HYBRID VEHICLE TESTING AND SIMULATION

FINAL REPORT -- DRAFT

P. T. KREIN
S. SPLATER
C. HIDROVO D. LOGUE

Power Affiliates Program
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

PAP-TR-97-4
May 1997
Abstract

Component tests have been completed on our hybrid electric vehicle system in support of system modelling, simulation, and analysis. Component models have been provided for the traction motor, the electronic inverter that drives the motor, cooling fans, control electronics, and other electrical components. Energy consumption due to drag, tire rolling resistance, and slopes has been modelled. Models for mechanical components such as the transmission follow measured performance of the test vehicle. The braking system is modelled to track the combination of regenerative brakes and hydraulic brakes. The battery pack has been modelled as a system to track its state of charge and energy performance. Simulation analysis, combined with results from road testing, has been used to compare operating strategies for the hybrid vehicle as a system. Excellent results have been obtained from simulation studies: actual road tests are tracked closely by simulations for equivalent drive cycles. The most appropriate system operating strategy for a hybrid vehicle system of the series type is identified in this work.

The Final Report includes two Master of Science theses prepared as part of the work, and two white papers prepared under this and follow-up work. These extra documents cover, respectively, engine testing and performance, system and component testing and performance, the simulation models and simulation tests, and an evaluation of hybrid system control strategies.
Summary

This report describes activities and results for the University of Illinois Hybrid Electric Vehicle (the "test car"), in support of modeling and simulation efforts at NREL. All vehicle sub-systems have been characterized in the laboratory and on the road. Each sub-system is modeled based on physical principles and characteristics, as well as on measured data. The final project effort has been to validate the complete system model by direct comparison between simulation studies and road tests. Excellent matches have been demonstrated, and validate the system-level model as well as the sub-system models. Details of tests and models of the vehicle's engine are provided in the M.S. thesis "Implementation of a multi-port fuel injection system and stoichiometry in a small V-twin engine," by Carlos Hidrovo. The modeling details are described in the M.S. thesis "Power consumption analysis of a practical series hybrid electric vehicle," by Scott A. Splater. Simulation studies and validation are described in a white paper by Daniel Logue. System-level control strategies are examined in a second white paper by Daniel Logue. These documents should be considered part of this Final Report.

In addition to the modeling task, hybrid system operating strategies were to be examined as part of the project. The strategy studies have been completed with compelling results, and we look forward to implementing the new strategies in our vehicle. In brief, the most appropriate way to operate a hybrid system of the series configuration is to command the engine-generator sub-system to slowly track the operating power demand of the vehicle, subject to a state-of-charge target for the batteries. More detail is provided in the second white paper by Daniel Logue.

The project has been a strong confirmation that there are very effective modeling methods for a complex system such as a hybrid vehicle. We were able to show that energy-based models, combined with steady-state models for various electrical components and sub-systems, give an excellent indication of detailed system performance.

Model Summary

Each major sub-system of the hybrid electric vehicle (HEV) must be included in a full system simulation to produce accurate results. The modeling strategies for each sub-system are summarized here.
1. Vehicle traction load. This includes aerodynamic drag, tire rolling resistance, the weight load on a slope, and the extra force needed for desired acceleration. The nature of these models is well known. Measured data from the test car are consistent with expected values for traction load. The model generates the axle torque needed to produce a certain speed, given a slope and acceleration.

2. Transmission and drive train. The transmission and drive train were modelled primarily from measured data. The results show power loss that is a function of speed and gear number. Given a vehicle speed and gear setting, the model allows a power value to be assigned to the transmission. The system simulation determines the gear setting based on expected driver action. Once the gear setting, transmission power, and traction power have been determined, required motor torque and speed are computed by the simulator.

3. Batteries. A circuit model is used to estimate electrical losses as a function of current. The measured characteristics of our lead-acid batteries are used to determine energy capacity as a function of charge or discharge rate and time. At each time step, the battery load is evaluated, and the energy is incremented or decremented according to capacity effects. The circuit model allows us to evaluate voltage over time.

4. Traction motor. A circuit model is used to relate electrical input to mechanical output of the motor, and to represent motor losses. The model enforces limits on both driving and braking torque. Model parameters were obtained from test data, and are consistent with the manufacturer’s data. The model is used iteratively: The required motor torque and speed values can be met only with a specific electrical input frequency. A few iteration steps allow this frequency to be determined. Once the frequency is known, the required motor voltage and current, and the expected motor losses can be found from the circuit model. If the required voltage is inconsistent with the available battery voltage, motor torque must be reduced and the car’s performance degrades.

5. Traction motor cooling fan. Our test car has a fan mounted on the traction motor shaft. Tests show that the fan’s power consumption is consistent with conventional models. It is a strong function of motor speed. When the motor speed is determined, the fan power adds to the motor
shaft load and increases the required motor current, power, and losses.

6. Main inverter electronics. The inverter converts battery energy into the ac voltage needed by the motor. A simple characteristic equation has been developed to model inverter losses. This is a new approach not previously published. It gives good estimates of inverter losses given a motor current value.

6. Control electronics. Controls and the general twelve volt system in our test car are independent of vehicle speed or operating conditions. Therefore, the model is simply a fixed power consumption.

7. Engine. In our test car, a closed-loop fuel control system is used to keep the engine close to its optimum fuel consumption under all normal conditions. The HEV operates to ensure that engine adjustments can be tracked by the closed-loop system. Laboratory measurements show that fuel consumption (for gasoline) of 235 g/kW·hr or less can be enforced if any engine transients occur over intervals of about one minute or more. In simulation, the sub-system control allows us to model the engine with fixed fuel consumption of 235 g/kW·hr if the engine power exceeds 5 kW. For a target power level, this fuel consumption requires a specific engine speed. The speed is used in conjunction with the generator model. In operation, the engine is shut off if the power drops below this level. Actual measurements suggest that fuel consumption as low as 220 g/kW·hr can be achieved, so the 235 g/kW·hr is conservative.

8. Generator. The generator is a permanent magnet synchronous machine, and a circuit model appropriate for this technology is used here. The resistance and inductance parameters for this model were measured by the manufacturer and confirmed in our laboratory. Given a desired engine speed and power, the voltage, current, and power loss can be determined for the generator.

9. Rectifier. The rectifier converts energy from the generator and delivers it to the battery pack. In the test car, a two-loop control structure is used. The inner loop enforces a specific rectifier current, with an engine speed compensation to ensure stability. The current actuation function is supported directly by the rectifier control hardware. The outer loop sets a target power level. Given battery voltage, the power is used to compute the command current to be found. Rectifier losses are proportional to current.
10. **Braking.** In this HEV, braking is simply "negative traction," and the traction motor model provides all the necessary information. The extra issue is that some portion of braking power will be absorbed by the conventional mechanical brakes in the vehicle. The interface between the regenerative brakes and hydraulic brakes is therefore of particular importance. For simulation studies, the braking action is assumed to be regenerative unless the torque requirement exceeds the motor’s capabilities. If extra torque is required, it is assigned to the conventional hydraulic brake system. The extra energy dissipated by the hydraulic brakes is modelled as a loss. The model is consistent with expected driver action: The driver modulates the brake pedal to achieve a desired braking torque, and the action is purely regenerative unless the pedal is pressed an extra distance.

System simulations that include these ten sub-system elements provide an excellent match to actual on-road performance of the vehicle. On this basis, we conclude that the important energy factors in the test car have been captured successfully in these elements.

**Discussion**

The electrical operation of a hybrid vehicle can be modelled accurately with quasi-steady-state concepts. The electrical time constants are so much faster than mechanical response times of a vehicle that the motor can be considered a lossy torque actuator. The inverter can be considered a lossy ac current source for the motor. Speed-dependent losses in fans and drive-train elements can be modelled independent of accelerations or traction forces. The quasi-steady-state approach supports energy-based system models, without requiring models of the detailed dynamics of pulse-width modulated inverters, nonlinear motor control algorithms, or other details of sub-system dynamic control structures. The quasi-steady modelling approach is well established in electromechanics and power electronics.

The auxiliary power unit (APU), consisting of the engine-generator set in this vehicle, is modelled as a power source into the batteries, subject to battery limitations. The modelling process can be quasi-steady because the power command to the APU set is deliberately set over long times intervals. The system operating principle is that the APU follows a moving average of the vehicle’s power demand, plus a small extra power if needed to achieve a target battery charge. With moving averages over intervals of
a minute or more, the engine can maintain ideal fuel-air mixtures throughout its operating range. The APU is essentially a steady system element on these time scales.

Details of the battery models and battery-drive interaction are essential for accurate performance predictions. For example, when the batteries are at a low state of charge, less voltage is available for the traction motor, and the available torque decreases. Under demanding drive cycles, it is important to determine whether the traction system can meet the power demands under all conditions. One important attribute of the test car is that it has sufficient power and torque under normal battery charge conditions to follow any of the standard Federal drive cycles.

The simulation model developed as part of the project takes the drive cycle (speed and time) as input, and simulates vehicle operation in a "best effort" attempt to track the cycle. The simulation outputs are the energy consumed and the fuel or battery charge consumed under the selected APU operating strategy. Examples are provided in the white papers.

Conclusions

A hybrid vehicle of the series configuration supports an unmatched combination of system capabilities. With recent advances in electric motor control and traction systems, it is straightforward to prepare a drive of 100 kW or higher rating for a passenger car. With appropriate engine controls, fuel consumption levels as low as 220 g/kW·hr have been demonstrated. With a system-level average power tracking strategy, the high performance of an electric traction drive can be combined with a high-efficiency low-emissions engine control without imposing high stress on storage batteries.

These conclusions are well-supported with model-based system simulations. The models have been developed from laboratory and on-road measurements made with the University of Illinois hybrid Electric Vehicle.
To: David Rausen, NREL
From: D. Logue, P. T. Krein, University of Illinois
Subject: HEV Simulation and Matlab Code
Date: May 12, 1997

This technical summary gives an in depth description of the HEV simulation program. Also included are simulation examples and model descriptions. The completed program is included on the enclosed 3.5" disk. All of the necessary files are contained in the hev directory on the disk. This directory should be copied to the computer on which the simulation is intended to run. The hev directory will then need to be added to the matlabrc file on this computer. The simulation may then be initiated by running the mnwin program from the MATLAB command line.

Program operation

A block diagram of the simulation is shown below in figure 1.

![Block diagram of HEV simulation.](image)

The particular models used in this diagram will be discussed in the next section. The Data File block above represents a file containing three vectors named time, vel, and grade. The time vector is the primary time vector which controls program execution. The time vector does not have to be uniform. The vel vector contains the velocity information which the program attempts to follow. Finally, the grade vector contains road grade data corresponding to each time instant. The grade data is expressed as an sin of the actual road grade angle. For instance, if the road grade is at a constant angle of 45°, then all of the components of the grade vector will be
\[
\sin(45°) = 0.707.
\]

These three vectors contained in the input file must have the names \textit{time}, \textit{vel} and \textit{grade}. The vectors may either be loaded by hand using the MATLAB command `load` or they may be loaded using the \textit{Load vectors} option under the \textit{File} pull-down of the HEV simulation main window. These vectors apply only to dynamic simulation. For steady state simulation, a velocity, grade or battery pack size sweep is selected from the Input Screen window under the \textit{Steady State} pull-down.

The dynamic simulation may be run by selecting run under the \textit{Calculate} pull-down and then selecting either Tracking APU or Exp APU. For steady state simulation the run option is selected under the \textit{Steady State} Pull-down. The program then uses the time and vel vectors to calculate the acceleration, the drive motor speed and the points at which the vehicle shifts gears. The vehicle mass is calculated using the number of batteries and the individual battery weight, both defined from the General control window, along with payload and vehicle weight, both defined in the vehicle control window. Using these parameters and the air density, vehicle frontal area, drag coefficient and tire resistance coefficient, all defined in the Vehicle control window, the road power is calculated using eq. (1) below

\[
P_{\text{road}} = \frac{1}{2} \rho C_d A_f v^3 + R_{\text{tire}} m g v + m g s v + m a v
\]  

where \( \rho \) is the air density, \( C_d \) is the drag coefficient, \( A_f \) is the vehicle frontal area, \( v \) is the velocity, \( R_{\text{tire}} \) is the tire resistance coefficient, \( g \) is the gravitational acceleration constant (9.81 m/s), \( s \) is the grade as defined above, and \( a \) is the acceleration.

Once the road power is calculated it is passed to the drive motor model which solves for the motor slip. It uses this value along with the constraint that the input voltage is proportional to the synchronous frequency to solve for the input voltage, input current and motor losses using the standard induction motor model with the shunt terms moved to the input terminals. All of the motor model parameters are available from the motor control window. Another option is available from the motor control window by choosing not to use the actual motor model. If this is done the simulation will use the linear model. In most cases, this model will perform as well as the actual model. Its advantages are that it is faster and when using actual road data, it handles slight motor overloads caused by uncertainties such as mechanical braking much better. The actual model will tend to cause the program to lock up if the motor's maximum torque is exceeded. A maximum torque may be applied to the linear model from the motor control window. If it is exceeded a warning message is given.

After the drive motor input current is calculated this value is used to extract electronic drive losses. As will be shown below, the electronic drive is configured as a diode pair per each phase. In this case, the drive motor is three-phase, so the electronic drive is modeled as six diodes. Each diode has a forward drop as well as an on resistance, so only the current is needed to calculate the power dissipation.

Once the total power demand is calculated, this is used in conjunction with the APU output power to determine the load on the batteries. The battery model is a parabolic fit of experimental
data for the Johnson Controls UPS 12-95 battery of watt-hour rating vs. power draw. Using the power demand on the batteries, the watt-hour rating is calculated. A ratio of this value to the maximum battery watt-hour rating is used to de-rate in real time the battery pack capacity. The maximum battery watt-hour rating is a variable named \textit{Whrref} and is currently set to 302 W\textperiodcentered hr. If the battery pack maximum energy density rating is exceeded a warning is given and program execution is halted. This rating may be set to any desired value from the General control window. The number of batteries and the individual battery weight may be set from this window as well. For steady state simulation, a battery pack size sweep is selected from the Input Screen under the Steady State pull-down. The program automatically calculates battery weight and battery number.

For dynamic simulation, the APU may be selected as either switching or tracking. In the switching case, the APU simply switches on full power when the battery SOC falls below the SOCL value set in the General control window. The APU shuts off when the battery SOC climbs above the SOCH value set in the General control window. The APU turns on and off with a time constant set from the general control window. Both SOCL and SOCH are percent values of full pack capacity. This particular switching program also turns on the APU if the power demand is above 20 kW for more than twenty seconds. This may be change by simple modifications to the program file mnloop2. In the tracking case, the APU output changes with a time constant set as above in an attempt to force the battery SOC to track the SOCH value.

It is also possible to select either a switching or tracking APU in the steady state case. There are differences in operation, however. The switching APU is assumed to be ideal. It switches with no delay. The tracking APU is assumed to supply all of the power demand such that the batteries never even come into play electrically. The battery weight is still an important factor. The same APU maximum output power used in the dynamic case is used in the steady state case. Also, in the steady state case, the fuel to energy conversion ratio of the APU may be set from the Input screen under the Steady State pull-down.

Lastly, the vehicle has to know when to shift gears and what ratios to use. This is done from the Gear Ratios control window. The ratios of motor speed to axle speed for each gear are set here. The shifting process may be done in one of two ways. The vehicle may be commanded to shift at either preset vehicle speeds or at a particular motor speed from this window. The preset vehicle speeds or the motor shift speed are set from this window. The shift speeds simply tell the simulation at what vehicle speeds to change gears. The motor shift speed tells the vehicle to shifter higher when the motor reaches this speed. The simulation is currently configured to have a five speed transmission. This may be reduced to any number of gears below five simply by setting the lowest gear's shift speed to zero. This causes the simulation to effectively ignore them.

Once the simulation has ran, variables to be plotted have to be selected. For the dynamic case, this is done by selecting the variables from under the Select Vars pull-down. When they are selected, a checkmark appears beside them. After the variables are selected, they may be plotted by choosing Plot Variables under the Graphics pull-down. They may also be plotted in a normalized fashion by choosing Plot Normalized instead of Plot Variables. This type of plotting
can be help in instances where one is interested in seeing when particular events occur relative to one another. After plotting, the variables may be printed in two ways. The first is to choose print from under the File pull-down. This will print the current window as it is shown. This is helpful when it is desired to add entities to the window such as a legend and have them show up when printing. The other printing method is to choose Print Variables under the Graphics pull-down. This method creates new axes for each plot and makes them bigger than is possible using MATLAB's command subplot. The disadvantage is that a temporary print window is created, losing any new objects placed on the main plot window. Finally, the variables may be printed to a file in vector form by choosing Print to file under the Graphics pull-down. This method will place the vectors in a *.mat file which may be retrieved later using either MATLAB's load command from the command line or the Load plots option under the File pull-down. Although this does not place the simulation in the same state as when the vectors were created, it does allow the vectors retrieved to be examined again.

For the steady state case, the variables to be plotted are selected by choosing the Select vars option under the Steady State pull-down. Doing this creates a window in which the desired variables may be selected by clicking the mouse over them. Once selected, the variables may be plotted by choosing Plot Variables from either the Select vars window or from under the Steady State pull-down. Selected variables may also be printed to a *.mat file and retrieved as in the dynamic case by selecting Print to file from under the Steady State pull-down.

**Modeling**

Descriptions of the models used in the program will now be given. These include the drive motor, electronic drive, battery pack, APU, drive motor fan, and mechanical braking.

**Motor model**

The motor model used is the standard induction motor model with the shunt terms moved to the input side as shown in figure 2 below.

![Induction motor model](image)

**Figure 2. Induction motor model.**

The shunt impedance is given by \( r_c \) paralleled with \( jX_\phi \), where \( X_\phi = \omega L_\phi \). The series impedance is \( r_1 + r_2 + j(X_1 + X_2) \) where \( X_1 = \omega L_1 \) and \( X_2 = \omega L_2 \), plus a resistive term whose power dissipation represents the actual motor shaft power. Here \( \omega \) is the motor electrical radian frequency and \( L_\phi \).
L₁ and L₂ are leakage inductances associated with the motor. The resistances r₁, r₂ and r₃ are called the shunt resistance, stator resistance and rotor resistance respectively. L₁, L₂ and L₃ are called the shunt inductance, stator leakage inductance and rotor leakage inductance respectively. Note that r₂ and X₂ are stator side equivalent rotor impedances. All of these single phase parameters may be changed from motor control window. The motor model may now be solved using the desired shaft power and equations (2), (3) and (4) below.

\[
s = \frac{\omega_s - \omega_m}{\omega_s}
\]  

(2)

\[
V_{i-n} = k\omega_s + V_{offset}
\]  

(3)

\[
P_{shaft} = \frac{3|V_{i-n}|r_2 s\omega_m}{\omega_s \left((sr_1 + r_2)^2 + s^2(X_1 + X_2)^2\right)}
\]  

(4)

Equation (2) is the well-known relationship between the slip s, the synchronous frequency ωₛ and the rotor mechanical frequency ωₘ. Equation (3) defines the single-phase input voltage as proportional to the synchronous frequency with proportionality constant k plus a constant offset voltage V_{offset}. This is a standard way to maintain constant flux density within the motor when the excitation frequency changes. Finally, eq. (4) defines the total shaft power. Using these equations along with the demanded road power allows the motor model to be solved for the motor losses, input voltage and input current.

**Electronic drive**

The electronic drive model is based on a three-phase IGBT PWM inverter. This inverter contains six IGBTs, two of which are on at any given moment. Associated with each of these devices is a forward voltage drop V_{forward} and an on state resistance r_{on}. Each leg of the inverter contains two IGBTs in series, therefore, the average power loss per leg is given by

\[
P_{loss(leg)} = \frac{2}{\pi} \int_0^{\pi} \sqrt{2}I_{rms} \left(V_{forward} + r_{on} I_{rms}\right) \sin(\theta) d\theta = \frac{4\sqrt{2}I_{rms}}{\pi} \left(V_{forward} + r_{on} I_{rms}\right)
\]  

(5)

The inverter has three legs, so the total inverter loss is three times the value given by eq. (5).

