Optimization of an Electric Bicycle

ME 555

April 22, 2016

Prepared By:

Isobel Kraft

Damiete Samuel-Horsfall

Advisor:

Dr. Bayrak

INTRODUCTION

Project Background

We chose to optimize an electric bicycle (EB) to maximize the range, or distance the bicycle can travel on a single charge. This electric bicycle consists of an generic road bike fitted with an electric motor mounted to the rear wheel and a battery housed on the down tube of the frame (between the legs). We chose to optimize range because, based on electric vehicle research, "range anxiety" is one of the main reasons why people choose to go with traditional vehicles. The same phenomenon can be applied to electric bicycles. If we wish to present a fully-electric bicycle, we want a design that will allow our customer to complete their daily bicycle commute without the need to pedal. Of course, the electric bicycle is able to be pedaled if the the motor cannot power the rider to their destination. However, we believe that by optimizing range, we will allow the customer to reach their destination emission-free and effortlessly.

We divided the system into two sub-systems: mechanical and electrical. The mechanical subsystem consists of the motor design and the electrical subsystem consists of the battery design. Since the system is optimized for maximum range, we also optimize each subsystem with that overall goal in mind. Both the motor and the battery are individually optimized to minimize battery charge loss. Optimizing charge loss allows the EB motor to be highly efficient and use power as "cleanly" as possible to lengthen the capability of the vehicle to be powered by the electric motor. Optimizing charge loss also allows the EB battery to discharge for as long a period as possible, thus allowing the EB to travel further. These subsystems are constrained by a number of aspects that restrict the optimization of each subsystem. The constraints, design variables, and parameters will be explained in the following sections. Each subsystem section will also present the models, the optimization method and optimization results.

To optimize at the system-level, we integrated the subsystems and optimized the system as a combination of the subsystem design variables and constraints. We coordinated the system using multidisciplinary feasible (MDF) method. The layout is seen in Figure 1 and the details of the system-level problem formulation, modeling, optimization method and results are in the System section of the report.

System Layout

Figure 1 below shows the system layout and MDF coordination. The subsystems, the motor and battery, are combined and linked by the subsystem objectives (battery charge loss). The system optimizer feeds the design variables to the linked, system-level objective function and returns the system objective function value and constraints. The motor subsystem uses MATLAB fmincon optimizer, the battery subsystem uses MATLAB Genetic Algorithm, and the system optimizer is MATLAB Genetic Algorithm. The optimization methods will be explained in greater detail later in the report.

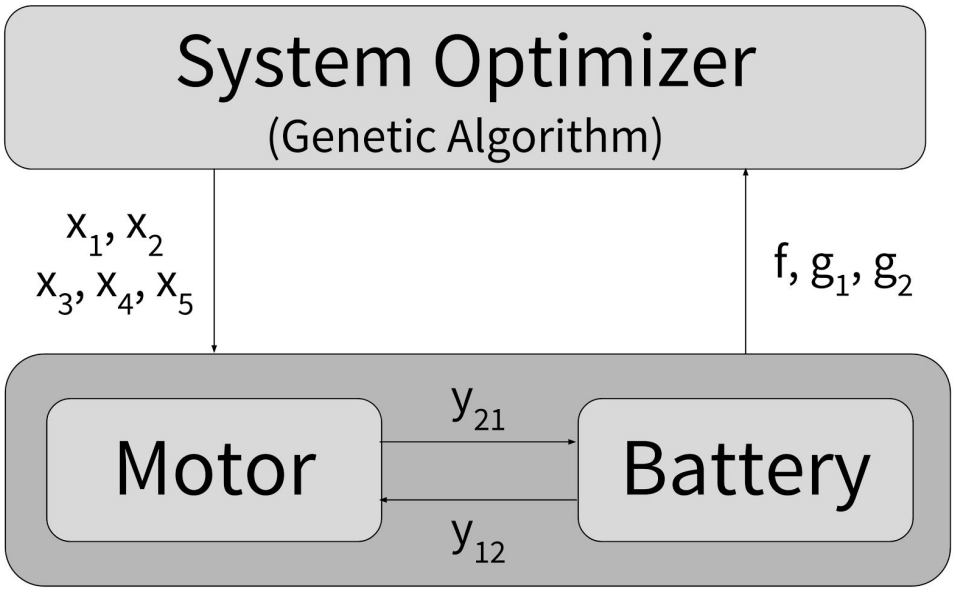

Figure 1: MDF System Coordination

SUBSYSTEM

Battery

Problem Formulation

Objective Function

The objective function for the battery subsystem is to minimize the battery charge loss. This means that the battery will discharge at the slowest rate possible to allow the EB to travel for a long a period as possible, reaching the furthest range possible. The battery charge variable is housed in the model as "State of Charge" (SoC).

$$\min_{x_1, x_2} \text{battery charge loss} = \Delta SoC = SoC_{initial} - SoC_{final}$$

In this project, the objective function value acts as a linking variable between the battery subsystem and the motor subsystem. The motor subsystem is also optimizing to minimize battery charge loss. The system-level optimization, discussed later in the report, shows how the subsystems were coordinated with this linking variable.

Design Variables

The variables for the objective function are limited to the variables that can be accessed given the model used for the subsystem. Out of the variables available, we chose variables that are aspects of the battery architecture: number of cells in series and number of branches in parallel. The number of cells in series increases the total voltage of the battery by 3.2V per cell. The number of branches in parallel increases the capacity of the battery by a factor of 0.5 Amp · hour (Ah) per branch.

Design Variable		Unit	Type
x_1	N_{series}, Number of cells in series	--	Discrete
x_2	$N_{parallel}$, Number of branches in parallel	--	Discrete

Table 1: Battery Subsystem Design Variables

Constraints

The constraints for the battery subsystem are integer constraints on the battery architecture variables. The number of cells is series is limited to 8 cells and the number of branches in parallel is limited to 12 branches. The cell constraint was determined by the geometric constraints of the battery pack housing. The battery pack will be affixed to the down tube of the bicycle that is approximately a 2-inch diameter. To insure that the battery pack will be able to be housed along the downtube, the battery thickness cannot exceed the diameter of the downtube. Based on "off-the-shelf" 8-series LiPo battery packs, the thickness of an 8-series battery is ~1.5 inches. This is the maximum thickness of the battery and, thus, the maximum number of cells in series. The branches in parallel was also determined by the geometric constraints of bicycle. The downtube length will be approximately 30 inches and each cell is about 2.5 inches long. This allows for a maximum of 12 branches to fit the battery pack on the downtube.

Constraint		Type
g_1	$x_1 \leq 8$	Discrete
g_2	$x_2 \leq 12$	Discrete

Table 2: Battery Subsystem Constraints

In addition to the discrete constraints, there is also a weight penalty embedded into the objective function code. As the battery is more powerful and has a higher energy output, the weight of the battery increases. Our research showed that the battery weight was correlated to energy output by 140 kg/W ▪ hr[1]. Therefore, the weight of the battery is calculated as follows:

$$Energy\ [Watt \cdot hr] = (Capacity)(N_{parallel})(V_{cell})$$

Where $V_{cell} = 3.2V$

$$Battery\ Weight = \frac{Energy}{140}$$

Parameters

The parameters for the battery subsystem were largely determined by the model used. The following section will go into greater detail about the model. The following table shows the parameters for the battery subsystem.

