29 Jan 1993

Optimizing Energy Usage in a Solar-Powered Vehicle

William R. Macneil

OPTIMIZING ENERGY USAGE IN A SOLAR-POWERED VEHICLE

FINAL REPORT FOR OURE

W. R. Macneil
Electrical Engineering, UMR
3/1/93

INTRODUCTION TO THE PROBLEM

The University of Missouri - Rolla (UMR) will be entering SUNRAYCE '93 and the World Solar Challenge and expects to excel in these solar-powered car races. Some of the factors involved in winning a race include efficiency, reliability, and high power. All of these are important considerations, but will be ineffective unless appropriately harnessed by a power management scheme appropriate to a race strategy. The power management scheme will be used to efficiently and effectively distribute available power to the car's systems throughout a race day. My goal was to produce a software framework into which the specifications and performance models of the UMR car, Sol Survivor, can be inserted as they evolve. Data is taken from a telemetry system and processed to produce "fuel gauges" for the driver. As of yet, a final vehicle is not available, so the specifications are estimates based upon worst-case scenarios. Detail is added to the software model as work proceeds.

Nature of the SUNRAYCE '93 event

SUNRAYCE '93 is a staged road rally over a seven day period. A winning time is determined by the total accumulated time over the week. If a day's portion of the race is not completed, a total time for that day is calculated. Daily distances range from 110 to 195 miles for a total of 1099 miles from Dallas to Minneapolis. Approximately 80% of the route has a 55 mph speed limit and much of the route consists of relatively smooth asphalt rural secondary roads with grades less than 4%. Short stretches of up to 8% grade are also present.

Available Power

The total power available to the car is one of the most important factors in vehicle performance, as has been the case for many previous solar car races. Power is available from sunlight and can be either transmitted to the motor to propel the vehicle or can be stored in the battery for later use. Power can be returned from the road via regenerative braking. The regenerative capabilities of our selected motor are not yet available.

Using a flat panel subset of an array, the power available from the sun at a specific time of the day for a surface area of one cell was calculated and made proportional to the effective surface area available on the actual array. A plot of this data taken on April 28, 1992 and a polynomial curve fit is shown on the graph in Figure 1.

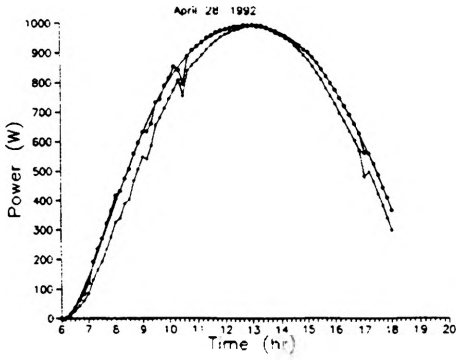


Figure 1. Solar Panel Power Output for Daylight Hours.

The slight spike-like dips in the curve are due to clouds passing overhead. Those points were removed from the data set before applying a curve fit to imitate ideal conditions. The curve fit equation used to estimate the instantaneous power available from the array during the day is given in Equation 1.

$$P(t) = 0.00882316\ t^6 - 0.68036\ t^5 + 21.4325\ t^4 - 352.414\ t^3 + 3159.69\ t^2 - 14330.4\ t + 25444.8 \quad (1)$$

where P is power in watts and t is time in hours on the 24 hour clock. The total charge acquired by the battery over a period of time, T, beginning at time, t, is given by Equation 2.

$$Q = \frac{t_{end} - t_{begin}}{Volts} \int_{t_{begin}}^{t_{end}} P(t)\partial t \quad (2)$$

where Q is charge in coulombs, V is voltage, t is duration of charging in seconds, and t_{begin} and t_{end} are the beginning and end times of charging (in seconds but based on the 24 clock). In our model, we will use interpolation to approximate the power curve. Summation is used to determine charge accumulation in the software which is covered after the next section.

