Battery Health Management System on BeagleBone Black

Presenter: Juan Castellanos-Smith

Electrical Eng. Graduate Student at the University of Texas at Dallas

Safety-Critical Avionics Systems Intern

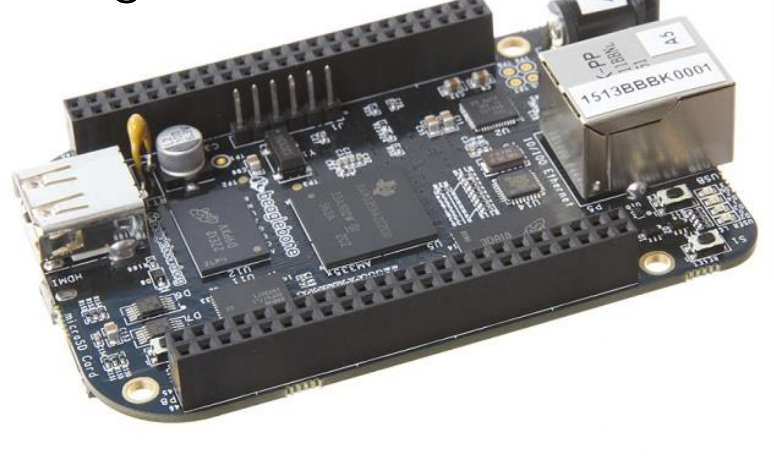
May 5, 2016

Battery Health Management (BHM)

- The BHM program uses an Unscented Kalman Filter (UKF) to estimate remaining battery charge over time. Battery current and voltage are inputs to the UKF.
- The BHM program is compiled from C source code and executed on the Linux based BeagleBone Black (BBB) computer.
- A Linduino (Arduino-compatible) drives the DC1894 analog-to-digital battery stack monitor and sends current and voltage to the BeagleBone Black.
- The DC1894 reads voltages from a battery of interest.
- A current sensor will be used to measure battery current.

Battery Health Management (BHM)

BeagleBone Black



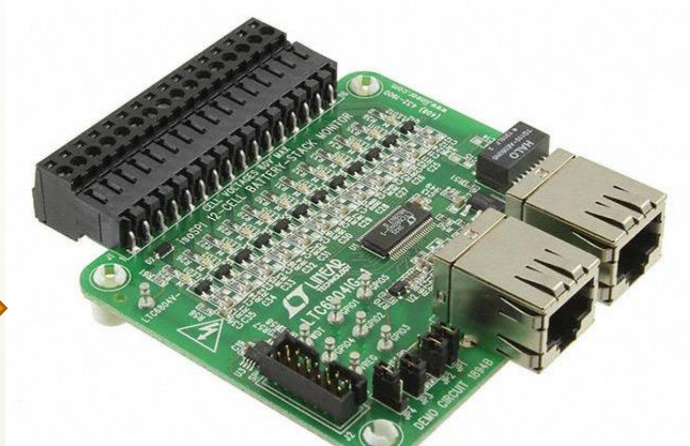
String over USB serial

Linduino

Battery Voltage

Battery Current

Battery



DC1894

Battery Model

(“Battery Charge Depletion Prediction on an Electric Aircraft”)

