1.a) Plot the *P* trace errors

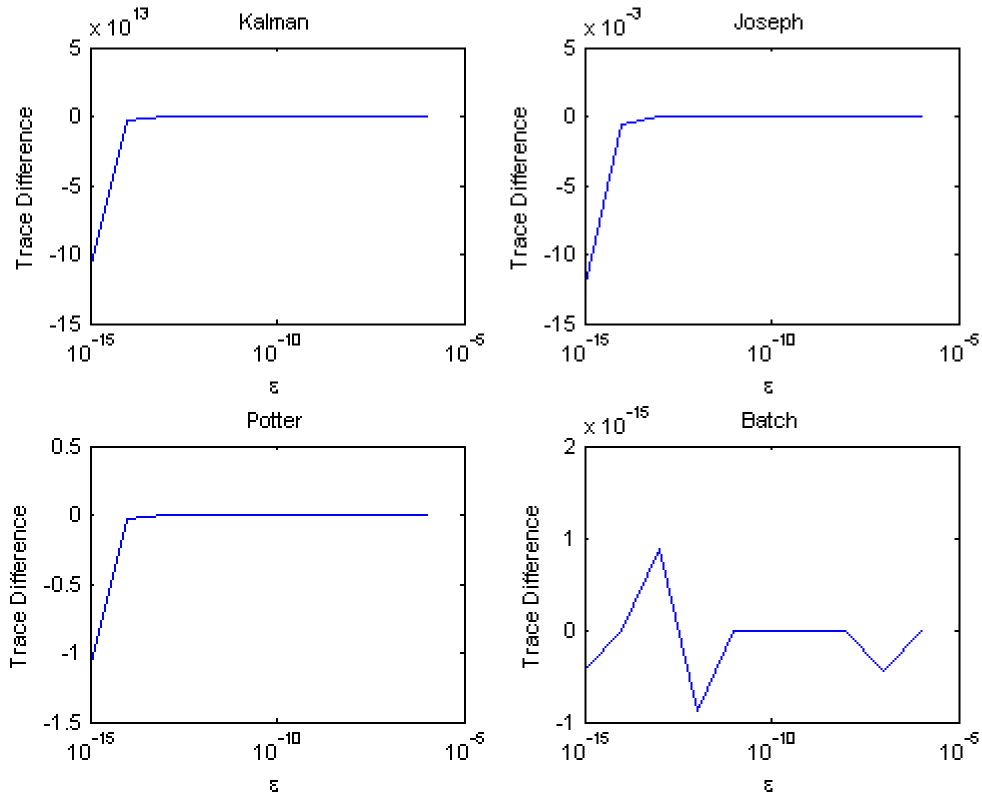


Illustration 1: semilogx() Plots of P2 Trace Error

