Appendix A *current and prototype electric scooters*

Country	ROC	ROC	ROC	Japan	Japan	Japan	USA
maker	ITRI	Shangwei	ENTER	TOKYO R&D	HONDA	YAMAHA	DORAN [a]
model	ZES 2000	SWAP	City Bike II	ES-600	CUV-ES	MEST	ECO-SCOOT
maximum speed (km/h)	50	80	55	55	60	50	33
climb capability (tan θ)	16 km/h on 18° slope	30 km/h on 30° slope	15° [b]	18° [b]	12° [b]	10° [b]	12° [b]
range (km) at 30 km/h	60	80	75	60	60 (35 km in "urban mode")	30	40
dry weight, kg	105	130	95	117	130	80	89
motor	DC brushless, 48 V	DC brushless, 48V	DC brushless	DC brushless	DC brushless (3.25 kW), 48 V	DC brushless	DC brush
battery	four sealed lead-acid	sealed lead-acid	lead-acid	maintenance-free lead-acid	Ni-Cd	maintenance-free lead-acid	lead-acid
battery capacity (Ah/V)	28/12	40/12	52/24	30/48	20/86.4	17/48	46/24
charging time (hr)	no data	6~8	5~8	8	8	8	~10
transmission	CVT (?)	CVT	single-stage reduction	CVT	CVT	no data	single-stage reduction
acceleration	0-30 m, 4.5 s	no data	no data	no data	0-200 m, 17.3 s	no data	no data
price (US$)	no data	2000	1460	4750	8150	no data	1900
notes	demo only	750 W rated power	in development	limited fleet sales	limited fleet sales	introduction	on market

[a] The Doran vehicle was bought out by the Sun Cat Motor Company in 1995 and renamed the Sol Gato
[b] Speeds on these "climb capability" slopes were not given

The data were based on tables from the following sources:

P. H. Jet Shu, Wei-Li Chiang, Bing-Ming Lin, Ming-Chou Cheng. "The Development of the Electric Propulsion System for the Zero Emission Scooter in Taiwan" Japan Society of Automotive Engineers. 1997. JSAE 9734403, SAE 92107

Shang Wei web site. "Shang Wei SWAP vs. Other Electric Scooters". http://www.shangwei.com/compar-e.htm. Accessed August 30, 1999

Appendix B *detailed stack cost/size analysis*

The DTI model outlined in *Detailed Manufacturing Cost Estimates for Polymer Electrolyte Membrane (PEM) Fuel Cells for Light Duty Vehicles* (August 1998) was used to calculate size, weight, and cost of a scooter fuel cell stack. The ultimate purpose was to use the methodology described in that report to produce reasonable estimates of size, weight, and cost that were more accurate than simply linearly scaling down automotive fuel cells.

In their report, DTI studied several different sizes of fuel cells and calculated manufacturing costs for the various components of the cell stack (auxiliaries like compressors and cooling pumps were not examined). Two sets of analyses were done, and in each case the area of the membranes in each cell was varied between six different sizes ranging from 116 cm^2 to 697 cm^2. In the first set, "equal voltage," the number of cells was held constant at 420 cells and thus power varied with membrane area. In the second set, "equal power," the number of cells was varied to keep total power at 70 kW_{elec}. Options for the cell unit design include "unitized" bipolar metallic separator plates stamped with flow fields on both sides, three-piece unipolar metallic plates, carbon-polymer composite plates, and amorphous carbon plates. The more conservative three-piece metallic separator plates were chosen here (see section B.2 for details)

Note: A May 1999 DTI study has examined a range of small fuel cells (among them, 3-5 kW stacks) for electricity and heat for individual residences. For a production quantity of 10,000 units, this study found the installed cost per kW to be about $4,500.[1] These figures are based on top-down cost analyses, and other companies who are designing home fuel cell systems have quoted figures on the order of $3,500 to $5,000 for a residential fuel cell system of a few kilowatts using batteries for peak power.[2]

These high prices for this size of fuel cell would seem prohibitive for the scooter, except for the fact that stationary power fuel cells must be designed very differently from automotive fuel cells; they must operate 24 hours, unlike vehicle engines which are only run a few times daily. Also, vehicle engines rarely reach top output. All in all, stationary fuel cell lifetimes must be longer and they must be designed more robustly. Another factor cited in the DTI report was that the larger production quantities for vehicles would help to drive down costs. So despite these recent high projections of cost for stationary fuel cell systems of similar size as the scooter studied here, predictions of fuel cells for vehicles are still on target.

B.1 "Blind" curve-fitting

Curves were fit to the original DTI results for automotive fuel cell, based on an equal-voltage study with 420 cells and stack power varying with different membrane area. Each of the six different cell

---

1 C. E. (Sandy) Thomas, Jason P. Barbour, Brian D. James and Franklin D. Lomax, Jr. Directed Technologies, Inc. "Analysis of Utility Hydrogen Systems & Hydrogen Airport Ground Support Equipment". Prepared for the Proceedings of the U.S. DOE Annual Hydrogen Program Review. May 1999.

2 Plug Power and American Power Corporation. Quote is from Ronald J. Wolk. "Fuel Cells for Homes and Hospitals" *IEEE Spectrum* May 1999 p. 45

membrane areas is listed in Table B.1 below.

Table B.1 DTI automotive stack parameters

Membrane area (cm^2)	Stack power (kW)	Cost ($/gross kW_e)	Weight (kg)	Volume (L)
116	31.5	$36	22.0	23.7
181	49.0	$29	29.4	32.5
258	70.0	$26	38.5	42.7
348	94.5	$23	48.1	54.2
452	122.5	$22	59.5	67.1
568	154.0	$21	72.4	81.3
697	188.9	$20	86.4	96.9

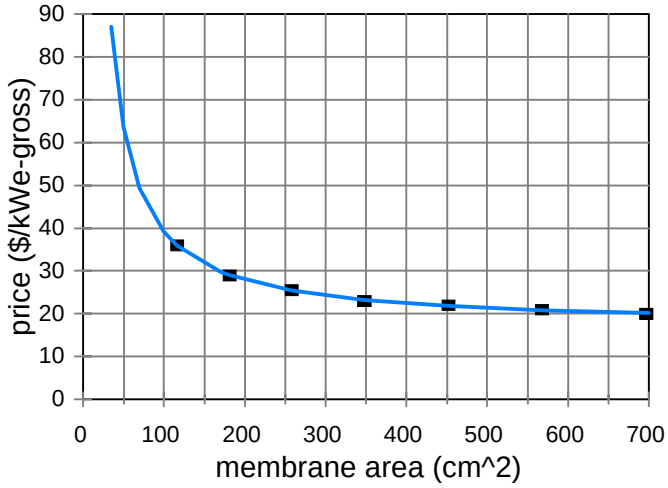
Note: the total weight was not given in the study, and was summed in the same way the stack weights were calculated later in section B.3. Volumes were given in terms of stack dimensions and converted here to liters.

The DTI automotive stack costs per kilowatt were plotted against cell active membrane area in Figure B.1, and fit with a hyperbola of the form

$$y = A / (x-B) + C$$

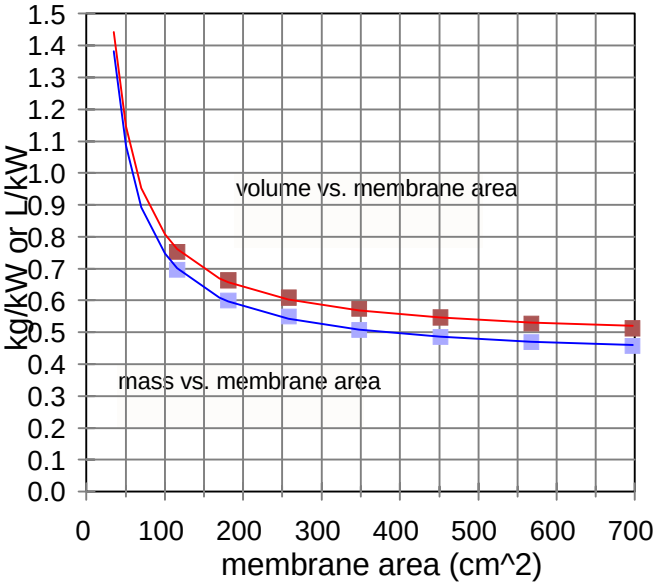
for cost per kilowatt “y” and active membrane area “x”, and fitting parameters A, B, and C.