**Battery pack**

The battery pack is modeled as a power delivery device without regard to exact values of terminal voltage or current. Battery modeling in a straightforward fashion is hindered by storage capacity being a non-linear function of the power drawn from the battery. This is especially true for the dynamic case where the load changes constantly. First, it will be shown how the battery model is developed and second, how this model is employed within the simulation.
The model constructed is based on the Johnson Controls UPS 12-95 battery. The batteries were charged and then discharged at various rates in a laboratory environment. In this way, watt-hour ratings were obtained as a function of power draw. The watt-hour rating is a measure of battery capacity. It is a multiplication of the power in watts by the time in hours required for discharge. Using this data, one can construct a plot as in figure 3.

![Figure 3. Battery model data.](image)

This data is then approximated by a line fit, in this case second order. So the watt-hour rating as a function of power draw is approximated by eq. (6).

\[
\text{Watt-hour rating} = (7 \times 10^{-6})P_{\text{draw}}^2 - 0.0904P_{\text{draw}} + 273.08
\]

(6)

For the steady state case, a single de-rating takes place: a certain power draw is required and the battery pack capacity is de-rated accordingly. In the dynamic case, however, it is not this simple. The energy taken from the battery pack at each time instant must be re-rated to account for the watt-hour rating change. Assume that the maximum pack capacity is given by \(Whr_{\text{ref}}\) the watt-hour reference. Now, assume at a particular time step the power draw is \(P_1\) with a corresponding watt-hour rating of \(Whr_1 < Whr_{\text{ref}}\). The energy removed as seen from the pack must be larger than the useable energy withdrawn. The useable energy taken is

\[
\Delta E_{\text{useable}} = P_1 \Delta t
\]

(7)

But, if we assume that the pack capacity is constant at \(Whr_{\text{ref}}\), then to account for the de-rating, the energy taken as seen from the pack must be larger and is given by
\[
\Delta E_{\text{pack}} = \frac{Whr_{\text{ref}}}{Whr_1} P_i \Delta t
\]  

(8)

This process acts in the reverse manner during charging. More energy must be supplied to the pack than is actually stored for later use. The following equation is used at each time step to calculate the new pack energy

\[
E_{\text{pack}}(k+1) = E_{\text{pack}}(k) - \left( \frac{Whr_{\text{ref}}}{Whr_1} \right)^{\text{sign}(P_i)} P_i \Delta t
\]  

(9)

where \( P_i \) is the power out of the pack at the time step \( k \). If the power out of the pack is positive, the pack is being discharged, and the energy removed is made larger by the watt-hour ratio. If the power out of the pack is negative, the pack is being charged, and the energy placed in the pack is made smaller by the watt-hour ratio. In this way the energy removed or given to the pack is re-rated at each time step.

Finally, there is a maximum power draw beyond which the batteries will fail to supply any useable energy. This the point at which the curve in figure three crosses a horizontal line at a watt-hour rating of one. This implies that the entire pack is discharged in one time step. As the discharge rate approaches this point battery damage becomes a possibility. The simulation allows for the battery's maximum output power to be set to any value below the model's maximum power capability.

**Auxiliary power unit**

The auxiliary power unit (APU) is modeled as a controllable power source. Its primary objective is match the average power demanded by the vehicle. This can be accomplished in two ways. The first is to switch the APU on and off in such a way that its average output matches the average power demand. The other way is to make the APU output track the average power demand. These two models will be discussed below.

The switching APU simply switches on full power when the battery state of charge (SOC) falls below a preset point. It then shuts down after the pack has re-charged to a second preset SOC. In this way the battery pack SOC always stays within a set of minimum and maximum bounds. To more closely model a realistic power source, the model has a user specified time constant associated with it which controls how fast the APU can be turned on or off. The APU output rises and falls in an exponential manner governed by this time constant. Finally, the APU can be commanded to come on after a preset power level has been sustained for a certain length of time. The power level and time length may be changed by altering the program mnloop2.

The output of the tracking APU model changes in order to force the battery pack SOC to stay at a certain level. This APU model also has the same time constant associated with it as does the switching APU. The battery pack also has a discharge time constant dependent upon the power
draw. This results in a second order system. This system will track unless the APU maximum output level is exceeded. It will then supply a constant maximum power.

Drive motor cooling fan

The cooling fan is almost an essential ingredient in an HEV application. During times of heavy acceleration, the drive motor may become excessively hot if a cooling fan is not present. A cooling fan may also be required in HEV applications utilizing an internal combustion engine for the APU. If the internal combustion engine is running near stoichiometric conditions, there may be excessive exhaust heat. A cooling fan is one viable solution to this problem.

The cooling fan is modeled as a power sink which is proportional to the square of the fan's angular velocity. So the fan power equation is

\[ P_{\text{fan}} = k_{\omega} \omega^2 \]  

(10)

The simulation's default value of \( k_{\omega} \) is set to match the University of Illinois' HEV cooling fan. This value is \( k_{\omega} = 2.93 \times 10^{-5} \).

Mechanical braking

This appears to be the most difficult element of the HEV to model. Without an extensive knowledge of the braking system, the power dissipated by the mechanical brakes is very difficult to ascertain. For the drive cycle comparisons performed with the University of Illinois' HEV, the best approximation seems to be a power proportional to the vehicle deceleration squared. The mechanical braking is approximated by

\[ P_{\text{braking}} = k_{\text{brake}} a^2 \]  

(11)

The value of \( k_{\text{brake}} \) may be changed in the program file named brake.

Examples of operation

Some examples of program operation will now be considered. These include dynamic and steady state simulations.

Dynamic simulation examples

The first two examples are contained in the datfiles directory on the enclosed 3.5" disk. They are called city1.mat and city2.mat. These files contain time and velocity data taken from the University of Illinois' HEV as it was driven in an urban environment. The grade vector is assumed to zero, although in reality it isn't. When a comparison is done between the simulated output and actual data taken from the vehicle, there will be some deviation in results due to the lack of grade information. Furthermore, the runs were done with no APU output and with the
vehicle in second gear. These files may be loaded by using the Load vectors option under the File pull-down. Let's first load the cityl.mat file. Once this is done the APU output value in the General control window needs to be set to zero. Set the first gear shift speed to 1 mph and the rest of the gear shift speeds above 50 mph. This may be done from the Gear Ratios control window and will force the simulated vehicle to remain in second gear. The default gear ratios are those of the University of Illinois' (UI) HEV. Then select the linear motor model from the Motor control window. This is done because the actual model may not converge or will take a long time to do so. This is caused by lack of grade information. At some points during the collection of velocity data in the vehicle, it was going down a hill during heavy acceleration. With no grade information, as the simulation program processes the velocity information, it appears as though the vehicle can accelerate faster than it actually can. If the full motor equivalent circuit is used, the torque capability of the model will be exceeded and it will not converge. Finally, set the number of batteries to 26, the number that the UI HEV contains and select Run then Tracking or Exp APU from under the Calculate pull-down. After it is finished running, select Velocity, the vehicle road speed, and Ppack, the power draw out of the battery pack, from under the Select Vars pull-down. Then select Plot Variables from under the Graphics pull-down. The velocity plot which should appear is shown below in figure 4. A comparison of simulated power out of the battery pack and actual data taken from the UI HEV is shown in figure 5.

![Velocity vs. time from file cityl.mat.](image)
Now, the plots for the file city2.mat. The velocity profile is shown in figure 6. The comparison of simulated power out of the battery pack and actual data is shown in figure 7.

Figure 5. Comparison of simulated and actual results.

Figure 6. Velocity vs. time for the file city2.mat.
The differences in the simulated and actual results are attributed to two factors. One, there was no grade information to feed the simulation. Two, it is very difficult to estimate the mechanical braking.

The next two simulations were ran using the file mph403hr.mat, which is also included on the 3.5" disk within the datfiles directory. This file contains a time vector 3 hours long with a corresponding velocity vector of a constant 40 mph. This file is 10,000 components in length and takes a fair amount of time to run. These were ran with a 300 s APU time constant, set from the General control window. The battery pack size is set at 12 batteries, the APU maximum output is set at 20 kW and the road grade to zero. The simulation was ran twice, once with the APU in switching mode and the once with the APU in tracking mode. The output for the tracking case is shown in figure 8 and the output for the switching case is shown in figure 9. In both cases, the delay imposed by the time constant is obvious.

**Steady state simulation examples**

Now, some steady state simulation examples will be given. The first will be an example of the velocity parameter being varied while the road grade is held at zero and the battery pack is a constant 20 MJ. This may be accomplished simply by opening the Input screen from the Steady State pull-down and making the initial grade zero and the initial pack size 20 MJ. As is stated on the input screen, those parameters which are being held constant take their values from their respective initial value blocks. The velocity is to be varied from 0 to 40 m/s. So set the initial velocity block to zero and the ending velocity block to 40. Make sure that when you change the value of a block that you click the mouse pointer on an area outside the block so that the block cursor stops blinking. This is will cause MATLAB to accept the value. If the cursor is still
blinking, the value will revert to the original value. This behavior is inherent to MATLAB itself. The APU output is set to 30 kW and the shift points are their default values. The APU can be set to either tracking or ideal from this screen. Now, choose Exit under the input screen's File pull-down. To run the simulation, choose run under the Steady State pull-down. After the simulation is finished running, choose the variables you wish to plot from the Select Variables option. After
selecting them, choose the Plot Variables option. Figure 10 shows the output you should observe for an ideal APU, and figure 11 shows the expected output for a tracking APU.

Figure 10. Output given a switching APU.

Figure 11. Output given a tracking APU.
The next simulation is a variation of grade from 0 to 10%. The initial grade is set to zero and the ending grade is set to 10. The pack size is left at 20 MJ and the velocity is set to a constant 15 m/s by changing the initial velocity block to 15. Figure 12 shows the output for a switching APU and figure 13 shows the output for a tracking APU.

Figure 12. Output given a switching APU.

Figure 13. Output given a tracking APU.
Finally, a demonstration of battery pack size variation will be given. The velocity is left at a constant 15 m/s and the grade is set to 1%. The battery pack size is varied from 20 MJ to 100 MJ. The output for a switching APU is shown in figure 14 and the output for a tracking APU is shown in figure 15.

Figure 14. Output given a switching APU.

Figure 15. Output given a tracking APU.
Finally, a comparison of constant speed simulations with constant speed road tests using the UI HEV will be given. These runs were done at constant speed, zero grade and with the APU output at zero. Figure 16 shows a comparison of the simulated battery pack output power and that of the actual UI HEV.

![Figure 16. Steady state comparison.](image)

There is an error value for all speeds and gears of about 5-10%. All except for the last column, the error may be attributed to transaxle loss in the UI HEV. It has been shown in the minutes right after start-up that the UI HEV transaxle can consume up to 1500 W. This falls off exponentially in time with a time constant of about 20 minutes.

The simulation does quite well overall. It is very easy to use once the user is familiar with its operation. All variables, other than function variables, are accessible from the command line. This means that any quantities the user would like to see that are not selectable for plotting from the variable selection windows may be plotted from the command line. The program files are straightforward and quite easy to change.
List of primary program files

aputc.m

This file performs the exponential rise and fall of the tracking APU. To do this it implements a first order transfer function with the given time constant. Given the time step, time constant, APU maximum output power and an initial power level it returns the new APU output power.

brake.m

Performs the function of emulating mechanical braking. Accepting the current velocity, road power, and acceleration it returns the estimated braking power.

carctrls.m

This file generates the Vehicle control window and the input of data through it.

chngstdy.m

This is the steady state input window generator.

deltae.m

This program calculates the re-rated energy taken from or given to the battery pack. It also contains the line fit for the battery model. Note that this equation is for a single battery, not the entire pack. It takes as parameters the current battery energy, the power draw, the battery watt-hour reference and the number of batteries. It then returns the re-rated energy value to add or subtract from the current battery energy content.

drvloss.m

This file calculates the electronic drive loss. It is given the rms motor current and returns the drive loss. The equation here is for a single IGBT and is multiplied by six to get the total power loss.

f.m

A file used by MATLAB’s fzero to solve the motor model equations. This file contains both the single-phase induction motor model and the single-phase linear model. The function takes no parameters in the ordinary fashion. It is called from the file motloss.m as the argument of MATLAB’s fzero function. Along with it, a precision argument is passed. The rest of the needed parameters are global variables, so f.m has access to them. In this manner, the motor model is solved.

findacc.m
This function finds the vehicle acceleration. It is passed the time and velocity vectors and it calculates and returns the acceleration.

(findcoef.m)

This file is used to find the road power equation coefficients. As stated in eq. (1) above, the road power is of the form

\[ P_{\text{road}} = c_1 v^3 + c_2 v + c_3 s v + c_4 a v \]

This program finds \( c_1, c_2, c_3, \) and \( c_4 \) and returns them. The function takes no arguments, it uses only global variables.

(gearbox.m)

This program generates the Gear Ratios input window.

(genctrs.m)

Generates the General control input window.

(ldvctrs.m)

This function is used to load a file containing the time, vel and grade vectors required by the simulation. If the vectors are not the same length, the shorter ones are padded with zeros. This may create problems if the time vector is not the longest one. The function takes no arguments and simply returns the three vectors.

(menctr1.m)

This file handles the selecting of variables to be plotted or printed to file for the dynamic portion of the simulation.

(mnloop1.m)

This file creates the power tracking APU for dynamic simulation. This file has its own value for Whrref which may be changed directly.

(mnloop2.m)

Creates the switching APU for dynamic simulation. This file contains its own value for Whrref which may be changed directly.

(mnwait.m)
Creates the wait screen which appears during the time the simulation is running.

`mnwin.m`

This is the main program file. It defines the main window and sets all of the variable default values. All global variables are defined here as well.

`motctrls.m`

This file creates the Motor control input window.

`motloss.m`

This file takes as arguments the motor speed in rad/sec, the road power, and the number of motor pole pairs. It returns the motor loss, motor current, motor voltage and motor slip. After the file `fm` is solved for the slip, this file solves the single-phase equivalent circuit for the other parameters. The proportional constant between the input voltage and the electrical frequency and the voltage offset are defined here.

`motspd.m`

This file uses the data from the Gear Ratios window and the vehicle speed to solve for the motor speed as well as the particular transmission gear.

`prdfind.m`

This file uses the time, vel and grade vectors along with road power equation coefficients to calculate the demanded road power for the dynamic portion of the simulation.

`proad_st.m`

This file uses the time, vel and grade vectors along with the road power equation coefficients to calculate the demanded road power for the steady state portion of the simulation.

`stdyst.m`

This is the main program file for the steady state portion of the simulation. It is used to perform the velocity, grade or pack size sweeps with either APU. This file also has its own value for Whrref which may be changed directly.

`whr_st.m`

Function used to calculate the new battery watt-hour rating.
Mnwin.m

%mnwin---Sets up main window
% This is the main simulation window program
% which contains the plot window and the
% main command menu.
% All of the global variables which the program
% uses are defined here.
% ****************To execute program****************
% Place files in Matlab's search path and execute
% this file by typing mnwin at the command prompt.

%Define globals
global comm; % Comment string
global tcons; % APU time constant
global MaxPB; % Maximum battery power
global BattWt; % Weight of one battery
global eff; % Empty eff parameter
global Bnum; % Number of batteries
global Eptot; % Pack total energy
global PAPU; % APU output maximum power
global Papu; % APU output actual power
global Epack; % Pack energy
global Ppack; % Power out of pack
global FanC; % Fan coefficient
global Pfan; % Fan power dissipation
global PBrake; % Mechanical braking power
global FEconversion; % Energy conversion (MJ/Gal)
global vel; % Velocity vector
global time; % Time vector
global grade; % Grade vector
global power; % Power vector
global acc; % Acceleration vector
global Proad; % Road power vector
global Wmot; % Motor speed vector
global gear; % Gear vector
global soc; % State of charge vector
global motorls; % Motor loss vector
global drivels; % Drive loss vector
global Slip; % Motor slip
global MotCurrent; % Motor current
global MotVoltage; % Motor voltage
global c1; % Vel. eq. coefficient
global c2; % Vel. eq. coefficient
global c3; % Vel. eq. coefficient
global c4; % Vel. eq. coefficient
global Aden; % Air density [kg/m^3]
global Afront; % Frontal area [ft^2]
global Cdrag; % Drag coefficient
global Tloss; % Tire-loss coefficient
global Vmass; % Vehicle mass [lb]
global Pmass; % Payload mass [lb]
global WheelRad; % Wheel radius [m]
global StatRes; % Stator resistance [ohm]
global StatInd; % Stator inductance [H]
global RotRes; % Rotor resistance [ohm]
global RotInd; % Rotor inductance [H]
global LeakRes; % Leakage resistance [ohm]
global LeakInd; % Leakage inductance [H]
global PoleNum; % Number of poles

global IGBTVdrop; % IGBT forward voltage drop

global IGBTRes; % IGBT on resistance

global Tmax; % Linear motor model maximum torque

global motmodflg; % Motor model type

% Steady state global variables

global stdyInivel; % Initial steady state velocity

global stdyEndvel; % Ending steady state velocity

global stdyInigrade; % Initial steady state grade

global stdyEndgrade; % Ending steady state grade

global stdyInipack; % Initial steady state pack size

global stdyEndpack; % Ending steady state pack size

global Num_stdy_pts; % Number of points in steady state analysis

global APU_type_flag; % Type of APU for steady state analysis

% 1=Ideal switching
% 2=Tracking

global Vary_type_flag; % Parameter to vary in steady state mode

% 1=vary velocity
% 2=vary grade
% 3=vary pack size

% Global steady state variables for replotting

global stdyvel;
global stdygrade;
global stdypacksz;
global pack_loss_st;
global mileage_st;
global Road_pwr;
global motls_st;
global drvis_st;
global motcur_st;
global motvol_st;
global slip_st;
global Pfan_st;
global Ptot_st;
global Wmot_st;
global gear_st;
global APU_on_time;
global APU_off_time;
global APU_period;

% Global plot flags

global fvel; fvel=0; % plot velocity flag

global facc; facc=0; % plot acc flag
Mnwin.m

global fgra; fgra=0; % plot grade flag
global fppa; fppa=0; % plot Ppack flag
global fepa; fepa=0; % plot Epack flag
global fsoc; fsoc=0; % plot SOC flag
global fpap; fpap=0; % plot Papu flag
global fpro; fpro=0; % plot Proad flag
global fpfa; fpfa=0; % plot Pfan flag
global fpbr; fpbr=0; % plot PBake flag
global fpow; fpow=0; % plot Power flag
global fgea; fgea=0; % plot gear flag
global fwmo; fwmo=0; % plot motor speed flag
global fmls; fmls=0; % plot motor loss flag
global fmtI; fmtI=0; % plot motor current
global fmtV; fmtV=0; % plot motor voltage
global fmSp; fmSp=0; % plot motor slip
global fdls; fdls=0; % plot drive loss flag

global Plt_Num_st; Plt_Num_st=0;

% Set shift globals

% 1st gear ratio
% 2nd gear ratio
% 3rd gear ratio
% 4th gear ratio
% 5th gear ratio
% Shift to 2nd speed
% Shift to 3rd speed
% Shift to 4th speed
% Shift to 5th speed
% Motor speed to shift at
% Shift flag 1-Vehichle speed
% 2-Motor speed

global plot_flag_cb; % Steady state plot flags

%---------Set default variable values--------------

Aden=1.2; % Air density [kg/m^3]
Afront=25; % Frontal area [ft^2]
Cdrag=.36; % Drag coefficient
Tloss=.01; % Tire loss coefficient
Vmass=2300; % Vehicle mass [lb]
Pmass=500; % Payload mass [lb]
WheelRad=.3; % Wheel radius [m]