HARDWARE CONFIGURATION

Five kilowatt hours of battery capacity is allowed for each vehicle in Sunrayce '93. These batteries must be of the lead-acid type. The lead-acid restriction is not present for the World Solar Challenge. Each team may begin the race with their batteries fully charged. The only charging methods allowed during the race are those derived from sunlight and regenerative braking. Solar energy is supplied by five separate solar panels, each of which requires its own maximum power point tracker (MPPT). There needs to be at least one MPPT per array section but more than that can be useful only up to the point at which the benefits are outweighed by the problems associated such as reliability and weight. For the purposes here, the number of MPPTs is of no concern. The model is simplified for the power management software by taking only the sum of the power coming from the array. That sum is the only useful value from the array once the particular hardware configuration is set. The basic power systems are laid out in Figure 3.

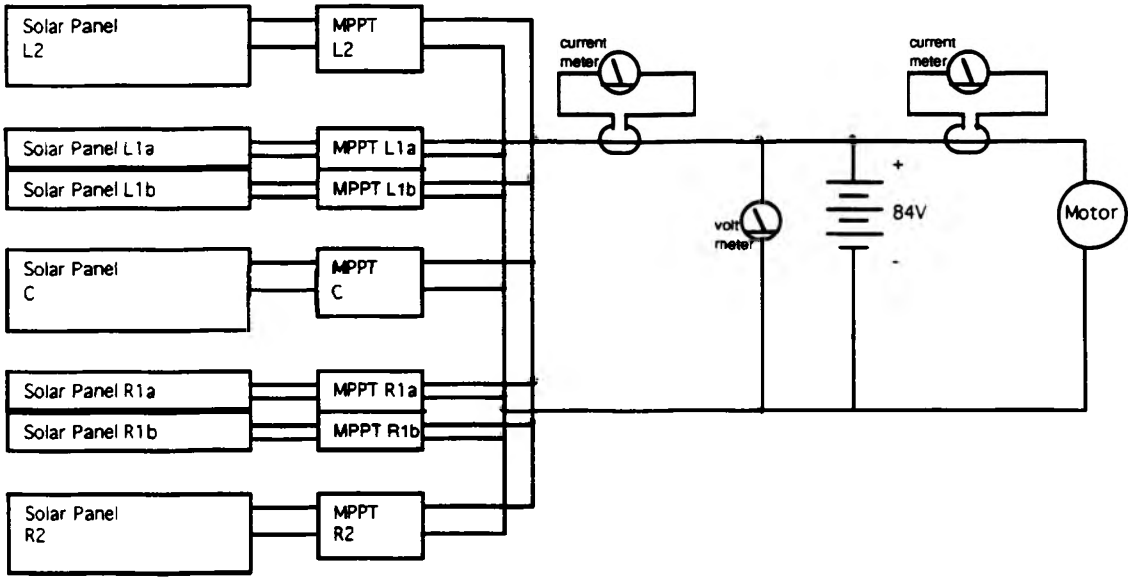


Figure 3. Main Power System for the UMR Solar Car.

The circle labeled motor on the diagram normally represents the motor controller and motor. The motor can be connected exactly as shown in the case of controller failure. There will also be backup units for the motor and controller. For the purposes of the software, the motor can be represented as one unit with controller transparent as above. Current measurements will be done by hall effect current sensors in order to eliminate loss.

Telemetry System

The easiest way to gain any benefits from the power management software is to use a fully automatic telemetry system. If necessary, data may be keyed in manually received by voice transmission. Manual data entry is not very efficient so automatic error correction schemes are being considered to avoid the fallback plan. The power management computer may also be put

inside Sol Survivor but the current plan is to use a radio based telemetry system.

Data from the systems on the car will be used two ways. Before the race, data will be accumulated from the car in order to construct a realistic model of vehicle performance. This data will be analyzed to aid in optimizing the performance of the vehicle during the race. Algorithms for optimization can checked for applicability. At the very least, the telemetry system will be used to create a gauging system to track system efficiencies and other instantaneous calculations. Race route time and energy consumption projections will also occur during the race to establish current consumption boundaries for the driver of the solar car.

Most of the data acquired by the telemetry system is straight-forward. An array of analog inputs are polled by a relay based multiplexer at set intervals and run through an analog to digital converter. A few other pieces of data originate from devices like digital counters such as the odometer/speedometer.