Parameter	Value
Rider Weight	70 kg
Frame Weight	10 kg
Wheel Inertia	0.02 kg·m^2
Tire Width	25 mm
Coulomb Friction Coefficient	0.013
Active Area Aerodynamic Drag[2]	0.452 m^2
Stiction Coefficient	1.2
Battery Cell Voltage	3.2 V
Maximum Temperature	1e30 ℃
Minimum Battery Temperature	-273.15 ℃

Table 3: Battery Subsystem Parameters

[1] "High energy, lightweight batteries", http://www.barnardmicrosystems.com/UAV/engines/batteries.html

[2] "Electric Bike Simulator", http://www.electricbikesimulator.com/index,en.html

Modeling

We chose to use Siemens LMS Imagine.Lab Amesim platform to model and simulate the battery subsystem. We started the model from an electric vehicle sample model provided by Amesim and changed the parameters of the vehicle to create a bicycle. By reducing the active area for drag, the friction coefficient, and tire size and removing the clutch, we were able to turn a generic 4-wheeled vehicle into a bicycle. The schematic for the entire system is in Figure 2 below.

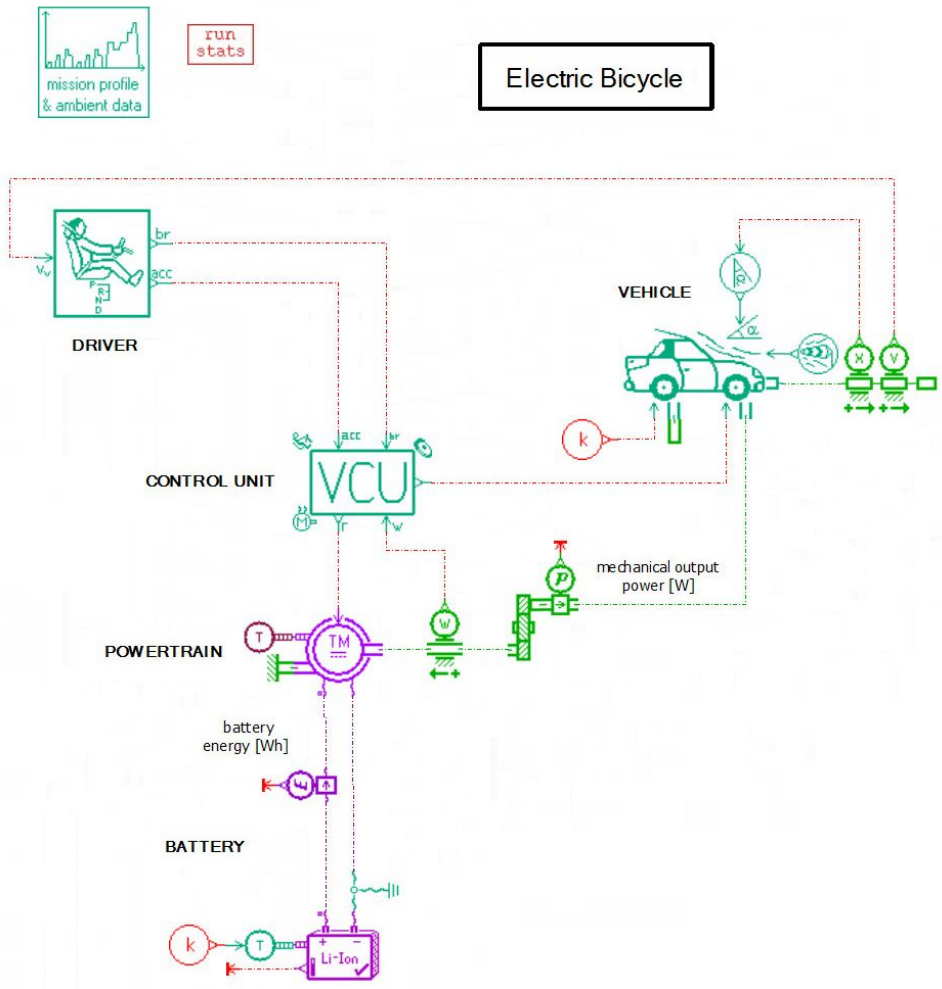

Figure 2: Amesim Model Schematic

For the battery subsystem, we chose to implement a Lithium-Ion polymer battery (Figure 3) that was available to us in the Amesim library. This component allows us to change the necessary variables of the battery to model the electric bicycle.

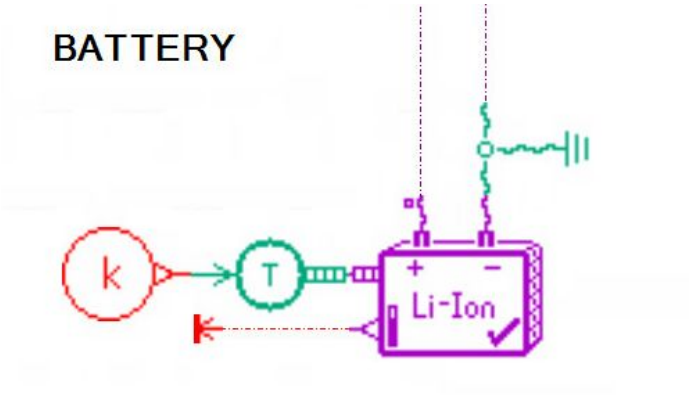

Figure 3: Amesim Battery Component

Another aspect of the Amesim model is the use of a mission profile. This is the profile that sets the vehicle velocity and the run-time of the simulation. For our purposes, it is only necessary to run the model for 5 minutes at various speeds between 0 and 20 miles per hour. This allows us to get a sample of the vehicle performance that can be optimized and extrapolated upon. Figure 4 shows the mission profile used for the simulation.

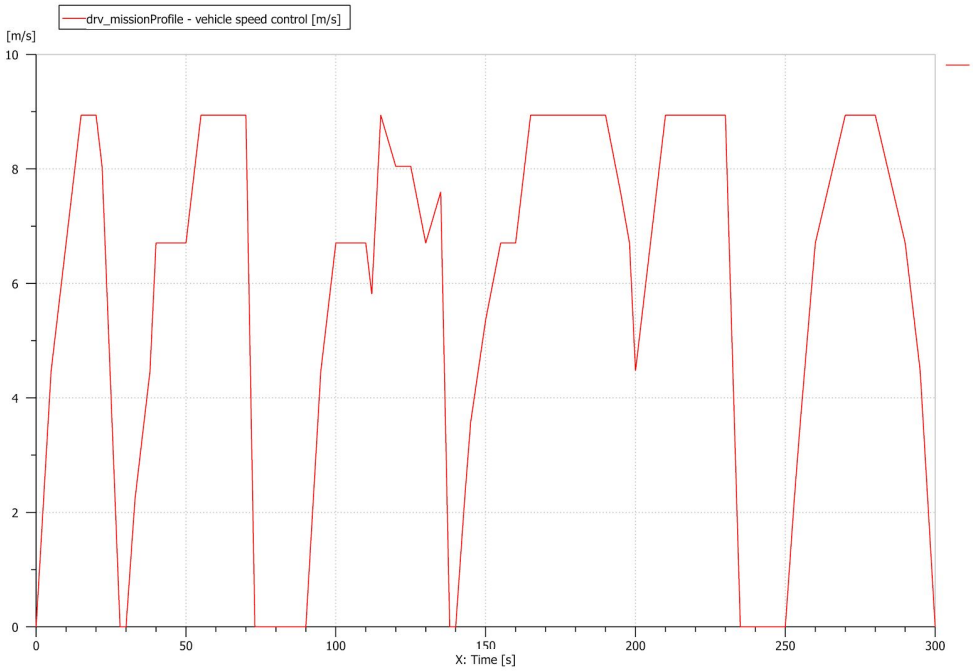

Figure 4: Amesim Mission Profile

Optimization

Method

Due to the discrete nature of the design variables and constraints for the battery subsystem, we used MATLAB Genetic Algorithm (GA) to optimize this subsystem. We used the following implementation of GA:

```
[x, fval, exitflag]=ga(@(x)SysObj(x),5,A,b,[],[],Lb,Ub,nlcon,intcon,options)
```

Where the inequality constraints were housed in the "$Ax \leq b$" matrices:

```
A = [1,0,; 0,1];
b = [8;12];
```

The lower bounds were:

```
Lb = [2; 5];
Ub = [8; 12];
```

The lower bounds were set to insure that Amesim model would have sufficient voltage to continue the simulation. There are no nonlinear constraints, so 'nlcon' was an empty matrix.The options were used to set a population size of 100 and for the GA to run 5 generations.