%******************************************************************************
tcons=30; % APU time constant [sec]
Bnum=26; % Number of batteries
PAPU=0; % APU output power
SOCL=.4; % SOC at which APU turns on
SOCH=.8; % SOC at which APU turns off
MaxPB=6000; % Maximum battery(1) output power [W]
BattWt=26; % Weight per battery [lb]
FanC=2.93e-5; % Fan coefficient Pfan=FanC*w^2, w in RPM
%
%******************************************************************************
Mnwin.m

StatRes=.0248;  % Stator resistance [ohm]
StatInd=114e-6;  % Stator Inductance [H]
RotRes=.01438;  % Rotor resistance [ohm]
RotInd=114e-6;  % Rotor Inductance [H]
LeakRes=16;  % Leakage resistance [ohm]
LeakInd=.00356;  % Leakage Inductance [H]
PoleNum=4;  % Number of poles
Tmax=103;  % Maximum Torque [N.m]
motmodflg=1;  % Motor model actual

**-----------------------------------------------**

IGBTVdrop=1.1;  % IGBT forward voltage drop [V]
IGBTRes=.0042;  % IGBT on resistance [ohm]

**-----------------------------------------------**

shftflg=1;  % Default is vehicle speed
Gear1Ratio=11.957;  % First gear ratio
Gear2Ratio=6.652;  % Second gear ratio
Gear3Ratio=4.674;  % Third gear ratio
Gear4Ratio=3.3566;  % Fourth gear ratio
Gear5Ratio=2.638;  % Fifth gear ratio
Shift1Vel=20;  % Speed to shift to 2nd [mph]
Shift2Vel=30;  % Speed to shift to 3rd [mph]
Shift3Vel=40;  % Speed to shift to 4th [mph]
Shift4Vel=100;  % Speed to shift to 5th [mph]
Motorshft=1800;  % Motor speed to shift at [RPM]

**-----------------------------------------------**

stdyInivel=0;  % Initial steady state velocity
stdyEndvel=50;  % Ending steady state velocity
stdyInigrade=.01;  % Initial steady state grade
stdyEndgrade=.05;  % Ending steady state grade
stdyInipack=20e6;  % Initial steady state pack size
stdyEndpack=75e6;  % Ending steady state pack size
Num_stdy_pts=100;  % Number of points in steady state analysis
APU_type_flag=1;  % Type of APU
Vary_type_flag=1;  % Type of variation
FEconversion=40;  % Energy conversion (MJ/Gal)

**-----------------------------------------------**

comm='figure';  % Default figure heading
plot_flag_cb=zeros{1,30);  % Checked flags

global hpwr;  % Main window is a global variable
% Create the main simulation window.
set(0,'DefaultFigureMenuBar','None');  % Disable all window menus
hpwr=figure;  % Create the main figure window
set(hpwr, 'units', 'normalized', 'pos', [.3 .27 .7 .73]);  % Size and position
set(hpwr, 'Color', 'black');  % Set main window BG color
set(hpwr, 'Name', 'HEV Simulation Program');  % Set main window name
set(hpwr, 'NumberTitle', 'off');  % Disable window numbering
set(hpwr, 'NextPlot', 'add');  % Place all plots in main window
hpwrfile=uimenu('Parent', hpwr, 'Label', 'File');  % Create file menu pull down
% Create load/create data from text file command
hpwrcr=uimenu('Parent',hpwrfile,'Label','Create vectors','Callback',
    [time,vel,grade,power]=retfile;');
% Create standard load vectors
hpwrlv=uimenu('Parent',hpwrfile,'Label','Load vectors','Callback', ...
    [time,vel,grade]=ldvctrs;);
% Load previously created data
hpwrldd=uimenu('Parent',hpwrfile,'label','Load plots','call','ldplts;');
% Writes output file
uimenu('parent',hpwrfile,'label','Data to file','Callback','prntvars');
% Setup print menu command
hpwrprn=uimenu('Parent',hpwrfile,'Label','Print','Callback','print');
% Menu close command
hpwrcls=uimenu('Parent',hpwrfile,'Label','Exit','Callback','close');

% Specific variables to calculate
hpwrcalc=uimenu('parent',hpwr,'label','Calculate'); % Calculate drop-down
% Calculate acceleration menu
uimenu(hpwrcalc,'label','Acceleration','call','acc=findacc(time,vel);');
% Calculate road power menu
uimenu(hpwrcalc,'label','Road power',...'call', '[c1,c2,c3,c4]=findcoef;Proad=prdfind(time,vel,grade,c1,c2,c3,'c4);');
% Calculate shift points menu
uimenu(hpwrcalc,'label','Shift points',...'call',[Wmot=zeros(1,size(vel,2));'
    'gear=zeros(1,size(vel,2));'
    for cnt=1:size(vel,2)'
    [Wmot(cnt),gear(cnt)]=motspd(ve
    l(cnt)); end']);
% Run main calculation loop menu
hpwrrn=uimenu('parent',hpwrcalc,'label','Run');
uimenu(hpwrrn,'label','Tracking APU','call','mnloop1;');
uimenu(hpwrrn,'label','Exp APU','call','mnloop2;');

% Plot variable list (creates a check mark beside selected variable)
hpwrwin=uimenu('Parent',hpwr,'Label','Select Vars'); %Window pulldown
menvel=uimenu(hpwrwin,'Label','Com Velocity','call','fvel=-fvel;menctr1');
menacc=uimenu(hpwrwin,'label','Acceleration','call','facc=-facc;menctr1');
mengra=uimenu(hpwrwin,'label','Com Grade','call','fgra=-fgra;menctr1');
menppa=uimenu(hpwrwin,'label','Ppack','separator','on',...
    'call','fppa=-fppa;menctr1');
menpap=uimenu(hpwrwin,'label','Papu','call','fpap=-fpap;menctr1');
Mnwin.m

menpro=uimenu(hpwrwin,'label','Proad','call','fpro=-fpro;menctrl'};
menpfa=uimenu(hpwrwin,'label','Pfan','call','fpfa=-fpfa;menctrl'};
menpbr=uimenu(hpwrwin,'label','Pbrake','call','fpbr=-fpbr;menctrl'});
menepa=uimenu(hpwrwin,'label','Epack','call','fepa=-fepa;menctrl'};
soc';
menmotls=uimenu(hpwrwin,'label','Motor loss','call','fmls=-fmls;menctrl'});
menrvls=uimenu(hpwrwin,'label','Drive loss','call','fdls=-fdls;menctrl'});
mengea=uimenu(hpwrwin,'label','Gear shifts','call','fgea=-fgea;menctrl'};
menwmo=uimenu(hpwrwin,'label','Motor spd [rev/s]','call','fwmo=-fwmo;menctrl'});

menmotI=uimenu(hpwrwin,'label','Motor Current','call','fmtI=-fmtI;menctrl'};
menmotV=uimenu(hpwrwin,'label','Motor Voltage','call','fmtV=-fmtV;menctrl'});
menmotS=uimenu(hpwrwin,'label','Motor Slip','call','fmSp=-fmSp;menctrl'});

%Create graphics pull-down
hpwrplot=uimenu('parent',hpwr,'label','Graphics'});
%Plot all selected variables.
pltn1=uimenu(hpwrplot,'label','Plot Variables','call','plotall'});
%Plot all selected variables normalized to magnitude one.
pltn2=uimenu(hpwrplot,'label','Plot Normalized','call','plotnm'});

%print all selected variables.
prnt1=uimenu(hpwrplot,'label','Print Variables','call','printall'});
%print selected variables normalized to magnitude one.
prnt2=uimenu(hpwrplot,'label','Print Normalized','call','prntnm'});
%Write variables to file
prnt3=uimenu(hpwrplot,'label','Print to file','call','tofl'});

%Create controls windows-----------------
hpwrct=uimenu('parent',hpwr,'label','Controls'});
%create general controls window.
ctgen=uimenu(hpwrct,'label','General','Callback','genctrls'});
%create vehicle parameters window.
ctcar=uimenu(hpwrct,'label','Vehicle','Callback','carctrls'});
%create motor parameters window.
ctmot=uimenu(hpwrct,'label','Motor','Callback','motctrls'});
%create output window.
ctout=uimenu(hpwrct,'label','Output','Callback','outctrls'});
%create gear parameters window.
ctgear=uimenu(hpwrct,'label','Gear Ratios','Callback','gearbox'});
Mnwin.m

%create toolbar window.
%cttool=uimenu(hpwrct,'label','Toolbar','Callback','rnctrls');

%Create Steady-State pull-down.
hpwrstdy=uimenu('parent',hpwr,'label','Steady State');
%Create steady-state parameters window menu command.
uimenu(hpwrstdy,'label','Input Screen','call','chngstdy;');
%Run steady state simulation
uimenu(hpwrstdy,'label','Run','call','stdyst;');
%Select variables to plot
uimenu(hpwrstdy,'label','Select vars','call','stdypvar;');
%plot steady-state variables.
uimenu(hpwrstdy,'label','Plot Variables','call','Plt_st;');
%print variables to file
uimenu(hpwrstdy,'label','Print to file','call','tofl_st;');
Findacc.m

%findacc --- This file finds the
%acceleration vector

function [acc]=findacc(k,time,vel)

acc=(vel(k)-vel(k-1))/(time(k)-time(k-1));

return;
Drvloss.m

%drvloss --- Calculates electronic drive loss. The drive is assumed to be a three phase drive using six IGBTs.

function [Ploss]=drvloss(Irms)

%Global variables
global IGBTVdrop; %Forward voltage drop
global IGBTRes; %On resistance

Ipeak=Irms*sqrt(2);

%loss per igbt
Pigbt=2*Ipeak/pi*IGBTVdrop+Ipeak.^2/2*IGBTRes/pi;

Ploss=6*Pigbt; %Six IGBTs
return;

The IGBTs are modeled as

\[ \text{IGBT} \text{On} | \text{IGBTVdrop} \]

The reverse diode is modeled identically.
Deltae.m
% deltaE --- Calculates re-rated energy out
% of back with a reference of Whrref

function [delE]=deltaE(Eold,P,tinc,Whrref,Bnum)
Pb=P/Bnum;
Whr=(7e-6)*Pb^2-.0904*abs(Pb)+273.1;

delE=(Whrref/Whr)^sign(P)*P*tinc;
return;

% I have the sign function present to make sure the
% the charging is de-rated as well as the dis-
% charging. When dis-charging, we want to derate
% by making the delE bigger (delE is what is subtracted
% from the Epack). For charging, we want the delE
% to be smaller since it is added. This creates
% a de-rating in both directions.
% batt model
function [Vsupplied, genls] = batmod(soc, Imot, Papu)

global IGBTRes; % IGBT on resistance
global IGBTVdrop; % IGBT forward drop
global Bnum; % Number of 12V batteries

% Find Vbat & Rbat using SOC
if (soc >= .39)
    vbat = 4/3 * (soc -.4) + 11.8;
    if (soc >= .8)
        Rbat = (soc -.8) * .5 + .182;
    else
        Rbat = .182;
    end
else
    vbat = 14.5 * soc + 6;
    Rbat = (.4 - soc) * 2 + .182;
end

Igen = -(Bnum * vbat - abs(Imot) * Rbat) / 2 / Rbat + .5 * sqrt((Bnum * vbat / Rbat - abs(Imot)) ^ 2 + 4 * Papu / 3 / Rbat);
genls = Igen^2 * .25 + 12 * sqrt(2) * abs(Igen) * 1 / pi + 6 * Igen^2 * .001;

% Peak dc voltage supplied to inverter
Vsupplied = Bnum * vbat - (Imot - Igen) * sqrt(2) * (IGBTRes + Rbat) - IGBTVdrop;
Aputc.m

% aputc.m --- APU transfer characteristic
% Takes the APU time constant, a reference,
% might be average road power, a time increment,
% an initial power,
% and returns the new APU output value.
% The APU response is modeled as a single
% time constant response.

function [APUout]=aputc(tinc,tcons,ref,initpow)
tcons=1/tcons; %Invert time constant for transfer function
yold=initpow; %initial conditions

%Calculate ynew from transfer function
APUout=(yold+tcons*ref*tinc)/(1+tcons*tinc);

return;
% tranls --file computes the transmission loss

function [loss]=tranls(motspd,gear,proad)

motspd=motspd*60; %need it in RPM

if gear==1
    pnld=.177*motspd+34.6;
    loss=pnld+.05*(proad-pnld);
elseif gear==2
    pnld=.139*motspd;
    loss=pnld+.05*(proad-pnld);
elseif gear==3
    pnld=.2704*motspd-87.7;
    loss=pnld+.05*(proad-pnld);
elseif gear==4
    pnld=.354*motspd-90;
    loss=pnld+.05*(proad-pnld);
elseif gear==5
    pnld=1.41*motspd+163;
    loss=pnld+.05*(proad-pnld);
end
Findcoef.m

% File to find coefficients to road power equation
% Given the parameters
% Air density [kg/m^3]
% Vehicle frontal area [ft^2]
% Drag coefficient
% Tire loss coefficient
% Vehicle and payload weight [lb]

function [c1,c2,c3,c4]=FindCoef

global Aden; % Air density [kg/m^3]
global Afront; % Frontal area [ft^2]
global Cdrag; % Drag coefficient
global Tloss; % Tire-loss coefficient
global Vmass; % Vehicle mass [lb]
global Pmass; % Payload mass [lb]
global BattWt; % Weight of one battery [lb]
global Bnum; % Number of batteries

PackWt=BattWt*Bnum; % Battery pack weight

% convert frontal area from ft^2 to m^2
Af=.092903*Afront;

% convert weights to kg
Vm=.4535924*Vmass;
Pm=.4535924*Pmass;

% Calculate road power coefficients
 c1=.5*Aden*Cdrag*Af;
 c2=Tloss*(Vm+Pm+PackWt)*9.81;
 c3=(Vm+Pm+PackWt)*9.81;
 c4=(Vm+Pm+PackWt);
 return;
%function to find road power/ file PrdFind
%using the coefficients found in file
%findcoef.m

function [roadpower]=PrdFind(k,time,vel,grade,c1,c2,c3,c4)
acc=findacc(k,time,vel);
roadpower=c1*vel(k).^3+c2.*vel(k)+c3.*grade(k).*vel(k)+c4.*acc.*vel(k);
return;
%******************Power Tracking APU***********************
%Flags user if system fails
global hpwr; %Make main window global

percomp=0.0;

% Set up wait indicator
%************************************************************************
waitbut=figure('visible','off');
set(waitbut,'units','normalized','pos',[.4 .4 .4 .2]);
set(waitbut,'color','red');
set(waitbut,'menubar','none');
set(waitbut,'NumberTitle','off');
set(waitbut,'name','Calculating.')
butax=axes;
set(butax,'visible','off');
set(waitbut,'visible','on');
text('HorizontalAlignment','center','pos',[.45 .85],...
'fontsize',36,'fontWeight','bold',...
'string','Wait',...
'color','black');

mnwaitcon = uicontrol('units','normalized','pos',[.6 .4 .05 .1],
'style','edit',...
'horiz','left','string',num2str(percomp),...
'backgroundcolor','red',...
'clipping','off','parent',waitbut);
uicontrol('units','normalized','pos',[.3 .1 .4 .2],
'style','frame',...
'backgroundcolor','yellow');

mnwaitbar=uicontrol('units','normalized','pos',[.3 .1 .001 .2],
'style','frame',...
'backgroundcolor','cyan');
text('units','normalized','fontsize',11,'HorizontalAlignment',...
'right','position',[.6 .41],...
'color','black','string','Percent completed:');
text('units','normalized','fontsize',11,'position',[.65 .41],...
'color','black','string',' %');

%************************************************************************
pause(1);

% define global variables
global Ppack;global Epack;global Papu;global soc;

[c1,c2,c3,c4]=findcoef; %Find road power coefficients

Eptot=1.0872e6*Bnum; %Find total batter energy referenced to 6 hour discharge
Whrref=302; %Reference W-hr rating to 6 hour value

lngth=size(vel,2); %general vector length
Mnloop1.m

% Set up initial conditions
Ps1=0;Es=0.8;Poffs=0;

% Define empty vectors
motorls=zeros(1,lngth);drivels=zeros(1,lngth);acc=zeros(1,lngth);
Papu=zeros(1,lngth);Proad=zeros(1,lngth);Ppack=zeros(1,lngth);
soc=zeros(1,lngth);Wmot=zeros(1,lngth);gear=zeros(1,lngth);
Epack=zeros(1,lngth);Epack(1)=Es*Eptot;Epack(2)=Es*Eptot;Epack(3)=Es*Eptot;
MotCurrent=zeros(1,lngth);MotVoltage=zeros(1,lngth);
Slip=zeros(1,lngth);Pfan=zeros(1,lngth);PBrake=zeros(1,lngth);
velnew=vel;Proad=zeros(1,lngth);genls=zeros(1,lngth);
soc(1)=Es;soc(2)=Es;soc(3)=Es;
Papu(1)=Ps1;Papu(2)=Ps1;Papu(3)=Ps1;

% Find road power vector
[c1,c2,c3,c4]=FindCoef;

xx=0;wtinc=lngth/10;
cntflag=0;
vflag=1;
flagL=0;
flagH=1;

cnt=3;
while cnt<=(lngth-2) %Start of main loop
  Proad(cnt)=PrdFind(cnt,time,velnew,grade,c1,c2,c3,c4);
  acc(cnt)=findacc(cnt,time,velnew); %Calculate acceleration vector
if Proad(cnt)<-40000 %this braking routine has been modified
  PBrake(cnt)=40000+Proad(cnt); %to find actual motor peak torque
end

% Routine to update wait indicator
xx=xx+1;
if xx>=wtinc
  percomp= cnt./size(vel,2)*100;
  set(mnwaitcon,'string',num2str(round(percomp)));
  barsize=percomp*.4/100;
  set(mnwaitbar,'pos',[.3 .1 barsize .2]);
drawnow;
  xx=0;
end
Mnloop1.m

[Wmot(cnt), gear(cnt)] = motspd(velnew(cnt)); % find motor speed [rev/sec] & gear

Pfan(cnt) = FanC * (Wmot(cnt) * 60).^2; % Find fan power dissipation

% Find motor loss, current, slip, and voltage
[motorls(cnt), MotCurrent(cnt), Slip(cnt), MotVoltage(cnt)] = motloss(2*pi*Wmot(cnt), Proad(cnt) - PBrake(cnt) + Pfan(cnt), PoleNum/2);

[Vs(cnt), genls(cnt)] = batmod(soc(cnt-l), MotCurrent(cnt), Papu(cnt-1));

% Increase velocity if it can be done
if (abs(Vs(cnt)) > abs(MotVoltage(cnt) * sign(Proad(cnt))) * sqrt(2) * sqrt(3)) & vflag == 1 & (vel(cnt) > velnew(cnt))
    velnew(cnt) = velnew(cnt) * 1.02 + .02;
end

% Decrease velocity if it can’t be held
elseif (abs(Vs(cnt)) < abs(MotVoltage(cnt) * sign(Proad(cnt))) * sqrt(2) * sqrt(3)) & acc(cnt) >= 0
    vflag = 0;
    velnew(cnt) = velnew(cnt) * .98;
    velnew(cnt+1) = velnew(cnt);
else
    vflag = 1;

end

 drivels(cnt) = drvloss(abs(MotCurrent(cnt))); % Find IGBT drive loss

% Find power out of pack
Ppack(cnt) = Proad(cnt) + motorls(cnt) + drivels(cnt) - Papu(cnt-1) + 250;
Ppack(cnt) = Ppack(cnt) + PBrake(cnt) + Pfan(cnt) + genls(cnt) + tranls(Wmot(cnt), gear(cnt), Proad(cnt));

% Make sure battery power capability isn’t exceeded
if Ppack(cnt) > (MaxPB * Bnum)
    disp('Exceeded battery power limit');
    break;
end

% Find new pack energy
Epack(cnt+1) = Epack(cnt) - deltaE(Epack(cnt), Ppack(cnt), (time(cnt+1) - time(cnt)), Whrref, Bnum);
if Epack(cnt+1) > Eptot
    Epack(cnt+1) = Eptot;
end