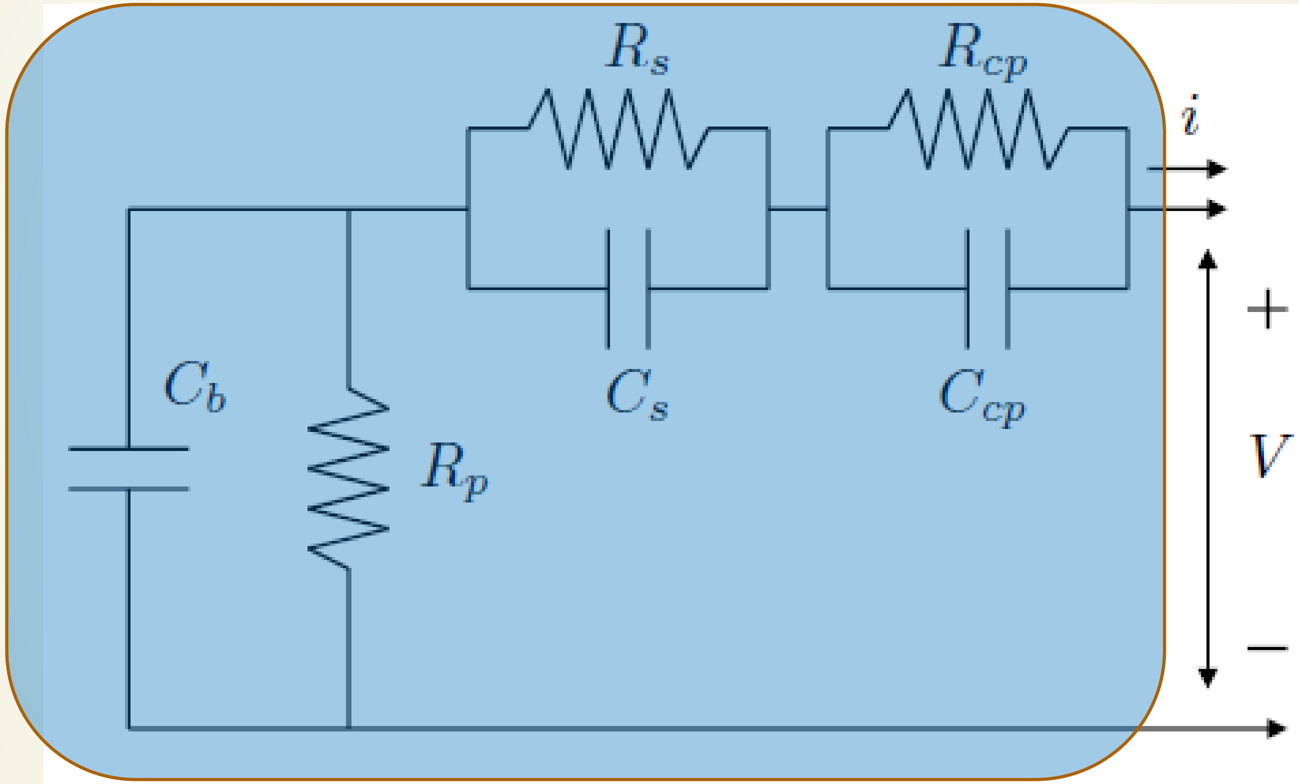


Figure 9. Equivalent circuit battery model

UKF Function in Matlab to be converted to C

UKF_bhm3/UKF_bhm3.m

function [ukfout] = UKF_bhm3(i, Param ,z_ukf, state_ukf, w_ukf, est_ukf, P, Q,R,dt,ndim,zdim)

Inputs:

- i = battery current
- Param = struct of battery model constants
- z_ukf = vector of measurements
- state_ukf = column 1 contains old model states,
  column 2 contains new states
- w_ukf = noise vector
- est_ukf = column 1 has old state estimates,
  column 2 has new estimates
- P = state estimate covariance matrix
- Q = process model noise covariance matrix, assumed constant
- R = measurement noise covariance matrix, assumed constant
- dt = time interval between measurements
- ndim = length of state vector
- zdim = length of measurement vector

UKF Function in Matlab to be converted to C

function [ukfout] = UKF_bhm3(i, Param ,z_ukf, state_ukf, w_ukf, est_ukf, P, Q,R,dt,ndim,zdim)

Outputs:

- ukfout is a struct
  - ukfout.P = updated estimate covariance;
  - ukfout.state_ukf = forecasted model states
  - ukfout.est_ukf = forecasted estimates
  - ukfout.SOC = state of charge

System Dynamics in Matlab

```
qmax = Param.qmax;
Cmax = Param.Cmax;
Ccb0 = Param.Ccb0;
Ccb1 = Param.Ccb1;
Ccb2 = Param.Ccb2;
Ccb3 = Param.Ccb3;
Rs = Param.Rs;
 Cs = Param.Cs;
Rcp0 = Param.Rcp0;
Rcp1 = Param.Rcp1;
Rcp2 = Param.Rcp2;
Ccp0 = Param.Ccp0;
Ccp1 = Param.Ccp1;
Ccp2 = Param.Ccp2;
 Rp = Param.Rp; %From Ed Hogge BHM Checkcase
```

Table 1. Parameter values used in equivalent circuit model

Parameter	Value	Parameter	Value
q_{max}	2.88×10^4 C	C_s	89.3 F
C_{max}	2.85×10^4 C	R_{cp0}	1.60×10^{-3} Ω
C_{Cb0}	19.4 F	R_{cp1}	8.45
C_{Cb1}	1576 F	R_{cp2}	−61.9
C_{Cb2}	41.7 F	C_{cp0}	2689 F
C_{Cb3}	−203 F	C_{cp1}	−2285 F
R_s	2.77×10^{-2}	C_{cp2}	−0.73 F