Figure B.1 Cost as a function of cell membrane area



Similarly, per-kilowatt data for weight and volume were plotted and regressed against curves of the same form as for the cost in Figure B.2:

Figure B.2 Weight and volume as a function of cell membrane area



These results were extrapolated to the membrane areas required for the three fuel cell sizes studied here: 1.1 kW, 3.2 kW, and 5.9 kW (corresponding to stack membrane areas of 35 cm^2, 100 cm^2, and 170 cm^2 as discussed in Chapter 4). This produced the following results:

Table B.2 Curve-fitting versus bottoms-up model

	Stack power	Curve-fitting result
Stack cost	1.1 kW	$96
	3.2 kW	$125
	5.9 kW	$176
Stack weight	1.1 kW	1.5 kg
	3.2 kW	2.4 kg
	5.9 kW	3.6 kg
Stack volume	1.1 kW	1.6 L
	3.2 kW	2.6 L
	5.9 kW	4.0 L

B.2 Size and volume

To improve upon this estimate, bottom-up calculations of size, weight, and cost were made by following the DTI procedure. First, the total size of the three fuel cells were calculated by first calculating the sizes of three-piece stainless steel cooler cells and active cells, and assuming a 2:1 active-to-cooler cell ratio:

The active cell requires one metal separator plate and two separate, unipolar plates etched with flow fields and gaskets separating the flow fields from the MEA, for a total thickness of 2.27 mm per active cell:

51 µm separator plate

76 µm anode flow field

1000 µm anode gasket

70 µm MEA

76 µm cathode flow field

1000 µm cathode gasket

[repeat with next separator plate]

The cooler cells are thinner, at 1.13 mm each:

51 µm separator plate

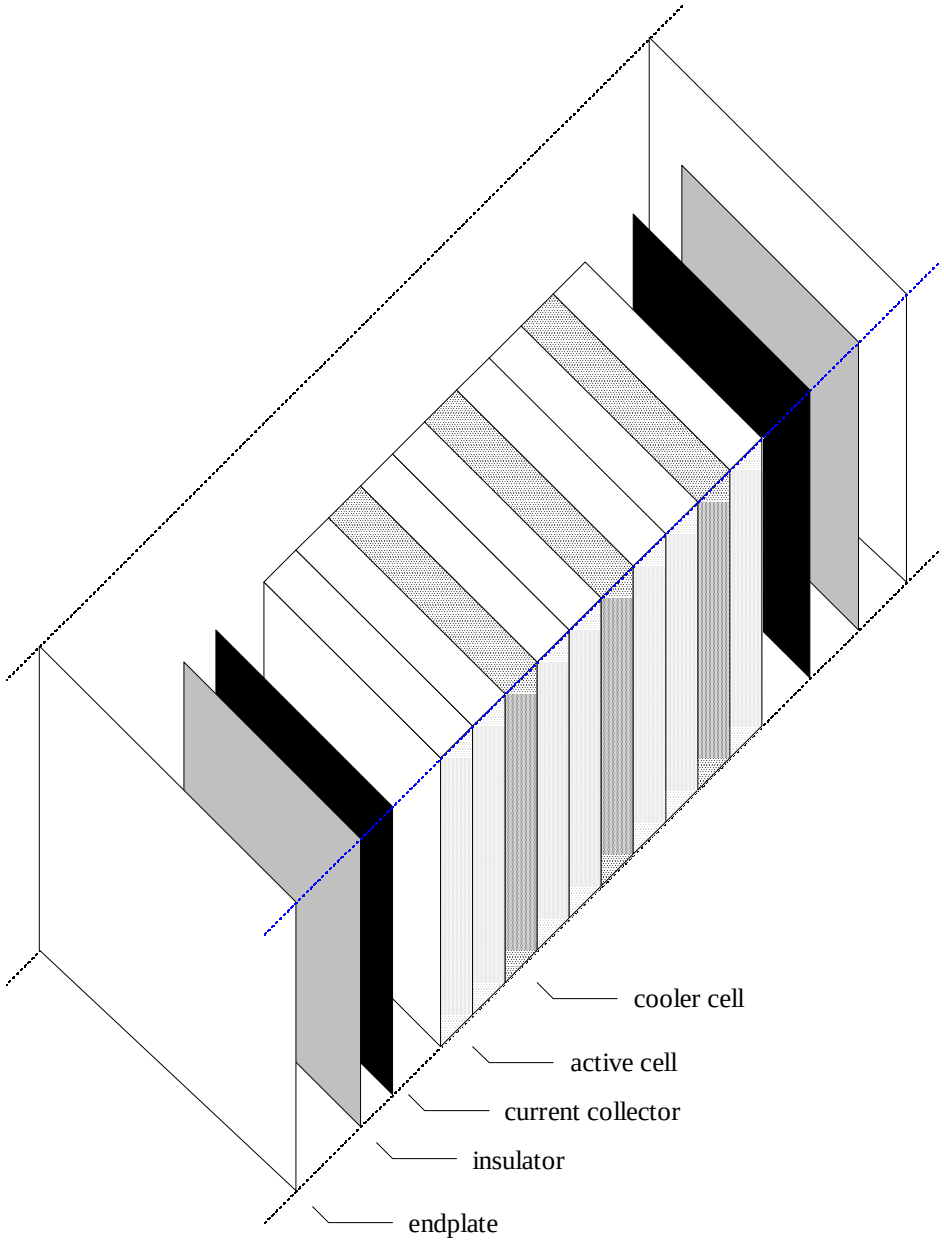
76 µm coolant flow field

1000 µm gasket

[repeat with next separator plate]

With 56 active cells and 28 cooler cells, this produced a total thickness of 15.9 cm. The arrangement of cells is described in Figure B.3 following:

Figure B.3 Diagram of stack assembly



Following the DTI procedure, the dimensions of the repeat components were calculated by taking the active area needed per cell, selecting a height and width, and adding 2.54 cm of inactive membrane area to each dimension to obtain a total membrane area. Next, a 5.1 cm manifold space was added to the width of the membrane (including inactive margin) to obtain the total stack face area, and then 1.27 cm was added to each side of the stack to account for the thickness of the plastic housing and produce the fuel cell stack's overall dimensions as listed in Table B.3 below. (Note that the original DTI stack designs contained two parallel strings of 210 cells, while the design presented here uses a single 56-cell stack.)

Table B.3 Stack dimensions

	5.9 kW	**3.2 kW**	**1.1 kW**
membrane active area	170 cm^2	100 cm^2	35 cm^2
active membrane dimensions,	10.0 cm x 17.0 cm (170 cm^2)	10.0 cm x 10.0 cm (100 cm^2)	5.0 cm x 7.0 cm (35 cm^2)
total membrane dimensions, including inactive margin	12.5 cm x 19.5 cm (244 cm^2)	12.5 cm x 12.5 cm (156 cm^2)	7.5 cm x 9.5 cm (71 cm^2)
total dimensions of stack face -- includes manifolding	17.6 cm x 19.5 cm (340 cm^2)	17.6 cm x 12.5 cm (220 cm^2)	12.6 cm x 9.5 cm (120 cm^2)
overall dimensions with plastic housing	20.2 cm x 22.1 cm x 17.5 cm	20.2 cm x 15.1 cm x 17.5 cm	20.2 cm x 12.1 cm x 17.5 cm
total volume	7.8 L	5.3 L	3.2 L

B.3 Weight of stack

B.3.1 Weight of non-repeat components

The weights of the non-repeat components (plastic stack housing, two endplates, two insulators, two current collectors, sixteen tie-rods) were calculated based on their dimensions and

extrapolation from the DTI study.

The stack housing mass was based on previously calculated stack housing volume and a density of 0.576 g•cm^{-3} for the plastic housing material.