% Update state of charge
soc(cnt) = Epack(cnt) / Eptot;

% Find average power demanded
% SOCH sets the equilibrium value of battery soc
Poffs = Poffs + 250 * (SOCH - soc(cnt));
Mnloop1.m

pav=Proad(cnt)+motorls(cnt)+drivels(cnt)+PBrake(cnt)+genls(cnt)+Pfan(cnt)+Poffs;
if soc(cnt)<SOCL
  flagL=1;flagH=0;
end
%5kW minimum on APU output
if soc(cnt)>SOCH
  flagH=1;flagL=0;
end
if pav<5000 & flagL==1
  pav=5000;
end
if pav<5000 & flagH==1
  pav=0;
end
%Set apu output accordingly
Papu(cnt+1)=aputc((time(cnt+1)-time(cnt)),tcons,pav,Papu(cnt));
if Papu(cnt+1)>PAPU
  Papu(cnt+1)=PAPU;
end
if Papu(cnt+1)<0
  Papu(cnt+1)=0;
end
cnt=cnt+1;
velnew(cnt)=velnew(cnt-1);
end

if velnew(cnt)>vel(cnt)
  velnew(cnt)=vel(cnt);
end
end %End of main loop

close;
Mnloop2.m

%mnloop1****APU on/off is exponential (switching APU)
%Flags user if system fails

global hpwr; %Make main window global
percomp=0.0;

%Set up wait indicator
%
waitbut=figure('visible','off');
set(waitbut,'units','normalized','pos', [.4 .4 .4 .2]);
set(waitbut,'color','red');
set(waitbut,'menubar','none');
set(waitbut,'NumberTitle','off');
set(waitbut,'name','Calculating.');
butax=axes;
set(butax,'visible','off');
set(waitbut,'visible','on');
text('HorizontalAlignment','center','pos', [.45 .85],
    'fontsize',36,'fontweight','bold',
    'string','Wait',
    'color','black');

mnwaitcon = uicontrol('units','normalized','pos', [.6 .4 .05 .1],
                      'style','edit',
                      'horiz','left','string',num2str(percomp),
                      'bgcolor','red',
                      'clipping','off','parent',waitbut);
uicontrol('units','normalized','pos', [.3 .1 .4 .2],
                      'style','frame',
                      'bgcolor','yellow');
mnwaitbar=uicontrol('units','normalized','pos', [.3 .1 .001 .2],
                      'style','frame',
                      'bgcolor','cyan');
text('units','normalized','fontsize',11,'HorizontalAlignment',
       'right','position', [.6 .41],
       'color','black','string','Percent completed:');
text('units','normalized','fontsize',11,'position', [.65 .41],
       'color','black','string','%');

pause(1);

%Define global vectors

global Ppack;global Epack;global Papu;global soc;

%Find road power coefficients
[c1,c2,c3,c4]=findcoef;

Eptot=1.0872e6*Bnum; %Find total batter energy referenced to 6 hour di
                       scharge
Whrref=273.1; %Reference W-hr rating to 6 hour value

length=size(vel,2); %general vector length

Page 1
% Set up data vectors
motorls=zeros(1,length); drivels=zeros(1,length);
Papu=zeros(1,length); Proad=zeros(1,length); Ppack=zeros(1,length);
soc=zeros(1,length); Wmot=zeros(1,length); gear=zeros(1,length);
Epack=zeros(1,length); Epack(1)=.77*Eptot; Epack(2)=.77*Eptot; Epack(3)=.7*Eptot;
MotCurrent=zeros(1,length); MotVoltage=zeros(1,length);
Slip=zeros(1,length); Pfan=zeros(1,length); PBrake=zeros(1,length);
Edel=zeros(1,length);
velnew=vel; Proad=zeros(1,length);
soc(1)=1; soc(2)=1; soc(3)=1;

% Find road power vector
[c1,c2,c3,c4]=FindCoef;

xx=0; wtinc=length/10;
cntflag=0;
cntflag=0; temp=0; flagL=0; flagH=1; APUon=0; APU1=0;
cnt=3; vflag=1

% Start main loop
while cnt<= (length-2)

% Update wait indicator
xx=xx+1;
if xx>=wtinc;
percomp= cnt./size(vel,2)*100;
set(mnwaitcon, 'string', num2str(round(percomp)));
barsize=percomp*.4/100;
set(mnwaitbar, 'pos', [0 barsize 0.2]);
drawnow;
xx=0;
end

Proad(cnt)=PrdFind(cnt,time,velnew,grade,c1,c2,c3,c4);
acc(cnt)=findacc(cnt,time,velnew); % Calculate acceleration vector

% Guess at mechanical braking power
if ((acc(cnt)<0)&(vel(cnt)>7))
  PBrake(cnt)=brake(vel(cnt),Proad(cnt),acc(cnt));
else
  PBrake(cnt)=0;
end
Mnloop2.m

[Wmot(cnt),gear(cnt)]=motspd(velnew(cnt)); %find motor speed[rev/sec] & gear
Pfan(cnt)=FanC*(Wmot(cnt)*60).^2; %Find fan power dissipation
&Pfan(cnt)
%Find motor loss, current, slip and voltage
[motorls(cnt),MotCurrent(cnt),Slip(cnt),MotVoltage(cnt)]=motloss(2*pi*Wmot(cnt),Proad(cnt)-PBrake(cnt)+Pfan(cnt),PoleNum/2);

[Vs(cnt),genls(cnt)]=batmod(soc(cnt-l),MotCurrent(cnt),Papu(cnt-l));

if (abs(Vs(cnt))>abs(MotVoltage(cnt)*sign(Proad(cnt)))*sqrt(2)*sqrt(3))
velnew(cnt)=velnew(cnt)*1.02+.02;
else
velnew(cnt+1)=velnew(cnt);
end

elseif (abs(Vs(cnt))<abs(MotVoltage(cnt)*sign(Proad(cnt)))*sqrt(2)*sqrt(3))
velnew(cnt)=velnew(cnt)*.98;
else
vflag=1;
end

[drlvls(cnt)]=drvloss(abs(MotCurrent(cnt))); %Find IGBT drive loss
Ppack(cnt)=Proad(cnt)+motorls(cnt)+drlvls(cnt)-Papu(cnt-l)*APU1+250;
%APU1 is flag first APU on.
Ppack(cnt)=Ppack(cnt)+PBrake(cnt)+Pfan(cnt)+tranls(Wmot(cnt),gear(cnt),Proad(cnt));

%Make sure battery power capability isn’t exceeded
if Ppack(cnt)>(MaxPB*Bnum)
disp('Exceeded battery power limit');
break;
end

%Find new pack energy
Epack(cnt+1)=Epack(cnt)-deltaE(Epack(cnt),Ppack(cnt),(time(cnt+l)-time(cnt)),Whrref,Bnum);
if Epack(cnt+1)>Eptot
Epack(cnt+1)=Eptot;
end
Mnloop2.m

Edel(cnt+1)=-deltaE(Epack(cnt),Ppack(cnt),(time(cnt+1)-time(cnt)),Whref,Bnum);
%Turn on APU if road power>20kW for more than 30 sec
if Ppack(cnt)>200000
  T20=T20+1;
  T20a=T20a-30;
  s20a=sign(T20a);
  if s20a>=0
    T20a=1;
  else
    T20a=0;
  end
else
  T20=0;T20a=0;
end

%Find SOC conditions
if ((Epack(cnt)/Eptot<=SOCL)&(flagL==0)) | ((T20a>0)&(Epack(cnt)/Eptot<=0.7))&(flagL==0))
  temp=cnt;APUon=1;flagL=1;flagH=0;APU1=1;cntup=cnt;
end
if ((Epack(cnt)/Eptot>=SOCH)&(flagH==0))
  temp=cnt;APUon=0;flagH=1;flagL=0;cntdn=cnt;
end

soc(cnt)=Epack(cnt)/Eptot;

%Act on SOC conditions
if APUon==1
  Papu(cnt+1)=aputc((time(cnt+1)-time(cnt)),tcons,PAPU,Papu(cnt));
else
  Papu(cnt+1)=aputc((time(cnt+1)-time(cnt)),tcons,0,Papu(cnt));
end

cnt=cnt+1;
velnew(cnt)=velnew(cnt-1);
if velnew(cnt)>vel(cnt)
  velnew(cnt)=vel(cnt);
end

end %end of main loop

close;
Motloss.m

% This function is used with f.m to solve the induction
% motor model for a given shaft speed and power.

function [loss,Iin,s,Vin]=motloss(Wm_slv,power,poles)

global Tmax; %Linear model maximum motor torque
global StatRes;
global StatInd;
global RotRes;
global RotInd;
global LeakRes;
global LeakInd;
global Wm;
global power_slv;
global poles;
power_slv=power;
Wm=Wm_slv;
r1=StatRes;r2=RotRes;
rs=LeakRes;
l1=StatInd;l2=RotInd;
l1=LeakInd;

%input voltag is a linear with excitation frequency
% plus a constant offset
k=.177;voff=1.2;

if ((abs(power)<10)||(abs(Wm_slv)<.1)) %vel condition to allow use of
    loss=0;Iin=0;Vin=0;s=0; %of actual road data.
    return;
end

if power/Wm>Tmax
    disp('Motor torque maximum exceeded')
end

%solve model
s=fzero('f',0,0.00001);

Ws=poles*Wm/(1-s);
if Ws<2*pi
    Ws=2*pi;
end

%solve for input current, a phasor input voltage %and the losses
I2=sqrt(abs(s)*abs(power_slv/3)/r2/abs(1-s));
Motloss.m

\[ \text{Vin} = I_2 \cdot \left( r_1 + \frac{r_2}{\text{abs}(s)} \right) + j \cdot W_s \cdot \left( l_1 + l_2 \right) \]

\[ I_{\text{in}} = \frac{\text{Vin} \cdot r_2 \cdot j \cdot W_s \cdot l_s}{j \cdot W_s \cdot l_s + r_s} + I_2 \]

\[ \text{loss} = 3 \cdot \text{abs}(\text{Vin})^2 / r_s + 3 \cdot I_2^2 \cdot (r_1 + r_2) \]
% Function used by fzero to solve induction motor model
function y=f(s)

global StatRes; %stator leakage resistance
global StatInd; %stator leakage inductance
global RotRes; %rotor leakage resistance
global RotInd; %rotor leakage inductance
global LeakRes; %shunt resistance
global LeakInd; %shunt inductance
global Wm; %motor speed
global power_slv; %shaft power
global poles; %number of pole pairs
global motmodflg; %motor model flag

r1=StatRes;r2=RotRes;
rs=LeakRes;
l1=StatInd;l2=RotInd;
l=LeakInd;

k=.102;voff=2.8;
%k=.177;voff=1.2;

%*****Actual motor model***************
if (motmodflg==1)
N=3*(k*poles*Wm./(1-s)+voff).^2*r2.*s*Wm.*(1-s).^3;
D=Wm*((s*r1+r2).^2.*(1-s).^2+s.^2*poles^2*Wm^2*(l1+l2).^2);
y=power_slv-N/D;
end

%*****R2/s approximation***************
if (motmodflg==0)
y=power_slv-3*(k*poles*Wm./(1-s)+voff).^2.*s.*(1-s)/r2;
end

return;
```matlab
function [speed,gear]=motspd(velocity)

%Set shift globals
global Gear1Ratio;  %gear ratios
global Gear2Ratio;
global Gear3Ratio;
global Gear4Ratio;
global Gear5Ratio;
global Shift1Vel;    %shift velocities [m/s]
global Shift2Vel;
global Shift3Vel;
global Shift4Vel;
global Shift5Vel;
global Motorshft;
global WheelRad;
global shftflg;     %1=vehicle speed, 0=motor speed

%Find constant that relates vehicle speed [m/s] to
%revolutions/sec for wheel.
wheelrev=1/(2*3.1415*WheelRad);

if shftflg==1 %Shift at vehicle speed points.
    Wrevpers=wheelrev*velocity;
end

%Convert mph shift speeds to m/s
sv1=Shift1Vel*0.44704;
sv2=Shift2Vel*0.44704;
sv3=Shift3Vel*0.44704;
sv4=Shift4Vel*0.44704;
sv5=Shift5Vel*0.44704;

if velocity<=sv4
    speed=Wrevpers*Gear4Ratio;
gear=4;
end

if velocity<=sv3
    speed=Wrevpers*Gear3Ratio;
gear=3;
end

if velocity<=sv2
    speed=Wrevpers*Gear2Ratio;
gear=2;
end
```

Motspd.m

%motspd --- give a velocity [m/s] of the car
% and the global shift point variables, and the
% tire radius, WheelRad, and the gear ratios
% (from wheel to motor) this function returns
% the gear and the motor speed [rev/sec].
speed=Wrevpers\*Gear2Ratio;
gear=2;
end

if velocity<=sv1
    speed=Wrevpers\*Gear1Ratio;
gear=1;
end

if velocity>sv4
    speed=Wrevpers\*Gear5Ratio;
gear=5;
end
end

if shftflg==0  %Shift at motor speed points.
motref=Motorshft/60;  %Convert motor speed shftpoint to rev/sec.

%Find the motor speed for each gear ratio
speed1=wheelrev\*velocity\*Gear1Ratio;
speed2=wheelrev\*velocity\*Gear2Ratio;
speed3=wheelrev\*velocity\*Gear3Ratio;
speed4=wheelrev\*velocity\*Gear4Ratio;
speed5=wheelrev\*velocity\*Gear5Ratio;

%Initialize with gear 5
speed=speed5;
gear=5;

if speed4<=motref
    speed=speed4;
gear=4;
end

if speed3<=motref
    speed=speed3;
gear=3;
end

if speed2<=motref
    speed=speed2;
gear=2;
end

if speed1<=motref
    speed=speed1;
gear=1;
end
end
% Global variables for plot routine
% velocity sweep vector
global stdyvel;
global stdygrade;
global stdypacksz;
global pack_loss_st;
global mileage_st;
global Road_pwr; % road power
global motls_st; % motor loss
global drvls_st; % drive loss
global motcur_st; % motor line current
global motvol_st; % motor terminal voltage
global slip_st; % motor slip
global Ptot_st; % total vehicle power
global Pfan_st; % fan power
global Wmot_st; % motor speed
global gear_st; % current gear ratio
global APU_on_time;
global APU_off_time;
global APU_period; % APU switching period
global BattWt; % battery weight
global Bnum; % number of batteries
global FanC; % fan constant
global FEconversion; % fuel conversion efficiency
percomp=0.0;
Bnum2=Bnum; % Save dynamic value

% Set up wait indicator

waitbut=figure('visible','off');
set(waitbut,'units','normalized','pos',[.4 .4 .4 .2]);
set(waitbut,'color','red');
set(waitbut,'menubar','none');
set(waitbut,'NumberTitle','off');
set(waitbut,'name','Calculating.'
)
butax=axes;
set(butax,'visible','off');
set(waitbut,'visible','on');
text('HorizontalAlignment','center','pos',[.45 .85],...
'fontsize',36,'fontweight','bold',...
'string','Wait',...
'color','black');

mnwaitcon = uicontrol('units','normalized','pos',[.6 .4 .05 .1],'style','edit',...
'horiz','left','string',num2str(percomp),...
'backgroundcolor','red',...
'clipping','off','parent',waitbut);
uicontrol('units','normalized','pos',[.3 .1 .4 .2],'style','frame',...
### Stdyst.m

```matlab
backgroundcolor', 'yellow');

mnwaitbar=uicontrol('units','normalized','pos', [.3 .1 .001 .2], 'style',
'frame', ...
'backgroundcolor', 'cyan');
text('units', 'normalized', 'fontsize', 11, 'HorizontalAlignment', ...
'right', 'position', [.6 .41],...
'color', 'black', 'string', 'Percent completed:');
text('units', 'normalized', 'fontsize', 11, 'position', [.65 .41],...
'color', 'black', 'string', ' %');

%****************************************************
pause(1);

%**********Start of main loop**********************

if Vary_type_flag==1 % Velocity is varied

% Set up velocity vector and find length
stdyinc=(stdyEndvel-stdyInivel)/Num_stdy_pts;
stdyvel=(stdyInivel+stdyinc):stdyinc:(stdyEndvel+stdyinc);
stdylen=size(stdyvel,2);

%********Wait indicator setup********************
xx=0;wtinc=stdylen/10;
cntflag=0;

Road_pwr=zeros(1,stdylen);motls_st=zeros(1,stdylen);
drvels_st=zeros(1,stdylen);motcur_st=zeros(1,stdylen);
slip_st=zeros(1,stdylen);motvol_st=zeros(1,stdylen);
Ppack_st=zeros(1,stdylen);mileage_st=zeros(1,stdylen);
pack_loss_st=zeros(1,stdylen);
APU_on_time=zeros(1,stdylen);
APU_off_time=zeros(1,stdylen);
APU_period=zeros(1,stdylen);
Pfan_st=zeros(1,stdylen);
Ptot_st=zeros(1,stdylen);
gear_st=zeros(1,stdylen);
Wmot_st=zeros(1,stdylen);

Whrref=302; %Reference W-hr based on 6hr discharge rate (per battery)
Nom_cap=302*3600; %Nominal capacity per battery

grade_st=stdyInigrade*ones(1,stdylen); %grade vector

Bnum=(stdyInipack/Nom_cap); %New number of batteries
[c1,c2,c3,c4]=FindCoef; %Find road power coefficients

if APU_type_flag==1 %Ideal switching APU

for cnt=1:stdylen
```

Page 2
%*******************Routine to update wait indicator*******************
******
xx=xx+1;
if xx>=wtinc
percomp= cnt./stdylen*100;
set(mnwaitcon, 'string', num2str(round(percomp)));
barsize=percomp*.4/100;
set(mnwaitbar, 'pos', [.3 .1 barsize .2]);
drawnow;
xx=0;
end
%*********************************************************************
******

Road_pwr(cnt)=Proad_st(stdyvel(cnt),grade_st(cnt),c1,c2,c3,c4); %find road power
[Wmot_st(cnt),gear_st(cnt)]=motspd(stdyvel(cnt)); %Find motor speed and gear
%Find motor loss, current, slip and voltage
[motls_st(cnt),motcur_st(cnt),slip_st(cnt),motvol_st(cnt)]=motloss(2*pi*Wmot_st(cnt),Road_pwr(cnt),PoleNum/2);
[drvls_st(cnt)]=drvloss(abs(motcur_st(cnt))); %Find IGBT drive loss
Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;

Ppack_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+drvls_st(cnt)+Pfan_st(cnt)+t ranls(Wmot_st(cnt),gear_st(cnt),Road_pwr(cnt)); %Power out of pack
Ptot_st(cnt)=Ppack_st(cnt);

Whrnew_st1=Whr_st(Ppack_st(cnt)/(stdyInipack/Nom_cap),Nom_cap);
Epack_dr_st=stdyInipack*Whrnew_st1/Whrref;

if (Ppack_st(cnt)>=PAPU)&&(Epack_dr_st<0)
    stdylen=cnt-1;
    break;
end

APU_off_time(cnt)=Epack_dr_st*(SOCH-SOCL)/Ppack_st(cnt);

%Find APU on time
Whrnew_st2=Whr_st(abs(PAPU-Ppack_st(cnt))/(stdyInipack/Nom_cap),Nom_cap);
Epack_dr_st=stdyInipack*Whrref/Whrnew_st2;
if Epack_dr_st<0
    stdylen=cnt-1;
    break;
end