Results

The results of the battery subsystem optimization were as follows:

$$x_1 = 8$$
$$x_2 = 12$$
$$\Delta SoC = 1.503$$

This means that the optimizer selected the upper bound of the design variables to minimize the battery charge loss. This makes sense because a more powerful battery will reduce the loss in charge in a given ride time. We attempted to find an interior optima by including the weight penalty. However, due to the relative weight of the battery compared to the bicycle and rider, the effect of the increased weight on performance was negligible. A graph of the results from the GA optimizer are shown in Figure 5.

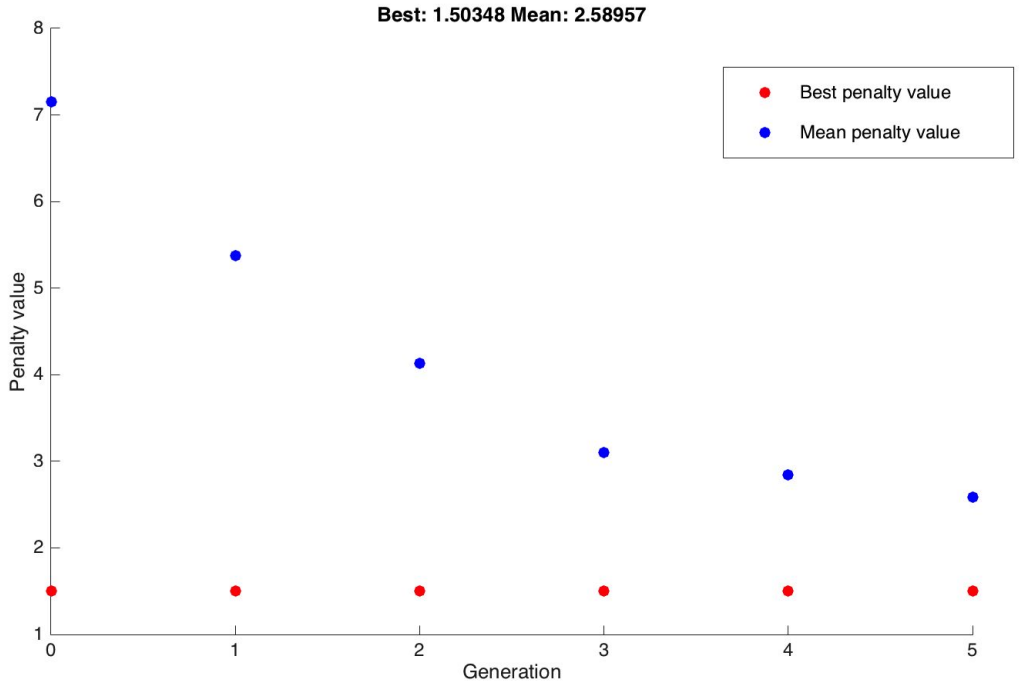

Figure 5: Battery Subsystem Optimization Results

In Amesim, the simulation at the optimized design variables produces the optimal change in state of charge, as depicted in Figure 6.

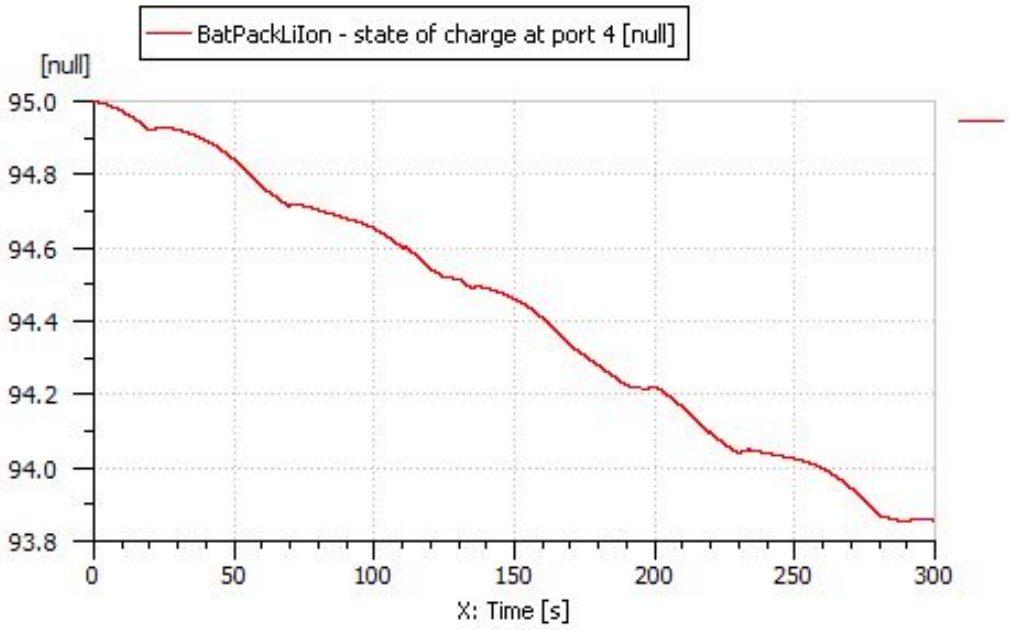

Figure 6: Battery Subsystem Optimization Amesim Results

Motor

Problem Formulation

The motor subsystem is the electromechanical component of the system with its main purpose to convert the electrical energy supplied by the battery into mechanical energy (kinetic) that will produce a displacement of the bicycle. There are several factors which oppose the motion of the vehicle: rolling resistance, aerodynamic drag, and internal motor resistance. The motor provides a torque to overcome these factors and draws charge from the battery in proportion to how much these factors are being overcome. The greater the torque produced and power output by the motor, the greater the amount of charge drained from the battery per unit time. Therefore, the goal for this subsystem minimize the amount of charge collected from the battery. The objective function is the difference in the state of charge of the battery before and after motion.

Objective Function

The objective function for the motor subsystem is to minimize the battery charge loss.

$$\min_{x_1, x_2, x_3} \text{battery charge loss} = \Delta SoC = SoC_{initial} - SoC_{final}$$

The objective function value also acts as linking variable between the subsystems. The battery subsystem is also optimizing to minimize battery charge loss. The system-level optimization, discussed later in the report, shows how the subsystems were coordinated with this linking variable.

Design Variables

Design Variable		Unit	Type
x_1	P_{max} (Maximum Power)	Watts	Continuous, local
x_2	T_{max} (Maximum Torque)	Nm	Continuous, local
x_3	Gear Ratio	--	Continuous, local

Table 4: Motor Subsystem Design Variables

Weight Constraint

The more powerful a motor system is the heavier it will be. The weight of the motor will add more weight to the entire vehicle which will would increase the subsystem objective function. In this case, the weight is used in a function which penalizes the use of more powerful motors. This is done by updating the weight of the car each time the value of the maximum power changes.