The sixteen tie-rods were estimated at 2.05 kg for DTI's default equal-voltage 52.5 cm stack. Since length is only 15.9 cm in this fuel cell, the tie-rod mass was pro-rated down to 0.6 kg.

The insulator and current collectors cap the ends of the array of cells and thus have an area equal to the total membrane area. They were regressed against total membrane area. The endplates, containing reactant and exhaust ports, tie rod holes, and serving as structural support for the stack, cover the entire face area of the stack. The endplate weights were regressed against stack face area. However, the regressions for the current collectors and endplates became negative at the low stack areas designed for the hybrid, so they were not used.

So, in the case of the 100 cm^2 and 35 cm^2 stacks, the current collector and endplate weights were simply taken to be the same as the weights for the DTI 116 cm^2 stack. For the 170 cm^2 stack, the current collector weight was interpolated between the weights for the collectors obtained for the DTI 116 cm^2 and 181 cm^2 stacks. The endplate, of area 344 cm^2, was set equal to the weight of the endplate for the DTI 116 cm^2 (452 cm^2 face area) stack.

The linear regression was retained for the insulator weights. The results for all of the parts were:

Table B.4 Non-repeat stack component weights

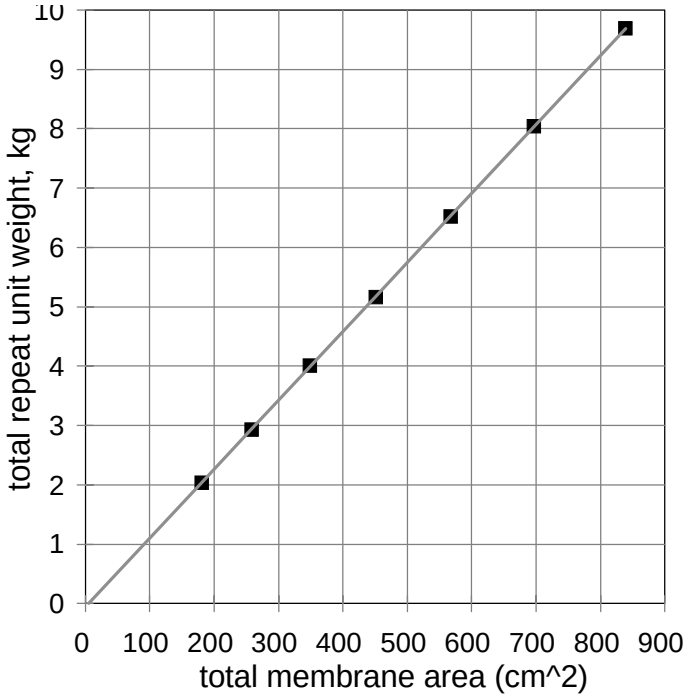
	5.9 kW	**3.2 kW**	**1.1 kW**
2 insulators	59 g	45 g	45 g
1 stack housing	1.32 kg	0.99 kg	0.66 kg
2 endplates	0.26 kg	0.26 kg	0.26 kg
2 current collectors	1.11 kg	0.71 kg	0.71 kg
16 tie rods	0.59 kg	0.59 kg	0.59 kg
total non-repeat weight	4.8 kg	3.6 kg	3.3 kg

B.3.2 Weight of repeat components

The repeat components were modeled as follows. First, the masses of the repeat units (separator plate, anode flowfield, cathode flowfield, gasket, MEA) were taken from DTI's values for the rubber gasket and MEA, and calculated from dimensions and densities for the separator plates and flowfields. A stainless steel density of 8 kg/L was used. This was done for each of the sizes studied in the DTI report; for example, the weights were 28.2 g for each active cell and 16.4 g for each cooling cell for a 116 cm^2 active area.

Then, the total weights of the repeat units were summed for all the 56 cooler cells and 28 active cells, and expressed as a function of total membrane area for the membrane sizes studied in the DTI study:

Figure B.4 Regression of total repeat unit weight



A straight line regression fit well to the repeat masses; because the majority of the weight comes from the stainless steel plates, and these plates' weights vary linearly with changing membrane area (because thickness is constant)

Table B.5 Total weight of stack repeat units

	5.9 kW	**3.2 kW**	**1.1 kW**
total membrane dimensions (including inactive)	12.5 cm x 19.5 cm	12.5 cm x 12.5 cm	7.5 cm x 9.5 cm
total membrane area	245.0 cm^2	157.3 cm^2	71.9 cm^2
total repeat unit weight	2.8 kg	1.8 kg	0.8 kg

B.4 Summary of stack weight and volume

The total stack weight and volume are summarized below with power densities.

Table B.6 Summary of size and weight results

	5.9 kW	**3.2 kW**	**1.1 kW**
membrane active area	170 cm^2	100 cm^2	35 cm^2
non-repeat mass	5.1 kg	3.6 kg	3.3 kg
repeat mass	2.8 kg	1.8 kg	0.8 kg
total mass	7.6 kg	5.4 kg	4.0 kg
volume	7.8 L	5.3 L	3.2 L
specific power	0.78 kW/kg	0.62 kW/kg	0.27 kW/kg
power density	0.76 kW/L	0.62 kW/L	0.34 kW/L

In comparison, Ballard reported in a 1995 press release a stack power density of 0.7 kW/L.

B.5 Cost

The costs were summed from minimum values of component costs as a function of cell membrane total area. In the case of this study, the total membrane areas were 244 cm^2, 156 cm^2, and 71 cm^2 for the 5.9 kW, 3.2 kW, and 1.1 kW hybrid scooters respectively. The insulators were the only parts actually regressed because the others became negative at the low membrane areas involved.

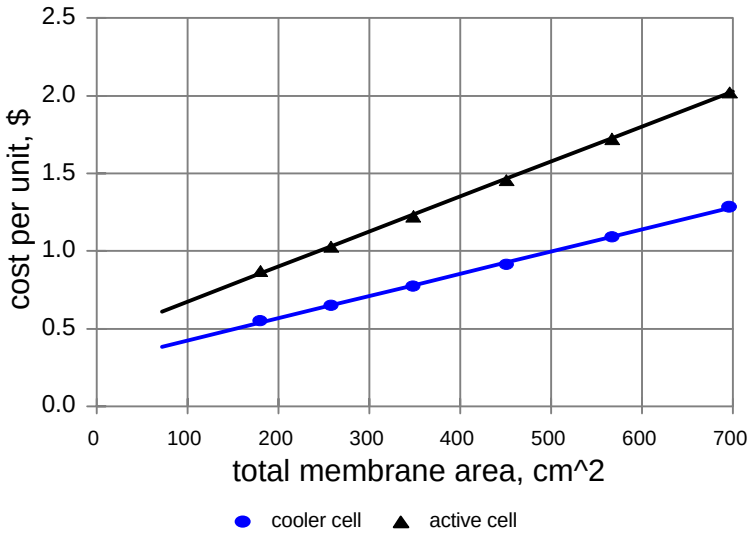
1. Again following DTI reported figures, stack housing prices were calculated at $1.16 per kg of material (based on the previous weight calculations) plus $15 assembly cost.

2. The insulators regressed to $0.24 each for the 5.9 kW stack, $0.20 each for the 3.2 kW stack, and $0.16 for the 1.1 kW stack.

3. The cost of the endplates became negative when regressed versus total membrane area, so the endplates were assumed to be the same cost ($4.02) as those for the smallest (116 cm^2) DTI case. For the same reason, "floor" values of $1.23 were chosen for the current collectors in the 100 cm^2 and 35 cm^2 stacks. The sixteen tie rods were calculated at $1.00 each.