APU_on_time(cnt)=Epack_dr_st*(SOCH-SOCL)/(PAPU-Ppack_st(cnt));
APU_period(cnt)=APU_off_time(cnt)+APU_on_time(cnt); %Apu cycling period
q1=APU_off_time(cnt)/APU_period(cnt);q2=1-q1;
pack_loss_st(cnt)=(q1*Whrnew_st1+q2*Whrnew_st2)/Whrref*100;
velmph=stdyvel(cnt)*2.23694;
mileage_st(cnt)=velmph/(APU_on_time(cnt)/APU_period(cnt)*PAPU)/3600*FEconversion*1e6;
end % (end for) End of velocity main loop
end % if/then switching APU
if APU_type_flag==2 %Tracking APU
for cnt=1:stdylen
    %***************Routine to update wait indicator***************
    xx=xx+1;
    if xx>=wtinc
        percomp= cnt./stdylen*100;
        set(mnwaitcon,'string',num2str(round(percomp)));
        barsize=percomp*.4/100;
        set(mnwaitbar,'pos',[.3 .1 barsize .2]);
        drawnow;
        xx=0;
    end
    %*********************************************************************
end

Road_pwr(cnt)=Proad_st(stdyvel(cnt),grade_st(cnt),c1,c2,c3,c4); %find road power
[Wmot_st(cnt),gear_st(cnt)]=motspd(stdyvel(cnt)); %Find motor speed and gear
%Find motor loss, current, slip and voltage
[motls_st(cnt),motcur_st(cnt),slip_st(cnt),motvol_st(cnt)]=motloss(2*P*i*Wmot_st(cnt),Road_pwr(cnt),PoleNum/2);
drvls_st(cnt)=drvloss(abs(motcur_st(cnt))); %Find IGBT drive loss
Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;
Ptot_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+drvls_st(cnt)+Pfan_st(cnt)+trans(Wmot_st(cnt),gear_st(cnt),Road_pwr(cnt)); %Total power demand
if Ptot_st(cnt)>=PAPU %Can’t hold this value
    stdylen=cnt-1;
    break;
end
pack_loss_st(cnt)=100;
velmph=stdyvel(cnt)*2.23694;
mileage_st(cnt)=velmph/(Ptot_st(cnt))/3600*FEconversion*1e6;
end %for/next

Page 4
if Vary_type_flag==2
if stdyInivel==0
    disp('Velocity is zero.');
    close(waitbut);
    return;
end

%Set up grade vector
stdyinc=(stdyEndgrade-stdyInigrade)/Num_stdy_pts;
stdygrade=stdyInigrade:stdyinc:stdyEndgrade;
stdylen=size(stdygrade,2);

%**********Wait indicator setup******************
xx=0;wtinc=stdylen/10;
cntflag=0;
%***********************************************

Road_pwr=zeros(1,stdylen);motls_st=zeros(1,stdylen);
drvis_st=zeros(1,stdylen);motcur_st=zeros(1,stdylen);
slip_st=zeros(1,stdylen);motvol_st=zeros(1,stdylen);
Ppack_st=zeros(1,stdylen);mileage_st=zeros(1,stdylen);
pack_loss_st=zeros(1,stdylen);
APU_on_time=zeros(1,stdylen);
APU_off_time=zeros(1,stdylen);
APU_period=zeros(1,stdylen);
Pfan_st=zeros(1,stdylen);
Ptot_st=zeros(1,stdylen);
gear_st=zeros(1,stdylen);

Whrref=302; %Reference W-hr based on 6hr discharge rate (per battery)
Nom_cap=302*3600; %Nominal capacity per battery
stdyvel=stdyInivel*ones(1,stdylen); %Velocity vector
velmph=stdyInivel*2.23694;

Bnum=(stdyInipack/Nom_cap); %New number of batteries
[c1,c2,c3,c4]=FindCoef; %Find road power coefficients

if APU_type_flag==1 %Switching APU
for cnt=1:stdylen

%*******************Routine to update wait indicator*******************
******
xx=xx+1;
if xx>=wtinc
percomp= cnt./stdylen*100;
set(mnwaitcon,'string',num2str(round(percomp)));
barsize=percomp*.4/100;
set(mnwaitbar,'pos',[.3 .1 barsize .2]);
drawnow;
xx=0;
end
%*********************************************************************
******

Road_pwr(cnt)=Proad_st(stdyvel(cnt),stdygrade(cnt),c1,c2,c3,c4);
[Wmot_st(cnt),gear_st(cnt)]=motspd(stdyInivel); %Find motor speed and gear
[motls_st(cnt),motcur_st(cnt),slip_st(cnt),motvol_st(cnt)]=motloss(2*pi*Wmot_st(cnt),Road_pwr(cnt),PoleNum/2);
[dvrls_st(cnt)]=drvloss(abs(motcur_st(cnt))); 
Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;

Ppack_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+dvrls_st(cnt)+Pfan_st(cnt)+trans(Wmot_st(cnt),gear_st(cnt),Road_pwr(cnt));
Ptot_st(cnt)=Ppack_st(cnt);

Whrnew_st1=Whr_st(Ppack_st(cnt)/(stdyInipack/Nom_cap),Nom_cap);
Epack_dr_st=stdyInipack*Whrref/Whrnew_st1;
if (Ppack_st(cnt)>=PAPU) & (Epack_dr_st<0)
    stdylen=cnt-1;
    break;
end

APU_off_time(cnt)=Epack_dr_st*(SOCH-SOCL)/Ppack_st(cnt);

Whrnew_st2=Whr_st(abs(PAPU-Ppack_st(cnt))/(stdyInipack/Nom_cap),Nom_cap);
Epack_dr_st=stdyInipack*Whrref/Whrnew_st1;
if Epack_dr_st<0
    stdylen=cnt-1;
    break;
end
APU_on_time(cnt)=Epack_dr_st*(SOCH-SOCL)/(PAPU-Ppack_st(cnt));
APU_period(cnt)=APU_off_time(cnt)+APU_on_time(cnt);

q1=APU_on_time(cnt)/APU_period(cnt);q2=1-q1;
pack_loss_st(cnt)=(q1*Whrnew_st1+q2*Whrnew_st2)/Whrref*100;
mileage_st(cnt)=velmph/(APU_on_time(cnt)/APU_period(cnt)*PAPU)\(\times\)3600*FE conversion*1e6;

end %Grade switching loop
end %end of if/then APU_type

if APU_type_flag==2 %Tracking APU
for cnt=1:stdylen

*********Routine to update wait indicator*********
xx=xx+1;
if xx>=wtinc
  percomp=cnt./stdylen*100;
  set(mnwaitcon,'string',num2str(\(\text{round}(\text{percomp})\)));
barsize=percomp*.4/100;
  set(mnwaitbar,'pos',[.3 .1 barsize .2]);
drawnow;
  xx=0;
end

*********
Road_pwr(cnt)=Proad_st(stdyInivel, stdygrade(cnt), c1, c2, c3, c4);
[Wmot_st(cnt), gear_st(cnt)]=motspd(stdyInivel); %Find motor speed and gear
[motls_st(cnt), motcur_st(cnt), slip_st(cnt), motvol_st(cnt)]=motloss(2*p i*Wmot_st(cnt), Road_pwr(cnt), PoleNum/2);
[drvls_st(cnt)]=drvloss(abs(motcur_st(cnt)));
Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;

Ptot_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+drvls_st(cnt)+Pfan_st(cnt); %Total power demand
if (Ptot_st(cnt)>=PAPU)\&\&(Epack_dr_st<=0) %Can’t hold this value
  stdylen=cnt-1;
  break;
end
pack_loss_st(cnt)=100;
velmph=stdyInivel*2.23694;
mileage_st(cnt)=velmph/(Ptot_st(cnt))/3600*FEconversion*1e6;
end %for/next
end % end of Tracking APU routine
end %If/then end Vary_type
if Vary_type_flag==3
if stdyInivel==0
    disp('Velocity is zero.');
    close(waitbut);
    return;
end

stdyinc=(stdyEndpack-stdyInipack)/Num_stdy_pts;
stdypacksz=stdyInipack:stdyinc:stdyEndpack;
stdylen=size(stdypacksz,2);

%***************Wait indicator setup***************
xx=0;wtinc=stdylen/10;
cntflag=0;
%************************************************

Road_pwr=zeros(1,stdylen);motls_st=zeros(1,stdylen);
drvls_st=zeros(1,stdylen);motcur_st=zeros(1,stdylen);
slip_st=zeros(1,stdylen);motvol_st=zeros(1,stdylen);
Ppack_st=zeros(1,stdylen);mileage_st=zeros(1,stdylen);
pack_loss_st=zeros(1,stdylen);
APU_on_time=zeros(1,stdylen);
APU_off_time=zeros(1,stdylen);
APU_period=zeros(1,stdylen);
Pfan_st=zeros(1,stdylen);
Ptot_st=zeros(1,stdylen);

Whrref=302; %Reference W-hr based on 6hr discharge rate (per battery)
Nom_cap=302*3600; %Nominal capacity per battery

if APU_type_flag==1
for cnt=1:stdylen

%******************************************Routine to update wait indicator******************************************

%********
xx=xx+1;
if xx>=wtinc
    percomp= cnt./stdylen*100;
    set(mnwaitcon, 'string',num2str(round(percomp)));
    barsize=percomp*.4/100;
    set(mnwaitbar, 'pos', [.3 .1 barsize .2]);
    drawnow;
    xx=0;
end
%******************************************

Bnum=(stdypacksz(cnt)/Nom_cap); %New number of batteries
```matlab
[\text{Stdyst.m}]

\text{\texttt{[c1,c2,c3,c4]=FindCoef; \%Find road power coefficients}}

\text{\texttt{Road_pwr(cnt)=Proad_st(stdyInivel,stdyInigrade,c1,c2,c3,c4);}}
\text{\texttt{[Wmot_st(cnt),gear_st(cnt)]=motspd(stdyInivel);}}
\text{\texttt{[motls_st(cnt),motcur_st(cnt),slip_st(cnt),motvol_st(cnt)]=motloss(2*pi*Wmot_st(cnt),Road_pwr(cnt),PoleNum/2);}}
\text{\texttt{[drvls_st(cnt)]=drvloss(abs(motcur_st(cnt)));}}
\text{\texttt{Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;}}
\text{\texttt{Ppack_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+drvls_st(cnt)+Pfan_st(cnt);}}
\text{\texttt{Ptot_st(cnt)=Ppack_st(cnt);}}
\text{\texttt{Whrnew_st1=Whr_st(Ppack_st(cnt))/(stdypacksz(cnt)/Nom_cap),Nom_cap);}}
\text{\texttt{Epack_dr_st=stdypacksz(cnt)*Whrnew_st1/Whrref;}}
\text{\texttt{if (Ppack_st>=PAPU) \&(Epack_dr_st<0)}}
\text{\texttt{stdylen=cnt-1;}}
\text{\texttt{break;}}
\text{\texttt{end}}
\text{\texttt{APU_off_time(cnt)=Epack_dr_st*(SOCH-SOCL)/Ppack_st(cnt);}}
\text{\texttt{Whrnew_st2=Whr_st(abs(PAPU-Ppack_st(cnt))/(stdypacksz(cnt)/Nom_cap),Nom_cap);}}
\text{\texttt{Epack_dr_st=stdypacksz(cnt)*Whrref/Whrnew_st2;}}
\text{\texttt{if Epack_dr_st<0}}
\text{\texttt{stdylen=cnt-1;}}
\text{\texttt{break;}}
\text{\texttt{end}}
\text{\texttt{APU_on_time(cnt)=Epack_dr_st*(SOCH-SOCL)/(PAPU-Ppack_st(cnt));}}
\text{\texttt{APU_period(cnt)=APU_off_time(cnt)+APU_on_time(cnt);}}
\text{\texttt{q1=APU_on_time(cnt)/APU_period(cnt);q2=1-q1;}}
\text{\texttt{pack_loss_st(cnt)=(q1*Whrnew_st1+q2*Whrnew_st2)/Whrref*100;}}
\text{\texttt{velmph=stdyInivel*2.23694;}}
\text{\texttt{mileage_st(cnt)=velmph/(APU_on_time(cnt)/APU_period(cnt)*PAPU)/3600*FE}}
\text{\texttt{conversion*1e6;}}
\text{\texttt{end \%End of battery loop}}
\text{\texttt{end \%switching APU}}
\text{\texttt{if APU_type_flag==2 \%Tracking APU}}
\text{\texttt{for cnt=1:stdylen}}
\text{\texttt{\%**************Routine to update wait indicator**************}}
```
**Stdyst.m**

```matlab
******
xx=xx+1;
if xx>=wtinc
  percomp= cnt./stdylen*100;
  set(mnwaitcon,'string',num2str(round(percomp)));
  barsize=percomp*.4/100;
  set(mnwaitbar,'pos',[.3 .1 barsize .2]);
  drawnow;
  xx=0;
end

%******************************************************************************

Bnum=(stdypacksz(cnt)/Nom_cap); %New number of batteries
[c1,c2,c3,c4]=FindCoef; %Find road power coefficients

Road_pwr(cnt)=Proad_st(stdyInivel,stdyInigrade,c1,c2,c3,c4);
[Wmot_st(cnt),gear_st(cnt)]=motspd(stdyInivel);
[motls_st(cnt),motcur_st(cnt),slip_st(cnt),motvol_st(cnt)]=motloss(2*pi*Wmot_st(cnt),Road_pwr(cnt),PoleNum/2);

[drvls_st(cnt)]=drvloss(abs(motcur_st(cnt)));
Pfan_st(cnt)=FanC*(Wmot_st(cnt)*60).^2;

Ptot_st(cnt)=Road_pwr(cnt)+motls_st(cnt)+drvls_st(cnt)+Pfan_st(cnt); %Total power demand

if (Ptot_st(cnt)>=PAPU)&&(Epack_dr_st<0) %Can't hold this value
  stdylen=cnt-1;
  break;
end
pack_loss_st(cnt)=100;
velmph=stdyInivel*2.23694;
mileage_st(cnt)=velmph/(Ptot_st(cnt))/3600*FEconversion*1e6;
end % end of Tracking APU routine
end %End of if/then APU_type
end %End of if/then Vary_type

Bnum=Bnum2; %Return original value
close(waitbut);
return;
```

Page 10
Whr_st.m
%function to calculate W-hr rating for steady
%state program

function [Whr]=Whr_st(power,nom_cap)
Whr=(7e-6)*power^2-.0904*power+273.1;
return;
Modeling, Simulation, and Experimental Validation for a Hybrid Electric Vehicle

Daniel Logue
University of Illinois
Department of Electrical and Computer Engineering

Chapter 1 Introduction

In recent years, there has been a strong movement toward the design of automobiles that are highly efficient, and at the same time, have a substantial reduction in tailpipe emissions as compared to contemporary vehicles. This movement is driven both by federal and state mandates and the willingness of the consumer to spend more for a vehicle that is environmentally friendly. Any replacement for today’s automobile must also be able to match its dynamic performance abilities. The hybrid electric vehicle (HEV) is capable of achieving all of these objectives. In addition, the HEV is not plagued by the purely electric vehicle’s range limitation.

However, new design tools and methods are needed to effectively design an HEV that is reliable and practical enough to hold the consumer’s interest. This thesis describes and HEV simulator that can be useful in both design and analysis. Its structure is based on that of the University of Illinois’ series HEV. The simulator is capable of both dynamic and steady state simulation. The simulator is useful for both design evaluation and basic parametric analysis.

Chapter 2 gives a detailed look at the subsystem models that the simulator uses. Chapter 3 explains the simulator structure and methodology. Chapter 4 contains
several simulation examples and details their implications on HEV design. Chapter 5 deals with implementation issues.
CHAPTER 2 COMPONENT MODELING

In a simulation of this nature, implementation of accurate component models is of vital importance. This chapter will explain in detail the models used in creating a functional series HEV simulation program. These models include: linear and nonlinear induction motor models, cooling fan, battery, tracking and switching APU models, mechanical braking, dynamic vehicle model, transmission, and electronic motor drive model.

2.1 Modeling vehicle dynamics

Vehicle dynamics modeling consists of modeling all of the forces opposing or assisting the vehicle's motion. The forces include: air drag force, rolling resistance, gravitational force due to the vehicle on a grade, and the vehicle traction force provide by the main drive motor. The forces are related to vehicle acceleration through Newton's second law as in eq. (2.1)

\[ \sum f_{\text{vehicle}} = m_f a \]  

(2.1)

where \( m_f \) is the equivalent mass of the vehicle when wheel rotational effects are considered, and \( a \) is the vehicle acceleration.

2.1.1 Force due to air drag
The air drag force opposes the vehicle's movement through the atmosphere. The drag force is generated by two different effects of the vehicle traveling through a fluid. These consist of a pressure gradient across the face of the vehicle and a friction force as air travels over the vehicle. Due to the shape of a typical automobile, the force due to the pressure gradient generally far outweighs the force owing to friction. The pressure drag coefficient is defined as

\[ C_d = \frac{f_{\text{drag}}}{\frac{1}{2} \rho A_f v^2} \]  

(2.1)

where \( f_{\text{drag}} \) is the drag force on the vehicle, \( \rho \) is the density of air, \( A_f \) is the frontal area of the vehicle and \( v \) is the vehicle speed assuming that the air is stationary. If the air is not stationary, its velocity may be added to the vehicle's velocity with the positive direction defined a the direction of vehicle travel. Rearranging eq. (2.1), the drag force is given by

\[ f_{\text{drag}} = \frac{1}{2} C_d \rho A_f v^2 \]  

(2.1)

Although the friction drag was assumed to be negligible, it is inherently included in the pressure drag. The drag coefficient for a vehicle is almost always determined experimentally. An air flow is imposed on the vehicle and the drag force is measured. If this force contains components resulting from both pressure drag and friction drag, when eq. (2.1) is solved for \( C_d \) it applies to sum of both components. This can be done because the friction drag is of exactly the same form as eq. (2.2).

### 2.1.2 Rolling resistance

Rolling resistance is created due to tire rotation under the weight of the vehicle. As the tires of the vehicle roll on the road's surface, they are continually deformed by the
weight of the vehicle. This deformation creates a force that opposes the vehicle motion. This force is independent of vehicle speed up to a certain limit. However, the rolling resistance is dependent on the tire pressure, since the pressure determines the severity of tire deformation. Assuming the tire pressure is constant, the rolling resistance may be expressed as

$$f_{tire} = R_{tire}mg$$  \hspace{1cm} (2.3)

where $R_{tire}$ is a constant which is experimentally determined, $m$ is the vehicle mass, and $g$ is the gravitational acceleration constant equal to 9.81 m/s$^2$.

### 2.1.3 Gravitational force due to road grade

When the vehicle encounters a non-zero road grade there is a gravitational force component in line with the vehicle's motion. This force can oppose or assist vehicle motion depending upon the whether the slope is positive or negative. Consider the situation depicted in fig. 2.1.

![Figure 2.1 Vehicle on a grade.](image-url)
Looking at fig. 2.1, the vehicle mass is represented by the block labeled $m$, the force of gravity in line with the vehicle’s motion is $f_g$, the vehicle velocity is $v$, the total downward force of the vehicle is $w$, and angle of the grade is $\theta$. The force $f_g$ is simply the dot product downward force generated by the vehicle weight and a unit vector in the direction of $f_g$. Evaluating this product gives

$$f_g = \vec{w} \cdot \hat{i}_g = mg \cos(90' - \theta)\hat{i}_g = mg \sin(\theta)\hat{i}_g$$

where the weight of the vehicle has been given as $mg$. The force $f_g$ shown in fig. 2.1 opposes the vehicle’s motion. If the angle $\theta$ had been negative, the force $f_g$ would aid the vehicle’s motion.

2.1.4 Force needed to overcome vehicle inertia

In order to accelerate the vehicle, a force must be applied in the direction of the acceleration. Also, as the vehicle accelerates, the wheels experience an angular acceleration. This implies that an additional force is necessary to accelerate the rotating mass.

The force required for linear acceleration of the vehicle mass is given by

$$f_{linear} = ma$$

where, once again, $m$ is the total vehicle mass.