This ensures that our optimizer includes the weight in the optimization process and controls it by simply attempting to minimize the objective function.

Velocity Profile Error Constraint

The difference between the desired velocity profile and the actual velocity indicates how much slower the EB is moving than the velocity profile has prescribed. When this error is integrated over the length of the simulation runtime, it represents the remaining the distance that the bike should have traversed if it had followed the velocity profile without error. Physically, this can be seen as the distance an individual would have to walk or pedal without assistance from the bike due to its depleted energy source. In reality a shorter distance is better; however, the charge is preserved better when the bike moves slower. Meaning, the optimum leads to a greater error in distance. Therefore, a limit of 300 m is placed on this error. This means a customer would have to apply manual effort for no more than 300 m, should the battery be depleted for a given time frame.

$$g_1 : \int error \cdot dt \leq 300$$

Cost Constraint

A more powerful and efficient motor is expensive. In addition to physical constraints, the cost of the motor is constrained by a price that the customer is not willing to go above. This number will be assumed for the purposes of preliminary simulations and then a hard cost requirement will be established later based on additional market information. The current cost model for the motor was obtained from the research done by Kang. M, Fienberg. F, and Papalambros. P[3] as also cited below.

$$Motor\ Cost = 16 * P(kW) + 385 \qquad \{\text{where P = motor power}\}$$

$$g_2 : C_{motor} - C_{max} \leq 0 \qquad \{\text{where } C_{max} = 500\}$$

Design Parameters

Parameter	Value
Rider Weight	70 kg
Frame Weight	10 kg

[3] Kang. M, Fienberg. F, and Papalambros. P, "Integrated Decision Making in Electric Vehicle and Charging Station Location Network Design", June 2015, p. 4

Wheel Inertia	0.02 kg·m^2
Tire Width	25 mm
Coulomb Friction Coefficient	0.013
Active Area Aerodynamic Drag[4]	0.452 m^2
Stiction Coefficient	1.2
Torque Time Constant	0.01 s
Minimum Voltage	0.001 V

Table 5: Motor Subsystem Parameters

Modeling

Our model is represented in the LMS Amesim software. The model is given by an electric vehicle which we have altered to resemble a bicycle. These alterations were done by changing values such as frontal area, tire width and height, rolling resistance, vehicle weight, etc., to liken the electric vehicle to an electric bicycle. More alterations will be made to make the model even more representative of an electric bicycle. The equations that describe the physics of the system are embedded in the software.

[4] "Electric Bike Simulator", http://www.electricbikesimulator.com/index,en.html

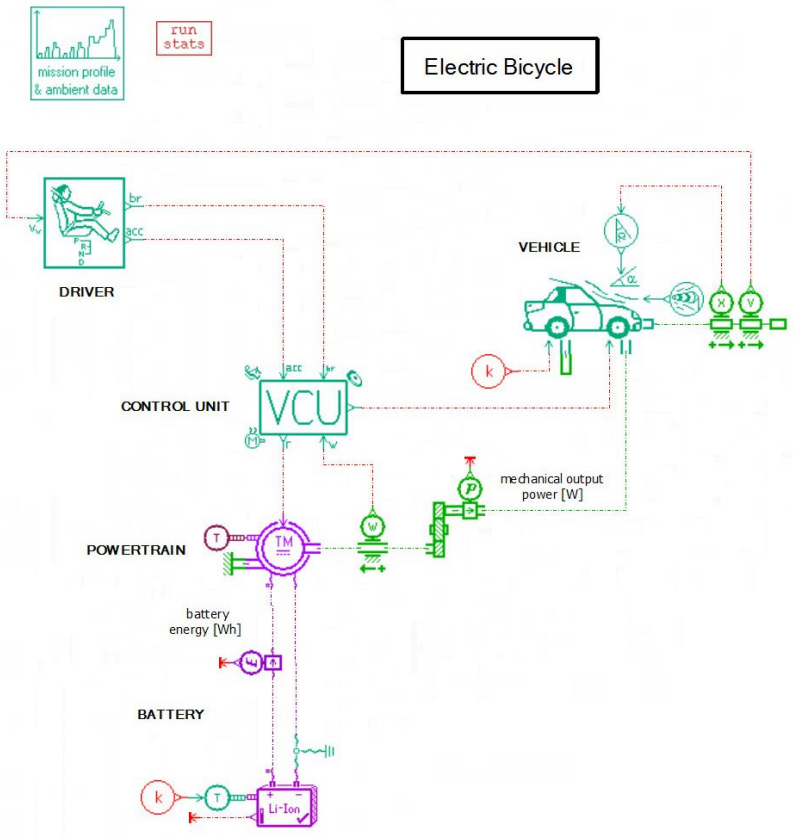

Figure 7: Amesim Model Sketch

Optimization

Method

Optimization is done using the matlab 'fmincon' function. Files from amesim are fed directly into matlab to make this possible. Now the fmincon function is able to change the variables and parameters as specified within the amesim model in order to obtain output values. The output values are the optimal value of 'dSoC' as well as the optimal design variables, maximum power and maximum torque. Plots are made in amesim to illustrate the results of the optimization, in order to ensure that the optima are truly reasonable.

Results

Running 'fmincon' with the error integral constraint set to 300 and the cost constraint set to 500, results were obtained using different initial points as shown in Table 6 below.

Run A	Run B

	Initial	$x_{optimal}$	Initial	$x_{optimal}$
Max Power	402	710.26	1000	580
Max Torque	5	10.447	5	5
Gear Ratio	2	1.748	1	2.4
ΔSoC	**1.271**		**.9419**	

Table 6: Motor Subsystem Optimization Results

These results show different minima obtained based on different initial values. This is due to the nature of the design space with multiple configurations. For instance, two configurations that have the same maximum power value, with one have a torque of 10 Nm and a gear ratio of 1, and the other having a torque of 5 Nm and a gear ration of 2, will essentially be the same. Moreover, the objective function is not very sensitive to changes in design variables once they fall within a certain threshold. This makes the design variable optima dependent on initial values. Within the design space, it is possible to have design variables that look very different but have similar objective function optima. As shown in the results in Table 6, the difference between the objective function optima is 0.33 which is small compared to the significant differences between the design variables for Run A and Run B.

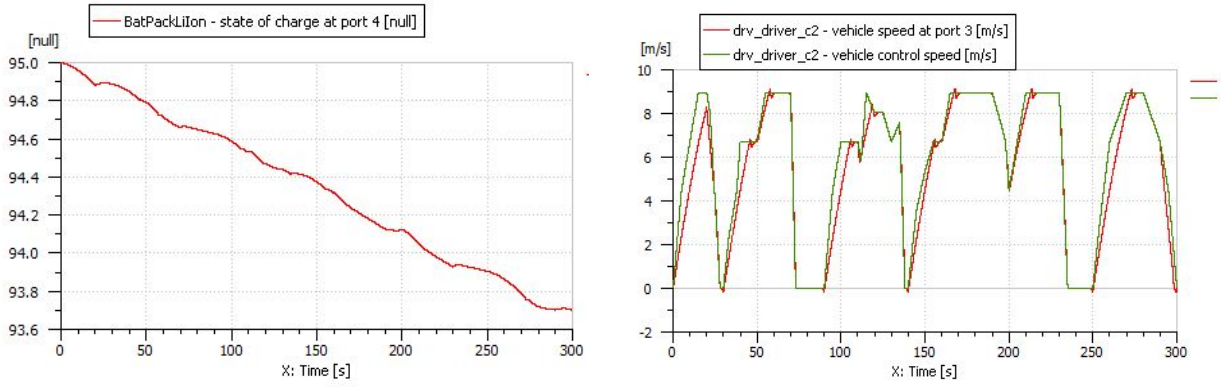

Figure 8: Fmincon Run A

Figure 8 (above) and Figure 9 (below) show the relevant plots for Run A and Run B, respectively. The plot on the left shows the state of charge of the battery and the plot on the right shows the plot of the ideal velocity profile (green) and the actual velocity of the bike (red). For Run A, the velocity of the bike closely follows the velocity profile and hence draws more charge from the battery than in Run B where the bike velocity does not follow the ideal velocity profile as closely, hence preserving more battery charge. The error integral constraint prevents this difference between ideal and actual velocities from becoming too large.