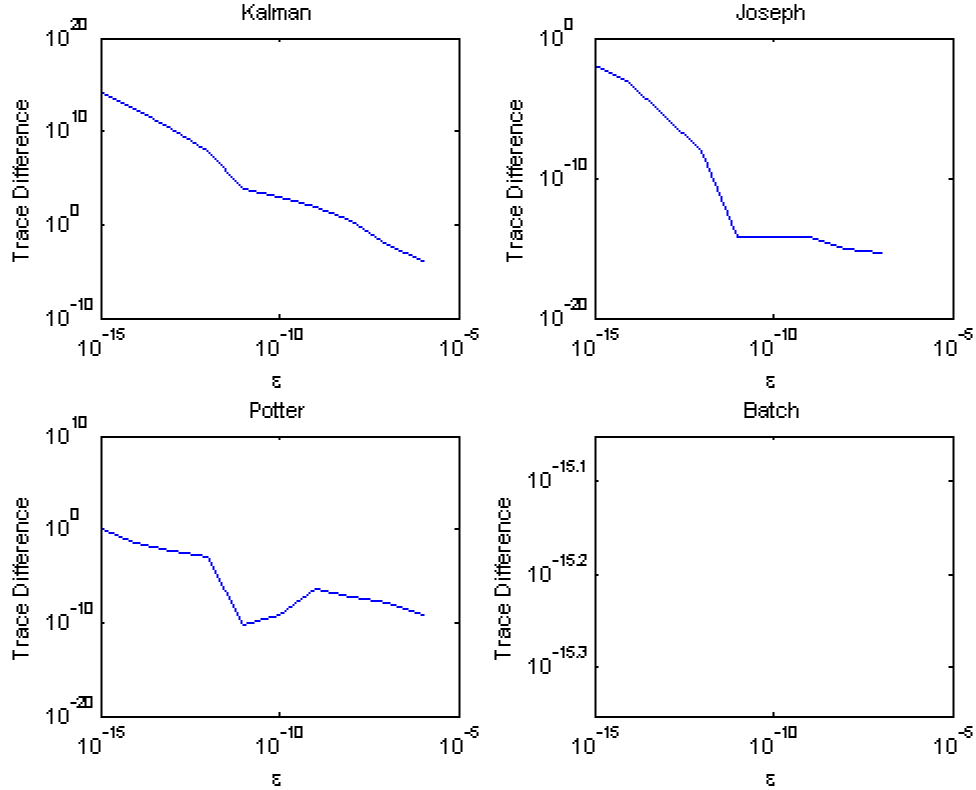


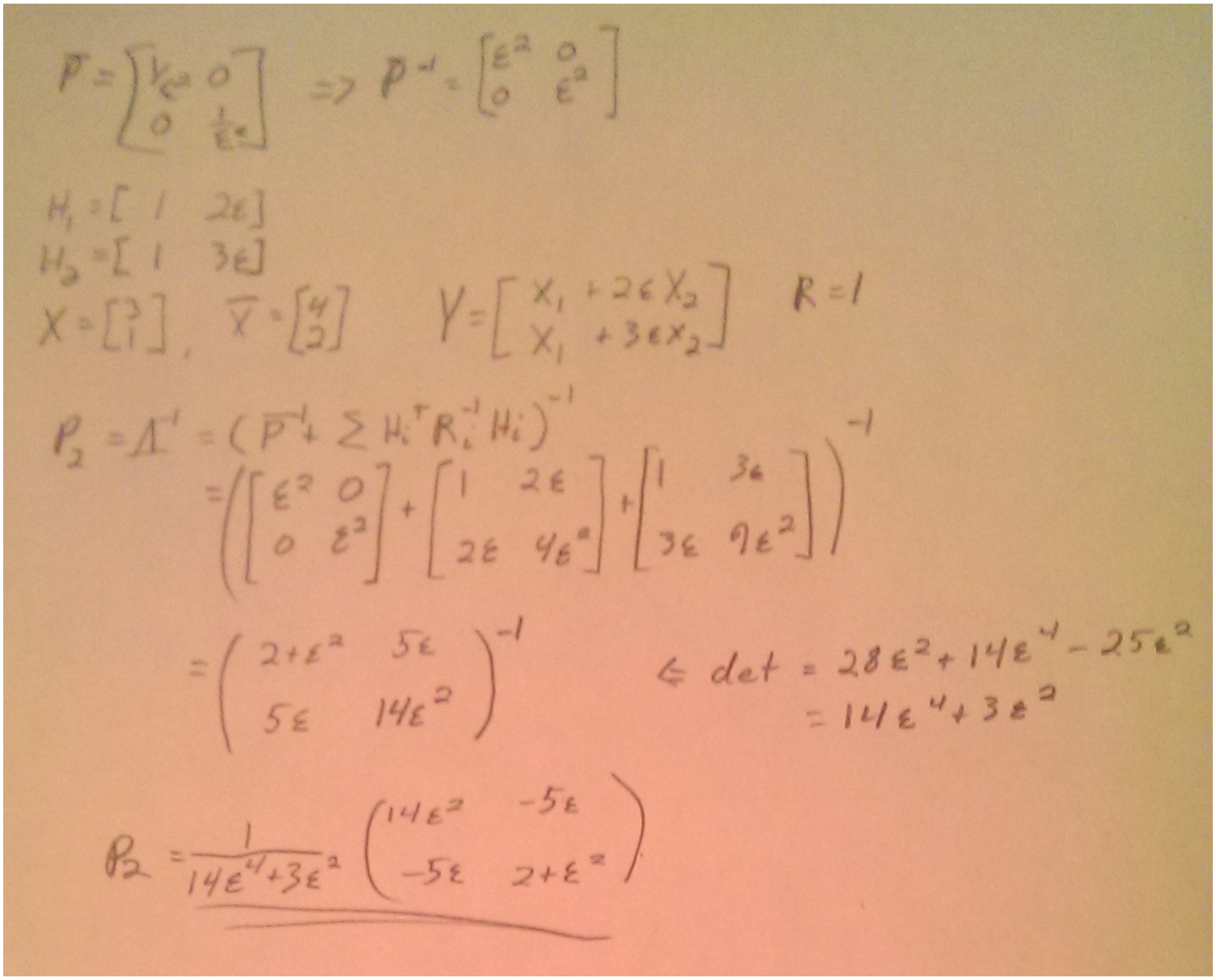
Illustration 2: loglog() Plots of P2 Trace Error

