates due to enrichment for high torques and rpms. The slopes of the NO graph are more subdued because enrichment at higher torques and rpm reduces the NO flowrate.

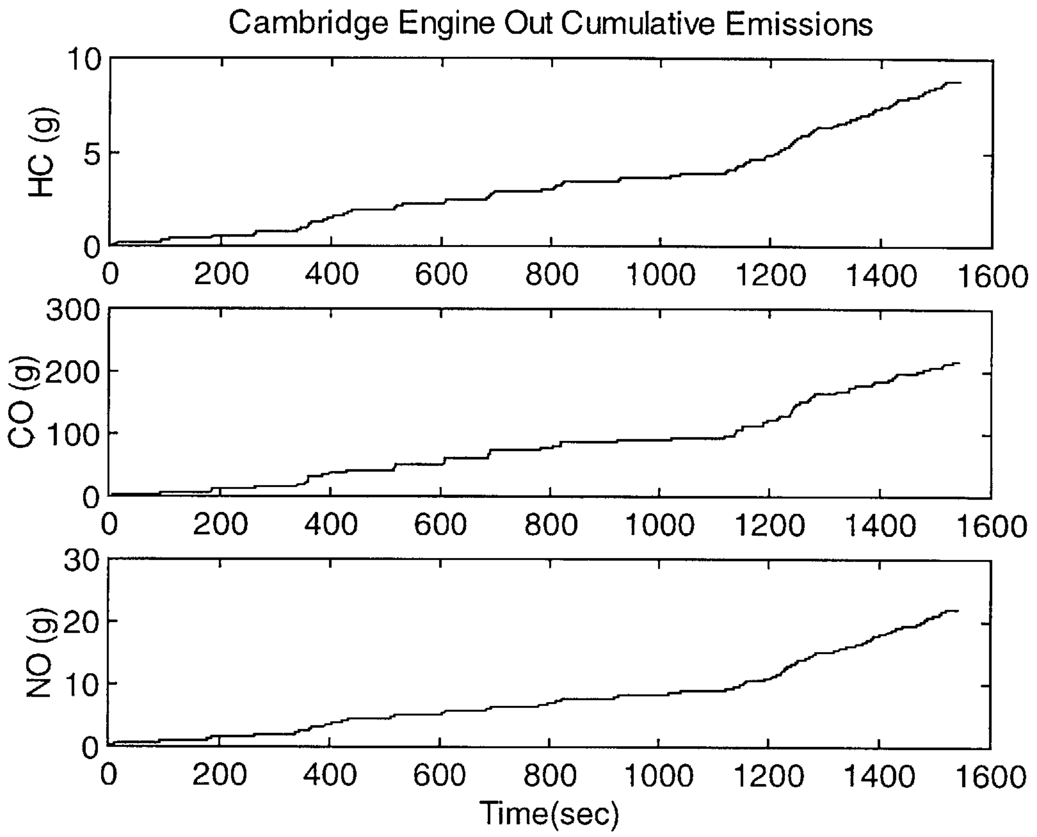


Figure 4.18: Cumulative emissions for the route from Cambridge to Route 128.

Now consider the plots of emission flowrates versus brake power in Figures 4.19-4.22. Major deviations from the corresponding graphs from the FTP data occur. The variations from a linear relationship for CO are seen in Figure 4.19. As power exceeds 30 kW, enrichment that occurs at high torque and rpm causes the CO emissions to reach up to 4 g/s. The fairly linear relationship that remains below 0.2 g/s CO are for operating conditions that remain within the 100 N-m and 3000 rpm boundary (refer to Figure 2.1d).

The “wild” data points above the linear relationship are the result of engine conditions outside this boundary. Otherwise, under 20 kW, the graph is a good duplicate of the Bag 1 graph (Figure 4.8).

A similar deviation from the linear can also be seen with the HC emissions (Figure 4.20). But in this case, the values of sHC don’t have as wide a range as sCO. This limits the spread of the HC flowrate as power increases. Below 30 kW, though, a concave down relationship, similar to the Bag 1 graph (Figure 4.7), can still be seen.

Figure 4.21 shows that for stoichiometric mixtures, the relationship for NO flowrate resembles the Bag 1 graph, Figure 4.9. The points that deviate below this line are the result of lower sNO values at torques above 110 N-m as predicted by the sNO map in Figure 2.4b. In this region, high enrichment occurs, greatly reducing the NO emissions. This occurs particularly when accelerating up the large hill in Belmont.

Figure 4.22 closely resembles the corresponding graph for fuel flowrate versus brake power for Bag 1 (Figure 4.10). The only difference being that the power surpasses 30 kW and reaches 70 kW while still maintaining the linearity of the flowrate.

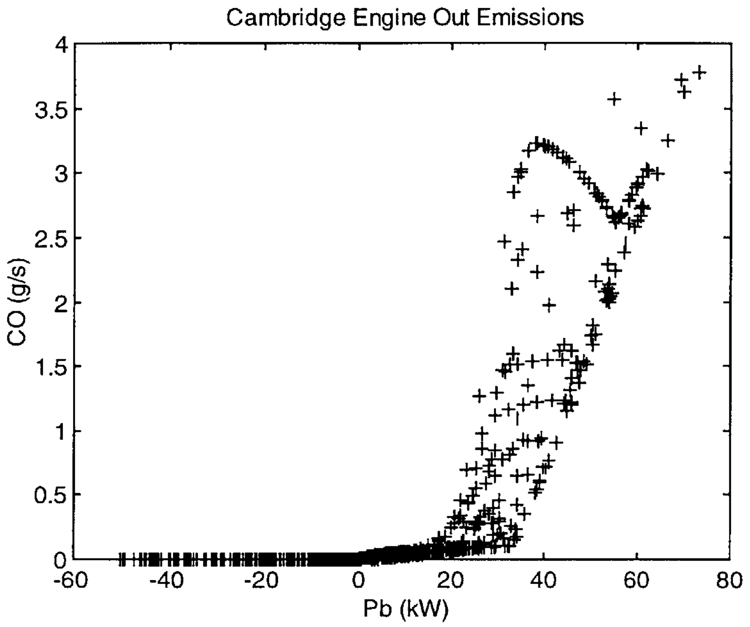


Figure 4.19: Flowrate of CO emissions against brake power for the Cambridge route.