The rotational acceleration is provided by the drive motor torque, but to be consistent, this torque is converted to a force on the wheel rim. A depiction of the wheel configuration is shown in fig. 2.2.
The wheel torque is given by

\[ T_{\text{wheel}} = J\alpha = f_{\text{road}} r_{\text{wheel}} \]  

(2.6)

where \( r_{\text{wheel}} \) is the wheel radius, \( a \) is the vehicle acceleration, \( J \) is the total moment of inertia of rotating parts, referred to the axle, and

\[ \alpha = \frac{a}{r_{\text{wheel}}} \]  

(2.7)

is the angular acceleration of the wheel. Substituting eq. (2.7) into eq. (2.6) the road force is given by

\[ f_{\text{road}} = \frac{Ja}{r_{\text{wheel}}} \]  

(2.8)

Therefore the total force due to the vehicle inertia and the wheel inertia is given by

\[ f_{\text{inertia}} = \left( m + 4 \frac{J}{r_{\text{wheel}}^2} \right) a \]  

(2.9)

2.1.5 Traction power

The total traction force provided by the vehicle is the sum of the forces derived in the sections above and is given by

\[ f_{\text{total}} = \frac{1}{2} C_d \rho A J v^2 + R_{\text{tire}} m g + m_i a + mg \sin(\theta) \]  

(2.10)

where \( m_i \) is the equivalent mass
battery energy is an integral of power draw with respect to time, the simulation program structure looks like

\[ E_{\text{pack}(k+1)} = E_{\text{pack}(k)} - \Delta E \]  

(2.27)

where \( E_{\text{pack}(k+1)} \) is the new battery energy (after one time increment) and \( E_{\text{pack}(k)} \) is the old battery energy. \( \Delta E \) is the amount of energy by which the pack energy changes and is given by

\[ \Delta E = k P_{\text{pack}} \Delta t \]  

(2.28)

where \( k \) is a weighting value that is necessary due to battery pack re-rating, \( P_{\text{pack}} \) is the power draw from the pack and \( \Delta t = t_{(k+1)} - t_{(k)} \).

The pack re-rating factor \( k \) is a function of the maximum watt-hour rating and the present watt-hour rating. If the demanded energy from the pack is \( \Delta E \) over the time interval \( \Delta t \), then the energy which must be subtracted from the current pack energy value is \( Whr_{\text{ref}} / Whr_{\text{present}} \Delta E \) where \( Whr_{\text{present}} \) is the watt-hour value for the present power draw. In other words, the latter amount of energy was withdrawn from the battery pack, but only \( \Delta E \) was useable. The re-rating may be done in a similar fashion for battery pack charging. If the energy supplied to the pack is \( \Delta E \) over the time interval \( \Delta t \), then the actual energy content that is added to the current battery pack energy value is given by \( Whr_{\text{present}} / Whr_{\text{ref}} \Delta E \). The complete expression for updating the battery pack energy is

\[ E_{\text{pack}(k+1)} = E_{\text{pack}(k)} - \left( \frac{Whr_{\text{ref}}}{Whr_{\text{present}}} \right)^{\text{sign}(P_{\text{pack}})} P_{\text{pack}} \Delta t \]  

(2.29)
This expression works because $P_{pack}$ is defined as positive for power out of the battery pack.

### 2.7.2 Battery terminal characteristic model

Along with the state of charge model, the overall battery model must also have a terminal characteristic model so that it can be realistically interfaced with the rest of the vehicle. The terminal model is composed of a dependent voltage source and a series resistance as shown in figure 2.8.

![Figure 2.8 Battery terminal model.](image)

The voltage source value, $V_{bat}$, and the series resistance, $R_{bat}$, both have a piece-wise dependence on the battery $SOC$.

### 2.7.3 Battery pack capacity

The battery model in fig. 2.7 pertains to a single battery, but the battery pack contains many batteries. The UI HEV contains 26 of the UPS 12-95 batteries. The total battery pack capacity is calculated as just the sum of the individual battery capacities. For example, the UI HEV contains 26 batteries, and the total pack capacity is calculated as

$$E_{pack} = (26)(273 \text{ W} \cdot \text{hr})\left(\frac{3600 J}{\text{W} \cdot \text{hr}}\right) = 25.6 \text{ MJ}$$
where 273 W-hr is obtained from the intercept of the line fit in fig. 2.7 with the W-hr rating axis.

In calculating the total pack W-hr re-rating, the power draw from the pack is assumed to be equally distributed between all the batteries contained in the pack. So the W-hr rating for a single battery is calculated using fig. 2.7 and this value is then multiplied by the number of batteries in the pack.

The source voltage and the series resistance for the model in fig. 2.8 are also initially calculated for a single battery. The total pack source voltage and series resistance are just those for a single battery multiplied by the number of batteries.

2.8 Auxiliary power unit modeling

The APU is the primary energy source for an HEV. This simulation models a series HEV which implies that the APU consists of some kind of power source connected to the batteries. The particular APU structure that this simulation as is that of an ICE driving a generator. The important thing to note here is that the APU is connected only to the battery pack, effectively decoupling the ICE dynamics from the vehicle driving dynamics. As stated earlier, the battery pack acts as a filter between the APU and the driving power requirements. This allows quasi-steady state APU operation, thus improving fuel economy and reducing exhaust emissions.

This section describes three different APU structures, the switching APU, the continuous tracking APU, and the discrete tracking APU. The switching APU is probably the most often thought of when the topic of HEVs is considered. This APU simply switches on to full output power and off to zero output power in order to constrain the battery pack energy to a specific range of values. The continuous tracking APU
attempts to continually supply the average power demanded for vehicle driving, thus attempting to hold the battery energy at a constant value. The batteries only supply power for transient driving conditions such as vehicle acceleration. The final APU structure, the discrete tracking APU, is very similar to the continuous case. The only difference is that the APU output power is constrained to a discrete set of values. Each of these structures have their own specific advantages and disadvantages which will be discussed in detail.

2.8.1 Switching APU structure

The switching APU is probably the simplest APU structure one can talk about. Both the APU control and operation are very straightforward. The battery state of charge determines when the APU is on and when it is off. The battery state of charge (SOC) is simply the current battery energy value divided by the maximum battery capacity and is usually expressed as a percentage. Therefore, in the switching APU case, the APU comes on when the batteries are at some low SOC value, \( SOC_{low} \), and the APU switches off when the battery state of charge reaches a higher SOC value, \( SOC_{high} \). The primary advantage of this technique is that the APU, when it is on, operates in a steady state mode. This allows fuel conversion efficiency to increase and exhaust emissions to decrease. A block diagram of a switching APU is shown in figure 2.9.
The SOC feedback line in fig. 2.9 simply supplies the APU with knowledge of the battery pack's SOC. It has probably already become apparent that the switching APU is just a special case of the discrete tracking APU with the output power constraint set containing only two points, the maximum APU output power and zero output. The reason it is being discussed as a separate case is because of its popularity with HEV designers as is evidenced by [6], [7]. It is necessary to look at the switching case in detail in order to evaluate its performance with regard to the other APU structures. In addition, the switching APU structure is used in steady-state simulation.

In general, the APU has some time constant associated with turn on and turn off. To simplify the analysis of the switching APU, this time constant will be ignored for the remainder of this section. In contrast to the definition of $P_{pack}$ in the section on battery modeling, here $P_{pack}$ is defined as the power into the battery pack and is given by

$$P_{pack} = P_{apu} - P_{demand} - P_{loss}$$  \hspace{1cm} (2.30)

where $P_{apu}$ is the power supplied to the pack by the APU, $P_{demand}$ is the demanded driving power, and $P_{loss}$ is a general loss function which includes battery losses. Therefore, the battery pack energy is given by

$$E_{pack} = \int P_{pack} \, dt$$  \hspace{1cm} (2.31)

Since the battery pack storage capability is of finite size, the average power into the pack must be zero over intervals of several hours or more

$$\left< P_{pack} \right> = 0$$  \hspace{1cm} (2.32)

Using eq. (2.32) along with a time average of eq. (2.30), eq. (2.33) is obtained.
Equation (2.33) states that the APU output switches on and off in such a way that the average output power matches the average demanded power plus any system losses. Since the APU is either on or off, the output is a square wave. If the demanded driving power is constant, then the switching period is given by

\[
T = \frac{\Delta E}{P_{apu,\text{max}} - P_{\text{demand}} - P_{\text{loss}}} + \frac{\Delta E}{P_{\text{demand}} + P_{\text{loss}}} = T_{on} + T_{off}
\]

where \( \Delta E = E_{\text{pack, max}}(SOC_{\text{high}} - SOC_{\text{low}}) \). This simplified form shows explicitly the pulse width modulated behavior of the APU. In general, it is not this simple because the demanded power is a function of time. The characteristics of a switching APU will be examined in more detail when simulation results are discussed in a later chapter.

2.8.2 Continuous tracking APU

The continuous tracking APU is defined to have a continuous spectrum of output power levels. An appropriate strategy is to command the APU output power to track the average demanded driving power plus any system losses. If the averaging window is long enough, the APU operates in a quasi-steady state, supporting low and high efficiency. However, the batteries are responsible for any transient power demands,
which in turn create battery losses. Therefore an extra control mechanism is required in order to force the battery $SOC$ to a desired level, $SOC_{ref}$. The block diagram of such a system is shown in fig. 2.10. Looking at fig. 2.10, the APU output power is given by

$$P_{apu} = \langle P_{demand} \rangle + K_{SOC} (SOC_{ref} - SOC)$$

(2.35)

where $K_{SOC}$ is constant. The first term on the right of eq. (2.34) commands the APU to supply the average demanded driving power, while the second term adds additional APU power in order to compensate for losses and to force the $SOC$ to $SOC_{ref}$.

The advantages of such a setup are that the batteries can be held near a fixed charge point in contrast to the switching case, and that the battery pack does not have to endure deep cycling. In the case of the switching APU, the batteries are charged with the APU at full power. The charge rate is the difference between the maximum APU output and the average power demanded. At low speeds, especially, the demanded power can be substantially lower than the APU maximum output. This results in heavy losses during charging. For the tracking case, the APU output is forced to match the demanded power, thus reducing battery losses. Furthermore, for a switching APU HEV, the batteries, by definition, are made to cycle between a low SOC value and a high SOC value. This deep cycling is detrimental to the life span of the batteries. Specifically, lead-acid batteries have an average deep cycle life of 500. Battery cycling is almost entirely eliminated in the continuous tracking APU case.

Interestingly, the battery pack size for the two different APU structures are chosen in fundamentally different ways. The pack size for the switching APU case is determined primarily on the basis of battery energy density. The pack energy capacity must be high
in order to prevent battery pack cycling and to avoid frequent APU starts. On the other hand, the pack size for the tracking APU case is determined primarily on the basis of battery power density. The pack must handle the power transients, but is not responsible for supplying power for extended driving periods.

Finally, as noted in the last section, the switching APU operates in steady state during its on-time to ensure high efficiency and reduce exhaust emissions. For an APU which has at its heart an ICE, this probably means that the ICE is being operated at stoichiometric conditions. In other words, the fuel-oxygen ratio is optimal. To create a practical tracking APU, the ICE must be capable of maintaining stoichiometric conditions even in quasi-steady state operation. Otherwise engine performance would be impaired. As will be shown in the chapter on Implementation, this task is quite feasible.

2.8.3 Discrete tracking APU

The discrete tracking APU is a discretized version of the continuous tracking APU. Whereas the continuous tracking APU has a continuous distribution of output power levels, the discrete APU has only a finite set of output power levels. The

![Discrete tracking APU structure.](image)

switching APU is a special case of the discrete tracking APU with only two power levels. The block diagram for the discrete tracking APU is shown in fig. 2.11 and is very similar
to the continuous case. The *discretize* block in fig. 2.7 provides a power command to the APU with a value that is within the discrete APU power level set that is closest to the actual desired APU power, $P_a$. In other words if the power range set of the APU is given by

$$P_{\text{discrete}} = \{P_1, P_2, \ldots, P_n\}$$

and $P_a$ is such that

$$P_m \leq P_a \leq P_{m+1} \quad \text{and} \quad |P_a - P_m| > |P_a - P_{m+1}|$$

then the output of the APU goes to the power level $P_{m+1}$. Naturally, as the number power levels contained within the APU range increases, the discrete APU case approaches that of the continuous case.

The discrete case would be applicable in situations where maintaining stoichiometric conditions over a continuous ICE operating range is difficult or impossible. This assumes, of course, that the ICE can be make the discrete transitions relatively quickly, ensuring that little time is spent in the transitory state. As the time required for the transitions increases, the overall system performance from an efficiency and emissions standpoint decreases.

**2.8.4 APU terminal characteristics**

The APU model used in this study represents an ICE driving a permanent magnet synchronous generator. The combination is assumed to be capable of delivering a continuous range of power levels between user-set minimum and maximum values. The conversion efficiency of the ICE is assumed to be constant over this range, consistent with experimental results on the UI HEV [1]. The generator is modeled in steady-state,
and so the winding resistance is the only parameter of importance in determining the
generator losses.

2.8.5 Fuel consumption

One of the most important quantities that the simulator should provide is that of
fuel consumption. The energy delivered by the APU and the distance traveled are
calculated over an extended time period. If the APU is cycling in steady state, this
corresponds to calculating the energy delivery and distance traveled over an integral
number of APU cycles. In all cases, the fuel consumption must be calculated in steady-
state. If it is not, then the fuel consumption becomes directly dependent on the initial
SOC of the battery pack \([x]\). For example, if the battery pack SOC is initially 100% and
the fuel consumption is calculated over a short interval before steady-state operation is
reached, then most of the driving energy is supplied by the battery pack and the fuel
consumption appears to be comparatively low. If the same scenario is repeated with the
initial battery pack SOC at 30%, depending on the particular APU control policy, the
APU may come on and charge the batteries in addition to supplying the necessary power
for driving. In this case, the fuel consumption would appear to be high. If the simulated
driving cycle is long enough to allow the APU system to reach steady-state, then the fuel
consumption may be calculated in a more objective manner.

As stated in section 2.8.4, the fuel conversion efficiency of the ICE is constant
over the ICE's operating range. Denote this conversion efficiency as \(F_c\) in units of \(g/kW\cdot hr\). The fuel consumption of the vehicle is then given by

\[
\text{fuel consumption} = \frac{\Delta E}{\Delta x} F_c
\]
where $\Delta E$ is the energy quantity delivered by the APU and $\Delta x$ is the distance traveled. Alternatively, the vehicle mileage, in units of miles per gallon, is the reciprocal of eq. (2.8.6) with the units converted accordingly.

### 2.8.6 APU response characteristics

Any time the APU is commanded to change, there is a response time associated with the change. For this simulation, the response is modeled with a single time constant. The form of the response is shown in eq. (2.8.6)

$$P_{out} = P_{com} \left(1 + \frac{s}{\tau_{apu}}\right)^{-1}$$

where $P_{out}$ is the APU output power, $P_{com}$ is the commanded power desired from the APU, and $\tau_{apu}$ is the time constant of the APU response. In this context, $s$ is no longer the drive motor slip, but the Laplace transform operator. A form of eq. (2.8.6) that is useable in the simulation program is derived in eqs. (2.8.6)-(2.8.6). In differential form, eq. (2.8.6) becomes eq. (2.8.6)

$$P_{out} = P_{com} - \frac{1}{\tau_{apu}} \frac{dP_{out}}{dt}$$

This equation may be discretized by replacing the derivative with a finite rate of change.

$$P_{k+1} = P_{com} - \frac{1}{\tau_{apu}} \frac{P_{k+1} - P_k}{\Delta t}$$

where $\Delta t$ is the program time step size, $P_{k+1}$ is the APU output power after the current time step, and $P_k$ is the current APU power. Rearranging eq. (2.8.6) gives

$$P_{k+1} = \frac{P_k + \tau \Delta t P_{com}}{1 + \tau \Delta t}$$

30
This response characteristic is used with all three APU models. The power tracking APU also has state of charge feedback which forces the battery pack SOC to track a reference value. This SOC feedback has a constant gain and is added directly to the $P_{com}$ value. This feedback has the form of

$$P_{soc} = K_{soc} \left( 1 - \frac{SOC}{SOC_{ref}} \right)$$

(2.x)

where $SOC_{ref}$ is the reference state of charge. The value can not change exceedingly fast because of the integral action of the battery pack. The overall response time of battery pack charging is dependent on the size of the pack.
\[ m_t = m + 4 \frac{J}{r_{wheel}^2} \]

The traction power may be found quite easily,

\[ dE = f_{total} \, dx \Rightarrow \frac{dE}{dt} = \frac{dx}{dt} = f_{total} \, v = P_{traction} \] (2.11)

Using eq. (2.11), the total traction power is

\[ P_{traction} = \frac{1}{2} C_d \rho A_f v^3 + R_{wire} m g v + m_i a v + m g v \sin(\theta) \] (2.12)

One of the primary advantages of an HEV is that it is capable of regeneration. This means that it can recapture kinetic energy by using the drive motor as a generator. By looking at eq. (2.12) one can determine which terms are purely dissipative and which are conservative. The velocity \( v \) is defined a positive for the direction in which the vehicle is traveling in eq. (2.12). Therefore, both of the first terms in eq. (2.12) are always positive, indicating that they are not conservative. However, the last two terms may become negative depending on the angle \( \theta \) and the whether or not the vehicle is accelerating or decelerating. A negative traction power implies that energy is being returned to the vehicle system.

2.2 Induction motor modeling

The induction motor model is based on the steady-state circuit model for a three-phase induction motor. The non-linear single phase equivalent induction motor model will be examined first. The approximate linear model analysis will be presented in the next section.

2.2.1 Non-linear induction motor model
The equivalent single-phase induction motor equivalent circuit is shown below in fig. 2.3 with the shunt terms moved to the input side as described in [1].

![Induction motor model](image)

The stator and equivalent rotor resistance are given by \( r_1 \) and \( r_2 \) respectively. Likewise, the stator and equivalent rotor reactive impedance are given by \( jX_1 \) and \( jX_2 \) respectively.

The shunt impedance is \( \frac{jX_\phi r_c}{jX_\phi + r_c} \) and the equivalent shaft load is given by \( \frac{1-s}{s} r_2 \).

Since this is a single-phase model, the actual shaft power is three times the power dissipated in this load. The parameter \( s \) is called slip and is defined as

\[
s = \frac{\omega_s - \omega}{\omega_s} \quad (2.10)
\]

where \( \omega_s \) is the synchronous speed of the motor and \( \omega \) is the actual speed. The shaft torque represented by fig. 2.3 is given by

\[
T_{\text{shaft}} = \frac{3|V_{i-n}|^2 r_2 s}{\omega_s \left\{ \left( sr_1 + r_2 \right)^2 + s^2 (X_1 + X_2)^2 \right\}} \quad (2.13)
\]

where \( T_{\text{shaft}} \) is the torque to the shaft in N\( \cdot \)m, \( s \) is the motor slip and \( \omega_s \) is the motor synchronous speed in rad/s. The shaft power is obtained as
where $\omega$ is the actual shaft speed in mechanical rad/s of a two pole machine and is given in terms of the motor slip by

$$\omega = (1-s)\omega_s$$

(2.15)

The motor speed in the HEV is controlled by a variable excitation frequency. In order for the torque curve to be translated on the speed axis in a consistent manner the motor terminal voltage must vary linearly with the excitation frequency [1]. Therefore, the terminal voltage $V_{l-n}$ must be

$$V_{l-n} = k\omega_s \frac{P}{2} + V_{offset}$$

(2.16)

where $k$ is constant of proportionality between the synchronous speed and the terminal voltage and $V_{offset}$ is an offset which allows the motor to be started from zero speed. $p$ is the number of motor poles.