System Dynamics in Matlab

```
SOC = 1 - (qmax-qb)/Cmax;
Cb = Ccb0 + Ccb1*SOC+Ccb2*SOC^2+Ccb3*SOC^3;
Ccp = Ccp0 + Ccp1*exp(Ccp2*SOC);
Rcp = Rcp0 + Rcp1*exp(Rcp2*SOC);
 %qhat = qmax - (qmax*0.99 - Cmax)*(1-SOC);
%Vb = qhat/Cb; %Diverges. Ed Hogge BHM Checkcase
Vb = qb/Cb;     %Works with this! Only diverges at very end.
Vcs = qcs/Cs;   %Ed Hogge BHM Checkcase
Vcp = qcp/Ccp; %Ed Hogge BHM Checkcase
Vp = Vb - Vcp - Vcs; %Ed Hogge BHM Checkcase
  ip = Vp/Rp; %Ed Hogge BHM Checkcase
  ib = ip + i; %Ed Hogge BHM Checkcase
  icp = ib - Vcp/Rcp; %Ed Hogge BHM Checkcase
    qbdot = -ib; %Ed Hogge BHM Checkcase
    qcpdot = icp; %Ed Hogge BHM Checkcase
    qcsnew = ib*Rs*Cs; %Ed Hogge BHM Checkcase
```

$$\mathrm{SOC} = 1 - \frac{q_{max} - q_b}{C_{max}} \quad (1)$$

$$C_b = C_{Cb0} + C_{Cb1} \cdot \mathrm{SOC} + C_{Cb2} \cdot \mathrm{SOC}^2 + C_{Cb3} \cdot \mathrm{SOC}^3 \quad (2)$$

$$C_{cp} = C_{cp0} + C_{cp1} \cdot \exp\left(C_{cp2}\left(\mathrm{SOC}\right)\right) \quad (3)$$

$$R_{cp} = R_{cp0} + R_{cp1} \cdot \exp\left(R_{cp2}\left(\mathrm{SOC}\right)\right) \quad (4)$$

$$\mathbf{y}^B = V_p = \left[\begin{array}{ccc} \frac{1}{C_b} & \frac{1}{C_{cp}} & \frac{1}{C_s} \end{array}\right] \cdot \mathbf{x}^B \quad (7)$$

$$\mathbf{x}^B = \left[\begin{array}{ccc} q_b & q_{cp} & q_{Cs} \end{array}\right]^T$$

$$\dot{\mathbf{x}}^B$$

System Dynamics in Matlab

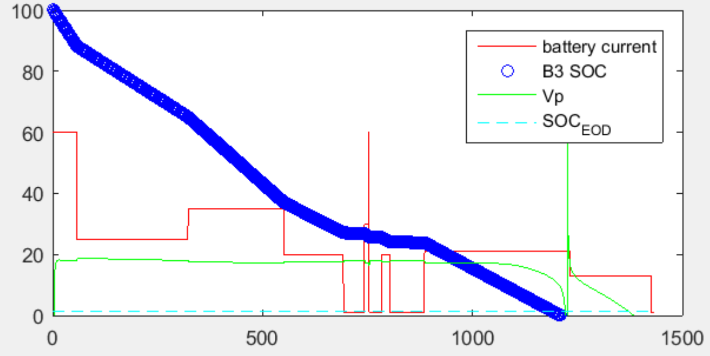
UKF_bhm3/UKF_bhm3.m

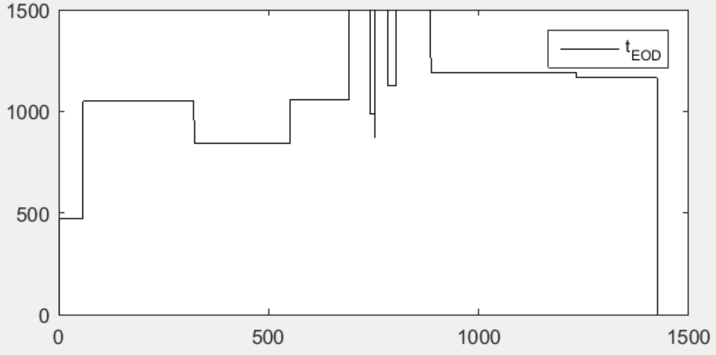
Validation of dynamics as compared to results in “Battery Charge Depletion Prediction on an Electric Aircraft”

