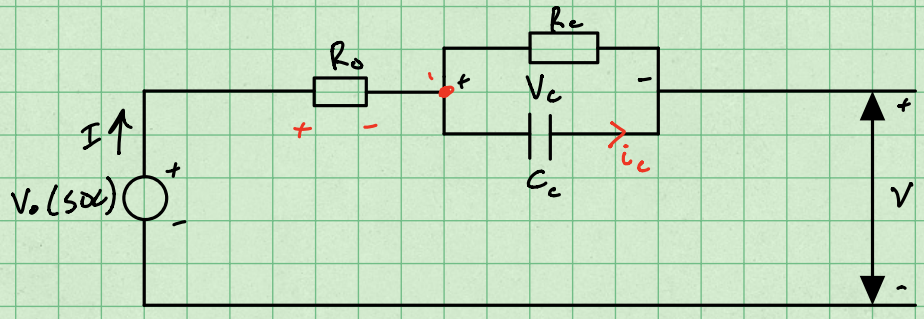
Problem 1)

a)



$$\dot{V}_c = -\frac{V_c}{C_c R_c} + \frac{I}{C_c} \quad (1)$$

$$\dot{SOC} = \frac{-I}{C_{bat}} \quad (2)$$

$$\dot{SOC} = [0]\,SOC + \left[-\tfrac{1}{C_{bat}}\right][I]$$

$$\begin{bmatrix} V \\ \dot{V}_c \end{bmatrix} = [\alpha]\,SOC$$

$$V - V_{oc}^{\circ} = -V_c + \alpha SOC - R_o I \quad (3)$$

$$\downarrow$$

$$\dot{x} = Ax + Bu + W$$

$$y = Cx + Du + V$$

where $A = \begin{bmatrix} -\frac{1}{C_c R_c} & 0 \\ 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} \frac{1}{C_c} \\ -\frac{1}{C_{bat}} \end{bmatrix}$ $C = [-1 \quad \alpha]$

$D = [-R_o]$

$y = V - V_{oc}^{\circ}$ $X = \begin{bmatrix} V_c \\ SOC \end{bmatrix}$ $u = I$

$W \sim N(0, 2.5 \cdot 10^{-7})$ $V \sim N(0, 10^{-4})$

$$\Downarrow$$

$$\begin{bmatrix} \dot{V}_c \\ \dot{SOC} \end{bmatrix} = \begin{bmatrix} -\frac{1}{C_c R_c} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_c \\ SOC \end{bmatrix} + \begin{bmatrix} \frac{1}{C_c} \\ -\frac{1}{C_{bat}} \end{bmatrix} I$$

$$y = V - V_{oc}^{\circ} = [-1 \quad \alpha] \begin{bmatrix} V_c \\ SOC \end{bmatrix} + [-R_o] I$$

discritize: $A = \begin{bmatrix} \exp\left[\frac{-T}{C_c R_c}\right] & 0 \\ 0 & 1 \end{bmatrix}$ $B = \begin{bmatrix} R_c \exp\left[\frac{-T}{C_c R_c}\right] \\ -T/C_{bat} \end{bmatrix}$

$C_c = [-1 \quad \alpha]$ $D_c = [-R_o]$

$$V_{c,k+1} = V_{c,k} \exp\left[\frac{-T}{C_c R_c}\right] + R_c\left(1 - \exp\left[\frac{-T}{C_c R_c}\right]\right) I_k$$

$$SOC_{k+1} = SOC_k + \left(\frac{-T}{C_{bat}}\right) I_k$$

$$y_k = V_k - V_{oc}^0 = -V_{c,k} + \alpha SOC_k - R_0 I_k$$

assume $V_{c,k}$ is predicted perfectly from model:

$$y_k = \underbrace{V_k - V_{oc}^0 + V_{c,k}}_{\text{model predic}} = \underbrace{\alpha SOC_k - R_0 I_k}_{\text{measurement update}}$$

c) $$P_\infty = A P_\infty A^T + Q - A P_\infty C^T (C P_\infty C^T + R)^{-1} C P_\infty A^T$$

let $A = 1 \quad C = \alpha$

$$P_\infty = P_\infty + Q - \alpha P_\infty (\alpha^2 P_\infty + R)^{-1} \alpha P_\infty$$

$$0 = Q - \frac{\alpha^2 P_\infty^2}{\alpha^2 P_\infty + R} \Rightarrow \quad Q = \frac{\alpha^2 P_\infty^2}{\alpha^2 P_\infty + R} \Rightarrow \quad \alpha^2 P_\infty^2 = \alpha^2 Q P_\infty + QR$$

$$0 = \alpha^2 P_\infty - \alpha^2 Q P_\infty - RQ$$

$$P_\infty = \leq 7.818 \cdot 10^{-6} \quad P_\infty > 0$$

$$P_\infty = 7.8 \cdot 10^{-6} \quad \text{(priori)}$$

$$L_\infty = P_\infty C^T (C P_\infty C^T + R)^{-1} = \frac{P_\infty \alpha}{\alpha^2 P_\infty + R} = 0.0491$$

from MATLAB $L_\infty = 0.0492$

$$P_\infty^+ = (1 - L_\infty \alpha)^2 P_\infty^- + L_\infty^2 R = 7.568 \cdot 10^{-6}$$

from MATLAB: $P_\infty^+ = 7.5683 \cdot 10^{-6}$

rounding error accounts for difference

P_∞^+ from MATLAB matches exactly the theoretical P_∞^+

d) The probability distributions match very closely after being properly scaled.