The first step in designing this system is to determine its requirements. They are:

- Physical wheel tracking method
- Wheel pulse recording device
- Recorded pulses transmission scheme
- Transmitted data interpretation
- Application of interpreted data

To accommodate all these requirements in a simple way, the system was designed as follows:

First a method of tracking the wheel is needed. I began experimenting with an optical shaft encoder. Very simply, this would consist of an IR transmitter/receiver pair which would provide pulses when the IR beam is allowed to pass through slots in the wheel. The pulse from the receiver would need to be conditioned to become a TTL-level square wave pulse. This scheme would allow high resolution wheel tracking because a large number of slots could be placed in an opaque disc on the wheel. However there is the problem of ambient light causing false signals. While researching this prospect, I discovered an IR receiver unit commonly used in VCR remote control receivers. It filters out all signals not in the IR bandwidth or signals not modulated at 40 kHz and produces clean square wave pulses at the output. The main remaining problem is that there is continual power consumption by the IR LED.

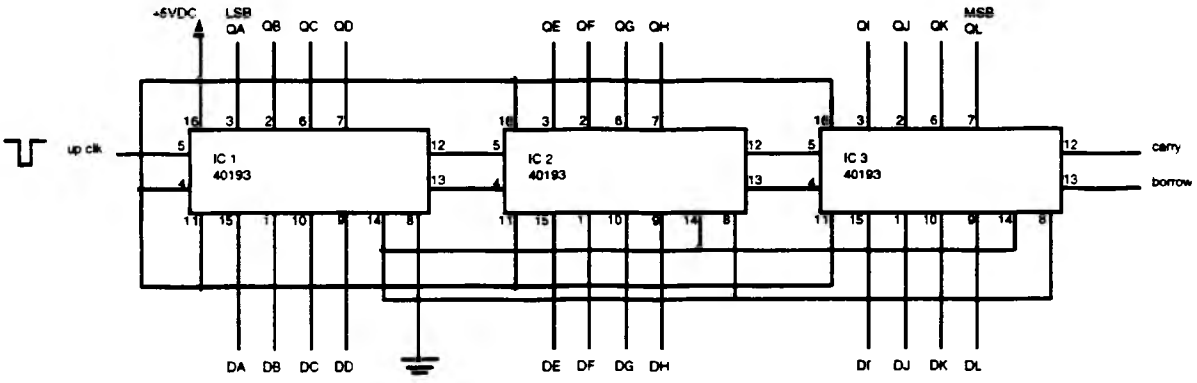
An alternative to optical encoding is the use of a Hall effect sensor used to track a magnet on one or more radial locations on the wheel. This is the method used by most commercial bicycle speedometers. The total sensor circuit would require less power and a breadboard mockup to fully try out the concept is currently under evaluation.

Individual wheel pulses cannot all be seen by the data acquisition hardware because pulses could be missed if the data acquisition system is polling different channels during the duration of the wheel pulse. The wheel pulses will then be counted by a 12-bit synchronous counter whose output can be simply polled by the telemetry system. Polling the output of a counting circuit can be unreliable if polling occurs during transition, so a synchronous counter was used to avoid timing glitches.

The 12 bit counter can only cycle through a sequence of 0-4095. 4096 revolutions of 20 inch wheels on the vehicle results in a raw trackable

distance of 4.062 miles. That distance will be evaluated once there is a working car to compensate for any depression in the tires that might cause a deviation from the 20 inch measurement. The software described later adjusts for "flipping" of the counter to produce the real odometer reading in software. In order for this flipping to be correctly interpreted, samples from the counter must be read by the software often enough that no more than one full cycle of the range of the counter takes place before the next sample is read. The counter itself (Figure 4) is made up of three 4-bit synchronous counters cascaded together.

Figure 4. 12-bit synchronous counter. (All inputs DA-DL are tied to GND)



Removal of one of these chips would result a counter cycle of too few pulses and risk a counter cycle being missed by the software. The speed of the telemetry transmissions should be fast enough for ordinary operation using a smaller counter limit, but the extra margin allows short user interrupts to not also require an odometer reset.

An offset can be added to the odometer to compensate for recalculations of position by causing a keyboard interrupt. The program will then prompt the user for a new odometer reading or odometer offset to correct any deviations from the race course and provide a reset function to the program.