MEA costs were simply the total membrane area multiplied by a cost of 52.3 $/$m^2$ predicted by DTI for mass-produced, low-cost technology. All subcomponents – MEA and cell hardware (gasket, separator plates, flow field plates) – were summed and then regressed against total membrane area for the repeat unit cost:

Figure B.6 Stack repeat unit regression



Finally, $10.00 was added for assembly, and a 10% contingency cost added at the end. The prices broke down as follows:

Table B.7 Cost summary

	price per component (number)	5.9 kW	3.2 kW	1.1 kW
REPEAT	**56 active cells**	56.04	37.75	29.55
	56 MEAs	83.56	34.10	11.94
	28 cooler cells	17.63	11.82	9.21
	assembly line machine cost per cell (total 84 cells)	8.40	8.40	8.40
	TOTAL	165.63	92.07	59.11
NON-REPEAT	**2 current collectors**	3.39	2.46	2.46
	16 tie rods	16.00	16.00	16.00
	2 insulators	0.47	0.40	0.33
	2 endplates	8.04	8.04	8.04
	1 stack housing	16.53	16.14	15.77
	TOTAL	44.44	43.04	42.59
SUMMARY	**final assembly and inspection**	10	10	10
	11% contingency cost	24.21	15.96	12.29
	total stack cost	$244	$161	$124

The costs per kilowatt for the stack are $103/kW for the 1.1 kW stack, $47/kW for the 3.2 kW stack, and $42/kW for the 5.9 kW stack.

These results were compared with the curve-fitting results listed in Table B.2:

Table B.8 Curve-fitting versus bottoms-up model

	Stack power	Curve-fitting result	Bottoms-up result
Stack cost	1.1 kW	$96	$124
	3.2 kW	$125	$176
	5.9 kW	$176	$244
Stack weight	1.1 kW	1.5 kg	4.0 kg
	3.2 kW	2.4 kg	5.4 kg
	5.9 kW	3.6 kg	7.4 kg
Stack volume	1.1 kW	1.6 L	3.2 L
	3.2 kW	2.6 L	5.3 L
	5.9 kW	4.0 L	7.8 L

The simplistic curve fits consistently overestimate stack performance (light weight, low cost, small volume). The more detailed model, which admittedly uses curve fitting in its numerous elements, more accurately captures the fact that parts not only increase in size, weight, and cost non-linearly as cell membrane size decreases, but that they increase more than expected from the initial fit.

Appendix C *radiator performance data*

The radiators used in the model are OEM coils produced by Lytron. The series has the following properties:

Table C.1 Lytron OEM coil weight and contained liquid volume

radiator model	dry (empty) weight	volume of coolant inside
M05-050	0.9 kg	115 mL
M05-100	1.8 kg	188 mL
M10-080	2.3 kg	320 mL
M10-160	3.6 kg	549 mL
M14-120	4.5 kg	606 mL
M14-240	7.3 kg	1090 mL

Figure C.1 shows performance curves of cooling factor (W/K) versus air flow rate (cfm) and coolant flow rate (gpm). Note that the M14-120 curve was accidentally printed twice on the data sheet from Lytron; the bottom figure is incorrect.

The source of the data in this appendix is the Lytron web site, “Lytron OEM Heat Exchangers (Radiators) Performance Curves” http://www.lytron.com/Catalog/oemperf.htm and “Lytron Manufacturers of Thermal Transfer Solutions”, http://www.lytron.com/Catalog/techwght.htm. Both sites were last accessed June 1999.

photocopy not included in PDF;

please see

http://www.lytron.com/Catalog/oemperf.htm

Appendix D *conversion factors*

1 gallon	3.785 L	
1 gallon per minute	0.06308 L/s	
1 cubic foot	28.32 L	
1 cubic foot per minute	0.472 L/s	
1 mole	22.5 L	(standard conditions)
1 atm	1.013 bar	
	760 mmHg	
	14.7 psi	
	101.3 kPa	
	407 in H_2O	
1 calorie	4.18 J	
1 BTU	1055 J	
1 horsepower	746 W	
1 ampere per ft^2 (“ASF”)	1.0764 $mA \cdot cm^{-2}$	
1 foot	30.48 cm	
1 pound	0.454 kg	
1 mile	1.609 km	

Appendix E *acronyms and abbreviations*

AIV	Aluminum Intensive Vehicle
BDC	Bottom Dead Center
CAFE	Corporate Average Fuel Economy
CFM	Cubic Feet per Minute
CSC	China Steel Corporation
DoD	Depth of Discharge
DFI	Direct Fuel Injection
DMFC	Direct Methanol Fuel Cell
ECE	Economic Commission for Europe
FHDS	Federal Highway Driving Schedule

FTP	Federal Test Procedure
FUDS	Federal Urban Driving Schedule
GCV	Gross Calorific Value (used in natural gas industry for HHV)
HHV	Higher Heating Value
ITRI	Industrial Technology Research Institute [Taiwan]
LHV	Lower Heating Value
LNG	Liquefied Natural Gas
MEA	Membrane Electrode Assembly
MIRL	Mechanical Industry Research Laboratory [ITRI]
MCFC	Molten Carbonate Fuel Cell
MPGE	Miles Per Gallon (Equivalent)
NCV	Net Calorific Value (lower heating value)
NGM	New Generation Motors
NiMH	Nickel Metal Hydride
NTD	New Taiwan Dollars
OEM	Original Equipment Manufacturer
PNGV	Partnership for a New Generation of Vehicles
PAFC	Phosphoric Acid Fuel Cell
PEMFC	Proton Exchange Membrane Fuel Cell (also Polymer Electrolyte Membrane Fuel Cell)
PM	Particulate Matter
PTFE	Polytetrafluoroethylene
ROC	Republic of China
SAE	Society of Automotive Engineers
SOC	State of Charge [of batteries]
SOFC	Solid Oxide Fuel Cell
TDC	Top Dead Center
THC	Total Hydrocarbons
TMDC	Taipei Motorcycle Driving Cycle
TSP	Total Suspended Particulates
UQM	Unique Mobility
USD	United States Dollars
VAC	Volts Alternating Current
VKT	Vehicle-Kilometers Traveled
VMT	Vehicle-Miles Traveled
ZES	Zero Emission Scooter