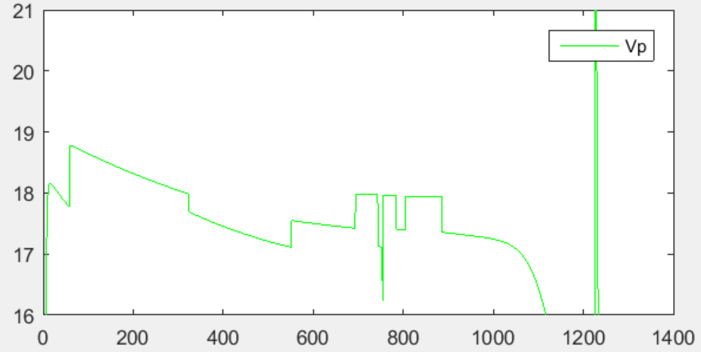
```
% Model state vector
x = [ qb + qbdot*dt;
      qcp + qcpdot*dt;
      qcsnew]; %Ed Hogge BHM Checkcase
```

$$\mathbf{x}^B = \begin{bmatrix} q_b & q_{cp} & q_{Cs} \end{bmatrix}^T \quad (5)$$

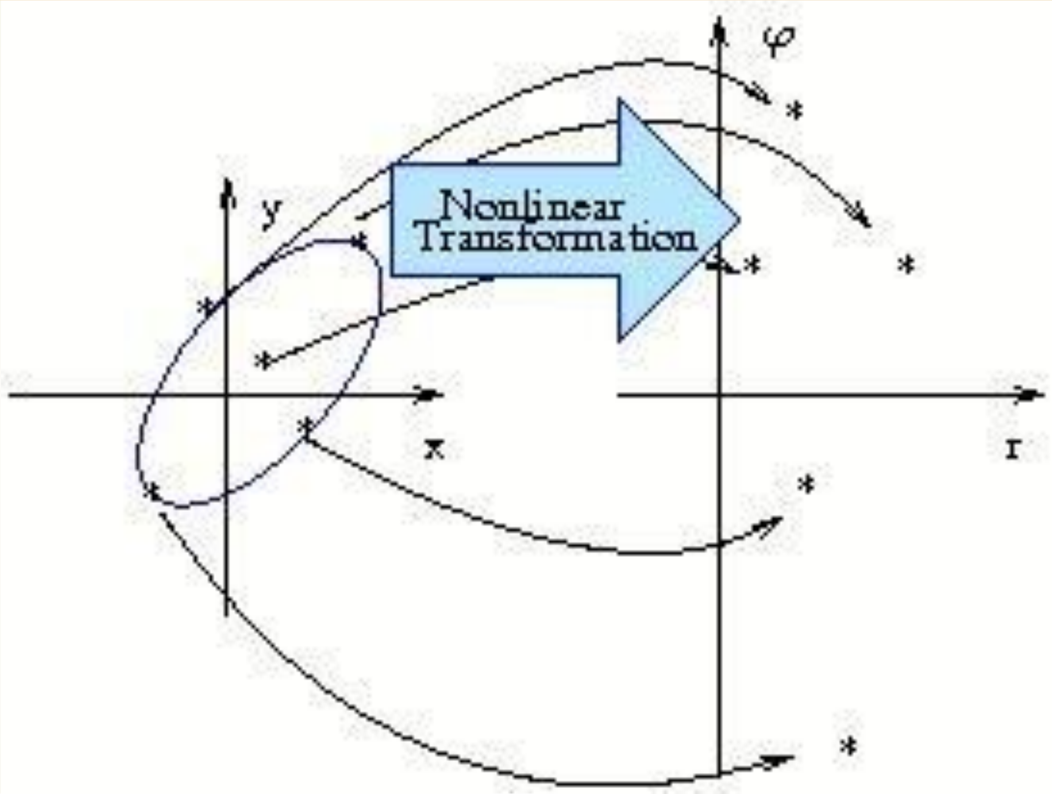
$$\dot{\mathbf{x}}^B = \begin{bmatrix} -\frac{1}{C_b R_p} & \frac{1}{C_{cp} R_p} & \frac{1}{C_s R_p} \\ \frac{1}{C_b R_p} & -\frac{1}{C_{cp} R_p R_{cp}} & \frac{1}{C_s R_p} \\ \frac{1}{C_b R_p} & \frac{1}{C_{cp} R_p} & \frac{1}{C_s R_p} \end{bmatrix} \mathbf{x}^B + \begin{bmatrix} i \\ i \\ i \end{bmatrix} + \xi \quad (6)$$