Given a particular shaft power, eqs. (2.14), (2.15) and (2.16) can be solved simultaneously for the resulting slip. This is done using a modified Newton routine. The routine actually solves for the motor slip using the shaft torque equation (2.13). The shaft torque is obtained by dividing the shaft power by the known shaft speed. The algorithm will solve for a corresponding slip on the torque-slip curve or, if the given torque is beyond the motor's capability, it will return the peak torque capability of the motor for the specified motor speed. The negative torque peak gives the maximum regenerative capability of the motor for braking. The routine is shown below in eqs.(2.17)-(2.21)

$$s_k = s_0 \quad \text{initial guess}$$

(2.17)
\[ \Delta T = T_{\text{shaft}} - T(s_k) \] (2.18)

\[ \Delta s = M \left[ \frac{dT(s)}{ds} \right]^{-1}_{s=s_k} \Delta T \quad \text{for} \quad \left[ \frac{dT(s)}{ds} \right]_{s=s_k} > \beta \] (2.19)

\[ \Delta s = \frac{M}{\beta^2} \left[ \frac{dT(s)}{ds} \right]_{s=s_k} \Delta T \quad \text{for} \quad \left[ \frac{dT(s)}{ds} \right]_{s=s_k} < \beta \] (2.20)

\[ s_{k+1} = s_k + \Delta s \] (2.21)

where \( s_0 \) is an initial guess for the slip, \( M \) is a scaling factor, \( T_{\text{shaft}} \) is the given shaft torque and \( \beta \) is a given constant. Assuming the given torque is achievable, eqs. (2.18)-(2.21) are repeated until \( T_{\text{shaft}} - T(s_k) < \epsilon \) where \( \epsilon \) is some specified tolerance. If \( |T_k - T_k| < \epsilon \), then the given torque is not on the curve and a minimum or maximum has been found. The ability to find extrema is achieved by using eq. (2.20) when the slope of the torque-slip curve approaches horizontal. If while searching, the algorithm strays from between the extrema, the search will diverge. If this occurs, the scaling factor \( M \) is reduced and the search is restarted with the original initial condition on \( s \).

Once the slip is obtained, the input line-to-neutral voltage and the synchronous motor frequency may be found using eqs. (2.15) and (2.16). The input current and the motor losses may now be found as follows

\[ i_2 = \frac{V_{l-n}}{r_1 + r_2 + j(X_1 + X_2) + r_2 \left( \frac{1 - s}{s} \right)} \] (2.17)

\[ i_{\text{shunt}} = \frac{V_{l-n}}{jX_f} + \frac{V_{l-n}}{r_c} \] (2.18)

\[ i_{\text{in}} = i_2 + i_{\text{shunt}} \] (2.19)
\[ P_{\text{loss}} = 3i_2^2(r_1 + r_2) + \frac{3V_{l-n}^2}{r_c} \]  

(2.20)

### 2.2.2 Linear induction motor model

The linear induction motor model is an approximation of fig. 2.3 and is valid for small values of slip [4]. The power equation for the linear model may be found by assuming the slip is small enough that all of the terms in eq. (2.14) that are second order in slip are zero. The resulting shaft power is given by

\[ P_{\text{shaft}} = \frac{3|V_{l-n}|^2s\omega}{\omega_1 r_2} \]  

(2.21)

![Figure 2.4 Comparison of torque curves.](image-url)
Using the demanded shaft power, eq. (2.21) is solved for slip. This slip is used in the manner described in the previous section to solve for all desired quantities. The primary disadvantage of the linear model as compared to the non-linear model is the loss of accuracy as the slip or torque increases. However, if this loss of accuracy can be tolerated, use of the linear model can decrease simulation time as compared to the nonlinear model. A comparison of torque vs. slip curves for the linear and non-linear cases for a typical induction motor is shown in fig. 2.4. Notice that when the slip is small the curves are almost identical.

2.3 Cooling fan model

The cooling fan for the main drive induction motor can be a sink of significant power. The cooling fan is intended primarily for cooling the drive motor when the motor is under heavy load. The fan may also be used to cool additional components as well. The fan power consumption may be derived qualitatively from momentum considerations. A force is arrived at by taking a derivative of the momentum \( p \) with respect to time as in eq. (2.22).

\[
f_{\text{air}} = \frac{dp}{dt} = \frac{\partial p}{\partial v} \frac{dv}{dt} + \frac{\partial p}{\partial m} \frac{dm}{dt} = m \frac{dv}{dt} + v \frac{dm}{dt}
\]  

(2.22)

This force is that on the air mass being moved through the fan. The power required to produce the air flow is the force on the air mass multiplied by the velocity of the air flow. The fan power is then given by

\[
P_{\text{fan}} = mv \frac{dv}{dt} + v^2 \frac{dm}{dt}
\]  

(2.23)
The angular velocity of the fan may be related to the velocity of the air flow. Also, the mass flow rate \( \frac{dm}{dt} \) is not constant, it is linearly dependent on the angular velocity of the fan. Everything else is constant, and so the fan power equation may be reduced to

\[
P_{\text{fan}} = k_1 \omega^3 + k_2 \omega
\]  

(2.24)

where \( \omega_{\text{fan}} \) is the angular velocity of the cooling fan. There may also be a correction term in \( \omega^2 \) which is not shown in eq. (2.24). This is in agreement with experimental data obtained from the University of Illinois' HEV [2].

![Figure 2.x Fan power vs. fan speed.](image)

The fan power is then a cubic function of the fan’s angular velocity. At low motor speeds the fan power is almost negligible, but at high motor speeds, the cooling fan is a sink of considerable power.

### 2.4 Transmission model

The purpose of the transmission is to better match the torque and speed capabilities of the drive motor to those required for driving. The transmission model is based on a standard five-speed transaxle. A specified set of gear ratios along with the
vehicle's speed are used to determine the shaft speed of the drive motor. Rotation of the gears in the viscous transmission oil creates a power loss that is speed dependent. While under load, the transmission also exhibits a constant marginal efficiency $\eta_{\text{trans}}$ [1]. The total transmission loss is then given by eq. (2.24).

$$P_{\text{trans loss}} = P_{\text{no load loss}} + (1 - \eta_{\text{trans}})(P_{\text{trans}} - P_{\text{no load loss}})$$  \hspace{1cm} (2.24)

where $P_{\text{no load loss}}$ is the speed dependent loss due only to the oil viscosity and $P_{\text{trans}}$ is the transmission input power. There is a different no load loss characteristic associated with each gear. The simulation model uses expression for these no load losses obtained by line fitting data obtained from the UI HEV. The actual no load losses for each gear are shown in figure x.x and Table 1 shows the line fitted approximations.

![Figure 2.x No load transmission loss.](image-url)
Table 2.1: Approximate expressions for no load loss.

<table>
<thead>
<tr>
<th>Gear</th>
<th>Approximate power loss (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P_{\text{loss}} = 0.177\omega_{\text{motor}} + 0.05(P_{\text{in}} - P_{\text{no load}}) + 34.6$</td>
</tr>
<tr>
<td>2</td>
<td>$P_{\text{loss}} = 0.139\omega_{\text{motor}} + 0.05(P_{\text{in}} - P_{\text{no load}})$</td>
</tr>
<tr>
<td>3</td>
<td>$P_{\text{loss}} = 0.270\omega_{\text{motor}} + 0.05(P_{\text{in}} - P_{\text{no load}}) - 87.7$</td>
</tr>
<tr>
<td>4</td>
<td>$P_{\text{loss}} = 0.354\omega_{\text{motor}} + 0.05(P_{\text{in}} - P_{\text{no load}}) - 90.0$</td>
</tr>
<tr>
<td>5</td>
<td>$P_{\text{loss}} = 1.410\omega_{\text{motor}} + 0.05(P_{\text{in}} - P_{\text{no load}}) + 163.0$</td>
</tr>
</tbody>
</table>

The model does not take into account the variation of gear oil viscosity with temperature.

2.5 Mechanical braking model

Hybrid electric vehicle braking is usually accomplished by regeneration. The drive motor act as a generator to provide a negative torque to the wheels, decelerating the vehicle and recovering kinetic energy. However, if the deceleration is too great, the required power to decelerate the vehicle may exceed the torque of the drive motor or the electronic drive. When this happens braking must be achieved in a different fashion. The alternative is generally standard mechanical brakes.

Mechanical braking modeling can be very troublesome because it is difficult to model the transition from regeneration to mechanical braking. In principle, the mechanical braking can be modeled by allowing the drive motor to accept its maximum torque limit and assigning the excess power to frictional loss in the mechanical brakes. This approximation is a valid reflection of the UI HEV's brake controls. The negative peak on the motor torque-slip curve is found using the Newton algorithm outlined in section 2.2.1.

2.6 Electronic drive model
The electronic drive is the interface between the batteries and the drive motor. It is used to convert the dc battery voltage into an ac source to power the drive motor. The particular drive that this simulation program models is a three-phase pulse width modulation (PWM) type. This class of drive is one of the most versatile and that is available today. Due to high power requirements of an electronic drive used in an HEV application, the modeled switches are insulated gate bipolar transistors (IGBTs). A circuit diagram showing the basic power structure of a PWM drive using IGBTs is shown in figure 2.5 below.

Figure 2.5 Electronic drive layout.

The voltages $+V_{in}$ and $-V_{in}$ are the battery positive and negative supply rails, respectively. The motor input terminals are labeled $V_a$, $V_b$ and $V_c$. The three-phase ac voltages used to drive the motor are constructed by using PWM to switch the IGBT transistors. At any time instant, one IGBT pulls one of the motor phases to the positive rail and one of the other IGBTs pulls another motor phase to the negative rail. Now that the basic circuit operation has been determined, it is necessary to look at the individual IGBT characteristics in order to determine the drive losses.
The IGBT acts as a combination of a power bipolar junction transistor (BJT) that is driven by a metal oxide semiconductor field effect transistor (MOSFET) [3]. The BJT is the power device while the MOSFET simply supplies the turn on current to the BJT. This allows the device to have the usual power characteristics of a BJT but with the much less demanding drive characteristics of a MOSFET. Here, the concern is not how the IGBTs are driven, it is their transfer characteristics that are of interest. The IGBT gate drive consumes very little power compared to the power loss from the on-state resistance and the forward voltage drop while the transistor is carrying current to the load.

The devices are modeled as a forward voltage drop $V_{\text{forward}}$ in series with an on-state resistance $r_{\text{on}}$. In addition, there is a reverse diode from emitter to collector added in IGBTs intended for use in inverters. Assuming that this diode is well matched to the transistor, it may be modeled in the same way. This implies that the drive is capable of not only supplying power to the motor, but, also, of regenerating power from the motor.

In operation, two IGBTs in fig. 2.5 are on at any given time for each motor winding. The average power loss per leg of the drive is given by

$$P_{\text{leg}} = 2I_{\text{rms}}^2 r_{\text{on}} + \frac{2\sqrt{2}I_{\text{rms}} V_{\text{forward}}}{\pi} \int_0^\pi \sin(\phi) d\phi = 2I_{\text{rms}}^2 r_{\text{on}} + \frac{4\sqrt{2}I_{\text{rms}} V_{\text{forward}}}{\pi}$$

(2.25)

The total power loss is three times $P_{\text{leg}}$ since there are three windings.

$$P_{\text{drive loss}} = 6I_{\text{rms}}^2 r_{\text{on}} + \frac{12\sqrt{2}I_{\text{rms}} V_{\text{forward}}}{\pi}$$

(2.26)

The power loss in the electronic drive can be substantial as shown in figure 2.6 below.
The values of $r_{on}$ and $V_{forward}$ used to generate fig. 2.6 are 4.2 mΩ and 1.1 V respectively. These values model the IGBTs in the UI HEV. The drive power loss is almost linear with line current indicating that the forward voltage drop is the source of most of the loss for this choice of $r_{on}$ and $V_{forward}$.

2.7 Battery pack modeling

The battery pack is an essential ingredient in the HEV structure. It acts as a power filter between the APU and the electronic drive. In order for an HEV to emulate a conventional vehicle, it must be able to respond to acceleration requests quickly. Transients of this nature are supplied by the ICE in a conventional vehicle. However, ICE transient behavior reduces fuel economy and increases exhaust emissions. In light of this reduced performance, it is not desirable to let the ICE supply power for acceleration transients. The responsibility of handling transient power demands is then shifted to the battery pack. For this simulation, the battery pack is modeled as a voltage source with a

$$P_{\text{drive loss}} = 6I_{\text{rms}}^2 r_{on} \sqrt{12V_{\text{rms}} V_{\text{forward}}} \over \pi$$
series resistance. The source voltage and the resistance are both dependent on the battery pack SOC. This particular model has two complementary components: the state of charge model and the terminal characteristic model, both of which will be described in the sections below.

2.7.1 State of charge battery model

The state of charge model is used to keep on track of the battery pack's stored energy. The model is constructed from experimental data based on the Johnson Controls UPS 12-95 lead acid battery. A plot of watt-hour rating vs. power draw is shown for a single UPS 12-95 battery in fig. 2.8.

![Graph of Watt-hour rating vs. power draw for the UPS 12-95 battery.](image)

\[
W = 7E-06P^2 - 0.0904P + 273.08
\]

\[
R^2 = 0.9613
\]

Figure 2.7 Watt-hour rating vs. power draw for the UPS 12-95 battery.

The data in fig. 2.8 was acquired by discharging batteries at a particular rate and keeping track of the amount of time required for discharge. Using the rate of discharge and the time required data, it is possible to calculate the corresponding watt-hour rating. The watt-hour rating is a measure of energy, and in this case it is a function of the
discharge rate. Watt-hour computation is accomplished by simply multiplying the discharge rate by the time required for discharge.

The rate of discharge mentioned above was calculated using a constant discharge current and assuming a constant voltage. In actual testing, the voltage varied from 12 V to 10 V. So strictly speaking, the watt-hour rating given above only an approximation of the actual watt-hour rating. In order to calculate the actual watt-hour rating, an integration over the voltage with respect to time would be required. This information is not available, and for the purposes of this thesis, the approximate model is sufficient.

In fig. 2.8, the watt-hour rating decreases monotonically with increasing power draw. The majority of lost energy is due to resistive losses within the battery. This is easily seen by noting that the line fit is a quadratic and decreases as power squared. If the voltage is constant, then this implies that the watt-hour rating decreases as current squared. This is consistent with resistive losses. The most important implication of this result is that the same watt-hour rule may be applied to battery charging as well as discharging. Qualitatively, this means that while discharging, one cannot retrieve the full energy content of the battery. Also, during charging, in order to achieve full charge, one must supply an energy quantity in excess of the full charge energy content.

The above results are derived in the context that the watt-hour rating applies to full discharge or full charge. Because the watt-hour re-rating is due to resistive losses, it may be applied to discrete energy quantities retrieved from or delivered to the battery. This real time re-rating of the battery pack is necessary for simulation. For simulation purposes, the batteries are assumed to have a maximum watt-hour rating, $Whr_{ref}$. This maximum value occurs where the curve in fig. 2.8 crosses the watt-hour axes. Since the
Chapter 3 Hybrid Electric Vehicle Simulation

This chapter will describe the HEV simulation program in detail. The simulation program is written in MATLAB 4.0 code and is event driven. Almost all interaction with the user takes place through parameter windows, although all subroutines and system variables are addressable from the command line as well.

3.1 Simulation overview

The simulator is capable of both dynamic and steady state simulation. It is important to establish the difference between the two. Dynamic simulation refers to cases in which the driving parameters, such as speed and road grade, change with time. However, all time step calculations assume steady-state operation of the individual sub-systems. This means that speed and other parameters are held constant over each time step. This is satisfactory so long as the time step is short compared to the actual vehicle dynamics but long compared to the dynamics of the components [6].

The steady-state simulator, on the other hand, assumes that velocity, road grade, etc. are truly constant. This removes the need for iterative simulation. The solution of losses and other parameters is immediate. In other words, nothing ever changes, so only one time step iteration need be done.

3.2 Dynamic simulation
It takes as inputs time, vehicle velocity, and road grade vectors. The vehicle is forced to track this given velocity profile as closely as the its capability will allow. If the vehicle cannot track the velocity profile, its velocity is reduced accordingly until a velocity is found that can be sustained. Figure 3.1 shows a block diagram of program flow. In the end, the simulation gives output vectors corresponding to the original time vector. There is a complete list of the available output variables in the appendix. Program flow will now be described using fig. 3.2.

The uppermost block represents the input vectors. The first vector is a time index. It does not have to regularly spaced, but the time steps should be much smaller than the expected vehicle dynamics and faster than sub-system dynamics. The second vector is a velocity vector. It corresponds directly to the time vector. In other words, for each time step, the corresponding vehicle speed is given by the velocity vector. The grade vector gives the road grade as \( \sin(\theta) \), where \( \theta \) is the angle of inclination of the grade, corresponding to each to time step. The road power can be found by using eq. (2.12) when a difference equation for acceleration is added. The transmission gear ratio can also be calculated, and the drive motor shaft speed found. With the shaft speed known, the cooling fan power is found using eq. (2.24). The sum of the traction power and the cooling fan power is equal to the required drive motor shaft power. Using the shaft power in conjunction with the shaft speed, the motor model can be solved for motor losses and the required terminal voltage and current by using eqs. (2.14)-(2.20). The electronic drive
Figure 3.2 Block diagram of program flow.
losses are then calculated using eq. (2.26). With drive current known, the required drive input dc voltage may be calculated. It is simply the known line-to-line peak drive motor voltage plus the voltage drop in the drive. This is the dc voltage that the battery pack must be capable of supplying at its terminals in order for the drive/motor combination to supply the necessary torque to the transmission.

The next step is to determine if the needed drive input dc voltage can be met by the battery pack at the drive current level. This is done using the APU supplied current, the drive current and the battery voltage and series resistance. The battery voltage and series resistance must first be determined using the battery SOC from the previous iteration. If the dc voltage available to the drive is less than that which is required, the velocity is reduced and the entire process is repeated until the required dc voltage level can be met. If the available dc voltage is higher than that which is required and the present velocity is lower than the originally requested velocity, the velocity is increased and the process is repeated. This repeated until the drive voltage level cannot be met or until the sustainable velocity is equal to that originally requested. Finally, once the available dc voltage is acceptable, the APU output is updated and the SOC battery model receives the sum of the APU output power minus the system power demand and losses. The battery state of charge is updated using eq. (2.29). Afterward, the time step is incremented and the entire process is repeated. This continues until the end of the time vector is reached.

Once the simulation has finished, any variable of interest can be plotted vs. the original time vector. The mechanical braking model was not included in the block diagram. Mechanical braking is accomplished by subtracting a portion of the power
supplied to the battery pack during braking. The portion that is subtracted is attributed to the mechanical brakes and is the difference between the braking power required and the maximum regeneration achievable by the drive/motor pair. The maximum regeneration is the lesser of the drive/motor current ratings and the breakdown torque of the drive motor.

3.3 Steady-state simulation

The steady-state simulator is used to simulate an HEV that is operating under fixed conditions. It takes as its input a set of minimum and maximum values. These values may correspond to vehicle velocity, road grade or vehicle pack size. A discrete range of values is generated with these minimum and maximum values as endpoints. The first point is used to calculate the required power and losses as in the dynamic simulator. In fact all applicable variables are calculated as in the dynamic simulation for this one point. Variables that have no steady-state meaning such as SOC are ignored. The motor ac voltage is assumed to be attainable, and this can be easily checked with the dynamic simulator. After the first point is calculated, the next point is processed in the same way until the maximum value is reached or the APU is no longer capable of supplying the required steady-state power.

There are two APU types available: the ideal switching APU and the ideal tracking APU. There is quite a difference between the two, however. The switching APU must create battery losses except for the single point where the output power is equal to that demanded by the vehicle. The switching APU is assumed to switch ideally, i.e. with no delay. In other words, the dynamic battery losses that occur during switching are ignored, and only the on/off state losses are considered. The switching APU is a
worst case when considering battery losses. This is because battery energy varies directly with charging/discharging time and battery losses vary as the square of the charging current. This implies that the battery charging time should be extended as long as possible. When the charging time is extended indefinitely, the APU output matches the demanded power, and the battery SOC remains constant. This results in the tracking APU. The tracking APU creates no battery losses, because, by definition, it matches the vehicle demanded power exactly at all points. These statements hold only during steady-state vehicle operation. Under dynamic driving conditions, the tracking APU/battery system exhibits losses as well, although they are still not as substantial as the switching case.