Appendix F *MATLAB simulation program listing*

This is a listing of the MATLAB m-file program used to simulate scooter performance over various driving cycles, `launch14.m`

```
% launch14.m
%
% Test bed to run various configurations through a specified driving cycle.
% Version 11 revises the efficiency calculation and changes baseparasitics
% Version 12 includes parasitic power in the final plot
% Version 13 refills the state of charge frequently - not just by regen
% Version 14 cleans up the code

more off; clear


%% STACK HEAT PARAMETERS
%
% stackmass is in kg, divided by two for the part in thermal contact
% with the cells; heatcapacity is in kJ/kg/C (water is 4.19,aluminum
% is 0.900, copper 0.386. We estimate 1.0 for stack) cooleff is in W/C
% and is different depending on which hybrid design is used

ambientTemp=40; initTemp=50;
stackmass=2.8; specheatcap=0.929;


%% CHOOSE TYPE OF HYBRIDIZATION (IF ANY)
%
% Select one of four configurations; adjust parameters accordingly
% The various hybrid versions reuqire different fuel cell areas,
% numbers of peaking power battery cells, and different parasitic power
% loads. "kickin" defines the power (watts) at which the battery kicks in.

disp(sprintf('Pick:\n  1 for pure FC\n  2 for 3.3 kW\n  3 for 1.1 kW\n  4 for
elec hybrid'));
hybridtype=input(' ?');
switch hybridtype
 case 1,
  cooleff=110; baseparasite=39.7; cellarea=170; numbolder=40;kickin=99000;
 case 2,
  cooleff=150; baseparasite=66.0; cellarea=100; numbolder=27; kickin=3020;
 case 3,
  cooleff=50; baseparasite=25.3; cellarea=35;  numbolder=47; kickin=1000;
 case 4,
  cooleff=35; baseparasite=0; cellarea=90;  numbolder=38; kickin=1830;
 otherwise,
  disp('Unknown option')
  keyboard
end


%% SET SCOOTER PHYSICAL PARAMETERS
%
% Crr     (coefficient of rolling resistance, dimensionless)
% Af      (frontal area, m^2)
% Cd      (drag coefficient, dimensionless)
% mass    (total mass of vehicle + driver)
% effd    (drive train efficiency - about 70%)
% paux    (auxiliary power, W)
```

```
g=9.81; rho=1.23;
Crr=0.014; mass=130+75;
Af=0.6; Cd=0.9;
effd=0.77; paux=60;


%% POLARIZATION CURVE
%
% Set number of cells in stack; area, in cm^2, of each stack. We define the
% polarization curve once, here, so we don't have to calculate efficiencies
% each time_inside_ the loop. Polarization curve is modeled with an
% analytic formula based on a least-squares fit to experimental data from
% Energy Partners (see reference in main body of thesis)
%
% power_for_density calculates gross power output for a given current
% density. vatmo is the voltage under atmospheric pressure for a given
% current density.

numcells = 56;
max_current_density=1800;
jaxis=[1:1:max_current_density];
for j=1:1:max_current_density,
 vatmo(j)=1.00-0.0260*log(j)-(2.015e-4)*j-(1.113e-5)*exp((6.00e-3)*j);
 power_for_density(j)=vatmo(j)*j*cellarea*numcells/1000;
end


%% DEFINE PEAKING BATTERY PARAMETERS
%
% We define efficiency and power here. The only reason we care about
% current is that we want to make sure the maximum charge/discharge
% current is not exceeded. Max current is in amps.
%
% numbolder   number of bolder peaking power cells
% capacity       maximum energy storable (J)
% batteryweight  weight of batteries (kg)
% effregen       fraction of kinetic energy available
% battenergy     current charge
% initSOC        initial state of charge (battenergy/capacity)
% currSOC        current state of charge
% regenerated    total energy regenerated so far (J)
% friction    total energy lost to friction in braking if no regen (J)

capacity=1*12*(numbolder/6)*60*60;
batteryweight=0.7173*(numbolder/6);
effregen=0.7;
initSOC=0.5; currSOC=initSOC;
battenergy=capacity*currSOC;
regenerated=0; friction=0;

%% DRIVING CYCLE DEFINITION
%
% Load in a driving cycle; uncomment to choose driving cycle.
% Note that FUDS must be converted to km/h as it is in mph.
% Timestep is defined here because different cycles have
% different time intervals.

%load ftp75.txt -ascii; vinput=ftp75; clear ftp75;
%timestep=1;

%load v2.txt -ascii; vinput=v2; clear v2;
%timestep=0.1;

%load v3.txt -ascii; vinput=v3; clear v3;
%timestep=1;

load realtmdc.txt -ascii; vinput=realtmdc; clear realtmdc;
```

```
timestep=1;

%load fuds.cycle -ascii; vinput=fuds; clear fuds;
%for i=1:1:size(vinput,1); vinput(i,1)=vinput(i,1)*1.609; end
%timestep=1;

%load j1082.txt -ascii
%vinput=j1082; clear j1082
%timestep=0.1;

%load ece40.txt -ascii; vinput=ece40; clear ece40;
%timestep=0.1;

v(1)=0; t(1)=0; cyclelength=size(vinput,1);

%% SMOOTH THE DRIVING CYCLE (BOX SMOOTH)
%
% (uncomment to use. Essentially we define temporary velocity vx
% and then overwrite the original vinput with vx when done smoothing)
%
% vx=vinput;
% vx(1,2)=vinput(1,2); vx(2,2)=vinput(2,2);
% vx(3,2)=vinput(1,2); vx(4,2)=vinput(2,2);
% vx(cyclelength,2)  =vinput(cyclelength,2);
% vx(cyclelength-1,2)=vinput(cyclelength-1,2);
% vx(cyclelength-2,2)=vinput(cyclelength-2,2);
% vx(cyclelength-3,2)=vinput(cyclelength-3,2);
% for i=3:1:cyclelength-1
%  vx(i,2) = ( 0.50*vinput(i,2) + ...
%              0.30*vinput(i-1,2)+0.20*vinput(i-2,2) );
% end
% vinput=vx;


%% CONVERT KM/H to M/S
for i=1:1:cyclelength
 t(i)=vinput(i,1)-timestep;
 v(i)=vinput(i,2)/3.6;
 if v(i)<0, v(i)=0; end
   %  Uncomment to clip if speed greater than 40 km/h
   %  if v(i)>(40/3.6), v(i)=40/3.6; end
 end

%% PAUSE TO ALLOW PLOTTING OF DRIVING CURVE
%
disp('plot your graph now')
keyboard

%% SMOOTH THE DRIVING CYCLE (LOW-PASS)
%
% note that the cutoff frequency - or rather cutoff period -
% occurs at To = 2pi/s0; periods below To are attenuated
% the smaller the To, the less smoothing. smaller s0 = more smoothing.
%
% If I wanted to close the loop I would use this
% [numc,denc]=feedback(num,den,gclose,1);

s0=1.5;  k=1; gclose=1;
num=k;
den=[1 s0];
v2=transpose(lsim(num,den,v,t))*s0;
v=v2; clear v2;


%% EXAMINE DRIVING CYCLE
%
% Evaluate driving cycle to calculate acceleration from finite
```

```
% differences of velocity; also, determine what fraction of the
% driving cycle is spent accelerating, decelerating, etc.
%
% plustime    if positive acceleration
% minustime   deceleration
% steadytime  steady velocity, no acceleration
%
% We also accumulate acceleration values to separate out
% average of positive accelerations and average of
% negative accelerations.

plustime=0; minustime=0; steadytime=0;
for i=2:1:cyclelength
 a(i)=(v(i)-v(i-1))/timestep;
 if a(i)>0,
  aplus(i)=a(i);
  aminus(i)=0;
  plustime=plustime+1;
 elseif a(i)<0,
  aminus(i)=a(i);
  aplus(i)=0;
  minustime=minustime+1;
 else
  aplus(i)=0;
  aminus(i)=0;
  steadytime=steadytime+1;
 end
end

clear vinput;
if ~((size(v) == size(a)) & (size(v) == size(t)))
 disp(sprintf('Error! Vectors v, a, and t are not the same size'));
 return
end

%% INITIALIZE OTHER VARIABLES
%
% bin is used to create a histogram of power consumption.
% fuelenergy accumulates the LHV of the fuel used
% distance is used to verify that the correct total distance is traveled

bin=zeros(1,30); fuelenergy=0; distance=0;


%% START THE MAIN LOOP

for i=1:1:cyclelength

 % p1 is the acceleration power
 % p2 is the rolling resistance power
 % p3 is the wind drag power
 % p4 is auxiliaries
 % p5 is parasitics
 % p6 is power supplied by FC to battery, if hybrid

 p1(i)=mass*a(i)*v(i);
 p2(i)=mass*g*v(i)*Crr;
 p3(i)=0.5*Cd*Af*rho*v(i)^3;
 p4(i)=paux;
 p5(i)=0; p6(i)=0;

 % We calculate output force in order to find torque needed.
 % Negative forces are discarded.
 % pwheels includes physical resistance power only, not aux. / para. / p6
 % p_no_parasitics is either equal to pwheels+auxiliary power+p6 recharge
 %  power, or if regenerative braking is taking place, it is equal to
 %  auxiliary power. Note that parasitic power p5 is not calculated until
```

```
%  later. p_no_parasitics includes output from the FC and

pwheels(i)=(p1(i)+p2(i)+p3(i))/effd;
if pwheels<0,
 p_no_parasitics(i)=p4(i);
else
 p_no_parasitics(i)=pwheels(i)+p4(i);
end


% we also record the instantaneous force in case we want to calculate
% torque. Here we discard values of force when v<0.

force(i)=mass*a(i) + mass*g*Crr + 0.5*Cd*Af*rho*v(i)^2;
if v(i)<0.01, force(i)=0; end


% Set up the histogram of pwheels by sorting each new data point
% into one of the bins. Bin(19) captures all powers greater