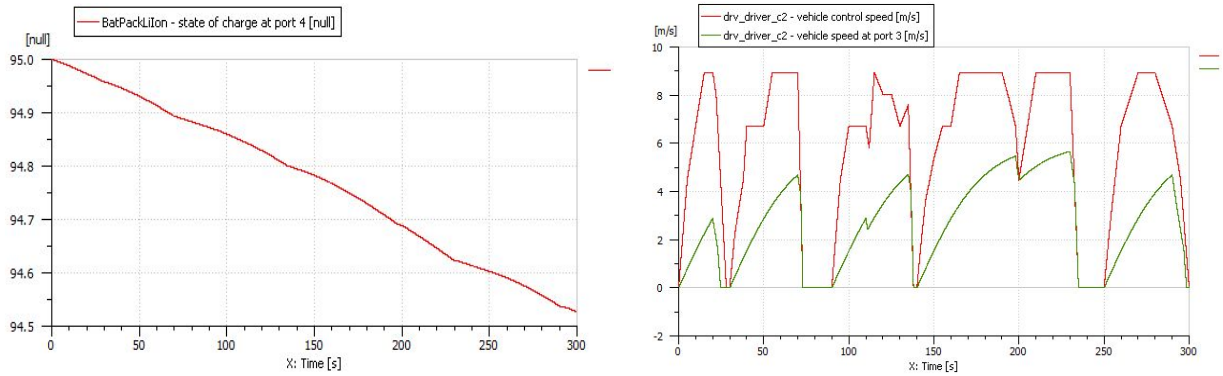

Figure 9: Fmincon Run B

SYSTEM OPTIMIZATION

Problem Formulation

As shown in the system layout, we chose to use multidisciplinary feasible coordination to combine our subsystems for the system-level optimization and analysis. Figure 8 below shows the MDF coordination representation.

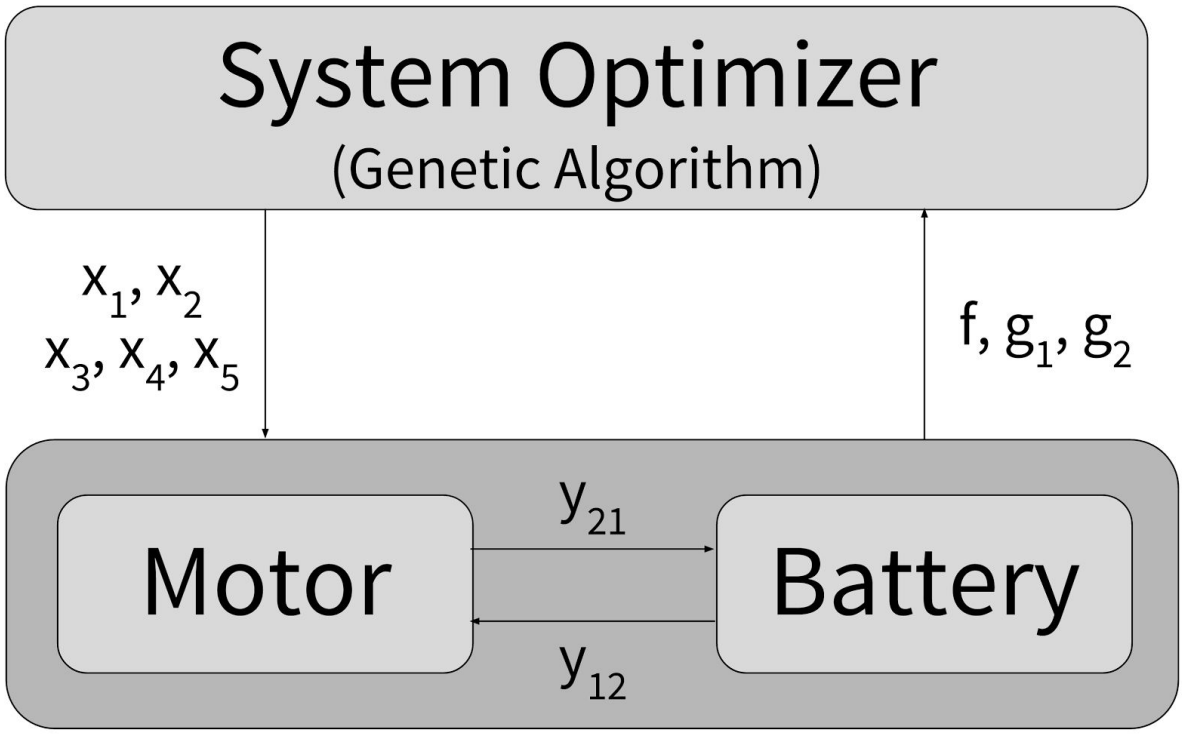

Figure 10: MDF System Coordination

Objective Function

The objective function for the EB system is to maximize the range on a single battery charge, or minimize negative range. Through extrapolating the range traveled in the 5-minute sample simulation using battery charge data, we can calculate the total range traveled during a single charge. The extrapolation method is as follows:

Given ΔSoC during the 5-minute sample and given that the decrease in charge is linear, we can calculate the slope of the this change in state of charge:

$$Slope_{charge} = \frac{\Delta SoC}{300\ s}$$

Using the slope and the initial state of charge, we can calculate the total time to deplete the battery:

$$Total\ Time = \frac{Charge_{intial}}{Slope_{charge}}$$

Given the range traveled during the 5-min period, we can calculate the slope of the range:

$$Slope_{range} = \frac{Range}{300\ s}$$

With the total time and range slope, we can calculate the total range traveled given a single charge:

$$Total\ Range = Slope_{range} \cdot Total\ Time$$

For the system-level optimization, the objective function is:

$$\min_{x_1, x_2, x_3, x_4, x_5} Range = -(Slope_{range} \cdot Total\ Time)$$

Design Variables

With MDF coordination, the design variables are a combination of the design variables used at the subsystem level.

Design Variables		Units	Type
x_1	N_{series}, Number of cells in series	--	Discrete
x_2	$N_{parallel}$, Number of branches in parallel	--	Discrete
x_3	P_{max} (Maximum Power)	Watts	Continuous
x_4	T_{max} (Maximum Torque)	Nm	Continuous
x_5	Gear Ratio	--	Continuous

Table 7: System Design Variables

Constraints

The constraints were also a combination of the subsystem-level constraints.

Constraint		Type
g_1	$x_1 \leq 8$	Discrete
g_2	$x_2 \leq 12$	Discrete
g_3	$\int error \cdot dt \leq 300$	Continuous
g_4	$C_{motor} \leq 500$	Continuous

Table 8: System Constraints

Parameters

The parameters were also a combination of the subsystem-level parameters.