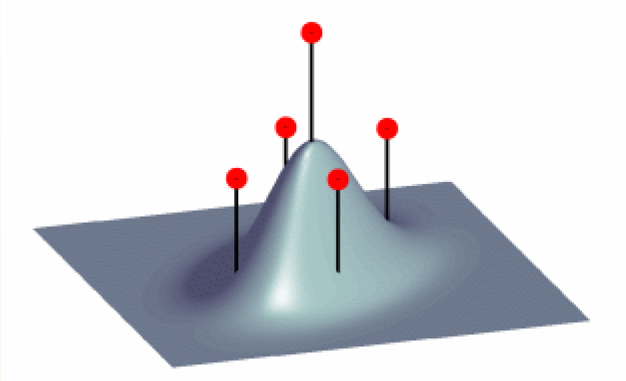






Unscented Kalman Filter





Unscented Kalman Filter

```
%% Selection of Sigma Points %%%%%%

W(1) = 0.99;                    %Sigma point array begins at 1 and not 0

SQRTM_P = real(sqrtm( 2*P/(1-W(1)) ));  %Square root of matrix is converted to real type

Chi(:,1) = xa; % xa is old state estimate feed back from algorithm

 for k = 2:ndim+1          %Generate sigma points with +sqrt. Increase indices by 1 since arrays don't start from 0

        Chi(:,k) = xa + SQRTM_P(:,k-1);

        W(k) = (1-W(1))/(2*2);

         Chi(:,ndim+k) = xa - SQRTM_P(:,k-1);

        W(ndim+k) = (1-W(1))/(2*2);

end
```

$$
\begin{aligned}
\boldsymbol{\mathcal{X}}_0 &= \bar{\mathrm{x}} & W_0 &= \kappa/(n+\kappa) \\
\boldsymbol{\mathcal{X}}_i &= \bar{\mathrm{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_i &= 1/2(n+\kappa) \\
\boldsymbol{\mathcal{X}}_{i+n} &= \bar{\mathrm{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_{i+n} &= 1/2(n+\kappa)
\end{aligned}
$$

Unscented Kalman Filter

```
%% Model Forecast Step %%%%%%%%%%%%%%%%%%
% Propagate Sigma Points through non-linear transform
 for k = 1:2*ndim+1         %Forecast sigma points
          ip = Vp/Rp;
          ib = ip + i;
          Chi_qcsnew = ib*Rs*Cs;    %%Consider finding qcsdot
          ChiF(:,k) = [ Chi(1,k) + qbdot*dt;    %Symbol for sigma pt is chi
            Chi(2,k) + qcpdot*dt;
          Chi_qcsnew];           %%% Not sure about this!!!
end
 %% Compute the mean and covariance of forecast
for k = 1:2*ndim+1         %Calculate mean of sigma points
          ChiF_mean = ChiF_mean + W(k)*ChiF(:,k);
end
for k = 1:2*ndim+1    %Calculate covariance matrix of sigma pts
          PX = PX + W(k)*(ChiF(:,k) - ChiF_mean)*(ChiF(:,k) - ChiF_mean)';
end
 PX = PX+Q;
```

$$\mathcal{Y}_i = \mathrm{f}\left[\mathcal{X}_i\right]$$

$$\mathbf{x}^B = \left[\begin{array}{ccc} q_b & q_{cp} & q_{Cs} \end{array}\right]^T$$

$$\dot{\mathbf{x}}^B$$

$$\bar{\mathrm{y}} = \sum_{i=0}^{2n} W_i \mathcal{Y}_i$$

$$\mathrm{P}_{yy} = \sum_{i=0}^{2n} W_i \left\{\mathcal{Y}_i - \bar{\mathrm{y}}\right\}\left\{\mathcal{Y}_i - \bar{\mathrm{y}}\right\}^T$$