% than 9000 W.

if      (pwheels(i)<500),   bin(1)=bin(1)+1;
 elseif (pwheels(i)<1000),  bin(2)=bin(2)+1;
 elseif (pwheels(i)<1500),  bin(3)=bin(3)+1;
 elseif (pwheels(i)<2000),  bin(4)=bin(4)+1;
 elseif (pwheels(i)<2500),  bin(5)=bin(5)+1;
 elseif (pwheels(i)<3000),  bin(6)=bin(6)+1;
 elseif (pwheels(i)<3500),  bin(7)=bin(7)+1;
 elseif (pwheels(i)<4000),  bin(8)=bin(8)+1;
 elseif (pwheels(i)<4500),  bin(9)=bin(9)+1;
 elseif (pwheels(i)<5000),  bin(10)=bin(10)+1;
 elseif (pwheels(i)<5500),  bin(11)=bin(11)+1;
 elseif (pwheels(i)<6000),  bin(12)=bin(12)+1;
 elseif (pwheels(i)<6500),  bin(13)=bin(13)+1;
 elseif (pwheels(i)<7000),  bin(14)=bin(14)+1;
 elseif (pwheels(i)<7500),  bin(15)=bin(15)+1;
 elseif (pwheels(i)<8000),  bin(16)=bin(16)+1;
 elseif (pwheels(i)<8500),  bin(17)=bin(17)+1;
 elseif (pwheels(i)<9000),  bin(18)=bin(18)+1;
else bin(19)=bin(19)+1;
end


%% HYBRIDIZATION
% battenergy   current energy in battery (J)
% battery(i)   an array storing battery energy over time (J)
% pfc          power from the fuel cell
% pbatt        battery out of cell. pbatt<0 is charging
% SOC          state of charge over time (fraction)
% batt_V       current battery voltage (V)
% batt_R       instantaneous battery resistance (ohms)
% lid          maximum current battery can be charged at for given SOC (A)
% negpower     power regenerated into the batteries (net of eff.) (W)

battery(i)=battenergy; negpower(i)=0;
pfc(i)=0; pbatt(i)=0; SOC(i)=0;
batt_V = 1.88+0.375*currSOC-0.176*currSOC*currSOC;
batt_R = (2.09-1.28*currSOC+1.07*currSOC*currSOC)/1000;
lid(i) = 1*0.979*(currSOC^(-4.44)-1);

% "overflow" measures power over "kickin". if overflow>0, require
% energy from the battery. if overflow<0, allow recharge. We restrict
% the battery so it never depletes below 20% capacity (to avoid
% battery damage) and never discharges faster than 80 amps.

overflow=p_no_parasitics(i)-kickin;
if overflow>0,
```

```
    if (battenergy-overflow*timestep/.95) > 0.20*capacity;,
        powerneed=overflow;
    else
        powerneed=(battenergy-0.20*capacity)*.95/timestep;
    end

      % count up current until desired power is reached.
      % Ensure discharge current under maximum of 80 A

      battcurr=0;
      while (battcurr*numbolder*(batt_V+battcurr*batt_R)) < powerneed,
       battcurr=battcurr+0.1;
      end;

    % diagnostic: display battcurr
    % battcurr
    if battcurr>80,
     battcurr=80;

     % diagnostic: flag if max charging exceeded
     % disp(sprintf('max charging current exceeded at time %5.0f',i));
    end

      % update with new energy, state of charge, voltage ,resistance
      % efficiency is a function of instantaneous voltage and resistance

      effbatt(i) = sqrt(0.95)*batt_V/(batt_V+battcurr*batt_R);
      battenergy=battenergy-powerneed/effbatt(i)*timestep;
      SOC(i)=battenergy/capacity; pbatt(i)=powerneed;
      batt_V = 1.88+0.375*SOC(i)-0.176*SOC(i)*SOC(i);
      batt_R = (2.09-1.28*SOC(i)+1.07*SOC(i)*SOC(i))/1000;

elseif pwheels(i)<0,
      % If powerneed is positive, and deceleration "power" exceeds
      % drag and rolling resistances, then we can regenerate into
      % the battery. pbatt is negative when charging, and we make sure
      % not to charge over 80% of capacity, or faster than lid(i),
      % to avoid battery damage

     friction=friction+pwheels(i);

      if (battenergy+abs(pwheels(i))*effregen*timestep*.95) < 0.80*capacity,
       powerneed=abs(pwheels(i))*effregen;
      else
       powerneed=abs((0.80*capacity-battenergy)*effregen/timestep/.95);
      end

      % count up current until desired power is reached.
      % Ensure charging current under maximum of lid(i)

      battcurr=0;
      while (battcurr*numbolder*(batt_V+battcurr*batt_R)) < powerneed,
       battcurr=battcurr+0.1;
      end;
      if battcurr>lid(i), battcurr=lid(i);
      % diagnostic: flag if max charging exceeded
      % disp(sprintf('Exceed charging lid at %5.0f',i));
      end

      effbatt(i) = sqrt(0.95)*batt_V/(batt_V+battcurr*batt_R);
      negpower(i)=powerneed*effbatt(i);
      battenergy=battenergy+negpower(i)*timestep;
      regenerated=regenerated+negpower(i)*timestep;
      SOC(i)=battenergy/capacity; pbatt(i)=-powerneed;
      batt_V = 1.88+0.375*SOC(i)-0.176*SOC(i)*SOC(i);
      batt_R = 2.09-1.28*SOC(i)+1.07*SOC(i)*SOC(i);
else
```

```
      % i.e. if neither regenerating nor drawing from battery.
      % first check to see if SOC is low and we should refill battery
      % we refill at a rate 400 W minus current power demand, only if
      % this is a hybrid and if current power demand is less than 400 W.

    if currSOC<0.55,
     p6(i)= 400 - p_no_parasitics(i);
     if (p6(i)<0 | hybridtype == 1), p6(i)=0;  end

     battcurr=0;
     while (battcurr*numbolder*(batt_V+battcurr*batt_R)) < p6(i),
      battcurr=battcurr+0.1;
     end;

     if battcurr>lid(i), battcurr=lid(i);
      % diagnostic: flag if max charging exceeded
      % disp(sprintf('Exceed charging lid at %5.0f',i));
     end

     effbatt(i) = sqrt(0.95)*batt_V/(batt_V+battcurr*batt_R);
     negpower(i)=p6(i)*effbatt(i);
     battenergy=battenergy+negpower(i)*timestep;
     SOC(i)=battenergy/capacity; pbatt(i)=-p6(i);
     batt_V = 1.88+0.375*SOC(i)-0.176*SOC(i)*SOC(i);
     batt_R = 2.09-1.28*SOC(i)+1.07*SOC(i)*SOC(i);
     if (hybridtype ~= 1),
      p_no_parasitics(i)=p_no_parasitics(i)+p6(i);
     end

   else
      % currSOC>0.4 so no need to refill. Instead, set SOC from the
      % previous value, or if this is the first time through the loop,
      % set it to initSOC. (We need to do this manually here because the
      % other options calculate a new value of SOC at the end
      % automatically)

    p6(i)=0;
    if i==1,
     SOC(i)=initSOC;
    else
     SOC(i)=SOC(i-1);
    end
   end

end
currSOC=SOC(i);

% calculate how much fuel cell has to output, equal to
% p_no_parasitics-pbatt if pbatt>0 (discharge),
% equal to p_no_parasitics if pbatt<0 (charge)

pfc(i)=p_no_parasitics(i)-0.5*(abs(pbatt(i))+pbatt(i));


%% CALCULATE PARASITIC POWER
% note that no parasitic power is appropriate for the battery-only case
%
% p5         Parasitics. Base plus 50-200 W blower depending on power.
% pfc        Gross power from fuel cell (or zinc-air battery), no peaking
% poutput    Total system output including peaking power

p5(i)=baseparasite+50+200*pfc(i)/5900;
if (hybridtype == 4), p5(i)=0; end
pfc(i)=pfc(i)+p5(i);
poutput(i)=p_no_parasitics(i)+p5(i);
```

```
 % Calculate current density j (mA) at which desired power is
 % attained. Calculate efficiency from vatmo and accumulate LHV
 % of fuel used. effp is the fuel cell efficiency. 100% efficiency
 % for zinc-air battery, because energy in that case is measured in
 % terms of _output_.

 j=1; while (power_for_density(j)<pfc(i)), j=j+1; end;

 effp(i)=vatmo(j)/1.481; currentdensity(i)=j;
 if (hybridtype == 4), effp(i)=1; end
 fuelenergy=fuelenergy+pfc(i)*timestep/effp(i);


 % Include metal hydride cooling. heat1 is waste heat generated.
 % heat2 is heat1 minus hydridecool, which is 28 kJ/mol times the
 % number of molesH2 per second. This formula works if current density
 % j is given in mA.

 heat(i)=(pfc(i)/effp(i))*(1-effp(i));
 molesH2=currentdensity(i)/1000*cellarea/96485*numcells/2;
 hydridecool=28000*molesH2;
 heat2(i)=heat(i)-hydridecool;
 if heat2(i)<0, heat2(i)=0; end


 % "coolpower" is the amount of heat that can be removed (watts)
 % by the system with given cooling factor "cooleff" given the
 % instananeous difference between the stack temperature
 % and ambient temperature. "Temp" is the current stack temperature
 % If stack temperature is below the critical thermostat level of
 % 50 degrees C, no coolant is circulated and we assume only a
 % small, unintentional loss of 1 W/K.

 if i==1,
  coolpower=(initTemp-ambientTemp)*cooleff;
 else
  if Temp(i-1)<50,
   coolpower=(Temp(i-1)-ambientTemp)*1;
  else
   coolpower=(Temp(i-1)-ambientTemp)*cooleff;
  end
 end


 % We calculate how much the stack temperature changes in this
 % time step, based on maximum cooling and the heat generated
 % heat2.

 deltaT=(heat2(i)-coolpower)/1000/(stackmass*specheatcap)*timestep;
 if i==1,
  Temp(i)=initTemp;
 else
  Temp(i)=Temp(i-1)+deltaT;
 end
  coolpowerstore(i)=coolpower;