b) Compare the behavior of the filters

As epsilon gets smaller:

- Kalman gets less accurate (it's the least accurate of the four). The covariance matrix isn't guaranteed to be positive definite (or symmetric) due to the iterative solution to obtain it.
- Joseph is the most accurate of the sequential processors, for longer. It ensures that the covariance matrix is positive definite and symmetric.
- Potter is still better than Kalman because it's a square-root algorithm; it computes and carries the better conditioned *W* instead of *P*. It does not force *P* to be positive definite or symmetric, so it is generally less accurate, although more machine precision might see this algorithm fairing better than Joseph, according to the trend.
- Batch is pretty much exact, given the precision of my computer. At least every-other one has an error of zero, leading me to believe that any deviation is due to rounding. This *P* is calculated in one step, so there aren't in-between calculations that accumulate errors. The zero-error causes the lines not to be plotted in the loglog() for Batch.

2.a) Derive the true covariance of the given system and measurement model in terms of epsilon.



$$\bar{P} = \begin{bmatrix} 1/\varepsilon^2 & 0 \\ 0 & \frac{1}{\varepsilon^2} \end{bmatrix} \Rightarrow \bar{P}^{-1} = \begin{bmatrix} \varepsilon^2 & 0 \\ 0 & \varepsilon^2 \end{bmatrix}$$

$$H_1 = [\,1 \quad 2\varepsilon\,]$$
$$H_2 = [\,1 \quad 3\varepsilon\,]$$
$$X = \begin{bmatrix} ? \\ 1 \end{bmatrix}, \quad \bar{X} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \qquad Y = \begin{bmatrix} X_1 + 2\varepsilon X_2 \\ X_1 + 3\varepsilon X_2 \end{bmatrix} \qquad R = 1$$

$$P_2 = \Lambda^{-1} = \left(\bar{P}^{-1} + \sum H_i^T R_i^{-1} H_i\right)^{-1}$$
$$= \left(\begin{bmatrix} \varepsilon^2 & 0 \\ 0 & \varepsilon^2 \end{bmatrix} + \begin{bmatrix} 1 & 2\varepsilon \\ 2\varepsilon & 4\varepsilon^2 \end{bmatrix} + \begin{bmatrix} 1 & 3\varepsilon \\ 3\varepsilon & 9\varepsilon^2 \end{bmatrix}\right)^{-1}$$

$$= \begin{pmatrix} 2+\varepsilon^2 & 5\varepsilon \\ 5\varepsilon & 14\varepsilon^2 \end{pmatrix}^{-1} \quad \Leftarrow \det = 28\varepsilon^2 + 14\varepsilon^4 - 25\varepsilon^2 = 14\varepsilon^4 + 3\varepsilon^2$$

$$P_2 = \frac{1}{14\varepsilon^4 + 3\varepsilon^2} \begin{pmatrix} 14\varepsilon^2 & -5\varepsilon \\ -5\varepsilon & 2+\varepsilon^2 \end{pmatrix}$$

Illustration 3: Derivation of True P

b) Plots of the state errors (computed - exact):