Unscented Kalman Filter – Converted to C code (UKF_bhm3.c)

```
%% Propagate the sigma points through the observation model
ZF = [1/Cb -1/Ccp -1/Param.Cs]*ChiF; % Expected measurement
for k = 1:2*ndim+1    %Calculate mean of expected measurements
          ZF_mean = ZF_mean + W(k)*ZF(:,k);
end
 for k = 1:2*ndim+1    %Covar of expected measurements
          PZ = PZ + W(k)*(ZF(:,k) - ZF_mean)*(ZF(:,k) - ZF_mean)';
end
PZ = PZ+R;
%% Compute the cross covariance between XF and Zf
for k = 1:2*ndim+1    %Cross covariance of sigma points and expected measurements
          PXZ = PXZ + W(k)*(ChiF(:,k) - ChiF_mean)*(ZF(:,k) - ZF_mean)';
end
%% Data Assimilation Step %%%%%%%%%%%%%%%%%%%%%
K = PXZ*PZ^(-1);
xa = ChiF_mean + K*(z_ukf - ZF_mean);
P = PX - K*PZ*K';
est_ukf(:,2) = xa;
ukfout = struct('P',P,'state_ukf',state_ukf,'est_ukf',est_ukf,'SOC',SOC);
```

$$\mathbf{y}^B = V_p = \begin{bmatrix} \frac{1}{C_b} & \frac{1}{C_{cp}} & \frac{1}{C_s} \end{bmatrix} \cdot \mathbf{x}^B \qquad (7)$$

$$\hat{\mathbf{z}}(k+1 \mid k) = \sum_{i=1}^{2n^a} W_i \boldsymbol{\mathcal{Z}}_i (k+1 \mid k)$$

$$\mathbf{P}_{\nu\nu}(k+1 \mid k) = \mathbf{R}(k+1) + \sum_{i=0}^{2n^a} W_i \{\boldsymbol{\mathcal{Z}}_i(k \mid k-1) - \hat{\mathbf{z}}(k+1 \mid k)\}\{\boldsymbol{\mathcal{Z}}_i(k \mid k-1) - \hat{\mathbf{z}}(k+1 \mid k)\}^T$$

$$\mathbf{P}_{xz}(k+1 \mid k) = \sum_{i=0}^{2n^a} W_i \{\boldsymbol{\mathcal{X}}_i(k \mid k-1) - \hat{\mathbf{x}}(k+1 \mid k)\}\{\boldsymbol{\mathcal{Z}}_i(k \mid k-1) - \hat{\mathbf{z}}(k+1 \mid k)\}^T$$