2.) $$SOC_{k+1} = SOC_k + \left(\frac{-T_s}{C_{bat}}\right) I_k + w_k$$

$$V_{c,k+1} = \exp\left[\frac{-T_s}{C_c R_c}\right] + R_c\left(1 - \exp\left[\frac{-T_s}{C_c R_c}\right]\right) I_k$$

$$V_k = V_{oc}(SOC_k) - V_{c,k} - R_o I_k + v_k$$

let $X_k = SOC_k$

$y_k = V_k + V_{c,k}$

$$f_k(x_k, u_k, w_k) = x_k - \frac{T_s}{C_{bat}} u_k + w_k$$

$$h_k(x_k, u_k, v_k) = V_{oc}(x_k) - R_o u_k + v_k$$

$$A'_{k-1} = \left.\frac{\partial f_k}{\partial x_k}\right|_{\hat{x}_{k-1|k-1}} = 1 \qquad E'_{k-1} = \left.\frac{\partial f_k}{\partial w_k}\right|_{\hat{x}_{k-1|k-1}} = 1$$

$$C'_k = \left.\frac{\partial h_k}{\partial x_k}\right|_{\hat{x}_{k|k-1}} = \left.\frac{dV_{oc}(x_k)}{dx_k}\right|_{\hat{x}_{k|k-1}} \qquad F'_k = \left.\frac{\partial h_k}{\partial v_k}\right|_{\hat{x}_{k|k-1}} = 1$$

Model Predict:

$$\hat{x}_{k|k-1} = f_k(\hat{x}_{k-1|k-1}, u_{k-1}, 0)$$

$$P_{k|k-1} = A'_{k-1} P_{k-1|k-1} A'^{T}_{k-1} + E'_{k-1} Q_{k-1} E'^{T}_{k-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - h_k(\hat{x}_{k|k-1}, u_k, 0))$$

$$L_k = P_{k|k-1} C'^{T}_k (C'_k P_{k|k-1} C'^{T}_k + F'_k R_k F'^{T}_k)^{-1}$$

$$P_{k|k} = P_{k|k-1} - L_k C'_k P_{k|k-1}$$

c) The distributions match very closely. This is because then nonlinear model is linearized at every step, preserving the gaussianity.

3) B) The probability distributions do not match.

Table of Contents

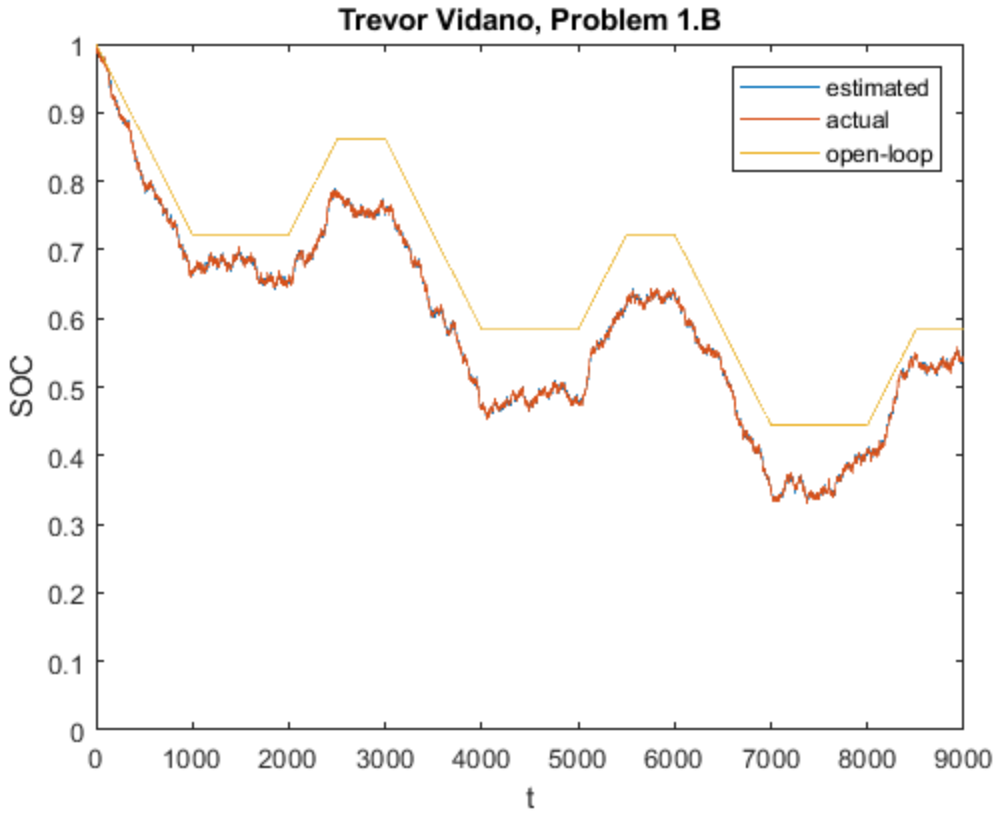
Main MATLAB script for HW 3 in MAE 298: Estimation