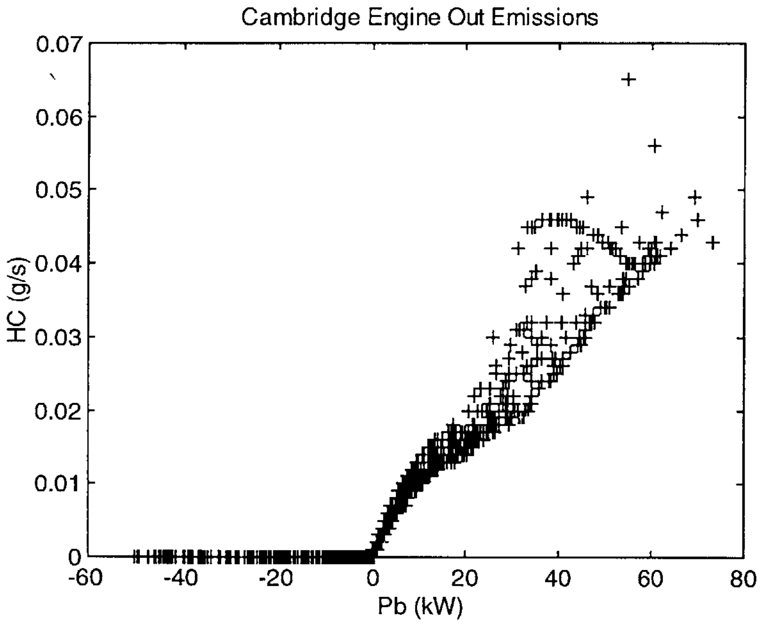


Figure 4.20: Flowrate of HC emissions against brake power for the Cambridge route.

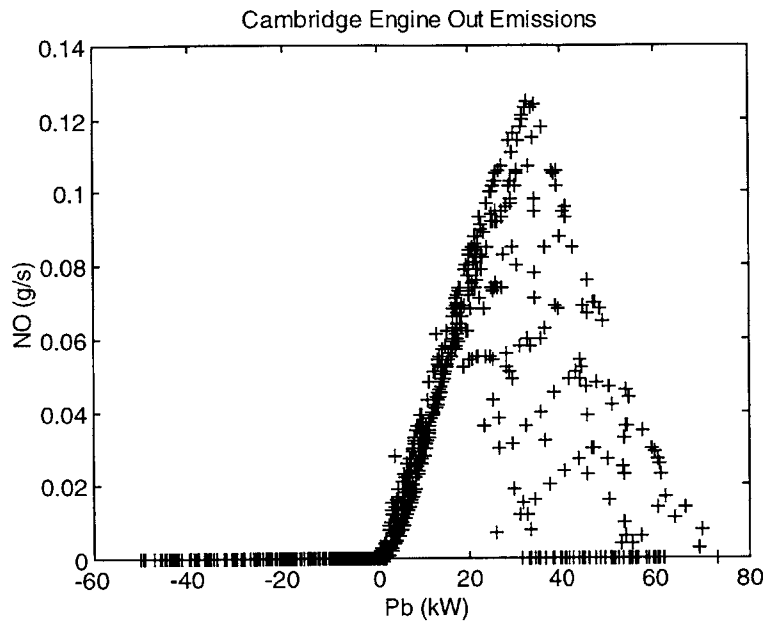


Figure 4.21: Flowrate of NO emissions against brake power for the Cambridge route.

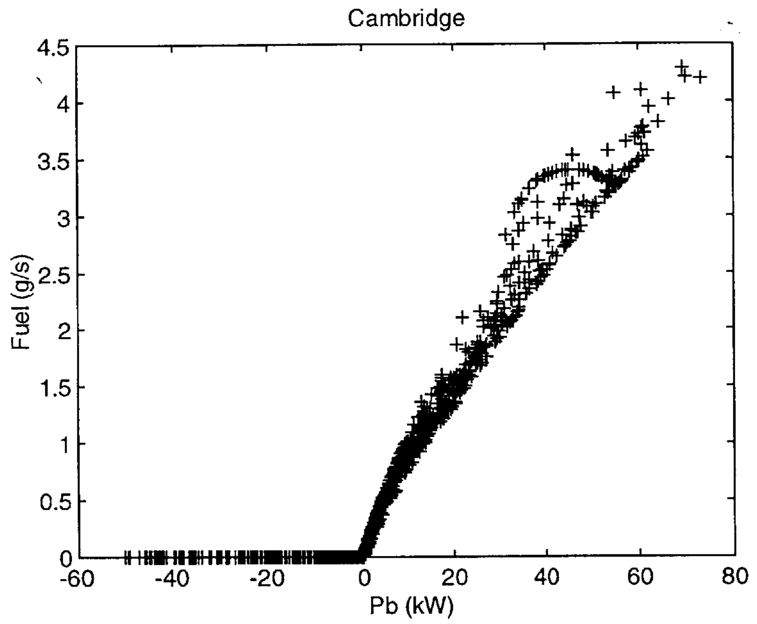


Figure 4.22: Fuel flowrate against brake power for the Cambridge route.

4.3 Conclusions

In this stage of the project, the main goal was to develop a model to create the time sequence of a vehicle's operating conditions and corresponding emissions from a given road profile. The models used to select gears and calculate the engine conditions follow an acceptable profile. Shifting points are reasonable and engine conditions are within expected operating conditions. By using quasi-steady engine-out emission maps, the basic function of the emission model was verified. The basic relationship of increasing emissions mass flowrates with power was followed for the majority of the species (as well as for the fuel mass flowrate). Also, the relationships between the CO and NO flowrates and enrichment (due to the engine control demands at high torque and rpm) were followed.

The next step is to determine how this quasi-steady emission profile is related to an actual emission profile for the same route. By identifying operating conditions where the emissions differ, the model can be modified to properly reflect real-world emission characteristics. Possibly there is a relationship between acceleration and flowrates or maybe the engine speed causes drastically different emission levels when under transient conditions. These questions are left to be answered in phase 2.

Appendix

Appendix A: Ford Contour Specifications

C_r	.015
C_d	.4
Mass	1258 kg
Frontal Area	2.4 m^2
Wheel Diameter	.6146 m
Displacement	.002 m^3
Maximum Torque	176 N-m
Idle	750 rpm
Maximum rpm	6000 rpm
Number of Gears	4
Final drive ratio	3.92
First G.R.	2.89
Second G.R.	1.57
Third G.R.	1.00
Forth G.R.	.7

Appendix B: Data File Formats

Car Specifications:

This is the format for the Contour information. The first line contains the drag coefficients, the mass, frontal area, wheel diameter, and engine displacement. Line 2 is the maximum brake torque, idle speed and maximum rpm. Line three contains the number of gears and the final drive ratio. The last lines are the transmission gear ratios.

```
.015 .4 1258 2.4 .6146 .002
176     750     6000
4 3.92
2.89
1.57
1.00
.7
```

Shifting schedule:

The first line contains the normalized torque values (the x-axis in Figure 3.3). The second line contains the upshift boundary points in terms of the normalized engine speed (the solid line in the figure). Accordingly, the third line is the downshift boundary (the dashed line in the figure).

```
-0.1   0      0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1
0.254  0.254  0.254  0.291  0.365  0.465  0.579  0.695  0.803  0.891  0.947  0.960
0.128  0.128  0.137  0.152  0.173  0.199  0.230  0.264  0.300  0.339  0.379  0.419
```