Parameter	Value
Rider Weight	70 kg
Frame Weight	10 kg
Wheel Inertia	0.02 $kg \cdot m^2$
Tire Width	25 mm
Coulomb Friction Coefficient	0.013
Active Area Aerodynamic Drag[5]	0.452 m^2
Stiction Coefficient	1.2
Battery Cell Voltage	3.2 V
Torque Time Constant	0.01 s
Minimum Voltage	0.001 V

Table 9: System Parameters

Modeling

To model the system, we used the same Amesim model used during the subsystem level optimization. By accessing variables for both the battery and the motor, we were able to optimize all parts of the system. Figure 11 shows the schematic for the entire Amesim model.

[5] "Electric Bike Simulator", http://www.electricbikesimulator.com/index.en.html

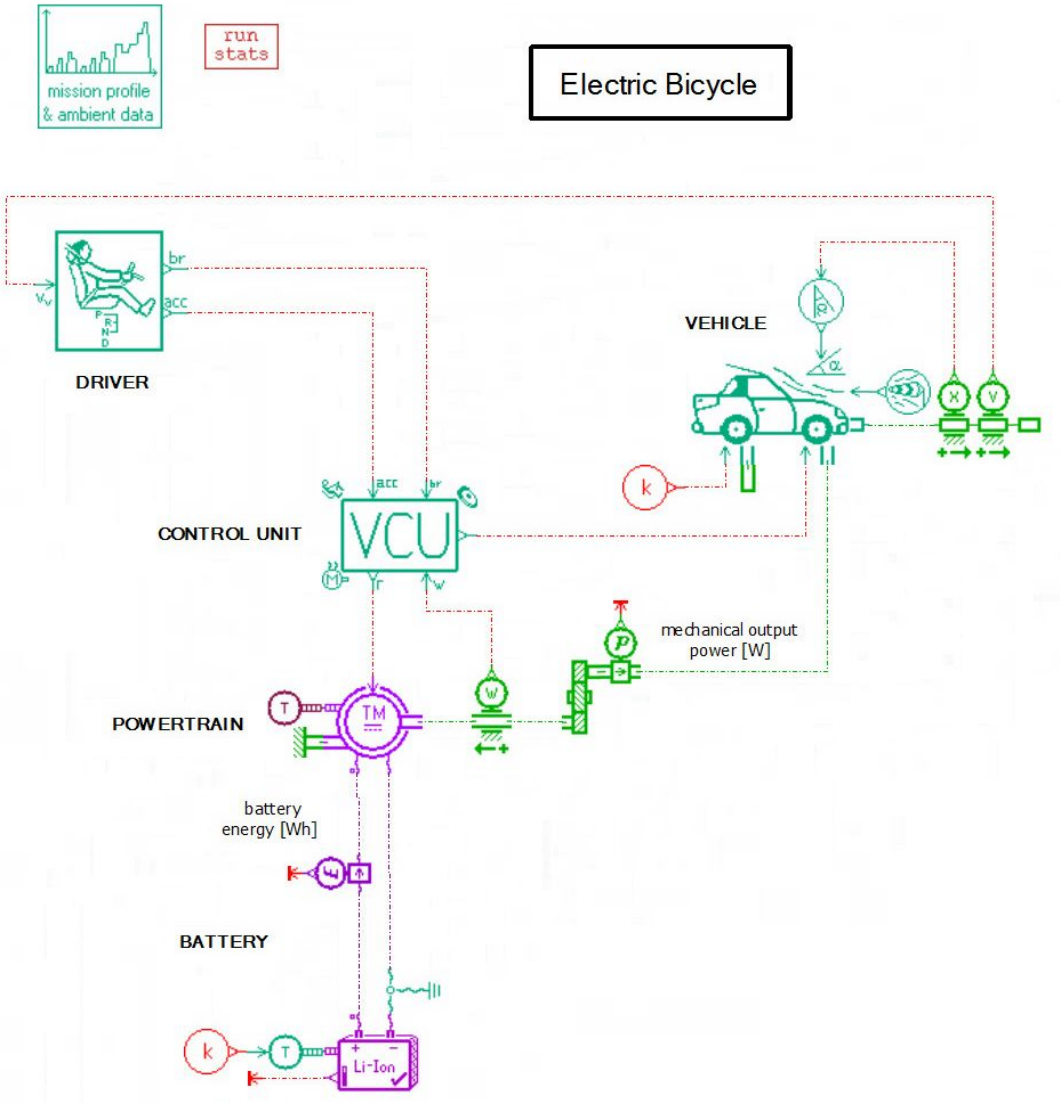

Figure 11: Amesim Model Schematic

Optimization

Method

To optimize at the system level, we chose to use MATLAB Genetic Algorithm because we were constrained by the discrete variables and constraints of the battery subsystem. We used the following implementation of GA:

```
[x, fval, exitflag]=ga(@(x)SysObj(x),5,A,b,[],[],Lb,Ub,nlcon,intcon,options)
```

Where the inequality constraints were housed in the "A*x ≤ b" matrices:

```
A = [1,0,0,0; 0,1,0,0];
b = [8;12];
```

The lower bounds were:

```
Lb = [2; 5; 400; 5; 1];
Ub = [8; 12; 1500; 30; 3];
```

The lower bounds were set to insure that Amesim model would have sufficient voltage, motor power, and torque to continue the simulation. Although we have nonlinear constraints from the motor subsystem, we did not house these constraints in 'nlcon'. Because we are calling Amesim in MATLAB to set the nonlinear constraint, the GA was not able to call Amesim as expected. To get around this obstacle, we represented the motor nonlinear constraints as a penalty on the overall system objective. The implementation is as follows:

```
TotalRange =
-(RangeSlope*TotalTime)+1e6*max(error_integral-300,0)+1e6*max(Cost-700,0);
```

This means that if either the velocity error integral constraint or the cost constraint is violated, a large number is added to the objective value. This prevents GA from selecting this objective value as the minima.

Results

The Genetic Algorithm produced the following optimized results:

$$x_1 = 8$$
$$x_2 = 12$$
$$x_3 = 1101.7\ W$$
$$x_4 = 9.969\ Nm$$
$$x_5 = 3$$

$$Total\ Range = 136686\ m$$

Figure 12 shows the GA optimizer output for an initial population of 100 and 5 generations.