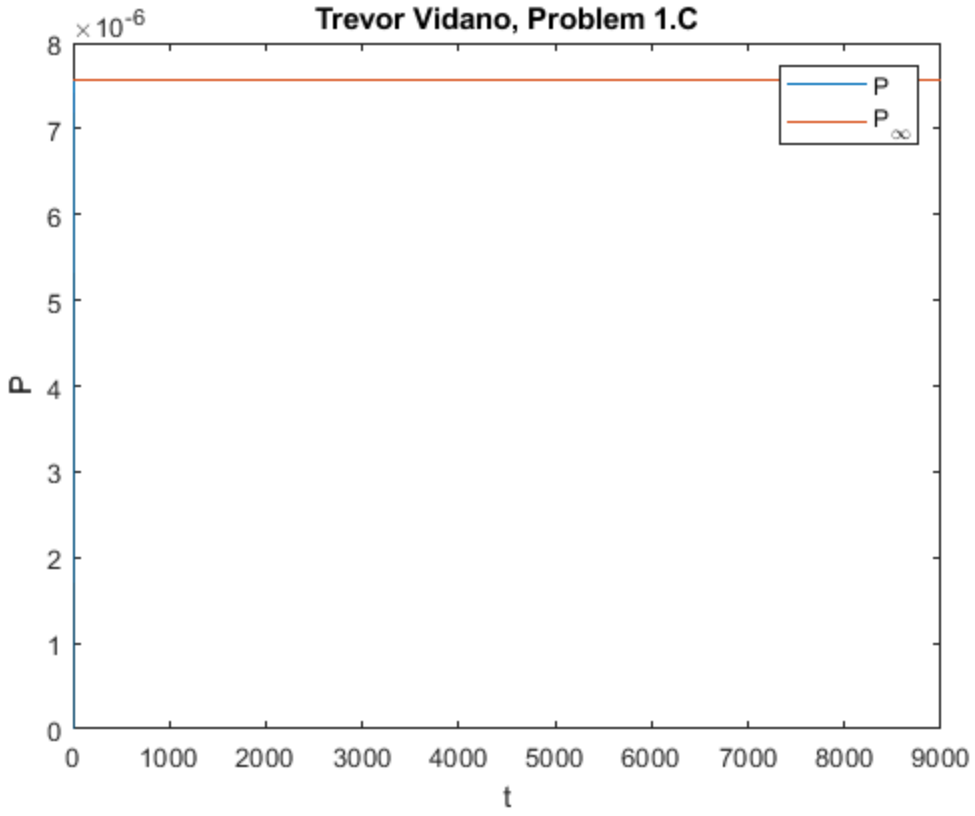
author: Trevor Vidano course: MAE 298: Estimation Spring Quarter 2021 MATLAB implementation of kalman filter and extended kalman filter on battery equivalent circuit model.