Pixel profile:

This is the beginning of the Cambridge to Route 128 pixel profile created from ImagePro Plus. The first line contains the number of pixel points in the file. Line 2 is the ratio of the x-axis: in this case there are 67 pixels per 1 mile. Accordingly, line 3 shows the elevation scale as 62 pixels per 50 feet. The rest of the lines contain the location of the pixels in the profile. The first number is the number of pixels from the left side of the image; the second number is the number of pixels from the top of the image (larger numbers indicate lower elevations).

```
708
1 67
50 62
44      424
45      424
46      424
47      425
48      425
49      425
50      425
```

Stop locations and speed limits data:
The first line is the number of stops along the route. The next lines within that number contain the mileage where each stop is located. The next line after these is the number of speed limit changes. Each line after that has the mileage and the new speed limit at that point. (The first mileage must be zero in order to assign the first speed limit.)

```
11
.17
.35
.45
.53
1.93
2.17
2.33
2.68
3.58
3.83
4.05
4
0 35
.57 50
3.15 35
4.13 55
```

Route profile:
This is the beginning of the route from Cambridge to Route 128, created from the pixel data and the stopping point data. The first line contains the number of data points in the file and the distance increment. The rest of the lines contain the distance traveled in miles, the elevation above the starting point in feet (a negative indicates an elevation below the starting point), a flag to indicate a stop location (a 0 indicates a stop), and the speed limit at that point in mph.

```
709   .014925373134
  .000000000000    0. 1 35
  .014925373134    0. 1 35
  .029850746269    0. 1 35
  .044776119403   -1. 1 35
  .059701492537   -1. 1 35
  .074626865672   -1. 1 35
  .089552238806   -1. 1 35
  .104477611940   -1. 1 35
  .119402985075   -1. 1 35
  .134328358209   -1. 1 35
  .149253731343   -2. 1 35
  .164179104478   -2. 1 35
  .179104477612   -2. 0 35
```

```
.194029850746   -2. 1 35
.208955223881   -2. 1 35
```

Velocity profile:
This is the beginning of Bag 1. The first line contains the number of data points and the standard time increment. Each line thereafter contains the time in seconds, the duration at that point and the speed in mph.

```
403   1
1     1     0
20    19    0
21    1     1.5
22    1     4.4
23    1     7.2
24    1     10.1
25    1     12.9
26    1     15.6
27    1     17.1
28    1     17.7
29    1     19.4
30    1     21.2
31    1     22
```

Emission Maps:
This contains the emission maps. The first line is the size of the matrix (in this case, 4x4). The second line contains the rpm values for the matrix (the x-axis) and the second line contains the torques (the y-axis). Since it is 4x4, there is a new data type every 4 lines. There are 4 sets of data: sfc, EIHC, EICO, and EINO. Speed increases across each line, and torque increases from the first line to the fourth in each set. The zeros are fillers in case the matrix is larger.

```
4
1500   2000   3000   4000   0   0   0   0   0   0
36     70     105    140    0   0   0   0   0   0
346.87 363.92 358.20 370.48 0   0   0   0   0   0
286.35 272.50 265.90 281.97 0   0   0   0   0   0
247.12 237.59 228.46 245.67 0   0   0   0   0   0
294.37 277.94 220.04 223.44 0   0   0   0   0   0
1.26   1.09   1.24   1.09   0   0   0   0   0   0
1.21   1.15   1.01   1.09   0   0   0   0   0   0
1.11   1.04   0.925  1.26   0   0   0   0   0   0
1.35   1.22   1.07   1.14   0   0   0   0   0   0
6.39   6.4    5      6.41   0   0   0   0   0   0
3.5    4.94   5.01   37.5   0   0   0   0   0   0
4.12   4.98   4.97   44.9   0   0   0   0   0   0
45.3   51.3   42.2   68.8   0   0   0   0   0   0
```

2.28	2.14	4.05	4.43	0	0	0	0	0	0
4.32	4.58	5.66	3.21	0	0	0	0	0	0
4.7	5.19	6.04	2.28	0	0	0	0	0	0
1.34	1.23	2.22	0.886	0	0	0	0	0	0