The subroutines written for use of the odometer/speedometer are recorded below.

```
function convertUnit (n: real; wasUnit, toUnit: unitype): real;
begin
 case (wasUnit) of
  'm': n := n;                    {* convert from meters (to meters)*}
  'k': n := n * 1000;             {* convert from km *}
  'i': n := n * 1609;             {* convert from miles *}
  'f': n := n * 1609 / 5280;      {* convert from feet *}
  'g': n := n * 0.9;              {* convert from grads (to degrees)*}
  '^': n := n;                    {* convert from degrees *}
  'r': n := n * 180 / Pi;         {* convert from radians *}
  's': n := n;                    {* from seconds to seconds *}
  'n': n := n * 60;               {* from minutes *}
  'h': n := n * 3600;             {* from hours *}
 end;
 case (toUnit) of
  'm': convertUnit := n;                {* convert to meters *}
  'k': convertUnit := n / 1000;         {* convert to km *}
  'i': convertUnit := n / 1609;         {* convert to miles*}
```

```
  'f': convertUnit := n * 5280 / 1609; {* convert to feet *}
  '%': convertUnit := n / 0.9;         {* convert to grads *}
  '^': convertUnit := n;               {* convert to degrees *}
  'r': convertUnit := n * Pi / 180;    {* convert to radians *}
  's': convertUnit := n;               {* to seconds *}
  'n': convertUnit := n / 60;          {* to minutes *}
  'h': convertUnit := n / 3600;        {* to hours *}
 end;
end;

function todaySeconds: real;
 var
  hour, minute, second, sec100: word;
begin
 GetTime(hour, minute, second, sec100);
 todaySeconds := 3600 * hour + 60 * minute + second + sec100 / 100;
end;

procedure decodeOdometerTelemetry (msbyt, lsbyt: byttype; var
            lastSampleTime, odometer, RPM: real);
 var
  revs, lastrevs, seconds: real;
begin
 seconds := todaySeconds - lastSampleTime;
 lastSampleTime := seconds + lastSampleTime;
 revs := 256 * msbyt + lsbyt;
 lastrevs := (convertUnit(odometer, 'm', 'r'));
 lastrevs := lastrevs - (4096 * trunc(lastrevs / 4096));
 if (revs > lastrevs) then
  begin
   odometer := odometer + (convertUnit((revs - lastrevs), 'r', 'm'));
   RPM := (revs - lastrevs) * 60 / seconds;
  end
 else
  begin
   odometer := odometer + convertUnit((4096 + revs - lastrevs), 'r',    'm');
   RPM := (4096 + revs - lastrevs) * 60 / seconds;
  end;
end;
```

Analog signals from sensors on the solar car are fed through a relay based 64 channel multiplexer. All the data is changed to either 8, 12, or 16 bit binary. This data acquisition system will allow a mixture of analog and digital channels but for the solar car's on board computer's purposes it will see all channels as digital.

The solar car's on board computer will consist of a single board Motorola 68HC11 prototyping board. The primary purpose for this computer will to be to channel data from the data acquisition system through the wireless serial data link to the chase vehicle computer.

The wireless serial data link will consist of a 9600 baud transmitter/receiver pair manufactured by Motorola. The data sending protocol functions as a normal wired serial link. Standard modem communication protocol can be used.

The chase vehicle computer will consist of a DOS based laptop. A storage device capable of containing all the race route topological and logistical data is needed. The main processor should be at least on the order of a high speed Intel 80486 or better. The bulk of the power optimization software will run here.

Once the data has been received at the chase vehicle computer, that data will be processed and gauges will be presented on laptop screen. Some of the

completed gauges include all raw data from telemetry as well as several calculated values including battery range in miles, maximum sustainable velocity on solar power alone, and basic system efficiency. These gauges will be described in the next section.

SOFTWARE CONFIGURATION

Each cycle of operation the software receives data via the telemetry equipment. This data is displayed to the user and processed to provide gauges to the user providing indications of economy and performance. The basic flowchart is shown in Figure 5.