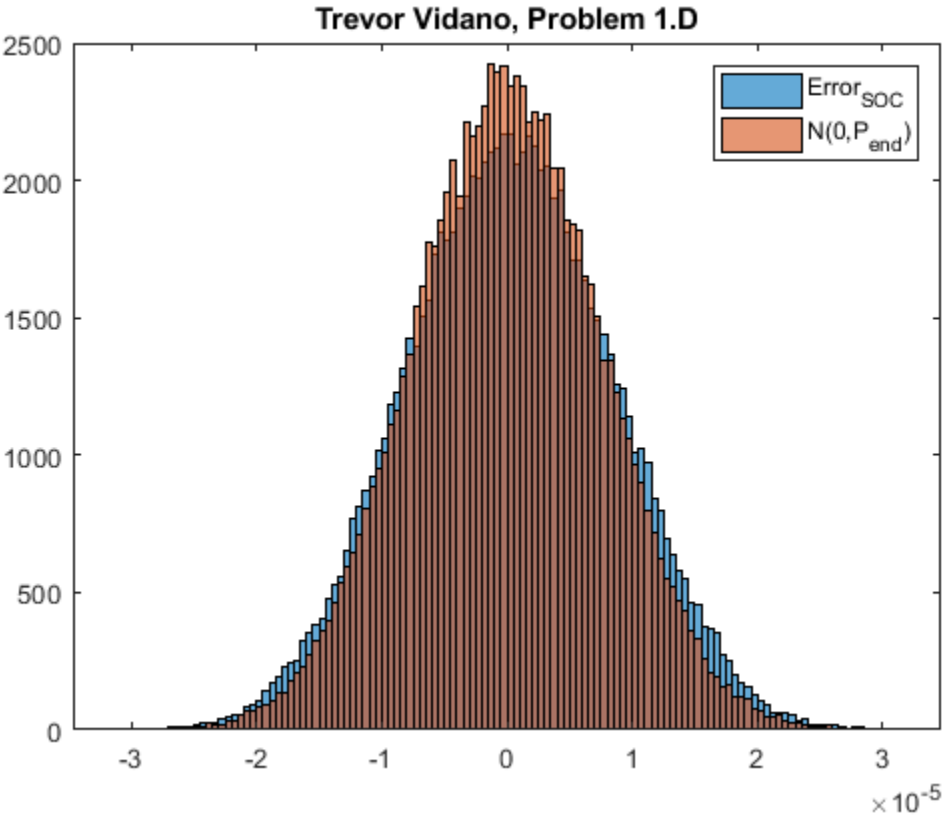
```
clearvars; close all;
% Battery parameters
C_bat= 5*3600;  % A-s
R0 = 0.01; % ohm
Rc = 0.015; % ohm
Cc = 2400; % F
alpha = 0.65; % V
Vocv0 = 3.435; % V

Q = 2.5E-7; % Process noise covariance
R = 1.0E-4; % Sensor noise covariance
Ts = 0.1;  % sampling period

%%%%%%%%%%%%%%%%%%%%%%%% State Space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Continuous time State Space Matrices
A_c = [-1/Cc/Rc,0;0,0];
B_c = [1/Cc;-1/C_bat];
C_c = [-1,alpha];
D_c = -R0;

% Discrete time State Space Matrics
F = expm(A_c*Ts);
G = [Rc - Rc*exp(-Ts/Cc/Rc); -Ts/C_bat];
H = C_c;
M = D_c;

% Helper functions:
%{
kalman_gain = @(P_priork,H_k,R_k) ...
    P_priork*H_k'*inv(H_k*P_priork*H_k' + R_k);
cov_prior = @(F_k0,P_postk0,Q_k0) ;
cov_post = @(H_k,K_k,P_priork,R_k,size) ...
    (eye(size) - K_k*H_k)*P_priork*(eye(size) - K-k*H_k)' +
 K_k*R_k*K_k';
prior_est = @(F_k0,x_postk0,G_k0,u_k0) F_k0*x_postk0 + G_k0*u_k0;
post_est = @(x_priork,K_k,y_k,H_k) x_priork + K_k*(y_k -
 H_k*x_priork);
```

```
%}
%%%%%%%%%%%%%%%%%%%%%%%% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
linearIV = matfile('IV_data_linear.mat');
u = linearIV.I';
t = linearIV.t';
SOC_act = linearIV.SOC_act';
V_measured = linearIV.V';

k_end = t(end)/Ts;

% scalar initilization:
SOC_post = 1;
SOC_open = SOC_post;
Vc = 0;
P_post = 0;
% matrix initialization:
% x_post(:,1) = [0;1];
% x_open(:,1) = x_post(:,1);
% P_post{1} = eye(2)*0;
for k = 2:k_end
%     % Full system estimation:
%     % Model Predict:
%     P_prior{k} = F*P_post{k-1}*F' + [0,0;0,Q];
%     K(:,k) = P_prior{k}*H'*inv(H*P_prior{k}*H' + R);
%     x_prior(:,k) = F*x_post(:,k-1) + G*u(:,k-1);
%     % Measurement Update:
%     y(:,k) = V_measured(k) - Vocv0;
%     x_post(:,k) = x_prior(:,k) + K(:,k)*(y(:,k) - (H*x_prior(:,k) +
 M*u(:,k)));
% %     P_post{k} = (eye(2) - K(:,k)*H)*P_prior{k}*(eye(2) -
 K(:,k)*H)' + ...
% %                 K(:,k)*R*K(:,k)';
%     P_post{k} = P_prior{k} - P_prior{k}*H'*inv(H*P_prior{k}*H' +
 R)*H*P_prior{k};
%     x_open(:,k) = F*x_open(:,k-1) + G*u(k-1);
%     P(k) = P_post{k}(4);

    % Estimation of only SOC:
    % Model Predict:
    P_prior(k) = 1*P_post(k-1)*1' + Q;
    K(k) = P_prior(k)*alpha/(alpha*P_prior(k)*alpha + R);
    SOC_prior(k) = 1*SOC_post(k-1) - Ts/C_bat*u(k-1);
    Vc(k) = Vc(k-1)*exp(-Ts/Cc/Rc) + Rc*(1 - exp(-Ts/Cc/Rc))*u(k-1);
    % Measurement Update:
    y(k) = V_measured(k) - Vocv0 + Vc(k);
    SOC_post(k) = SOC_prior(k) + K(k)*(y(k) - (alpha*SOC_prior(k) -
 R0*u(k)));
    P_post(k) = (1 - K(k)*alpha)^2*P_prior(k) + K(k)^2*R;

    % Open loop estimate:
    SOC_open(k) = SOC_open(k-1) - Ts/C_bat*u(k-1);
end
```

```
% Compare Differences to gaussian process:
gauss = normrnd(0,P_post(end),[1,length(P_post)]);
SOC_errors = SOC_post - SOC_act;
SOC_errors = SOC_errors*(max(gauss)/max(SOC_errors));

%%%%%%%%%%%%%%%%%%%%%%% Generate Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure();
plot(t,SOC_post,t,SOC_act,t,SOC_open); xlabel('t'); ylabel('SOC');
legend('estimated','actual','open-loop');
title('Trevor Vidano, Problem 1.B');
ylim([0,1.0]);
figure();
plot(t,P_post,[t(1),t(end)],[7.568e-6,7.568e-6]); xlabel('t');
 ylabel('P');
legend('P','P_{\infty}');
title('Trevor Vidano, Problem 1.C');
figure();
histogram(SOC_errors); hold on;
histogram(gauss);
legend('Error_{SOC}','N(0,P_{end})');
title('Trevor Vidano, Problem 1.D');
```