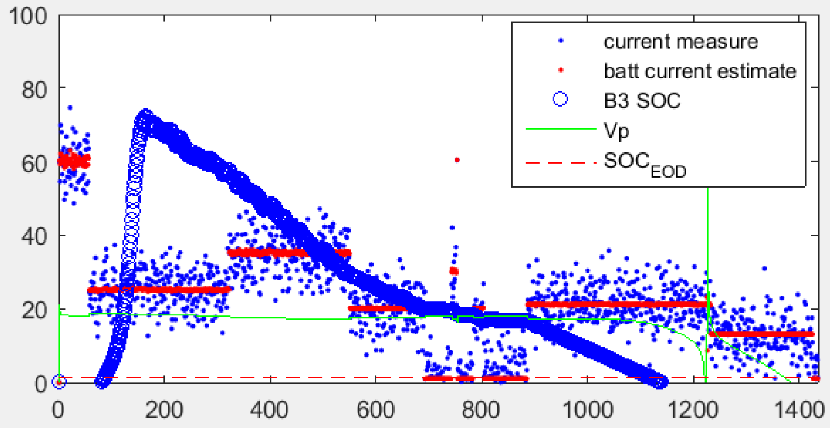
BeagleBone Black Output on SD Card

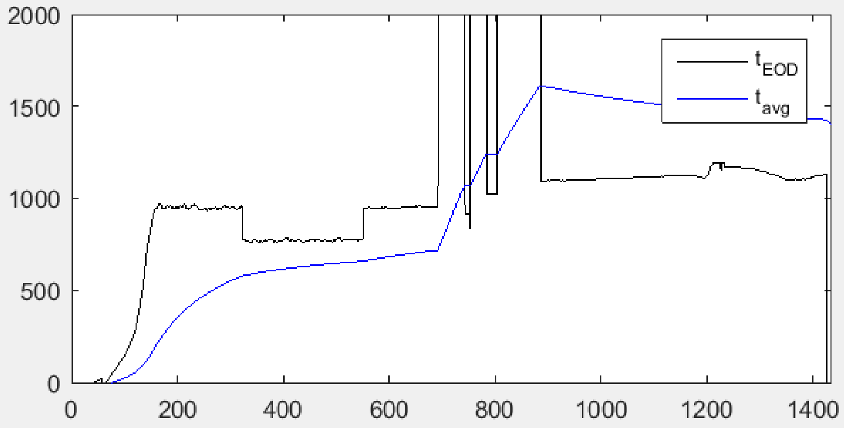
```
time ,        dt,        i,        V,     SOC,          t_EOD
0.384020, 0.384020, 0.002000, 0.200000, 1.000000, -326215.066870
...
34.808678, 0.378297, 59.000000, 19.544800, 0.932649, 3.702595
35.186949, 0.378271, 59.000000, 19.544600, 0.931866, 4.232829
...
112.751668, 0.378590, 59.000000, 19.544901, 0.778342, 398.632246
113.129971, 0.378303, 0.002000, 19.544600, 0.777559, 8396619.992
113.508321, 0.378350, 59.000000, 19.544701, 0.777559, 398.940899
...
191.829817, 0.378352, 59.000000, 19.544701, 0.623250, 416.592375
192.208053, 0.378236, 59.000000, 19.544701, 0.622467, 416.606263
192.586373, 0.378320, 59.000000, 19.545000, 0.621684, 416.620196
192.964917, 0.378
```

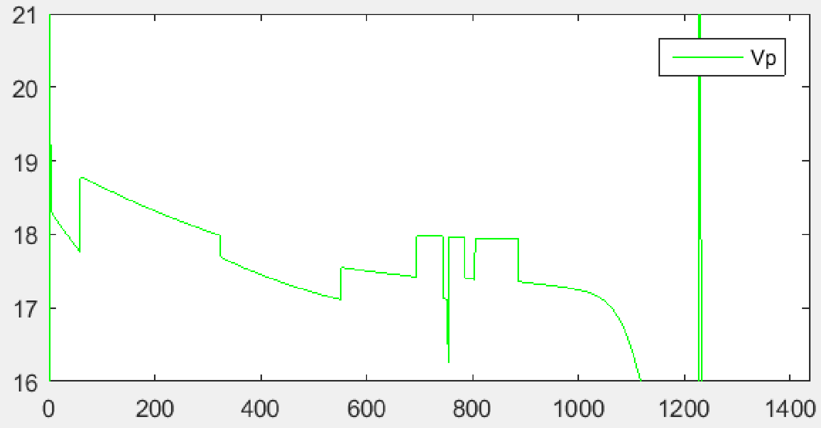
Future Work: UKF and EOD refinement

/UKF_bhm4/UKF_bhm3.m

Not yet implemented in C on BeagleBone Black







Future work: UKF weights tuning

```
%% Model Forecast Step
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Propagate Sigma Points through non-linear transform

...

 for k = 1:2*ndim+1        %Forecast sigma points

            ip = Vp/Rp;

            ib = ip + i;

            Chi_qcsnew = ib*Rs*Cs;    %%Consider finding qcsdot

            ChiF(:,k) = [ Chi(1,k) + qbdot*dt;    %Symbol for sigma pt is chi

              Chi(2,k) + qcpdot*dt;

            Chi_qcsnew];          %%% Not sure about this!!!

end

 %% Compute the mean and covariance of forecast

for k = 1:2*ndim+1        %Calculate mean of sigma points

            ChiF_mean = ChiF_mean + W(k)*ChiF(:,k);

end

for k = 1:2*ndim+1    %Calculate covariance matrix of sigma pts

            PX = PX + W(k)*(ChiF(:,k) - ChiF_mean)*(ChiF(:,k) - ChiF_mean)';

end

 PX = PX+Q;
```

From "Optimal Estimation of Dynamics Systems" 2nd by Crassidis

$$W_0^{\text{mean}} = \frac{\lambda}{L+\lambda} \tag{3.259a}$$

$$W_0^{\text{cov}} = \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) \tag{3.259b}$$

$$W_i^{\text{mean}} = W_i^{\text{cov}} = \frac{1}{2(L+\lambda)}, \quad i = 1, 2, \ldots, 2L \tag{3.259c}$$

Thank you!