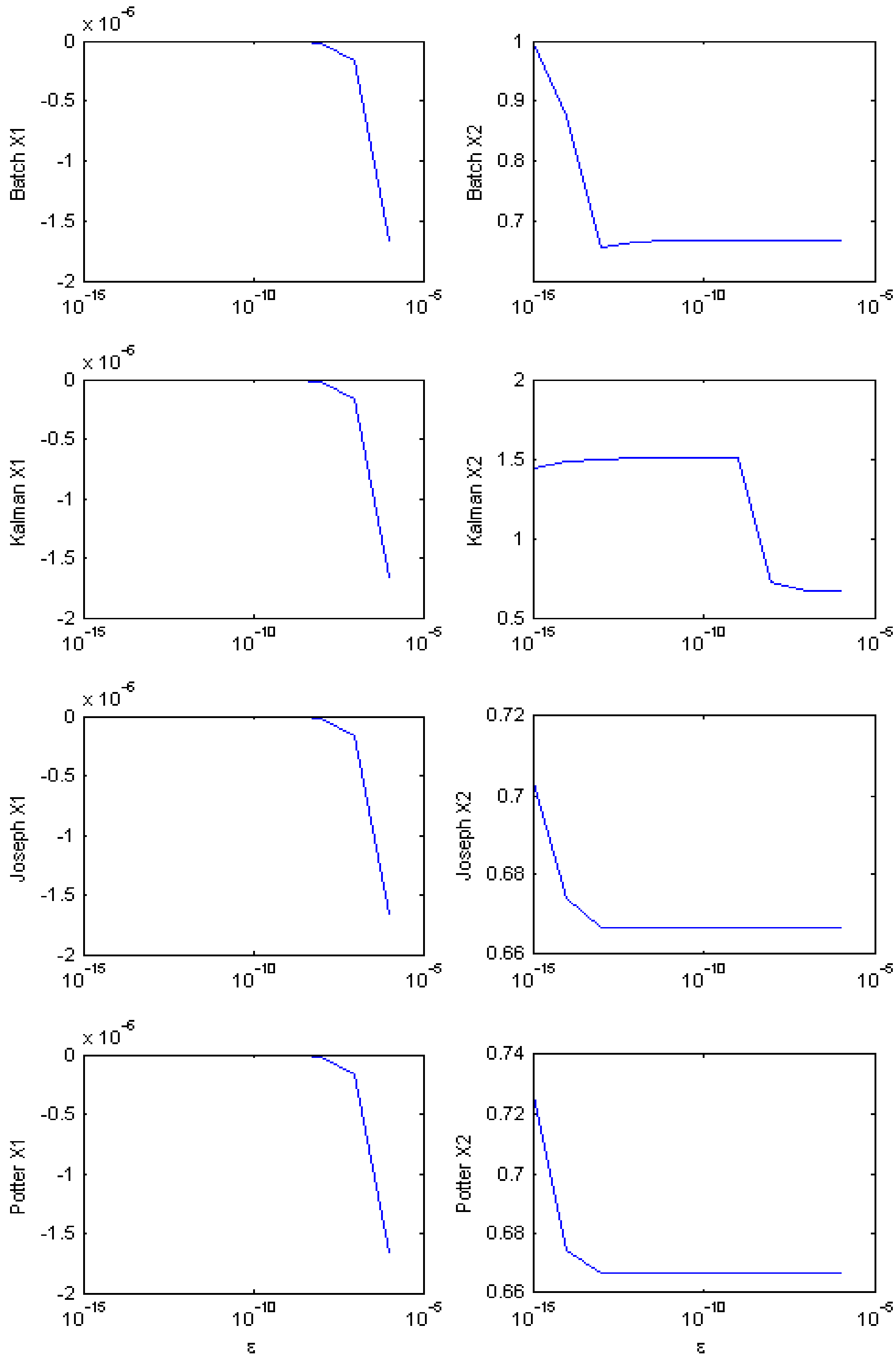


Illustration 4: semilogx() of State Variables Error

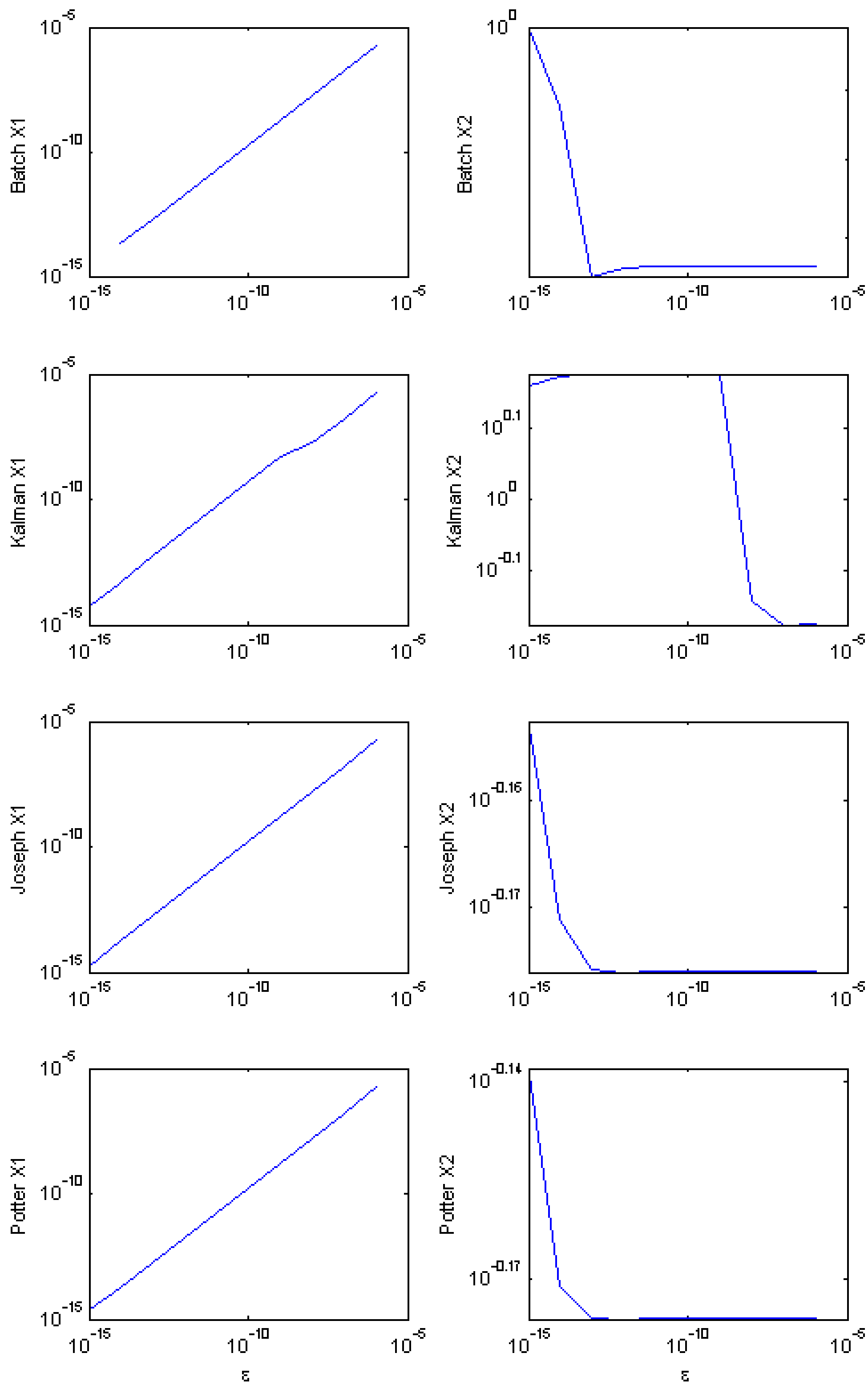


Illustration 5: loglog() of State Variables Error

c) Compare and contrast the performance of the filters

For X1, all the filters performed very closely. As epsilon got smaller, the solution became more accurate. This is because X1's variance is very small (much more so than X2's, which is why the errors are so much smaller). However, all of the X2s increased in error with smaller epsilon, due to the measurement model being less and less able to “utilize” the X2 term with finite machine precision. For X2, Potter and Joseph methods had the least error overall, due to their well-calculated *P*. Kalman continued to hold the same magnitude of error for all values of epsilon. Batch held a similar value to Joseph and Potter until epsilon=10-13, where its error drastically increased. This would be due to a very small information matrix causing numerical errors when taking the inverse.