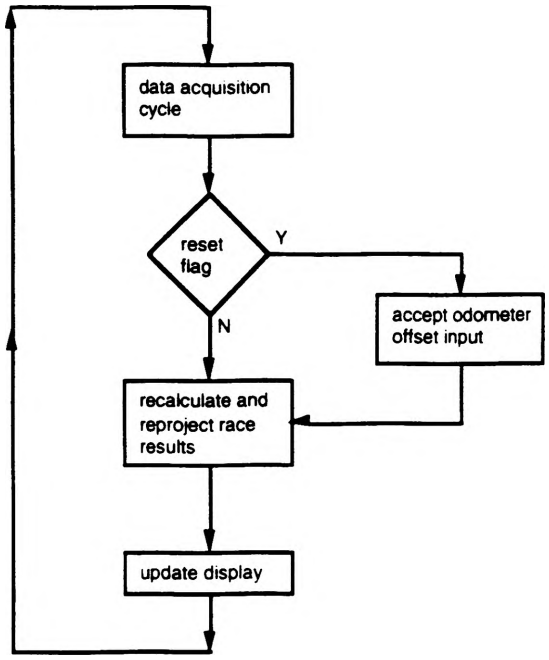


Figure 5. Power Optimization Software Flowchart.

In the event of an unscheduled detour, it is possible for the internal position of the car to be misplaced. In this case the user can reset the program to the correct position by an odometer offset to avoid projected race miscalculations.

Gauges and Corresponding Calculations

An obvious gauge that the race team would want to have access to is a calculation of instantaneous sustainable velocity supported by solar power only. Using Equation 3 supplied during the EE301e course:

$$F = m\,a + m\,g\,(k_r \cos\Theta + \sin\Theta) + \rho\ C_d\,A(v+w)^2\,(0.05097) \quad (3)$$

the road force applied for a particular instant can be calculated. The variables from the equation including the estimates for constants being used for the current model of the car are as follows:

F	road force	(N)
v	velocity	(m/s)
w	opposing wind velocity	(m/s)
Θ	slope	(degrees)
a	acceleration	(m/s^2)
g	acceleration due to gravity	9.8 m/s^2
m	mass	343 kg (760 lbs)
C_d	drag coefficient	0.15
A	frontal area	1.3 m^2
k_r	rolling resistance coefficient	0.0065
ρ	air density	1293 kg/m^3

From this the power requirements for the vehicle moving at a constant velocity are found by Equation 4.

$$P = F\,v \tag{4}$$

By inserting the road force equation into the above equation and solving for v using Mathematica, the maximum velocity sustainable by a specific power input is given in the following function.

```
function pwrOf(a,b:real):real;
    {* a**b *}
    begin
        pwrOf:=Exp(Ln(a)*b);
        end;

function vFromP(p,wind,theta,gravity,mass,Cd,Area,kr,airD:real):real;
    {* instantaneous v at a given P *}
    {* Solve[f v^3 + g v^2 + h v - p == 0,v] *}
    var x,y,f,g,h:real;
    begin
        x:=mass*gravity*(kr*Cos(theta) + Sin(theta));
        y:=0.5 * airD * Cd * Area;
        f:=y;
        g:=2 * y * wind;
        h:=x + (y * wind *wind);
        vFromP:= (-g/3.0/f) - (pwrOf(2,(1/3)) * (-(g*g) + 3*f*h)) /
            (3*f* pwrOf( (-2*g*g*g + 9*f*g*h + 27*f*f*p +
             pwrOf(3,(3/2)) * f * Sqrt(-(g*g*h*h) + 4*f*h*h*h - 4*g*g*g*p -
            18*f*g*h*p + 27*f*f*p*p))    ,(1/3) )) +
            (pwrOf( (-2*g*g*g + 9*f*g*h + 27*f*f*p +
            pwrOf(3,(3/2)) * f * Sqrt(-(g*g*h*h) + 4*f*h*h*h - 4*g*g*g*p +
            18*f*g*h*p + 27*f*f*p*p) )    ,(1/3)  ) /(3* pwrOf(2,(1/3)) *f));
        end;
```

It is also desirable to have gauges relating system efficiency. It would allow the driver to know parameters that keep him within a high-efficiency range. It would also be useful in identifying power losses in the drive system. Overall drive-system efficiency can be found by dividing the power output