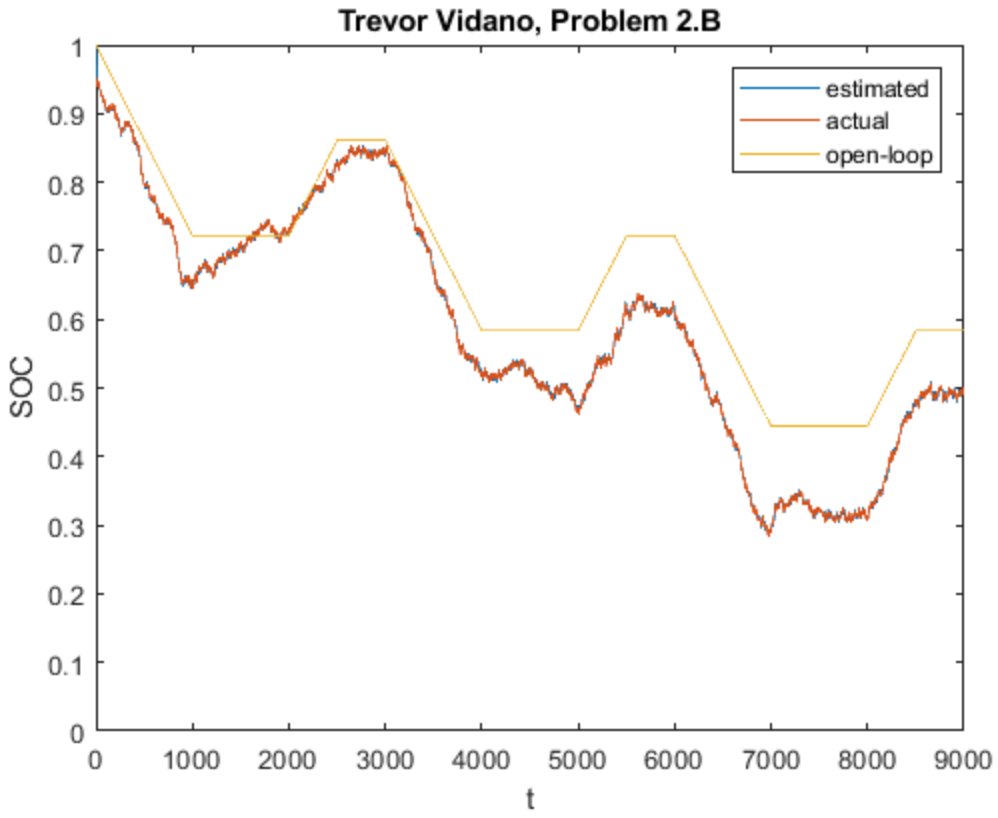

Trevor Vidano, Problem 1.C
×10^-6
8
7
6
5
4
3
2
1
0
P
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
t
P
P∞