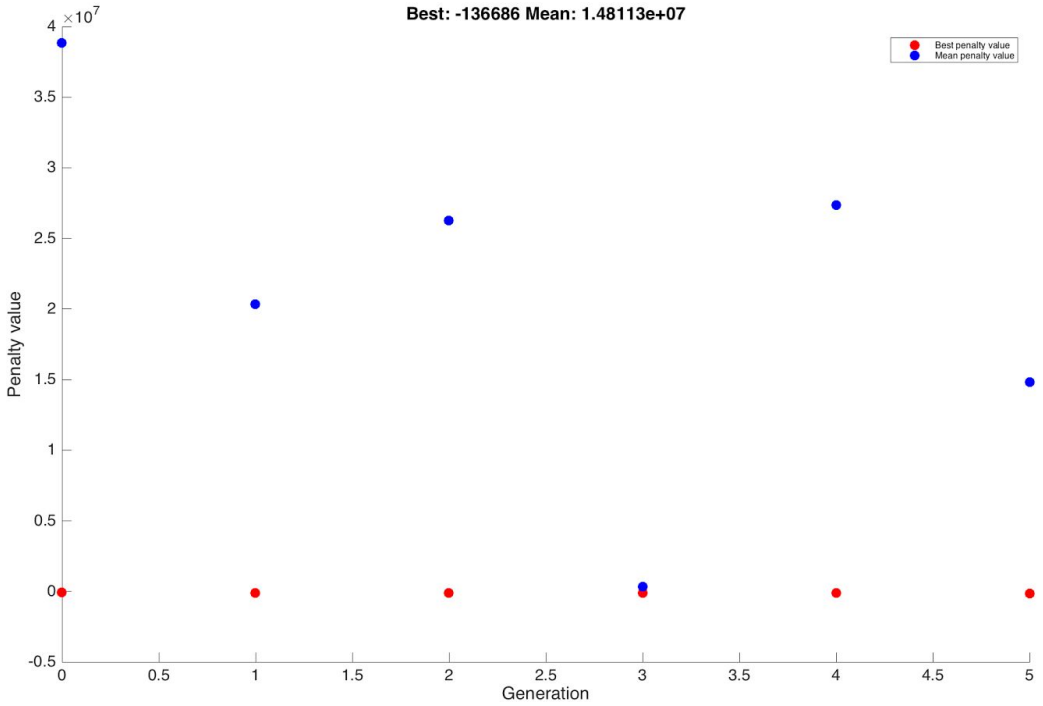

Figure 12: System Optimization Results

The difference between the system and subsystem optima can be explained in a similar way as has been done for the differing values within the motor subsystem optimization. The design space accommodates several different configurations that produce different design variable optima but similar objective function optima. Hence, the optimal values given in the system optimization are different from the ones given in the subsystem, although they produce similar objective results. We imagine the design space to be similar to that of an egg holder function given the presence of several local optima that can be encountered. Additionally, the genetic algorithm has more of an ability to search globally than does 'fmincon' and so vastly different design variable optima may be obtained by using the genetic algorithm for system level optimization.

The battery design variables were optimized to the upper bound of the battery architecture constraints. These results are the same as was seen in the subsystem optimization. This is to be expected because the EB will travel the further with the largest battery possible and the weight penalty attached to this has negligible effect on the system.

POST-ANALYSIS

Parametric Study

To perform a parametric study, we chose to alter the vehicle weight and velocity parameters. We ran the system-level optimization with the vehicle weight at 100 kg and 60 kg, as opposed to the original 80-kg value. This heavier weight accounts for a heavier frame and/or rider. The same is said for the lighter weight. We also replaced the mission profile in Amesim with velocity profiles going 10% faster and 10% slower. The results are in Table 10 below.

Parameter	Range	x(1), x(2), x(3), x(4), x(5)
100 kg weight	123834 m	8, 12, 1370.7, 9.832, 3
60 kg weight	156718 m	8, 12, 926.12, 7.004, 3
10% faster	153728 m	8, 12, 929.167, 8.376, 3
10% slower	161943 m	8, 12, 506.87, 5.226, 2

Table 10: Parametric Study Results

As expected, at a 100-kg vehicle weight, the range is less than the 80-kg range. Similarly, the 60-kg vehicle weight range is greater than the 80-kg weight range. This is explained simply; the heavier the vehicle the more power required to move it the same distance and vice versa. By lessening the load, the vehicle can go further with less strain and by increasing the load the vehicle can not go as far.

Also, as expected, the velocity profile with 10% faster speeds also goes farther than the original velocity profile. However, for the profile with 10% slower speeds, the vehicle is reaching an even further range than the original profile and the 10% faster profile. While this may be counter-intuitive results, we believe that the slower speed of travel is allowing for the battery to discharge rate and thus allowing the EB to travel longer. This happens to result in a greater distance in a single charge. However, we do not consider this to be a better result because we require the EB to travel at speeds that are appealing to the customer. If the customer could travel 160,000m+ but at a slower speed than they could pedal, we doubt that they would purchase this EB. By constraining the EB to the velocity profile we prescribed, we are optimizing the range for an appropriate traveling speed.

Sensitivity Analysis

We performed a sensitivity analysis on our system by relaxing the nonlinear constraints and running the GA optimizer. The results of these runs are in Table 11 below.

Constraints		Range	x(1), x(2), x(3), x(4), x(5)
Chosen Constraint	$g_3 \leq 300$ $g_4 \leq 500$	136686 m	8,12, 1101.7, 9.9691, 3
Relax g_3	$g_3 \leq 500$ $g_4 \leq 500$	137751 m	8, 12, 476.219, 10.844, 3
Relax g_4	$g_3 \leq 300$ $g_4 \leq 700$	145825 m	8, 12, 480.107, 6.627, 3

Table 11: Sensitivity Analysis Results

These results show that by individually relaxing the cost constraint and the velocity error integral constraint, the EB is able to travel further than it was with the chosen constraint values. This indicates that both the velocity error integral constraint and the cost constraint are active. However, due to the nature of the Genetic Algorithm we are not able to make any conclusions on the relative magnitude of each constraint. The algorithm we used does not have an Augmented Lagrangian nonlinear constraint solver, hence we do not have access to true or approximate Lagrange Multipliers. Thus,we cannot determine which constraint has a greater effect on the objective value.

APPENDIX A

Battery Subsystem Codes

Objective Function: BatObj.m

```
function f = BatObj(x)
%Battery subsystem objective function
%INPUT: x(1): Nseries (# cells in series)
%       x(2): Nparallel (# cells in parallel)
%       x(3): Cap (capacitance)
%OUTPUT: dSoC

Nseries = x(1);
Nparallel = x(2);

Cap = Nparallel*.5;

Weight = 97.2/1000; %Vehicle weight (tonne), no battery weight

Wh = (Cap*x(1)*3.2);

BatWeight = Wh/140/1000; %Battery weight (tonne) assuming 140 Wh/kg

TotalWeight = Weight+BatWeight;

addpath(fullfile(getenv('AME'),'scripting','matlab','amesim'));

[sts, res]=system('AMELoad
H:\ME_555_Project\Electric_Vehicle_Amesim_041616\ElectricVehicleAmesim\Electric
Vehicle5');
if sts~=0
    disp('Loading error!')
    disp(res)
end

ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 number of elements in
series in one branch', Nseries);
ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 number of branches in
parallel', Nparallel);
ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 element nominal
capacity [Ah] ', Cap);
ameputp('ElectricVehicle5', 'DRVVEH4A instance 1 total vehicle mass [tonne]',
TotalWeight);

[Results, VarNames]= amerun('ElectricVehicle5', 0, 300);

timeval = amegetvar(Results,VarNames,'time [s]');
```

```
SoC = amegetvar(Results,VarNames,'LIION_ESSBATPDY01_1 state of charge at port 4
[null]');

dSoC = SoC(1)-SoC(end);

f = dSoC


[sts, res]=system('AMESave
H:\ME_555_Project\Electric_Vehicle_Amesim_041616\ElectricVehicleAmesim\Electric
Vehicle5');


end
```

Genetic Algorithm Script: BatObj.m

```
%Calls GA for BatObj
tic;
A = [1,0; 0,1];
b = [8;12];

Lb = [2,5];
Ub = [8; 12];
intcon = [1,2];
options = gaoptimset('PopulationSize', 100, 'Generations', 5,
'PlotFcns',@gaplotbestf);

[x, fval, exitflag] = ga(@(x)BatObj(x),2,A,b,[],[],Lb,Ub,nlcon1,intcon,
options)
toc
```