calculated from vehicle speed by the electrical power consumption of the motor. This calculation is performed in this section of code:

```
function roadForce(var a,theta,v,wind,airD:real):real;
      {* Returns force required to propel a vehicle under specified
      conditions of acceleration, rolling resistance,
      and aero drag in Newtons*}
      var accelF,rollF,dragF,m,g,kr,Cd,Area:real;
      begin
           m:=(290+170+300)/2.2; {*lb's converted to kg *}
            g:=9.8; {* m/s^2*}
           kr:=0.0065;
           Cd:=0.15;
           Area:=1.3;
           accelF:=m * a;
           rollF:=m*g*(kr*Cos(theta) + Sin(theta));
           dragF:=airD*Cd*Area*Sqr(v+wind)*0.5;
           roadForce:=accelF + rollF + dragF;
           end;

function driveEfficiency (a, theta, RPM, wind, airD, Volts, Imot: real): real;
      var
            mechPwr, elecPwr,v: real;
      begin
            v := RPM * 1609 / 190080; {* m/s *}
            mechPwr := v * roadForce(a, theta, v, wind, airD);
            elecPwr := Volts * Imot;
            driveEfficiency := mechPwr / elecPwr;
            end;
```

The results of these functions have been added to the screen display.

POWER OPTIMIZATION

The first step in power optimization is to calculate the energy usage during the day. Before this is possible a reasonable profile of the race course for that day must be available to the software. Other members of the Team will drive the race route to accumulate data. Samples will be taken every six feet and the data will include factors such as hill grade.

Calculations done on the data are done as estimates. Since the samples are taken every five feet it is reasonable to apply instantaneous values from those points to all points within those next five feet. Solar energy input, for example, can be estimated safely in this way. Hill grade or slope will also be applied to the five foot range as well. This idea is illustrated in Figure 6.

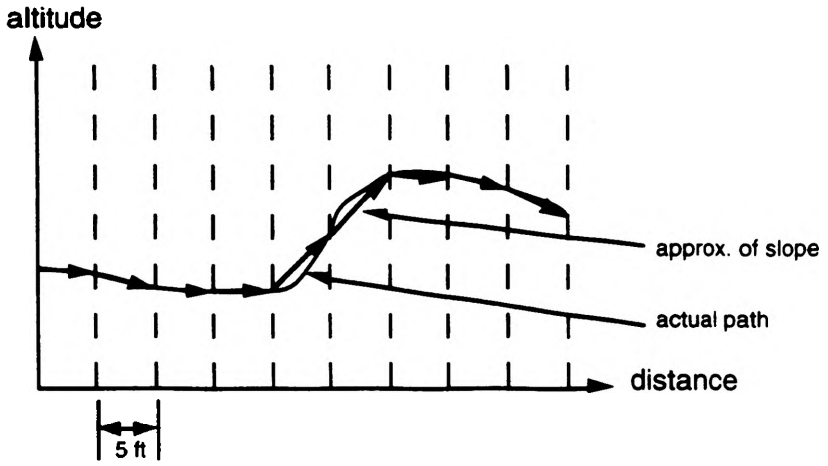


Figure 6. Race course data curve estimation.

The telemetry routines will keep track of the position of the vehicle on the race route and each reprojection will begin at that position using the current time and battery level. From that starting point the software will step through the remaining data points until the finish line. The projection software procedure is shown below:

```
procedure ReProjectSetV (var v, projBat, projTime, projOdom: real; hereStep, lastStep:
integer;      gravity, mass, Cd, Area, kr: real);
var
      theta, dist, a, wind, cloudCover, airD, F, mechP, driveEff, solarP, motorElecP:
      real;
      stepTime: real;
      index: longint;
begin
      {* Race route projection is done on the basis data to be recorded by a scout
            vehicle. *}
      {* This vehicle will drive the route a couple months before the race and take
            *}
      {* measurements of grade every 6 feet.  projBat in Joules.  projTime in
            seconds.*}
      {* set v *}
      cloudCover := 0.0;
      for index := hereStep to lastStep do
            begin
                  theta := convertUnit(grade(index), '%', 'r');
                  dist := convertUnit(6, 'f', 'm');
                  a := 0.0;
                  wind := 0.0;
                  airD := airDensity;
                  F := roadForce(a, theta, v, wind, airD, gravity, mass, Cd, Area,
                        kr);
                  mechP := F * v;
                  driveEff := 0.9;
                  motorElecP := mechP / driveEff;
                  solarP := projArrayPwr(projTime, theta, cloudCover);
                  stepTime := dist / v;
                  projBat := projBat - (stepTime * motorElecP) + (stepTime *
                        solarP);
                  projTime := projTime + stepTime;
                  projOdom := projOdom + dist;
            end;
end;
```