Appendix C: Fortran Code

The following pages contain the FORTRAN code for the model. The main program is presented first; the subroutines follow in alphabetical order. To change the input files used by the model, the subroutine 'openfiles' was edited.

```
      program profiler

C*********************************************************************
C*Todd Haugsjaa
C*The purpose of this program is to take data in the format of
C*distance, elevation, whether or not there is a stop light at each
C*distance increment and the speed limit and transform it into a
C*velocity profile against 1 second increments.  From this, brake
C*power, gear and engine speed and emission levels are determined.
C*
C*It assumes a sinusoidal acceleration and deceleration to calculate
C*the velocity profile and assumes that the car is capable of
C*producing the correct power (which is calculated from velocity,
C*accel,etc.).
C*
C*It is also possible to follow a prescribed velocity profile with no
C*elevation changes.
C*
C*********************************************************************

C*DATA IN RELATION TO DISTANCE TRAVELED, d
C*y=elevation (ft)
C*light=1 if no light, =0 when the car should stop
C*limit=speed limit (mph)
C*dd=the distance increment between readings (mi)
C*alpha=angle(degrees) calculated from elevation change and dd
      real d(5000),y(5000),light(5000),limit(5000),dd,alpha(5000)

C*CAR DATA
C*ratio=array of gear ratios
C*Cr=coeff of rolling friction
C*Cd=coeff of aero. drag
C*M=mass of car (kg)
C*Area=frontal area of car    (m^2)
C*diameter=tire diameter (m)
C*displacement=engine displacement volume (m^3)
C*nogears=number of gears on car
C*torquemax is the car's maximum producible torque
C*rpmmax is the car's maximum producible engine speed
C*pertorque contains an array percentage of maximum torques
C*up and dn are the engine speeds where a shift is requested for
C*   12 torques referenced from pertorque
C*idle is generally the lowest engine speed

      real ratio(5),Cr,Cd,M,Area,diameter,displacement
      integer nogears
      real torquemax,pertorque(12),
      &      idle,rpmmax,up(12),dn(12)
C*DATA IN RELATION TO TIME, time (sec)
C*duration=the length of time spent at each value (sec)
C*a=acceleration of car (mi/hr^2)
C*v=velocity (mph)
C*x=distance traveled (mi)
C*Pb=brake power (W)
C*rpm=engine speed      (rpm)
C*bmep=brake mean effective pressure (Pa)
C*torque=brake torque (N-m)
C*gear=what gear the car is in
      real time,duration,a,v,x,Pb,
      &      rpm,bmep,torque
      integer gear
```

```
C*DATA FOR READING VELOCITY PROFILE's (BAGs)
      real time2(5000),duration2(5000),v2(5000)

C*EMMISIONS
C*mrpm=map rpm (in percentage of total)
C*mload=map torque (in percentage of total)
C*grid=grid size of emission map (4 means 4X4)
C*sfc=map spec fuel cons (g/kWhr)
C*sfhc=map specific emissions(g/kWhr)
C*eihc=map (% hc/fuel)
      real mrpm(10),mload(10),sfc(10,10),
      &      sfhc(10,10),sfco(10,10),sfno(10,10),
      &      eihc(10,10),eico(10,10),eino(10,10)
      integer grid

C*INCREMENTING,etc.
C*n=number of distance data points
C*t=number of time data points for velocity profile (bag)
C*files=1 for default files;=0 for user entered
C*dt=increment between time data (.25 .5 or 1 sec)
C*velprof=y for reading velocity profile (bag)=n for distance profile
      integer n,t
      real dt
      character (1) velprof
C**********************************************************
      call openfiles(velprof,dt)
      call readfile(n,dd,d,y,light,limit,Cr,Cd,M,Area,
     &       diameter,displacement,nogears,ratio,
     &       torquemax,rpmmax,pertorque,up,dn,idle,
     &       mrpm,mload,velprof,time2,duration2,v2,t,dt,
     &       grid,sfc,eihc,eico,eino)

      if (velprof.eq.'n') then
      call compute(nogears,ratio,d,y,dd,limit,light,
      &            n,dt,Cr,Cd,M,Area,diameter,displacement,
      &            torquemax,rpmmax,pertorque,up,dn,idle,
      &          mrpm,mload,grid,sfc,eihc,eico,eino)
      else
      call compute2(v2,nogears,ratio,
      &            time2,duration2,dt,Cr,Cd,M,Area,diameter,displacement,
      &            torquemax,rpmmax,pertorque,up,dn,idle,t,
      &          mrpm,mload,grid,sfc,eihc,eico,eino)

      endif
      end    program
      subroutine compute(nogears,ratio,d,y,dd,limit,light,
     &           n,dt,Cr,Cd,M,Area,diameter,displacement,
     &           torquemax,rpmmax,pertorque,up,dn,idle,
     &          mrpm,mload,grid,sfc,eihc,eico,eino)

C*********************************************************************
C*This is the main computing routine - it turns a distance profile
C*into a requested velocity profile by following an acceleration
C*profile.  It then calculates the engine speed, torque and brake
C*power required to follow that.  A call to requestgear is made to
C*figure out which gear the car should be in.
C*Also, emission levels are determined through a call to a subroutine
C*********************************************************************

C*max, min and coast variables indicate peak acceleration (mi/hr^2)
C*and the time required to reach that peak (sec). For accelerating,
C*braking, and coasting to slow down, respectively.
      real amax,tmax,amin,tmin,acoast,tcoast

C*DISTANCE DATA
      real d(5000),y(5000),light(5000),limit(5000),dd,
     &      alpha(5000),height,angle
C*TIME DATA
      real time,duration,a,v,x,Pb,rpm,bmep,torque

C*CAR DATA
C*row=air density (Pa)
C*g=gravity (m/s^2)
      real ratio(5),Cr,Cd,M,Area,diameter,displacement,row,g
      real torquemax,rpmmax,pertorque(12),up(12),dn(12),idle
      integer gear,nogears

C*variables for emissions
C*mrpm and mlaod are the map values of the engine speed and load
C*sfhc, sfco, sfno values are calculated (g/kWhr)
C*hc,co,no,fuel (g/s) and cumulative (grams) numbers are calculated
C*sfc and ei values are from the maps
C*grid=the grid size of the maps. (e.g. if grid=4, then it is a 4x4
C*matrix of points
      real mrpm(10),mload(10),
     &      sfc(10,10),fuel,cummfuel,
     &      sfhc(10,10),hc,cummhc,
     &      sfco(10,10),co,cummco,
     &      sfno(10,10),no,cummno,
     &      eihc(10,10),eico(10,10),eino(10,10)
      integer grid

C*INCREMENTERS AND FLAGS
C*stopped is used to indicate if the car has stopped after a stop
C*has been requested (needed since it can take 2 or more
C*distance increments)
C*j=distance data incrementer
C*status=-2,-1,0,1 for braking,coasting,cruising,accelerating
C*int=interval: how far along the acceleration sine wave
      logical stopped
      integer j,status,n
      real dt,int

C*USED TO FIND UP COMING STOPS
C*k=data location of next stop