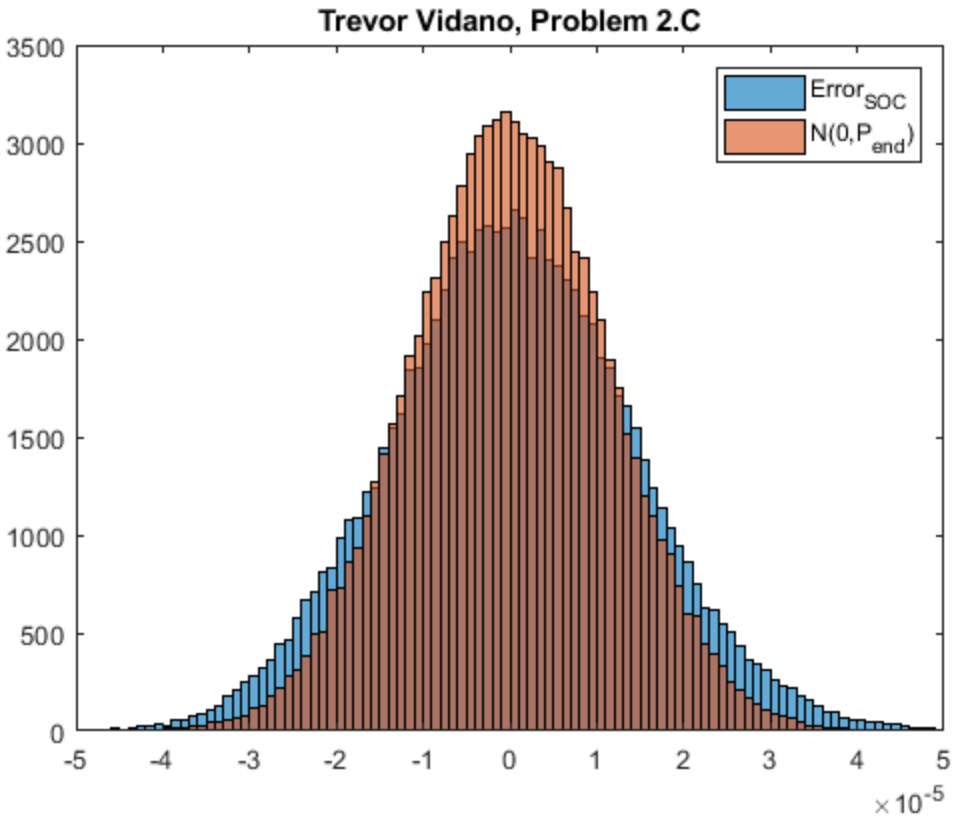

Trevor Vidano, Problem 1.D
2500
2000
1500
1000
500
0
-3
-2
-1
0
1
2
3
×10^-5
Error_SOC
N(0,P_end)

Problem 2: EKF

```
clearvars;
C_bat= 5*3600;  % A-s
R0 = 0.01; % ohm
Rc = 0.015; % ohm
Cc = 2400; % F
alpha = 0.65; % V
Vocv0 = 3.435; % V

nonlinIV = matfile('IV_data_nonlinear.mat');
u_non = nonlinIV.I';
t_non = nonlinIV.t';
SOC_trueNon = nonlinIV.SOC_act';
V_measNon = nonlinIV.V';

VocTable = matfile('OCV_table');
soc_intpts_OCV = VocTable.soc_intpts_OCV;
OCV_intpts = VocTable.OCV_intpts;
dVocTable = matfile('OCV_slope_table');
soc_intpts_OCV_slope = dVocTable.soc_intpts_OCV_slope;
OCV_slope_intpts = dVocTable.OCV_slope_intpts;

Q = 2.5E-7; % Process noise covariance
R = 1.0E-4; % Sensor noise covariance
Ts = 0.1;  % sampling period
k_end = t_non(end)/Ts;
x_post = 1.0;
P_ekf_post = 0.0;
Vc_non = 0.0;
x_open = x_post;
for k = 2:k_end
    % Model Prediction:
    x_prior(k) = x_post(k-1) - Ts/C_bat*u_non(k-1);
    P_ekf_prior(k) = P_ekf_post(k-1) + Q;
    Vc_non(k) = Vc_non(k-1)*exp(-Ts/Cc/Rc) + Rc*(1 - exp(-Ts/Cc/
Rc))*u_non(k-1);

    x_open(k) = x_open(k-1) - Ts/C_bat*u_non(k-1);

    % Measurement Update:
    C_prime(k) =
 interp1(soc_intpts_OCV_slope,OCV_slope_intpts,x_prior(k));
    L(k) = P_ekf_prior(k)*C_prime(k)/(C_prime(k)^2*P_ekf_prior(k) +
 R);
    y_non(k) = V_measNon(k) + Vc_non(k);
    Voc(k) = interp1(soc_intpts_OCV,OCV_intpts,x_prior(k));
    x_post(k) = x_prior(k) + L(k)*(y_non(k) - (Voc(k) -
 R0*u_non(k-1)));
    P_ekf_post(k) = P_ekf_prior(k) - L(k)*C_prime(k)'*P_ekf_prior(k);
end
% Compare Differences to gaussian process:
gauss_non = normrnd(0,P_ekf_post(end),[1,length(P_ekf_post)]);
```

```
x_errors = x_post - SOC_trueNon;
x_errors = x_errors*(min(gauss_non)/min(x_errors));


%%%%%%%%%%%%%%%%%%%%%%% Generate Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
figure();
plot(t_non,x_post,t_non,SOC_trueNon,t_non,x_open); xlabel('t');
 ylabel('SOC');
legend('estimated','actual','open-loop');
title('Trevor Vidano, Problem 2.B');
ylim([0,1.0]);
figure();
histogram(x_errors); hold on;
histogram(gauss_non); hold off;
xlim([-.5e-4,.5e-4]);
legend('Error_{SOC}','N(0,P_{end})');
title('Trevor Vidano, Problem 2.C');
```