Using the projection methods shown above, the software could use an iterative method to find the best constant value. For projection estimates using a set velocity, the initial try could be the speed limit. The next try should be half that. This bisection method should continue until the software has narrowed in on a speed range acceptable to the user. The goal each time is to find the shortest possible elapsed time that does not drain the battery past a set safety limit. Depending on the type of battery used (as this will vary from race to race) the safety limit will be different. Lead-acid batteries will be used at Sunrayce'93 in all solar cars. In our strategy, enough time is available each day to begin the next racing day fully charged. An arbitrary value can be used until the completed battery model is used.

For projection estimates using a set battery drain, or a set motor current, the constant power setting will also be approximated using bisection method. The initial (guess) value will be the maximum possible power draw, that is the power draw at maximum acceleration.

Each sampling cycle, a variety of optimization methods could be tried and the best results of those will be displayed to the user, to be conveyed to the driver. In addition to a power or speed boundary, the user will receive gauges presenting elapsed time to finish and foreseeable range of the vehicle.

The driver will then try to maintain the parameters given to him such as upper and lower speed limits and/or constant battery power drain. The driver will of course have to vary these with some discretion to compensate for stop lights and other traffic interruptions. As the day goes on, and the car gets closer to that day's finish line, the instructions given to the driver will reflect an increasingly improving model.

RESULTS AND CONCLUSIONS

It will, of course, be necessary to finish development of the power management algorithms and extract data from the telemetry system before a final conclusions can be given; however, this is a continuing study.

The results of the power management program will be seen through the results of Sunrayce '93. Development of this program will continue over the next several as the final vehicle is completed.

FURTHER STUDY

As in all computer simulations, there is always a limit as to how accurate the results can be predicted. Part of this inaccuracy is due to a limit on how well factors can be measured or predicted. For example, hill grade can be measured precisely for a specific point on the race course. On the other hand, cloud cover is a very difficult factor to predict accurately except in the very short term. Additionally, an unexpected detour in the route would throw off projected results until the program is alerted of the change.

The basic shell of the simulation program has been completed as a framework for which to add factors that lend increasing degrees of accuracy. In order for the program to be complete, two major additional problems need to be dealt with:

1) The car itself needs to be completed so the final hardware parameters for size and weight and other similar factors can be measured and mathematically modeled. The finally selected batteries and motors must be evaluated in a "hands-on" fashion so precise models can calculated. The

current software uses an ideal battery model which puts no limits on current draw. Additionally, battery behavior is extremely complex because the history of the battery determines future behavior.

2) The entire race route needs to be charted. Data taken from the actual route itself will be used for the program to continually replot the projected time and energy consumption during the race in order to warn the Team of potential power mismanagement hazards and to attempt to calculate optimized speeds for the course at different points.

The route will actually be recorded by an array of instruments placed inside the support van that will be driven on the route. Every five feet, a 20 inch bicycle wheel will trail from the rear of the van recording RPMs. At this same sampling frequency, hill grade, and electronic compass readings will also be taken. Cloud cover can only be effectively dealt with on the short term (ex: daily projection) Daily weather pattern predictions are expected to be available via internet news updates, accessed on the road by the support vehicles on-board computer with a cellular modem.

ACKNOWLEDGMENTS

Many people were involved in the Solar Car Project here at UMR. There were a large number of people involved in the design and construction of the car and several people responsible for allowing us to even have a project on this campus such as our advisor, Dr. Norman Cox. I would also like to mention some of the other people who contributed to making the Power Management System as a whole possible, such as the Student Project Leader Jeff Shapiro and Dwayne Jolley who are preparing to gather the topology and other race route information; Matt Spaethe, Rob Zeigler, and Jeff Shapiro who determined the solar cell model; Rick Jenkins, John Stone, Chuck Kincy, Rob Zeigler who designed and built the 68HC11 aspect of the data acquisition system on the car; and Tom Sullivan and Shawn Thompson who organized the Electrical Committee and who provided some of the telemetry equipment information.

REFERENCES

GM SunRaycer Case History. General Motors Electric Vehicles Division., 1992.

Sunrayce '93 Regulations March 1992. US Department of Energy, National Renewable Energy Laboratory., 1992.