Motor Subsystem codes

Objective Function: MotObj.m

```
function[dSoS]= MotObj(x)
addpath(fullfile(getenv('AME'),'scripting','matlab','amesim'));

[sts, res]=system('AMELoad H:\ElectricVehicleAmesim\ElectricVehicle5');
if sts~=0
    disp('Loading error!')
    disp(res)
end

Pmax_mot= x(1);
Tmax_mot = x(2);
Gear_ratio=x(3);
```

```
weight = 0.001*(80+2+0.01*Pmax_mot);
%ameputgpar('ElectricVehicle4', 'Pmax_mot', x1);
ameputp('ElectricVehicle5', 'EMDTMF03 instance 1 rescaled machine maximum power
[W]', Pmax_mot);
ameputp('ElectricVehicle5','DRVVEH4A instance 1 total vehicle mass [tonne]',
weight);
%ameputgpar('ElectricVehicle4', 'Tmax_mot', x2);
ameputp('ElectricVehicle5', 'EMDTMF03 instance 1 rescaled machine maximum
torque [Nm]', Tmax_mot);

ameputp('ElectricVehicle5','RN001 instance 1 gear ratio [null]',Gear_ratio)

[Results, VarNames]= amerun('ElectricVehicle5', 0, 300);

timeval = amegetvar(Results,VarNames,'time [s]');

SoS = amegetvar(Results,VarNames,'LIION_ESSBATPDY01_1 state of charge at port 4
[null]');
numel(SoS);
dSoS = SoS(1)-SoS(end)

% timeval = amegetvar(Results,VarNames,'time [s]');
error = amegetvar(Results,VarNames,'DRVDRVA00A_1 error on speed [m/s]');

dt = timeval(2:end)-timeval(1:end-1); % Calculate dt = tnext-tcurrent
error = error(1:end-1); % remove last element in the error vector

error_integral = sum(dt.*error);  % integration of error over time.

[sts, res]=system('AMESave H:\ElectricVehicleAmesim\ElectricVehicle5');
end

addpath(fullfile(getenv('AME'),'scripting','matlab','amesim'));
[sts, res]=system('AMELoad H:\ElectricVehicleAmesim\ElectricVehicle5');
if sts~=0
    disp('Loading error!')
    disp(res)
end
```

Fmincon Script: FminconMotor.m

```
x0= [745 7];
% p=[3.3 p2 p3]; % These p values are assumed for now and will be specified
                 % later on
A=[];
b=[];
Aeq=[];
beq=[];
lb=[400;5];
```

```
ub=[1500; 30];
opts= optimoptions('fmincon','Algorithm','sqp');
[xopt, fopt, exitflag]= fmincon(@(x)sampleCode(x), x0, A, b, Aeq, beq, lb,ub,
@(x)nlcon(x), opts );
```

EB System Codes

Objective Function: SysObj.m

```
function [TotalRange] = SysObj(x)
%System objective function
%INPUT: x(1): Nseries (# cells in series)
%       x(2): Nparallel (# branches in parallel)
%       x(3): Pmax_mot (Maximum motor power W)
%       x(4): Tmax_mot (Maximum motor torque Nm)
%       x(5): Gear_ratio (Gear ratio for motor)
%OUTPUT: Range

%Design Variables
Nseries     = x(1);
Nparallel   = x(2);
Pmax_mot    = x(3);
Tmax_mot    = x(4);
Gear_ratio  = x(5);

%Battery capacity, dependent on Nparallel
Cap = Nparallel*.5;

%SYSTEM WEIGHT CALCULATIONS
Weight = 100/1000; %Vehicle weight (tonne), no battery weight

%Battery watt.hr
Wh = (Cap*Nparallel*3.2);

%Component Weights
BatWeight = Wh/140/1000; %Battery weight (tonne) assuming 140 Wh/kg
MotWeight = (0.01*Pmax_mot)/1000; %Motor weight


%Sum of frame+battery+motor weight
TotalWeight = Weight+BatWeight+MotWeight;

%Amesim called in GA script
% addpath(fullfile(getenv('AME'),'scripting','matlab','amesim'));
% [sts, res]=system('AMELoad
H:\ME_555_Project\Electric_Vehicle_Amesim_041616\ElectricVehicleAmesim\Electric
Vehicle5');
% if sts~=0
```

```
%     disp('Loading error!')
%     disp(res)
% end

%Weight
ameputp('ElectricVehicle5','DRVVEH4A instance 1 total vehicle mass [tonne]',
TotalWeight);

%Motor Parameters
ameputp('ElectricVehicle5', 'EMDTMF03 instance 1 rescaled machine maximum power
[W]', Pmax_mot);
ameputp('ElectricVehicle5', 'EMDTMF03 instance 1 rescaled machine maximum
torque [Nm]', Tmax_mot);
ameputp('ElectricVehicle5','RN001 instance 1 gear ratio [null]',Gear_ratio)

%Battery Parameters
ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 number of elements in
series in one branch', Nseries);
ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 number of branches in
parallel', Nparallel);
ameputp('ElectricVehicle5', 'LIION_ESSBATPDY01 instance 1 element nominal
capacity [Ah] ', Cap);

%Run Simulation
[Results, VarNames]= amerun('ElectricVehicle5', 0, 300);

%Error constraint calcluation for motor
error = amegetvar(Results,VarNames,'DRVDRVA00A_1 error on speed [m/s]');
timeval = amegetvar(Results,VarNames,'time [s]');
dt = timeval(2:end)-timeval(1:end-1); % Calculate dt = tnext-tcurrent
error = error(1:end-1); % remove last element in the error vector
error_integral = sum(dt.*error);  % integration of error over time.

%Cost constraint calculation
Cost= 16*Pmax_mot*0.001 + 385;

%State of Charge
SoC = amegetvar(Results,VarNames,'LIION_ESSBATPDY01_1 state of charge at port 4
[null]');
dSoC = SoC(1)-SoC(end);


%Calculate time to 0 charge
Time = timeval(end)-timeval(1);
ChargeSlope = dSoC/Time;
TotalTime = SoC(1)/ChargeSlope;

%Range
```

```
range = amegetvar(Results,VarNames,'DRVVEH4A_1 vehicle linear displacement at
port 5 [m]');
drange = range(end);
RangeSlope = drange/Time;

TotalRange =
-(RangeSlope*TotalTime)+1e6*max(error_integral-300,0)+1e6*max(Cost-700,0);

setappdata(0,'ei',error_integral);
setappdata(0,'c',Cost);
end
```

Genetic Algorithm Script: GAforBatObj.m

```
%Calls GA for BatObj
tic;
A = [1,0; 0,1];
b = [8;12];

Lb = [2,5];
Ub = [8; 12];
intcon = [1,2];
options = gaoptimset('PopulationSize', 100, 'Generations', 5,
'PlotFcns',@gaplotbestf);

[x, fval, exitflag] = ga(@(x)BatObj(x),2,A,b,[],[],Lb,Ub,nlcon,intcon, options)
toc
```

APPENDIX B

Motor Subsystem Error

Integral Constraint Activity

	Run A		Run B	
	Initial	$x_{optimal}$	**Initial**	$x_{optimal}$
Max Power	402	710.26	402	402
Max Torque	5	10.447	5	5
Gear Ratio	2	1.748	2	2
ΔSoC	**1.271**		**0.773**	

Table 12 : Motor Subsystem Optimization Results

Run A was performed with an error integral constraint of 300, while for Run B the error integral constraint was relaxed to 20000. It can be observed that when the constraint was relaxed, a better objective function optima was achieved. This indicates that the constraint is active.