C*stoplocation=distance data point where car can physically stop
```

```
      integer k
      real stoplocation,k1

C*********************************************
      row=1.2
      g=9.81
      amin=   -6*3600
      tmin=   3
      amax=   4.5*3600
      tmax=   3
      acoast= -1*3600
      tcoast= 3

C*first increment is set as follows:
      time=0
      duration=dt
      x=0
      v=0
      a=0
      Pb=0
      rpm=idle
      gear=1
      bmep=0
      torque=0
      cummhc=0
      j=2
      alpha(j)=asind((y(j)-y(j-1))/dd/5280)
      angle=alpha(j)
      height=y(j)
      call emissions(mrpm,mload,a,rpm,
      &      torque,duration,Pb,rpmmax,torquemax,
      &      grid,eihc,hc,cummhc,eico,co,cummco,eino,no,cummno,
      &      sfc,fuel,cummfuel)
      call writefile(time,duration,x,v,a,angle,Pb,gear,rpm,
      &      bmep,torque,hc,cummhc,co,cummco,no,cummno,light(1),height,
      &      fuel,cummfuel)
      call nextstop(k,j,n,light)

C*this prevents a crash at the end of the route
      y(n+1)=y(n)

      int=0
      status=1
      do while (j.le.n)
      alpha(j+1)=asind((y(j+1)-y(j))/dd/5280)
      do while (x.le.d(j))
            stoplocation=(-(v**2)/(2*amin)+x)
            if ((stoplocation.ge.(k-2)*dd).and.(k.ne.n))then
C*start deceleration
                  stopped=.false.
                  status=-2
            else
                  stopped=.true.
            endif
            call find_a_v_x(stopped,status,limit,int,dt,amin,acoast,
      &           amax,tmin,tmax,tcoast,a,v,x,j,k,n,time,duration,light,
      &
      dd,angle,alpha,height,y,torquemax,Pb,Cd,Area,g,M,Cr,row,
      &           d,rpm,ratio,idle,diameter,gear,torque)
            call emissions(mrpm,mload,a,rpm,
      &           torque,duration,Pb,rpmmax,torquemax,
     &             grid,eihc,hc,cummhc,eico,co,cummco,eino,no,cummno,
     &             sfc,fuel,cummfuel)
C*This makes values more neutral for the time at a stop by inserting
C*another set of time data containing the values required to reach
C*that stop and giving the duration of the stop zero values for
C*engine conditions, etc.
            if (duration.gt.dt) then
                  time=time-duration+dt
                  call writefile(time,dt,x,v,a,angle,Pb,gear,rpm,
     &
     bmep,torque,hc,cummhc,co,cummco,no,cummno,light(j),
     &                  height,fuel,cummfuel)
                  time=time+duration
                  Pb=0
                  torque=0
                  bmep=0
                  a=0
            endif
            call writefile(time,duration,x,v,a,angle,Pb,gear,rpm,
     &            bmep,torque,hc,cummhc,co,cummco,no,cummno,light(j),
     &            height,fuel,cummfuel)
C    print*, v,a,status,stopped,light(j)
            call requestgear(torque,rpm,v,torquemax,rpmmax,
     &            pertorque,up,dn,gear,status,nogears)
     enddo
     j=j+1
     enddo
C*convert mi/gram into mpg (density =750 g/L)
     cummfuel=x/cummfuel*750*3.785
     print *,'fuel mileage for the trip: ',cummfuel
     close(8)
     return
     end
```

```
      subroutine compute2(v2,nogears,ratio,
     &            time2,duration2,dt,Cr,Cd,M,Area,diameter,displacement,
     &            torquemax,rpmmax,pertorque,up,dn,idle,t,
     &          mrpm,mload,grid,sfc,eihc,eico,eino)

C*******************************************************************
C*This turns a velocity profile into
C*acceleration, brake power, etc. A call to requestgear is made to
C*figure out which gear the car should be in.
C*******************************************************************

C*TIME DATA
      real time2(5000),duration2(5000),a,v2(5000),x,Pb,
     &      rpm,bmep,torque,angle,height

C*CAR DATA
C*row=air density (Pa)
C*g=gravity (m/s^2)
      real ratio(5),Cr,Cd,M,Area,diameter,displacement,row,g
      real torquemax,rpmmax,pertorque(12),up(12),dn(12),idle
      integer gear,nogears

C*INCREMENTERS AND FLAGS
C*i=time incrementer
C*j=distance data incrementer
C*status=-1,0,1 for braking,cruising,accelerating
      integer i,status,t
      real dt

C*VARIABLES FOR EMISSIONS (described in compute.for)
      real mrpm(10),mload(10),
     &      sfc(10,10),fuel,cummfuel,
     &      sfhc(10,10),hc,cummhc,
     &      sfco(10,10),co,cummco,
     &      sfno(10,10),no,cummno,
     &      eihc(10,10),eico(10,10),eino(10,10)
      integer grid

C***********************************************
      row=1.2
      g=9.81

C*first increment is set as follows:
      i=1
      x=0
      a=0
      Pb=0
      rpm=idle
      torque=0
      Pb=0
      bmep=0
      gear=1
      cummhc=0
      angle=0
      height=0
      status=1
C*dummy variable for velocity profile
      light=1

      call emissions(mrpm,mload,a,rpm,
     &      torque,duration2(i),Pb,rpmmax,torquemax,
     &      grid,eihc,hc,cummhc,eico,co,cummco,eino,no,cummno,
     &      sfc,fuel,cummfuel)
      call writefile(time2(i),duration2(i),x,v2(i),a,angle,Pb,
     &      gear,rpm,bmep,torque,hc,cummhc,co,cummco,no,cummno,light,
     &      height,fuel,cummfuel)

      do while (i.lt.t)
      i=i+1
C*present conditions(i) are a result of past conditions (i-1)
      a=(v2(i)-v2(i-1))/dt*3600
      x=.5*a*(dt/3600)**2+(v2(i)*dt/3600)+x
      if (a.lt.0) then
            status=-1
      elseif (a.gt.0) then
            status=1
      else
            status=0
      endif
      rpm=v2(i)*ratio(gear)/3.14159/diameter*26.8224
      if (rpm.lt.idle) then
            rpm=idle
      endif
      Pb=(Cr*M*g*cosd(angle)+
     &      .5*row*(v2(i)*.44704)**2*Cd*Area+
     &      M*a*1.2418e-4+M*g*sind(angle))*v2(i)*.44704

      torque=Pb/rpm/(3.14159/30)
      call emissions(mrpm,mload,a,rpm,
     &      torque,duration2(i),Pb,rpmmax,torquemax,
     &      grid,eihc,hc,cummhc,eico,co,cummco,eino,no,cummno,
     &      sfc,fuel,cummfuel)
      call writefile(time2(i),duration2(i),x,v2(i),a,angle,Pb,
     &      gear,rpm,bmep,torque,hc,cummhc,co,cummco,no,cummno,light,
     &      height,fuel,cummfuel)
      call requestgear(torque,rpm,v2(i),torquemax,rpmmax,
     &      pertorque,up,dn,gear,status,nogears)
      enddo
      cummfuel=x/cummfuel*750*3.785
      print *,'fuel mileage for the trip: ',cummfuel
      close(8)
      return
      end
      subroutine emissions(mrpm,mload,a,rpm,
     &      torque,duration,Pb,rpmmax,torquemax,
     &      grid,eihc,hc,cummhc,eico,co,cummco,eino,no,cummno,
     &      sfc,fuel,cummfuel)

C*******************************************************************
C*   This determines the emission level for each dt in
C*   grams/sec based on the engine speed and load.
C*******************************************************************