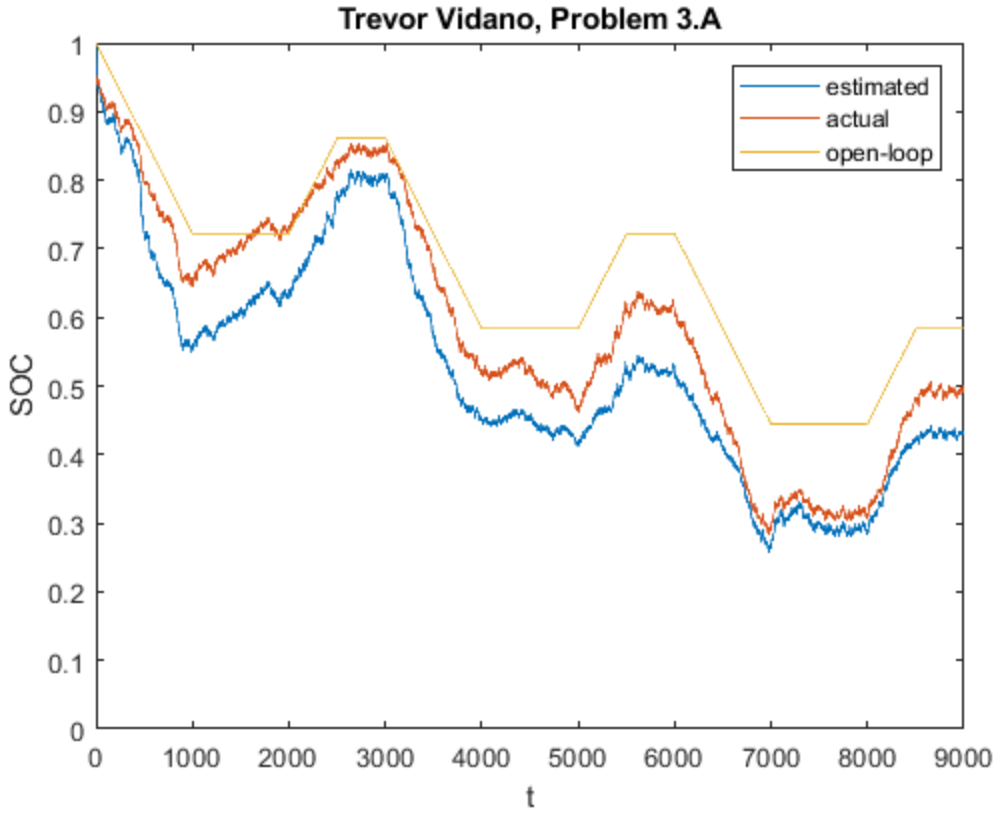


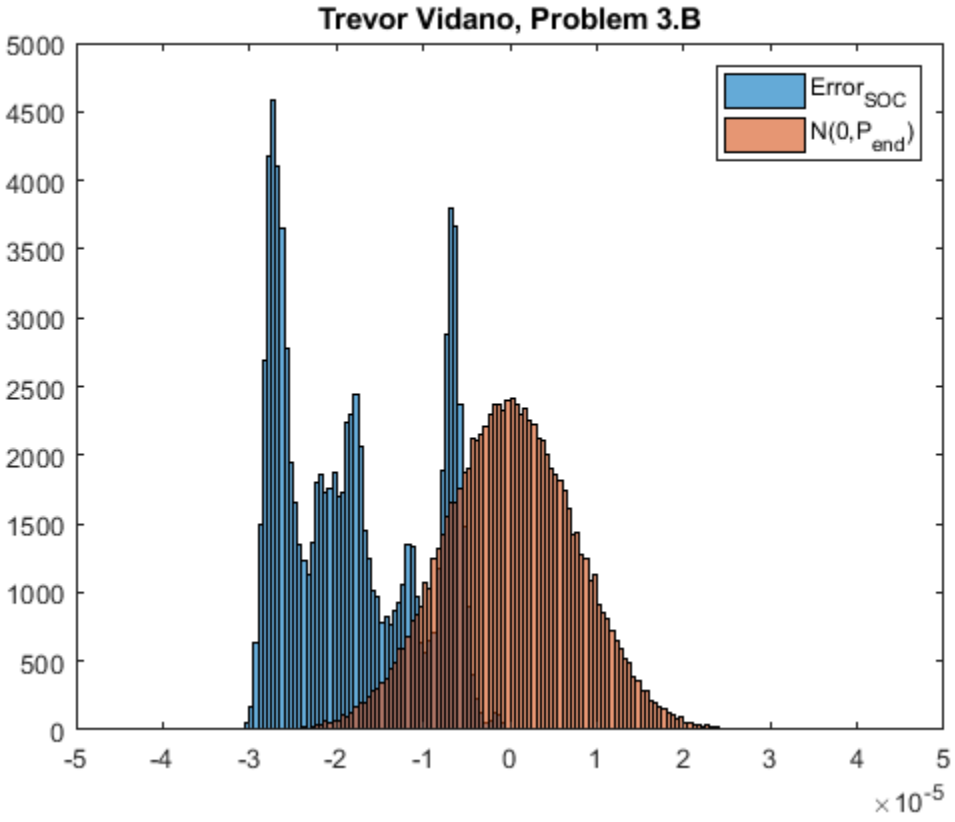


Problem 3: EKF

```
C_bat= 5*3600;  % A-s
R0 = 0.01; % ohm
Rc = 0.015; % ohm
Cc = 2400; % F
alpha = 0.65; % V
Vocv0 = 3.435; % V

nonlinIV = matfile('IV_data_nonlinear.mat');
u_non = nonlinIV.I';
t_non = nonlinIV.t';
SOC_trueNon = nonlinIV.SOC_act';
V_measNon = nonlinIV.V';

VocTable = matfile('OCV_table');
soc_intpts_OCV = VocTable.soc_intpts_OCV;
OCV_intpts = VocTable.OCV_intpts;
dVocTable = matfile('OCV_slope_table');
soc_intpts_OCV_slope = dVocTable.soc_intpts_OCV_slope;
OCV_slope_intpts = dVocTable.OCV_slope_intpts;

Q = 2.5E-7; % Process noise covariance
R = 1.0E-4; % Sensor noise covariance
Ts = 0.1;  % sampling period
```

```
k_end = t_non(end)/Ts;
x_post = 1.0;
P_ekf_post = 0.0;
Vc_non = 0.0;
x_open = x_post;
for k = 2:k_end
    % Model Prediction:
    x_prior(k) = x_post(k-1) - Ts/C_bat*u_non(k-1);
    P_ekf_prior(k) = P_ekf_post(k-1) + Q;
    Vc_non(k) = Vc_non(k-1)*exp(-Ts/Cc/Rc) + Rc*(1 - exp(-Ts/Cc/
Rc))*u_non(k-1);

    x_open(k) = x_open(k-1) - Ts/C_bat*u_non(k-1);

    % Measurement Update:
    L(k) = P_ekf_prior(k)*alpha/(alpha^2*P_ekf_prior(k) + R);