% this marks the end of the master loop
end


%% CALCULATE SUMMARY VARIABLES
%
% Note mpge fuel economy is calculated using 21.56 mpge ~ 1 km/kWh
% Also, overall conversion efficiency is electricity divided by enthalpy
%  of hydrogen - is not net of parasitics.

totaltime    = max(t);
```

```
avgspeed     = mean(v);
distance     = avgspeed*totaltime;
fueleconomy  = distance*21.56/(fuelenergy/3600);
fueleconomy2 = (distance/1000)/(fuelenergy/3600000);
avgeffp      = mean(effp);
[maxpower,i1]= max(poutput);
batteryecon  = avgspeed*3.6/(mean(pfc)/1000);

%% PRINT OUT NUMERIC RESULTS

disp(sprintf(' '))
disp(sprintf(' ------------ simulation results ---------- '))
disp(sprintf(' '))
disp(sprintf('avg speed (km/h) : %5.2f', avgspeed*3.6));
disp(sprintf('total dist. (m)  : %5.0f', distance));
disp(sprintf('Max power from the engine occurs at t=%5.1f and is %5.0f
W',t(i1),maxpower));
disp(sprintf('avg power from engine is (W) : %5.1f', mean(poutput)));
disp(sprintf('avg power, no parasitics (W) : %5.1f', mean(p_no_parasitics)));
disp(sprintf('avg power from FC is (W)     : %5.1f', mean(pfc)));
disp(sprintf('Overall Efficiency           : %5.1f
%%',mean(pfc)*100/(fuelenergy/max(t)) ))
disp(sprintf('Hydrogen LHV usage rate  (W) : %5.1f', fuelenergy/totaltime));
disp(sprintf('Equiv. fuel economy          : %5.1f mpge',fueleconomy))
if (hybridtype == 4),
 disp(sprintf('Fuel economy battery no para : %5.1f km/kWh
battery',batteryecon))
else
 disp(sprintf('Fuel economy 2               : %5.3f km/g
hydrogen',fueleconomy2*33/1000))
end

disp(sprintf(' '))
[y3,i3]=max(a);
[y4,i4]=min(a);
disp(sprintf('Max accel occurs at t=%5.1f and is %5.2f m/s^2',t(i3),y3));
disp(sprintf('Min accel occurs at t=%5.1f and is %5.2f m/s^2',t(i4),y4));
disp(sprintf('Std dev of acceleration is %5.2f', std(a)  ));
disp(sprintf('Max pwheels is %5.2f', max(pwheels)  ));
disp(sprintf(' '))

disp(sprintf('Max fuel cell power output is %5.2f W',max(pfc)     ));
disp(sprintf('Max battery   power output is %5.2f W',max(pbatt)   ));
disp(sprintf('Max battery   power in     is %5.2f W',-max(-pbatt) ));
disp(sprintf('Max battery energy         is %5.2f kJ',max(battery)/1000));
disp(sprintf('Min battery energy         is %5.2f kJ',min(battery)/1000));
disp(sprintf('Total kJ regenerated       is %5.2f kJ',(regenerated/1000)));
disp(sprintf('Area per cell at 614 mW/cm2 = %5.2f cm2',max(pfc)/.614/56));

% PLOT 1. BATTERY vs. FUEL CELL
subplot (311), plot (t,battery)
xlabel ('time, seconds')
ylabel ('battery energy, J')
grid on

subplot (312), plot (t,pfc)
xlabel ('time, seconds')
ylabel ('FC power, W')
grid on

subplot (313), plot (t,pbatt)
xlabel ('time, seconds')
ylabel ('battery power, W')
grid on


% avgposaccel is sum of positive accelerations divided by total cycle time
```

```
% avg acceleration is sum of positive and negative accelerations divided
%  by total cycle time

keyboard;
disp(sprintf(' '))
disp(sprintf('avg accel   power: %5.1f', mean(p1)));
avgposaccel=0;
for i=1:1:cyclelength,
 if p1(i)>0, avgposaccel=avgposaccel+p1(i); end
end
avgposaccel=avgposaccel/cyclelength;
disp(sprintf('avg pos accel    : %5.1f', avgposaccel ));
disp(sprintf('avg rolling power: %5.1f', mean(p2)));
disp(sprintf('avg drag    power: %5.1f', mean(p3)));
disp(sprintf('avg parasit power: %5.1f', mean(p5)));
disp(sprintf(' '))


% nonzerotime measures time spent in motion, i.e. not stationary
% "avg nonzero" refers to power when scooter is in motion
% plustime, calc'ed earlier, measures time when scooter is accelerating
% avg positive accel power is mean acceleration when scooter is accelerating

nonzerotime=cyclelength;posaccelpower=0;
for i=1:1:cyclelength
 if v(i)<0.001, nonzerotime=nonzerotime-1;end
 if p1(i)>0, posaccelpower=posaccelpower+p1(i); end
end
disp(sprintf('avg positive accel   power: %5.1f', posaccelpower/plustime));
disp(sprintf('avg nonzero  rolling power: %5.1f', sum(p2)/nonzerotime ));
disp(sprintf('avg nonzero  drag    power: %5.1f', sum(p3)/nonzerotime ));
disp(sprintf(' '))


% PLOT 2. plot velocity versus time, power versus time, parasitics

orient portrait
set(gcf,'paperposition',[0.8 3 7 5])

subplot (311), plot (t,v*3.6)
xlabel ('time, seconds')
ylabel ('speed, km/h')
orient landscape
grid on

subplot (312), plot (t,poutput/1000)
xlabel ('time, seconds')
ylabel ('power output, kW')
grid on

subplot (313), plot (t,p5)
xlabel ('time, seconds')
ylabel ('parasitic power, W')
grid on

keyboard
clf


% PLOT 3. plot heat characteristics versus time

subplot(311),plot(t,heat2)
title('heat generation after hydride')
ylabel('heat power, W')
xlabel('time ,s')
grid on
```

```
subplot(312),plot(t,Temp)
title('stack temperature')
ylabel('temperature, C')
xlabel('time ,s')
grid on

subplot(313),plot(t,coolpowerstore)
title('cooling power')
ylabel('cooling, W')
xlabel('time ,s')
grid on



%PLOT 4
% plot breakdown of various powers. include parasitics, regen'ed
% power. The zeroing is necessary to get the patch polygon filled in

keyboard
p1(cyclelength-1)=0;p2(cyclelength-1)=0;p3(cyclelength-1)=0;
p4(cyclelength-1)=0;p5(cyclelength-1)=0;
p1(cyclelength)=0;p2(cyclelength)=0;p3(cyclelength)=0;
p4(cyclelength)=0;p5(cyclelength)=0;

for i=1:1:cyclelength,
 if p1(i)>0,
  pospower(i)=p1(i);
 else
  pospower(i)=p1(i);
% COMMENT THIS SECTION OUT to show negative acceleration powers
  pospower(i)=0;
 end
end

clf; subplot(111); hold on

% plot (t,-negpower, 'k')
t(cyclelength)=0;

handle_accel=patch(t,p5+p4+(p2+p3+pospower)/effd,'r');
handle_drag =patch(t,p5+p4+(p2+p3)/effd,'b');
handle_roll =patch(t,p5+p4+p2/effd,'g');
handle_ppara =patch(t,p5+p4,'m');
handle_paux =patch(t,p4,'y');

set(handle_accel,'EdgeColor','r')
set(handle_drag, 'EdgeColor','b')
set(handle_roll, 'EdgeColor','g')
set(handle_ppara, 'EdgeColor','m')
set(handle_paux, 'EdgeColor','y')

patch( [570 570 980 980 570], [4600 5850 5850 4600 4600], 'w')

patch( [600 600 650 650 600], [5650 5800 5800 5650 5650], 'r')
patch( [600 600 650 650 600], [5450 5600 5600 5450 5450], 'b')
patch( [600 600 650 650 600], [5250 5400 5400 5250 5250], 'g')
patch( [600 600 650 650 600], [5050 5200 5200 5050 5050], 'm')
patch( [600 600 650 650 600], [4850 5000 5000 4850 4850], 'y')
patch( [600 600 650 650 600], [4650 4800 4800 4650 4650], 'w')

text (670,5750,'acceleration power')
text (670,5550,'aerodynamic drag')
text (670,5350,'rolling resistance')
text (670,5150,'parasitic power')
text (670,4950,'auxiliary power')
text (670,4750,'regenerated power')
```

```
grid on
xlabel ('time, seconds')
ylabel ('power, W')
keyboard



%PLOT 5 plot breakdown of powers _without_ parasitics
%the zeroing is necessary to get the patch polygon filled in

clf; subplot(111); hold on

handle_accel=patch(t,p4+(p2+p3+p1)/effd,'r');
handle_drag =patch(t,p4+(p2+p3)/effd,'b');
handle_roll =patch(t,p4+p2/effd,'g');
handle_paux =patch(t,p4,'y');

set(handle_accel,'EdgeColor','r')
set(handle_drag, 'EdgeColor','b')
set(handle_roll, 'EdgeColor','g')
set(handle_paux, 'EdgeColor','y')

patch( [570 570 980 980 570], [5000 5850 5850 5000 5000], 'w')

patch( [600 600 650 650 600], [5650 5800 5800 5650 5650], 'r')
patch( [600 600 650 650 600], [5450 5600 5600 5450 5450], 'b')
patch( [600 600 650 650 600], [5250 5400 5400 5250 5250], 'g')
patch( [600 600 650 650 600], [5050 5200 5200 5050 5050], 'y')

text (670,5750,'acceleration power')
text (670,5550,'aerodynamic drag')
text (670,5350,'rolling resistance')
text (670,5150,'auxiliary power')

grid on
xlabel ('time, seconds')
ylabel ('power, W')
keyboard


%PLOT 6 battery power versus total power
%the zeroing is necessary to get the patch polygon filled in

clf; subplot(111); hold on

plot (t,ones(1,cyclelength)*max(pfc),'b')
plot_total=poutput;
plot_fc=pfc;
t(cyclelength)=0;
plot_total(cyclelength)=0;
plot_fc(cyclelength)=0;
plot_total(cyclelength-1)=0;
plot_fc(cyclelength-1)=0;

handle_total=patch(t,plot_total,'k');
handle_pfc  =patch(t,plot_fc,'g');

set(handle_total,'EdgeColor','k')
set(handle_pfc,  'EdgeColor','g')

grid on
xlabel ('time, seconds')
ylabel ('power, W')
axis ([0 950 0 6000])

%UNUSED OPTIONS
%
```

```
%set(gca, 'YTickLabel', str2mat('0.0','0.1', '0.2', '0.3', '0.4', '0.5','0.6',
'0.7', '0.8', '0.9', '1.0'))
%set(gca, 'XTick', 0,50,100,150,200,250,300,350,400,450,500,550,
600,650,700,750, 800,850, 900, 950)
%set(gca, 'XTick', [0:50:950])
```