C*data against time
      real a,rpm,torque,duration,Pb

C*incrementers and constants
      integer j,k

C*engine vales
C*max indicates the engines upper limits
C*ratio indicates the current ratio of rpm or toruqe to its max value
      real rpmmax,torquemax,rpmratio,torqueratio

C*variables for emissions     (described in compute.for)
      real mrpm(10),mload(10),
     &      sfc(10,10),fuel,cummfuel,
     &      eihc(10,10),hc,cummhc,
     &      eico(10,10),co,cummco,
     &      eino(10,10),no,cummno
      integer grid
C***********************************************************
      rpmratio=rpm/rpmmax
      torqueratio=torque/torquemax

      j=2
      do while ((rpmratio.gt.mrpm(j)).and.(j.lt.grid))
      j=j+1
      enddo
      k=2
      do while ((torqueratio.gt.mload(k)).and.(k.lt.grid))
      k=k+1
      enddo

      call interpolemiss(rpmratio,mrpm,torqueratio,mload,
     &      eihc,j,k,hc)
      call interpolemiss(rpmratio,mrpm,torqueratio,mload,
     &      eico,j,k,co)
      call interpolemiss(rpmratio,mrpm,torqueratio,mload,
     &      eino,j,k,no)
      call interpolemiss(rpmratio,mrpm,torqueratio,mload,
     &      sfc,j,k,fuel)

      if (Pb.gt.0) then
C*creates (g/s) from the emiss index (g emm/100g fuel)
C*and sfc(g fuel/kWhr)
      hc=(hc/100)*fuel*Pb/1000/3600
      co=(co/100)*fuel*Pb/1000/3600
      no=(no/100)*fuel*Pb/1000/3600
C*creates g/s from g/kWhr
      fuel=fuel*Pb/1000/3600
      else
C*if negative power, then no emissions from engine
      hc=0
```

```
co=0
no=0
fuel=0
endif
cummhc=cummhc+hc*duration
cummco=cummco+co*duration
cummno=cummno+no*duration
cummfuel=cummfuel+fuel*duration

return
end
```

```
      subroutine find_a_v_x(stopped,status,limit,int,dt,amin,acoast,
      &
      amax,tmin,tmax,tcoast,a,v,x,j,k,n,time,duration,light,dd,
      &
      angle,alpha,height,y,torquemax,Pb,Cd,Area,g,M,Cr,row,d,
      &            rpm,ratio,idle,diameter,gear,torque)
C*********************************************************
C*This varies the goal speed requested and calculates the physics of the
C*vehicle and engine conditions.
C*********************************************************

      USE MSFLIB

      logical stopped
      integer status,j,k,n,jinc
      real limit(5000),light(5000),int,dt,time,duration,
      &      amin,acoast,amax,tmin,tmax,tcoast,
      &
      a,v,x,dd,torquemax,Pb,angle,alpha(5000),height,y(5000),Cd,Area,
      &      g,M,Cr,row,d(5000),rpm,ratio(5),idle,diameter,
      &      ranval,variation
      integer gear

C*******************************************

C*this if-then will vary the speed limit (possibly imposed by traffic)
C*The 'call seed' will vary the random sequence each time the program is
run
C     call seed (RND$TIMESEED)
C*variation is the percentage deviation allowed around the speed limit
      variation=.15
C*jinc is the number of j's between changes.
C*In this case, about every .2 miles
      jinc=nint(.2/dd)
      if (mod(j,jinc).eq.0) then
          call random(ranval)
      limit(j)=((ranval*2-1)*variation+1)*limit(j)
      else
      limit(j)=limit(j-1)
      endif

      if (.not.stopped) then
      status=-2
C*it may take more than two distance increments to stop
      else
      if (v.lt.limit(j)) then
             status=1
      else if      (v.gt.limit(j)) then
             status=-1
      else
             status=0
      endif
      endif
C*this case will make calls to find the acceleration requested
      select case (status)
      case (1)
             call rate(amax,tmax,a,v,limit(j),int,dt)
      case (0)
             a=0
      case (-1)
             call rate(acoast,tcoast,a,v,limit(j),
     &                         int,dt)
     case (-2)
            call rate(amin,tmin,a,v,0.0,int,dt)
     endselect
     v=v+a*dt/3600
     x=.5*a*(dt/3600)**2+(v*dt/3600)+x
C*If the physics take the vehicle beyond the current distance
C*increment, the next increment is used to calculate the engine
C*conditions.
     if (x.gt.d(j)) then
     angle=alpha(j+1)
     height=y(j+1)
     else
     angle=alpha(j)
     height=y(j)
     endif
     rpm=v*ratio(gear)/3.14159/diameter*26.8224
     if (rpm.lt.idle) then
     rpm=idle
     endif
     Pb=(Cr*M*g*cosd(angle)+
     &.5*row*(v*.44704)**2*Cd*Area+
     &M*a*1.2418e-4+M*g*sind(angle))*v*.44704

     torque=Pb/rpm/(3.14159/30)

C*if the torque requested exceeds the maximum torqe
C*the car can produce, the acceleration is limited
C*by the max torque, and the car may slow down
     if (torque.gt.torquemax) then
     torque=torquemax
     Pb=torque*rpm*3.14159/30
     a=(Pb/v/.44704
     &      -Cr*M*g*cosd(angle)
     &      -.5*row*(v*.44704)**2*Cd*Area
     &      -M*g*sind(angle))/M/1.2418e-4
     v=a*dt/3600+v
     x=.5*a*(dt/3600)**2+v*dt/3600+x
     endif

     if (v.le.2e-4) then
     v=0
     duration=60
     stopped=.true.
     status=1
     call nextstop(k,j,n,light)
     else
     duration=dt
     endif
     time=time+duration
     return
     end
```

```
      subroutine interpolate(i1,i2,x1,x2,i3,x3)

C*******************************************************
C*    i1    i3    12
C*    x1    x3    x2
C*The object is to find x3
C*******************************************************

      real x1,x2,x3,i1,i2,i3

C*******************************************************
      x3=x1+(i3-i1)/(i2-i1)*(x2-x1)

      return
      end
      subroutine interpolate2d(i1,i2,j1,j2,x1,x2,x3,x4,i5,j5,x5)

C*****************************************************
C*     i1     i5     i3
C*    j1     x1     x5a    x2
C*    j5            x5
C*    j2     x3     x5b    x4
C*The object is to find x5 after finding x5a and x5b
C*****************************************************

      real i1,i2,i5,j1,j2,j5,x1,x2,x3,x4,x5,x5a,x5b

C*****************************************************

      call interpolate(i1,i2,x1,x2,i5,x5a)
      call interpolate(i1,i2,x3,x4,i5,x5b)
      call interpolate(j1,j2,x5a,x5b,j5,x5)

      return
      end
      subroutine interpolemiss(rpmratio,mrpm,torqueratio,mload,
     &                                    memiss,j,k,emiss)
C*************************************************
C*This feeds the emission levels into the interpolate2d subroutine
C*************************************************
      real rpmratio,mrpm(10),
     &      torqueratio,mload(10),
     &      memiss(10,10),emiss
      integer j,k

      real x1,x2,x3,x4,x5,i1,i2,i5,j1,j2,j5
C*************************************

      x1=memiss(j-1,k-1)
      x2=memiss(j,k-1)
      x3=memiss(j-1,k)
      x4=memiss(j,k)

      i1=mrpm(j-1)
      i2=mrpm(j)
      j1=mload(k-1)
      j2=mload(k)