The switching APU steady-state on and off times are calculated using eq. (2.34). Two pack re-ratings must be done in this case. One corresponds to the APU in the off state in which the power out of the pack is used, call it \( Whr_{off} \). There is also a watt-hour rating corresponding to the APU power minus the demanded power, call it \( Whr_{on} \). Once again, the battery pack SOC is to remain between two user set levels, \( SOC_{high} \) and \( SOC_{low} \). Therefore, the APU on and off times are given by

\[
T_{off} = \frac{E_{pack} \left( \frac{Whr_{off}}{Whr_{ref}} \right) (SOC_{high} - SOC_{low})}{P_{demand} + P_{loss}} \tag{3.1}
\]

\[
T_{on} = \frac{E_{pack} \left( \frac{Whr_{ref}}{Whr_{on}} \right) (SOC_{high} - SOC_{low})}{P_{APU} - P_{demand} - P_{loss}} \tag{3.2}
\]

In eq. (3.1), \( E_{pack}(SOC_{high}-SOC_{low}) \) is the amount of energy that is taken from the batteries during the APU off time. This energy quantity must be scaled by \( Whr_{off} / Whr_{ref} \) to
account for losses. Finally, the energy divided by the power draw gives the off time. The same thing happens in eq. (3.2) except the energy supplied to the pack is scaled by $\text{Whr}_{\text{ref}} / \text{Whr}_{\text{on}}$.

The tracking APU is assumed to track ideally, meaning that the battery pack supplies no net power. This results in no battery losses. The demanded power and losses are calculated as above and the APU output is set equal to them.

As mentioned above, the steady-state simulator is also capable of doing battery pack size sweeps and road grade sweeps.
CHAPTER 4 SIMULATION RESULTS

This chapter will compare some simulations with actual results obtained driving the UI HEV, discuss possible causes of simulation error and finally present simulation examples directed toward solving HEV design problems. Specifically, the last sections use simulations in an attempt to determine the optimal APU type.

4.1 Comparison of simulation results with the UI HEV

This section is devoted to the comparison of simulation results with that of actual data taken while driving the UI HEV. The data was acquired by driving the UI HEV in an urban environment and on the highway and using a data logger to obtain the vehicle speed and battery pack voltage and current. This allows the power out of the battery pack to be found as a function of time. The velocity profile obtained with the data logger is then given to the simulator along with the vehicle parameters such as weight, gear ratios, etc. The power out of the battery pack is then found using the simulator and the results compared to the actual pack power. The APU is not active in any of these tests.

First, a couple of urban driving examples will be examined. Figure 4.1 shows the first velocity profile. Figure 4.2 shows the actual and simulated power out of the battery pack resulting from this driving cycle. Incidentally, the negative velocity values are due to the vehicle rolling backward.
Figure 4.1 Urban driving velocity profile.

Figure 4.2 Actual and simulated power out of the pack.
Figure 4.3 Urban driving velocity profile.

Figure 4.4 Actual and simulated power out of the pack.
The next comparison is also an urban driving cycle very similar to the first. The velocity profile is shown in fig. 4.3. The actual and simulated power out of the battery pack is shown in fig. 4.4. Both driving cycles were performed with the vehicle in second gear.

The next example is one of highway driving. The vehicle is held in third gear and is driven at a speed of approximately 40 mph. The vehicle velocity profile is shown in fig. 4.5 and the actual and simulated power out of the battery are shown in fig. 4.6.

![Figure 4.5 Highway driving velocity profile.](image_url)
Figure 4.6 Actual and simulated power out of the pack.

Figure 4.7 Steady-state power comparison.
The next example is steady-state comparison of battery pack power. The actual and simulated power for constant speed and gear ratio are shown in fig. 4.7 and are listed in Table 4.1.

<table>
<thead>
<tr>
<th>Gear/Speed</th>
<th>$P_{\text{actual}}$</th>
<th>$P_{\text{simulated}}$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd/40 mph</td>
<td>9300 W</td>
<td>9290 W</td>
<td>0.11%</td>
</tr>
<tr>
<td>2nd/38 mph</td>
<td>8420 W</td>
<td>8480 W</td>
<td>0.71%</td>
</tr>
<tr>
<td>3rd/41 mph</td>
<td>8760 W</td>
<td>9080 W</td>
<td>3.65%</td>
</tr>
<tr>
<td>3rd/37 mph</td>
<td>7630 W</td>
<td>7580 W</td>
<td>0.66%</td>
</tr>
<tr>
<td>3rd/42 mph</td>
<td>9270 W</td>
<td>9260 W</td>
<td>0.11%</td>
</tr>
<tr>
<td>4th/43 mph</td>
<td>9540 W</td>
<td>8670 W</td>
<td>1.36%</td>
</tr>
</tbody>
</table>

The actual power out of the pack in fig. 4.7 was obtained by driving in one direction on the highway and using the data logger to record the pack voltage and current once each second. The vehicle was then driven in the opposite direction and the same data taken. The average pack power for both directions was then averaged to arrive at the values shown in the figure. The average speed of the two drives was given to the simulator and it returned the simulated values given in the figure.

4.2 Error analysis

All of the battery pack power comparisons in the previous section show some error. This section is devoted solely to determining possible sources of the error. The four major sources or error are the absence of road grade information, the assumption that the ambient air is stationary, velocity measurement error, and unmodeled transmission loss.

4.2.1 Power error due to road grade variation

Road grade variations can be sources or sinks of considerable power. As discussed in chapter 2, the power due to road grade is given by
\[ P_g = mgs \]  

(4.1)

where \( s \) is the sin of the angle of road inclination. The mass of the UI HEV, including two 170 lb passengers, is 1733 kg. The power requirement due to road grade then becomes

\[ P_g = 17000s \]  

(4.2)

where, as previously explained, \( s = \sin(\theta) \), where \( \theta \) is the angle of inclination of the grade. Figure 4.x shows traction power demand as a function of vehicle speed and road grade with \( s \) expressed as a percent.

![Figure 4.8 Power demand vs. road grade and speed.](image)

4.2.2 Power error due to ambient wind speed being non-zero

All of the simulation comparisons assumed the ambient windspeed to be zero. The power due to windspeed variation is given by

\[ P_{\text{wind}} = \frac{1}{2} \rho C_d A_f [ (v_{\text{vehicle}} + v_{\text{wind}})^3 - v_{\text{vehicle}}^3 ] (v_{\text{vehicle}} \bullet \hat{i}_{\text{wind}}) \]  

(4.3)
where \( v_{\text{vehicle}} \) is the vehicle speed, and \( v_{\text{wind}} \) is the ambient wind speed. Figure 4.9 shows the traction power error vs. ambient windspeed for several vehicle speeds. The direction of the windspeed is opposite that of the vehicle’s motion.

![Figure 4.9 Traction power vs. ambient windspeed.](image)

### 4.2.3 Power error due to acceleration measurement error

The data logger used to record the UI HEV’s velocity vs. time inevitable has some error associated with measurement. This error in velocity measurement can greatly affect the calculated acceleration. The acceleration is calculated as

\[
a = \frac{v_k - v_{k-1}}{\Delta t}
\]  

(4.4)

and as \( \Delta t \) gets very small, the resulting acceleration error can be very large. The traction power due to vehicle acceleration is given by

\[
P_{\text{acc}} = m_i a v
\]  

(4.5)
where, as defined in section 2.1.5, $m_i$ is the vehicle equivalent mass. Figure 4.10 shows $P_{acc}$ vs. vehicle speed for various velocity measurement error values. The vehicle mass is taken as 1733 kg.

![Graph showing $P_{acc}$ vs. vehicle speed for various velocity measurement error values.](image)

Figure 4.10 $P_{acc}$ vs. vehicle speed.

4.3 Simulation examples

***************NEEDS COMPLETION***************

4.4 APU comparison

This section looks at the three different APU types described in section 2.8 in an attempt to discover which is the best. This will be done by simulation of several driving cycles with each APU type. Battery losses and cycling and complexity of APU control will be the criteria on which the decision will be based.

4.4.1 Switching APU

The switching APU is one whose output power assumes only two values; it is either on at maximum power or off. The state of the output is determined explicitly by
the battery pack SOC. The first simulation is that of a constant 40 mph for 2 hrs. The vehicle is held in second gear. The pack size is approximately 13 MJ, the maximum APU output is 30 kW, and the APU on/off time constant is 30 s. $\text{SOC}_{\text{high}}$ and $\text{SOC}_{\text{low}}$ are 80% and 40% respectively. All other parameters common to this simulation and the others in this section are shown in the appendix. The simulated battery pack energy and APU output are shown in fig. 4.11.

![Pack energy and APU output](image)

**Figure 4.11** Simulated battery pack energy and APU output.

The average APU output power is 14.08 kW and the pack cycles at 20 min intervals for this road speed. The average vehicle demanded power is 7.05 kW and the resulting mileage is 31.7 mpg. This low mileage is a direct result of the battery pack watt-hour rating being cut in half during charging. The charging rate is $30 \text{ kW} - 7.05 \text{ kW} = 23 \text{ kW}$.
A cycling time of 20 min implies that the battery pack would need to be reconditioned at 6500 mile intervals assuming that the batteries fail after 500 cycles [10], [11]. If the pack size were doubled, this number of miles would, in theory, be slightly over 13,000 miles. A battery pack this size is fully capable of handling most accelerations demanded of conventional vehicles. This implies that the size of a battery pack of an HEV based on a switching APU would be chosen on the basis of energy density and not power density. It must be large enough to prevent excessive cycling.

The second simulation example is one of urban driving. The pack size is 16.3 MJ and all other parameters are the same as in the previous example. The velocity profile is one obtained while driving the UI HEV. The original driving routine was approximately 450 s long. In this instance, it has been repeated nine times over and given to the

![Figure 4.12 Urban driving cycle repeated nine times.](image-url)
simulator. The velocity profile is shown in fig. 4.x. Although the velocity vector is too long to effectively display, the repetition of the velocity profile is obvious. The simulated total vehicle demanded power and the APU output are shown in fig. 4.12. Finally, the battery pack output power and the pack energy are shown in fig. 4.13. The average vehicle demanded power is 3.2 kW and the average APU output power is 8.6 kW.

Figure 4.13 Total demanded power and APU output.
This gives a mileage of only 19.9 mpg. Once again, this low mileage is caused by the mismatch between the average demanded power and the high APU output power. The APU switching is easy to see by looking at the power out of the pack in fig. 4.14.

4.4.2 Continuous tracking APU

The same two examples above will now be repeated with a tracking APU. The first example is one at a constant 40 mph. The maximum APU output is 30 kW, the battery pack size is 13 MJ and the APU time constant is 5 min. The simulated pack energy and APU output power are shown in fig. 4.15.
The average demanded vehicle power is 7.05 kW and the average APU output is the same. The resulting mileage is 63 mpg, double that for the switching case. The high mileage is due to the absence of battery loss due to charging and discharging. Also, battery cycling has been eliminated.

The second example is now repeated for the tracking case. The only difference is that the cycle has now been repeated 27 times instead of just nine. This was done so that steady-state could be achieved with respect to the APU. The maximum APU output is 30 kW, the pack size is 16.3 MJ and the APU time constant is five minutes. The power out of the pack is shown in fig. 4.16 and the APU output power and pack energy are shown in fig. 4.17.
The average APU output power is 6.4 kW giving a mileage of 26.7 mpg compared to the 19.9 mpg for the switching case. Battery cycling has also been all but eliminated.
4.4.3 Discrete tracking APU

The discrete tracking APU is one whose output can take on a range of discrete power values. The first example of constant 40 mph will now be repeated with a discrete tracking APU. The maximum APU output is 30 kW, the switching time constant is 30s, and the APU can only produce multiples of 2000 W. All other parameters remain the same. Figure 4.18 shows the pack energy and the APU output for the discrete tracking APU. The resulting average APU output power is 7.18 kW and the mileage is 61.9 mpg. Even with 2 kW resolution, this APU can almost match the continuous tracking APU.
4.5 Conclusions

So, apparently, the tracking APU is quite superior to the switching APU. The tracking APU supports increased efficiency and almost eliminates battery cycling altogether. Also, for the switching APU, the pack size is determined by battery energy density. The capacity must be large enough to prevent excessive cycling. This will probably mean that the pack is oversized with regard to power requirements. With the tracking case, the battery pack size is based on power requirements which may imply a pack of reduced size.
The efficiency of the switching APU HEV decreases even more as the maximum APU output power increases. However, this decrease in efficiency must be tolerated if the vehicle top speed is to be increased. Figure 4.x shows this behavior explicitly.

![Figure 4.x Mileage for tracking and switching APUs.](image)

The heavy line represents an ideal tracking APU with a 45 kW maximum output. The lighter lines represent switching APUs with various maximum output powers. The vertical portions represent the highest steady state speed that may be supported indefinitely at a given APU power level. The mileage of an HEV with these particular APU implementations is shown vs. the vehicle steady-state velocity. As would be expected, the efficiency of the switching APUs matches that of the tracking APU only when the vehicle demanded power is such that the switching APU is on continuously.
Continuous tracking may present some control problems. Varying ICE parameters in order to vary APU output power might prove to be difficult. Chapter 5 is devoted to this issue. If continuous tracking is not possible an approximation can be made by implementing a discrete tracking APU with several power steps. However, in some cases this might not prove to be any easier than implementing the continuous tracking system.
CHAPTER 5 IMPLEMENTATION ISSUES

The tracking APU was shown in chapter 4 to be superior to the conventional switching APU owing to the reduction of both battery pack losses and battery cycling. The only drawback is that the implementation of a tracking APU is more complex than the switching APU. The switching case requires holding stoichiometric conditions at only one operating point while the tracking case requires stoichiometric conditions over a continuous range of operating points. However, if the ICE is allowed to change only in a quasi-static manner, then tracking becomes feasible. A prototype of this nature has been constructed for the University of Illinois’ HEV and will be described in detail in the following sections.

5.1 ICE fuel injection system

The ICE used in the University of Illinois’ HEV is a Kawasaki FD 620D small industrial gasoline powered V-twin engine. The engine was originally equipped with a single throttle body fuel injection unit. The engine had a split intake manifold and a single fuel injector to supply both pistons. Since the engine has four stroke action, each piston receives an injection of fuel every other revolution of the crank shaft. This original fuel injection system suffered from fuel maldistribution between the pistons and was
eventually replaced by a dual runner multiport fuel injection system [8]. A system then had to be designed that could drive two fuel injectors instead of the one.

A fuel injector is simply a fuel valve actuated by an electromechanical solenoid. When the coil is energized, the valve opens and allows pressurized fuel to spray into the cylinder. The duration of the valve opening determines the amount of fuel supplied. This duration in turn is determined by the on time, or pulse width, of the excitation signal supplied to the solenoid. There must now be some way of calculating the injector pulse width to obtain the correct air-fuel mix. This is generally accomplished by using fuel maps accompanied by exhaust oxygen sensor feedback. A fuel map is essentially a look-up table of injector pulse width values vs. engine operating parameters. Use of the map gives an approximate pulse width value needed to obtain the correct fuel mix. In late-model cars, exhaust oxygen sensor feedback is used to fine tune the proper air-fuel ratio.

Oxygen sensors are placed in vehicle exhaust systems in an attempt to obtain the oxygen content present in the exhaust gasses. The sensors deliver a voltage which is related to the air/fuel ratio. Feedback from the oxygen sensors is used to control the fuel injection system to try to achieve stoichiometric operation. Stoichiometric engine operation occurs when the air/fuel ratio is such that there is just enough oxygen present to oxidize all of the fuel. This results in high efficiency and reduced exhaust emission gasses. Now, define [9]

$$\lambda = \frac{(F / A)_{\text{actual}}}{(F / A)_s} \quad (5.1)$$

as the air/fuel equivalence ratio where \((F/A)_{\text{actual}}\) is the actual air/fuel ratio and \((F/A)_s\) is the stoichiometric air/fuel ratio. This implies that at stoichiometric engine operation,
$\lambda = 1$. When $\lambda < 1$, the air/fuel mixture is said to be lean, meaning that there is not enough fuel present. If $\lambda > 1$, there is excess fuel and the mixture is said to be rich.

It is unfortunate that the oxygen sensor output voltage is not a linear function of $\lambda$. Instead, the output voltage is given by [9]

$$\frac{\text{V}_o}{\text{RT}} = \frac{\text{p}''_{\text{O}_2}}{\text{p}'_{\text{O}_2}}$$

(5.2)

where $\text{V}_o$ is the sensor output voltage, $R$ is the universal gas constant, $T$ is the temperature, $F$ is the Faraday constant, $p''_{\text{O}_2}$ is the ambient oxygen partial pressure and $p'_{\text{O}_2}$ is the equilibrium oxygen partial pressure in the exhaust stream. The oxygen partial pressure in the exhaust stream is a strong function of $\lambda$ near $\lambda = 1$. This means that $\text{V}_o$ is a steep function of $\lambda$ near $\lambda = 1$. The relationship between $\text{V}_o$ and $\lambda$ looks very much like that of figure 5.1.

![Figure 5.1 O2 sensor output voltage vs. equivalence ratio.](image-url)
This steep curve presents an extreme problem in using $V_o$ to control the fuel injection. Even small fluctuations in the oxygen content in the exhaust gases can cause the output voltage to swing full scale. As a result, direct feedback of the oxygen sensor output is not useful. This method was attempted and was not successful. Most present day automobiles use the oxygen sensor as a discrete rather than an analog sensor for exactly this reason. It is used as an indicator to determine whether the air/fuel mixture is rich or lean, but not the extent to which it is rich or lean. The injection duration is dithered around the $\lambda=1$ value to produce stoichiometric mixtures over intervals of many cycles. In the end, a slightly different approach was taken. The oxygen sensor output is averaged and integrated, and the result used as a control signal. The system block diagram is shown in figure 5.2.

![System Block Diagram](image)

**Figure 5.2 Fuel injection control system.**

The control system shown in fig. 5.2 shows this integral control approach to force the oxygen sensor output to track the reference input voltage $V_{ref}$. The output of the integrator controls the pulse-width of the output of a pulse generation circuit. The fuel injectors are then driven by the pulse generator. The injectors, aside from having to inject
the correct amount of fuel, must also inject at the proper time. The timing is determined by the stock electronic control module (ECM). As was mentioned above, the engine originally had a single fuel injector to feed both cylinders. This means that the stock ECM would provide a fuel injection pulse for every revolution of the crank shaft. This is not the case for two injectors; each must inject every other revolution of the crank shaft. Therefore, the pulse train from the stock ECM was split, a timing pulse going to each injector driver.

The low pass filter averages the oxygen sensor output. The time constant of the filter is approximately five seconds. The same is true for the integrator time constant. This means that the one should not change the operating conditions of the engine over time scales of ten seconds or so. If the engine's throttle is changed too fast, the control circuit will not be able to enforce stoichiometric operation. So long as the engine operates in a quasi-steady-state fashion, stoichiometric conditions may be maintained. This implies that this approach is not useful for conventional automobiles for which the fast driving dynamics are directly coupled to the engine dynamics. However, in a series HEV application, a ten second response time is not a performance concern.

Fortunately, unless the engine stalls, this system is completely stable. Even if the \( \lambda \) is forced lean or rich to the extent that the \( V_o \) saturates, the system will recover given sufficient time. In addition, \( \lambda \) can be driven to any value, at least over the range 0.9 to 1.1, not just the stoichiometric value. The value of steady-state value of \( \lambda \) may be set at any value by simply adjusting \( V_{ref} \).

In summary, a conversion efficiency of 220 g/kW-hr was obtained over range of power values from 6 kW to 12 kW and a range of speeds from 2000 RPM to 2300 RPM.
Using this control method, it possible to set $\lambda$ to a precise value at or near $\lambda=1$. It is also possible to set $\lambda$ to other specific values in order to meet catalyst, exhaust temperature, or power requirements. The maximum achievable output power was found to be 16 kW. For a more detailed analysis of the results given here, refer to [9].
Figure A1. Fuel injector pulse-width generation circuit.
Figure A3 Fuel injection integral control circuit.
REFERENCES


Figure A2 Fuel injector driver circuits.