Appendix G *a prototype scooter*

The prototype scooter illustrated here is being constructed by graduate student Arne LaVen at the Desert Research Institute. Interested parties should contact him at arnel@dri.edu or the following address:

Arne LaVen

Energy and Environmental Engineering Center

Desert Research Institute

5625 Fox Ave.

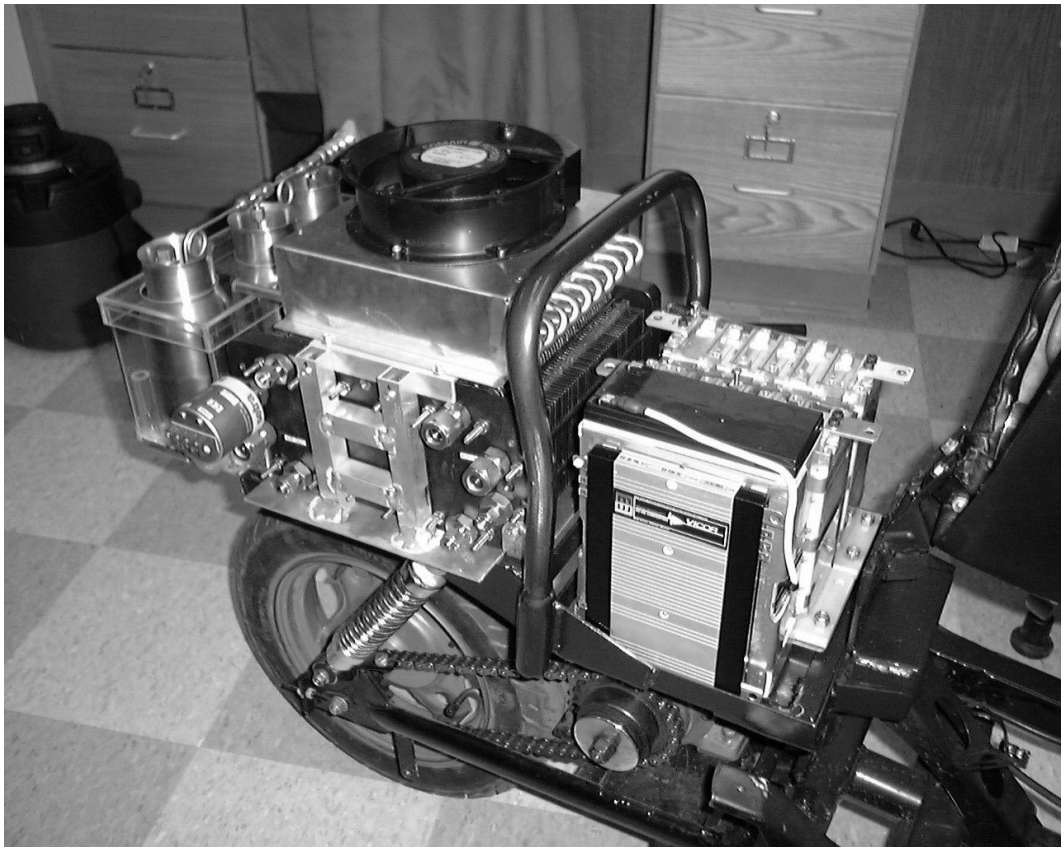
Reno, NV 89506

The photographs below were from Arne LaVen in July of 1999.

Figure G.1 Fuel cell scooter prototype

This is the scooter frame without body. Note the blower attached to the front of the scooter, piping air to the fuel cell over the rear wheel. The power system is shown in more detail in Figure G.2:

Figure G.2 Closeup of scooter power system



The seat has been removed and reveals the fuel cell, underneath a radiator and a fan. At the far back of the scooter are three metal hydride hydrogen canisters. The rear wheel is attached by a chain to the electric motor, which itself is below a DC-to-DC converter. Behind the converter are two ultracapacitors for peaking power.