      i5=rpmratio
      j5=torqueratio

      call interpolate2d(i1,i2,j1,j2,x1,x2,x3,x4,i5,j5,x5)
      emiss=x5
      if (emiss.lt.0) then
      emiss=0
      endif
      return
      end
      subroutine nextstop(k,j,n,light)

      integer j,k,n,light(5000)
C***************************************
C*This subroutine looks ahead in the distance data to find the
C*next stop
C***************************************

C*can't have two consecutive stops in distance data
      k=j+2
      do while ((light(k).ne.0).and.(k.lt.n))
      k=k+1
      enddo
C     print *,'stop at ',k
      return
      end
      subroutine openfiles(velprof,dt)
C*****************************************************
C*This subroutine asks which files are to be used by the program
C*and whether it will follow a velocity profile or a road profile.
C*If it is a road profile, it can convert from pixels to distance if
C*needed.
C*****************************************************
      integer files
      real dt
      character(1) velprof,dopixels
      character (30) filename1, filename2,filename3,filename4,
      &                         filename7,filename8,filename9
C*****************************************************
      files=1
      if (files.eq.1) then
      dopixels='n'
      velprof='y'
C*time increment, dt, for road profile must be .25, .5 or 1 second
      dt=.5
C*road or velocity profile
      filename1='bag1.txt'
C*car characteristics
      filename2='contoura.txt'
C*shifting
      filename3='shifting.txt'
C*emisssions
      filename4='con_emm.txt'
C*output file
      filename8='bag1.out'
      else
      print *,'is this a velocity profile (ftp-style) [y,n]?'
      read(*,*) velprof
      print *,'what file has(will have) road profile or velocity
      &profile?'
      read(*,*) filename1
      print *,'what file has car characteristics?'
      read(*,*) filename2
      print *,'what file has shifting schedule?'
      read(*,*) filename3
      print *,'what file has emissions maps?'
      read(*,*) filename4
      print *,'output to what file?'
      read(*,*) filename8
      endif
      open (1, file=filename1)
      open (2, file=filename2)
      open (3, file=filename3)
      open (4, file=filename4)
      open (8, file=filename8)

      if (velprof.ne.'y') then
      if (dopixels.eq.'y') then
             print *,'what file has pixel map?'
             read(*,*) filename7
             open (7, file=filename7)
             print *,'what file has stop/limit data?'
             read(*,*) filename9
             open (9, file=filename9)
             call pixel
             close(7)
             close(9)
```

```
          rewind(1)
      endif
      endif

      return
      end
      subroutine pixel
C*********************************************************************
C*This reads the pixel outline and creates the road profile.
C*Saved to a file to store stopage and limit info for later runs.
C*********************************************************************

C*x=distance in pixels
C*y=height in pixels (in distance from top of image!)
C*mi/pix1=distance/pixels used to calculate ratio
C*ft/pix2=height/pixels used to calc ratio
C*x1=first distance (pixels)
C*x2=new pixel distance read (can have multiple y's for one x - use
C*   last y)
C*dd=mi/pix1
C*dh=ft/pix2
C*h=height in ft.
C*stops(i)=location of stops in mi
C*location(i)=location of new speed limits
C*limit(i)=speed limit corresponding to location(i)
C*d=current distance in mi
C*n=number of distance data
C*i=incrementer for inserting stops
C*j=incrementer for inserting limits
C*s=number of stops
C*m=number of limits
C*y1=first height in pixels

      real x,y,mi,pix1,ft,pix2,x1,x2,dh,h,stops(20),location(20),y1
      double precision dd,d
      integer n,i,j,s,m,limit(20)

C*********************************************
C*stop locations read from file
      read (9,*) s
      i=1
      do while (i.le.s)
      read (9,*) stops(i)
      i=i+1
      enddo

C*Speed limits read from file
      read(9,*) m
      j=1
      do while (j.le.m)
      read (9,*) location(j),limit(j)
      j=j+1
      enddo

C*pixel points read from file
      read (7,*) n
      read (7,*) mi, pix1
      read (7,*) ft, pix2
      read (7,*) x1, y1
      dd=mi/pix1
      dh=ft/pix2

C*first distance increment.  actual height doesn't matter, just
C*difference
      d=0
      h=0
      light=1
```

```
C*write road profile in miles/feet to new file
      write (1,50) n+1,dd
      write (1,100) d,h,light,limit(1)
50    format (i4,x,f16.12)
100   format (f16.12,x,f6.0,x,i1,x,i2)
      x=x1
      x2=x1
      i=1
      j=1
      do while (x2.lt.(n+x1))
      read (7,*) x2,y
      do while (x2.eq.x)
             read (7,*) x2,y
      enddo
      x=x2
      d=(x2-x1)*dd
      h=(y1-y)*dh
      if ((d.ge.stops(i)).and.(i.le.s)) then
             light=0
             i=i+1
      else
             light=1
      endif
      if ((d.ge.location(j)).and.(j.le.m)) then
             j=j+1
      endif
      write (1,100) d,h,light,limit(j-1)
      enddo
      return
      end
      subroutine rate(accel,tau,a,v,goal,int,dt)

C*******************************************************************
C*This determines the acceleration rate based on an acceleration
C*profile chosen to resemble a sine wave.  Initially, the rate
C*increases until it reaches the max rate at which point it holds
C*steady until the difference between the goal velocity and the
C*present velocity requires a slackening of the rate.
C*******************************************************************

C*accel=the maximum acceleration (or decel) rate
C*tau=the time taken to reach the max acceleration
C*goal=the velocity requested by the speed limit or stop
      real accel,tau,a,v,goal,dt,int

C*************************************
C*if the direction of acceleration changes (due to close stops,etc.)
C*the sine curve must restart
      if (accel*a.le.0) then
      int=0
      endif
C*advance the sine wave if below tau or if the goal speed is close
      if ((int.lt.tau).or.(int.gt.tau).or.
      &      ((abs(goal-v)).le.(abs(accel/2/3600*(tau-dt))))) then
      int=int+dt
      endif
      a=accel/2*(1-cos(int/tau*3.14159))
      if (int.eq.(tau*2)) then
      int=0
      endif

      if ((abs(goal-v)/dt*3600).lt.abs(a)) then
      a=(goal-v)/dt*3600
      endif
      return
      end
      subroutine readfile(n,dd,d,y,light,limit,Cr,Cd,M,Area,
     &       diameter,displacement,nogears,ratio,
     &       torquemax,rpmmax,pertorque,up,dn,idle,
     &       mrpm,mload,velprof,time2,duration2,v2,t,dt,
     &       grid,sfc,eihc,eico,eino)