    y_non(k) = V_measNon(k) - Vocv0 + Vc_non(k);
    Voc(k) = interp1(soc_intpts_OCV,OCV_intpts,x_prior(k));
    x_post(k) = x_prior(k) + L(k)*(y_non(k) - (alpha*x_prior(k) -
 R0*u_non(k-1)));
    P_ekf_post(k) = (1 - L(k)*alpha)^2*P_ekf_prior(k) + L(k)^2*R;
end
% Compare Differences to gaussian process:
gauss_non = normrnd(0,P_ekf_post(end),[1,length(P_ekf_post)]);
x_errors = x_post - SOC_trueNon;
x_errors = x_errors*(min(gauss_non)/min(x_errors));

%%%%%%%%%%%%%%%%%%%%%% Generate Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
figure();
plot(t_non,x_post,t_non,SOC_trueNon,t_non,x_open); xlabel('t');
 ylabel('SOC');
legend('estimated','actual','open-loop');
title('Trevor Vidano, Problem 3.A');
ylim([0,1.0]);
figure();
histogram(x_errors); hold on;
histogram(gauss_non); hold off;
xlim([-.5e-4,.5e-4]);
legend('Error_{SOC}','N(0,P_{end})');
title('Trevor Vidano, Problem 3.B');
```


Trevor Vidano, Problem 3.A
estimated
actual
open-loop
SOC
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000


Trevor Vidano, Problem 3.B
Error SOC
N(0,P end)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
-5
-4
-3
-2
-1
0
1
2
3
4
5
×10-5

Published with MATLAB® R2020b