HW8 Problem 1

Table of Contents

John Clouse

Initialize

```
close all
logx_fig = figure;
loglog_fig = figure;
% Epsilon values
len = 10;
eps = ones(len,1)*1e-5; % start at 1e-6 after the loop iterates
for ii = 1:len
    eps(ii:end) = eps(ii:end)/10;
end
std_dev = 1./eps;
R = 1;


Beta = @(e) 1-2*e+2*e*e*(2+e*e);
P2_exact_trace = zeros(len,1);
for ii = 1:len
    e = eps(ii);
    P2_exact_trace(ii) = trace([1+2*e*e, -(1+e);-(1+e), 2+e*e]/Beta(e));
end
clear e
```

CKF P2

```
P2_ckf_trace = zeros(len,1);
for ii = 1:len
    H1 = [1 eps(ii)];
    P1_ap = eye(2)*std_dev(ii)*std_dev(ii);
    K1 = P1_ap*H1'*inv(H1*P1_ap*H1' + R);
    P1 = (eye(2) - K1*H1)*P1_ap;

    %P1 is now P2_ap
    P2_ap = P1;
    H2 = [1 1];
    K2 = P2_ap*H2'*inv(H2*P2_ap*H2' + R);
    P2 = (eye(2) - K2*H2)*P2_ap;
    P2_ckf_trace(ii) = trace(P2);
```

```
end

diff_ckf = P2_exact_trace - P2_ckf_trace;
HW8_plot(eps, diff_ckf, 1, 'Kalman', logx_fig, loglog_fig);
```

Joseph P2

```
I = eye(2);
P2_joseph_trace = zeros(len,1);
for ii = 1:len
    H = [1 eps(ii); 1 1];
    P_ap = I*std_dev(ii)*std_dev(ii);
    for jj = 1:2
        K = P_ap*H(jj,:)'*inv(H(jj,:)*P_ap*H(jj,:)' + R);
        P = (I-K*H(jj,:))*P_ap*(I-K*H(jj,:))' + K*R*K';
        P_ap = P; % a priori for next measurement
    end
    P2_joseph_trace(ii) = trace(P);
end

diff_joseph = P2_exact_trace - P2_joseph_trace;
HW8_plot(eps, diff_joseph, 2, 'Joseph', logx_fig, loglog_fig);
```

Potter P2

```
P2_potter_trace = zeros(len,1);
for ii = 1:len
    H = [1 eps(ii); 1 1];
    P_ap = I*std_dev(ii)*std_dev(ii);
    W_bar = sqrt(P_ap);
    for jj = 1:2
        F = W_bar*H(jj,:)';
        alpha = inv(F'*F + R);
        gamma = 1/(1+sqrt(R*alpha));
        K = alpha*W_bar*F;
        W = W_bar-gamma*K*F';
        W_bar = W; % sequential update
    end
    P2_potter = W*W';
    P2_potter_trace(ii) = trace(P2_potter);
end

diff_potter = P2_exact_trace - P2_potter_trace;
HW8_plot(eps, diff_potter, 3, 'Potter', logx_fig, loglog_fig);
```

Batch P2

```
P2_batch_trace = zeros(len,1);
for ii = 1:len
    H = [1 eps(ii); 1 1];
    info_mat = inv(I*std_dev(ii)*std_dev(ii));
    info_mat = info_mat + H(1,:)'*inv(R)*H(1,:);
```

```
    info_mat = info_mat + H(2,:)'*inv(R)*H(2,:);
    P2_batch = inv(info_mat);
    P2_batch_trace(ii) = trace(P2_batch);
end

diff_batch = P2_exact_trace - P2_batch_trace;
HW8_plot(eps, diff_batch, 4, 'Batch', logx_fig, loglog_fig);
```

Published with MATLAB® R2013b

HW8 Problem 2

Table of Contents

John Clouse

Initialize

```
clear
close all
logx_fig = figure;
loglog_fig = figure;
% Epsilon values
len = 10;
eps = ones(len,1)*1e-5; % start at 1e-6 after the loop iterates
for ii = 1:len
    eps(ii:end) = eps(ii:end)/10;
end
std_dev = 1./eps;
R = 1;

X_exact = [3;1];
X_ap0 = [4;2]; % a priori state
```