C*******************************************************************
C*This reads the files containing the road profile and car data
C*******************************************************************

C*data against distance
C*n=number of data points to be read from distance data

      real d(5000),y(5000),light(5000),limit(5000),dd
      integer n

C*car data and incrementer
      real ratio(5),Cr,Cd,M,Area,diameter,displacement,transaxle
      integer i,nogears
      real idle,torquemax,rpmmax,pertorque(12),up(12),dn(12)

C*variables for emissions
C*sfc is in g/kWhr
C*ei.. is as a percentage (gEmmis/gFuel *100)
C*grid determines size of the map. Will be a matrix of grid x grid
      real mrpm(10),mload(10),sfc(10,10),
     &       eihc(10,10),eico(10,10),eino(10,10)
      integer grid

C*VELOCITY PROFILE VALUES
C*velprof=y if following a velocity profile
C*time2, duration2, v2: these arrays have a '2' to
C*differentiate them from the values for the route profile.
C*time2init=the initial time
C*t=the number of time data points
      character (1) velprof
      real time2(5000),duration2(5000),v2(5000),dt,time2init
      integer t
C*****************************************************
C*reads road (or velocity) profile
      if (velprof.eq.'y') then
      read (1,*) t,dt
      read  (1,*) time2init,duration2(1),v2(1)
      time2(1)=0
      i=2
      do while (i.le.t)
             read (1,*) time2(i),duration2(i),v2(i)
             time2(i)=time2(i)-time2init
             i=i+1
      enddo
      else
      i=1
      read (1,*) n, dd
      do while (i.le.n)
             read(1,*) d(i),y(i),light(i),limit(i)
             i=i+1
      enddo
      endif

C*reads physical car data
      read (2,*) Cr,Cd,M,Area,diameter,displacement
      read (2,*) torquemax,idle,rpmmax
      read (2,*) nogears,transaxle
      i=1
      do while (i.le.nogears)
      read(2,*) ratio(i)
      ratio(i)=transaxle*ratio(i)
      i=i+1
      enddo

C*reads shifting schedule for car
      read(3,*) pertorque(1),pertorque(2),pertorque(3),
      &           pertorque(4),pertorque(5),pertorque(6),
      &           pertorque(7),pertorque(8),pertorque(9),
      &           pertorque(10),pertorque(11),pertorque(12)
      read(3,*) up(1),up(2),up(3),up(4),up(5),up(6),
      &           up(7),up(8),up(9),up(10),up(11),up(12)
      read(3,*) dn(1),dn(2),dn(3),dn(4),dn(5),dn(6),
      &           dn(7),dn(8),dn(9),dn(10),dn(11),dn(12)


C*reads emission maps [emiss(speed,load)]
      read(4,*) grid
      read(4,*) mrpm(1),mrpm(2),mrpm(3),mrpm(4),mrpm(5),
      &             mrpm(6),mrpm(7),mrpm(8),mrpm(9),mrpm(10)
      read(4,*) mload(1),mload(2),mload(3),mload(4),mload(5),
      &             mload(6),mload(7),mload(8),mload(9),mload(10)

      i=1
      do while (i.le.10)
      mrpm(i)=mrpm(i)/rpmmax
      mload(i)=mload(i)/torquemax
      i=i+1
      enddo

      i=1
      do while (i.le.grid)
      read(4,*) sfc(1,i),sfc(2,i),sfc(3,i),sfc(4,i),
      &                   sfc(5,i),sfc(6,i),sfc(7,i),sfc(8,i),
      &                   sfc(9,i),sfc(10,i)
      i=i+1
      enddo
      i=1
      do while (i.le.grid)
      read(4,*) eihc(1,i),eihc(2,i),eihc(3,i),eihc(4,i),
      &                   eihc(5,i),eihc(6,i),eihc(7,i),eihc(8,i),
      &                   eihc(9,i),eihc(10,i)
      i=i+1
      enddo
      i=1
      do while (i.le.grid)
      read(4,*) eico(1,i),eico(2,i),eico(3,i),eico(4,i),
      &                   eico(5,i),eico(6,i),eico(7,i),eico(8,i),
      &                   eico(9,i),eico(10,i)
      i=i+1
      enddo
      i=1
      do while (i.le.grid)
      read(4,*) eino(1,i),eino(2,i),eino(3,i),eino(4,i),
      &                   eino(5,i),eino(6,i),eino(7,i),eino(8,i),
      &                   eino(9,i),eino(10,i)
      i=i+1
```

```
enddo

close(1)
close(2)
close(3)
close(4)
return
end
```

```
      subroutine requestgear(torque,rpm,v,torquemax,rpmmax,pertorque,
      &                  up,dn,gear,status,nogears)
C***********************
C*This selects the gear based on the torque/rpm shift curves
C************************

      real torque,rpm,v
C*shifting data
C*newup,newdn=rpm where shift is requested - interpolated by torque
      real torquemax,rpmmax,pertorque(12),up(12),dn(12),newup,newdn
      integer status,nogears,k,gear

C**************************
      k=2
      do while (((torque/torquemax).gt.pertorque(k)).and.(k.lt.12))
      k=k+1
      enddo
      if (k.ne.12) then
      call interpolate(pertorque(k-1),pertorque(k),
      &            up(k-1),up(k),torque/torquemax,newup)
      call interpolate(pertorque(k-1),pertorque(k),
      &            dn(k-1),dn(k),torque/torquemax,newdn)
      else
      newup=up(k)
      newdn=dn(k)
      endif

      if ((status.le.-1).and.((rpm/rpmmax).le.newdn).and.
      &  (gear.gt.1)) then
      gear=gear-1
      endif

      if (status.ge.0) then
      if (((rpm/rpmmax).lt.newdn).and.(gear.gt.1)) then
            gear=gear-1
      else if (((rpm/rpmmax).gt.newup).and.(gear.lt.nogears)) then
            gear=gear+1
      endif
      endif
      if (v.eq.0) then
      gear=1
      endif
      return
      end
      subroutine writefile(time,duration,x,v,a,angle,Pb,gear,rpm,
     &      bmep,torque,hc,cummhc,co,cummco,no,cummno,light,height,
     &      fuel,cummfuel)

C**************************************************************
C*This writes the time increment calculations into a file
C**************************************************************
       integer gear
      real time,duration,a,v,x,Pb,rpm,bmep,torque,
     &      hc,cummhc,co,cummco,no,cummno,
     &      light,angle,height,fuel,cummfuel
C**************************************************************

      write (8,100) time,duration,a,v,x,
     &      height,angle,gear,rpm,Pb,
     &      torque,hc,cummhc,co,cummco,
     &      no,cummno,fuel,cummfuel,light
     &
100   format (f7.2,x,f5.2,x,f7.0,x,f5.2,x,f7.4,x,
     &           f6.0,x,f5.1,x,i1  ,x,f5.0,x,f7.0,x,
     &           f6.1,x,f6.3,x,f6.2,x,f6.3,x,f6.2,x,
     &           f6.3,x,f6.2,x,f6.3,x,f6.1,x,f2.0)
      return
      end
```

References

1. Cheng, W.K., Hamrin, D., Heywood, J.B., Hochgreb, S., Min, K., and Norris, M., "An Overview of Hydrocarbon Emissions Mechanisms in Spark-Ignition Engines," SAE paper 932708, SAE International Congress & Exposition, Detroit, MI

2. Butler, J.W., Jesion, G., St. Denis, M.J., Cicero-Fernandez, P., and Winer, A.M., "Effects of In-Use Driving Conditions and Vehicle/Engine Operating Parameters on 'Off-Cycle' Events: Comparison with Federal Test Procedure Conditions," Air & Waste Management Association, Vol. 44, pp. 31-8, Jan. 1994

6201-21