CKF State

```
X_ckf = zeros(2,len);
diff_ckf = zeros(2,len);
for ii = 1:len
    H1 = [1 2*eps(ii)];
    P1_ap = eye(2)*std_dev(ii)*std_dev(ii);
    K1 = P1_ap*H1'*inv(H1*P1_ap*H1' + R);
    X_est1 = X_ap0 + K1*(H1*X_exact - H1*X_ap0);
    P1 = (eye(2) - K1*H1)*P1_ap;
    X_ap1 = X_est1;

    %P1 is now P2_ap
    P2_ap = P1;
    H2 = [1 3*eps(ii)];
    K2 = P2_ap*H2'*inv(H2*P2_ap*H2' + R);
    X_est2 = X_ap1 + K2*(H2*X_exact - H2*X_ap1);
    P2 = (eye(2) - K2*H2)*P2_ap;
    X_ckf(:,ii) = X_est2;
```

```
    diff_ckf(:,ii) = X_exact - X_ckf(:,ii);
end

row = 2;
HW8_P2_plot(eps, -diff_ckf, row, 'Kalman', logx_fig, loglog_fig)
```

Joseph State

```
I = eye(2);
X_joseph = zeros(2,len);
diff_joseph = zeros(2,len);
for ii = 1:len
    H = [1 2*eps(ii); 1 3*eps(ii)];
    P_ap = I*std_dev(ii)*std_dev(ii);
    X_ap = X_ap0;
    for jj = 1:2
        K = P_ap*H(jj,:)'*inv(H(jj,:)*P_ap*H(jj,:)' + R);
        P = (I-K*H(jj,:))*P_ap*(I-K*H(jj,:))' + K*R*K';
        X_est = X_ap + K*(H(jj,:)*X_exact - H(jj,:)*X_ap);
        P_ap = P; % a priori for next measurement
        X_ap = X_est;
    end
    X_joseph(:,ii) = X_est;
    diff_joseph(:,ii) = X_exact - X_joseph(:,ii);
end
row = 3;
HW8_P2_plot(eps, -diff_joseph, row, 'Joseph', logx_fig, loglog_fig)
```

Potter State

```
X_potter = zeros(2,len);
diff_potter = zeros(2,len);
for ii = 1:len
    H = [1 2*eps(ii); 1 3*eps(ii)];
    P_ap = I*std_dev(ii)*std_dev(ii);
    X_ap = X_ap0;
    W_bar = sqrt(P_ap);
    for jj = 1:2
        F = W_bar*H(jj,:)';
        alpha = inv(F'*F + R);
        gamma = 1/(1+sqrt(R*alpha));
        K = alpha*W_bar*F;
        W = W_bar-gamma*K*F';
        X_est = X_ap + K*(H(jj,:)*X_exact - H(jj,:)*X_ap);
        W_bar = W; % sequential update
        X_ap = X_est;
    end
    X_potter(:,ii) = X_est;
    diff_potter(:,ii) = X_exact - X_potter(:,ii);
end
row = 4;
HW8_P2_plot(eps, -diff_potter, row, 'Potter', logx_fig, loglog_fig)
```

Batch State

```
X_batch = zeros(2,len);
diff_batch = zeros(2,len);
for ii = 1:len
    H = [1 2*eps(ii); 1 3*eps(ii)];
    info_mat = inv(I*std_dev(ii)*std_dev(ii));
    info_mat = info_mat + H(1,:)'*inv(R)*H(1,:);
    info_mat = info_mat + H(2,:)'*inv(R)*H(2,:);

    N = inv(I*std_dev(ii)*std_dev(ii))*X_ap0;
    N = N + H(1,:)'*inv(R)*H(1,:)*X_exact;
    N = N + H(2,:)'*inv(R)*H(2,:)*X_exact;
    X_est = inv(info_mat)*N;
    X_batch(:,ii) = X_est;
    diff_batch(:,ii) = X_exact - X_batch(:,ii);
end
row = 1;
HW8_P2_plot(eps, -diff_batch, row, 'Batch', logx_fig, loglog_fig)
```

Published with MATLAB® R2013b