ASEN 5070-Stastistical Orbit Determination-HW 11

Zach Dischner

12-2-2012

1 Homework 11 - Sequential Filter Examination

1.1 Problem 1 - $\Phi(18340, 0)$

I performed the Kalman (sequential) filter on the given observation data, and provide the final state transition matrix after 1 pass.

```
K>> Phi(:,:,end)
ans =
  Columns 1 through 8
     -0.61643       -11.09      -10.322      -4174.4      -9188.8        10552   1.8282e-07  -5.5451e+07
      -2.8515      -18.654      -18.303      -7704.6       -16119        18730    3.241e-07  -5.8774e+07
       5.0177       34.499       33.101        13508        28583       -32615  -5.6852e-07   1.0978e+08
    0.0007999    0.0068075    0.0063366       3.6194       5.5988      -6.4226  -1.1174e-10         9732
    0.0052584     0.035949     0.033613       14.037        30.68      -34.034  -5.9283e-10   1.2195e+05
    0.0033362     0.022887     0.021063       8.8626       18.771      -20.505  -3.7482e-10        70925
            0            0            0            0            0            0            1            0
            0            0            0            0            0            0            0            1
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
  Columns 9 through 16
       1.3609            0            0            0            0            0            0            0
        2.212            0            0            0            0            0            0            0
      -4.5774            0            0            0            0            0            0            0
  -0.00082049            0            0            0            0            0            0            0
   -0.0044845            0            0            0            0            0            0            0
   -0.0030313            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
            1            0            0            0            0            0            0            0
            0            1            0            0            0            0            0            0
            0            0            1            0            0            0            0            0
            0            0            0            1            0            0            0            0
            0            0            0            0            1            0            0            0
            0            0            0            0            0            1            0            0
            0            0            0            0            0            0            1            0
            0            0            0            0            0            0            0            1
            0            0            0            0            0            0            0            0
            0            0            0            0            0            0            0            0
  Columns 17 through 18
            0            0
            0            0
            0            0
            0            0
            0            0
            0            0
            0            0
            0            0
```

```
0          0
0          0
0          0
0          0
0          0
0          0
0          0
0          0
1          0
0          1
```

While the size is ungainly to examine closely, each value was checked with the solution, and verified to be correct.

To show the correctness, I found the maximum relative difference between the provided solution and my computed solution. The difference is extremely small, and can be attributed to numerical precision in the machine.

$$\boxed{\Delta_{rel,max} = [0.00064402]}$$

1.2 Problem 2 - State Deviation Vector $\hat{x}$

Next, I found the state deviation vector $\hat{x}$ after one iteration through the Kalman filter, and mapped it back to the epoch.

$$\hat{x}_0 = \begin{bmatrix} -0.0054022 \\ -0.30433 \\ -0.15502 \\ 0.040911 \\ 0.032783 \\ -0.014743 \\ -8.7659e+06 \\ -6.5631e-07 \\ 0.15604 \\ 1.8204e-06 \\ 1.347e-06 \\ -2.5081e-07 \\ -10.556 \\ 9.9579 \\ 5.7963 \\ -5.7542 \\ 2.3104 \\ 1.4786 \end{bmatrix}$$

These values were found for an integration tolerance of $1e^{-}13$, and I noticed that the solution obtained varies with the tolerances used. A range of solutions can be found, all on the same orders of magnitude as shown above, it all depends on the tolerances. When iterated, the solution converges within a few passes for all small tolerance values.

1.3 Problem 3 - Compare Solution to Batch Results

State deviation vectors after one pass through the sequential and batch filers are provided below. Both are found for an integration tolerance of $1e^{-13}$. In addition, the sequential result is provided when using the Joseph formulation.

$$\hat{x}_{0,Kalman} = \begin{bmatrix} -0.0054022 \\ -0.30433 \\ -0.15502 \\ 0.040911 \\ 0.032783 \\ -0.014743 \\ -8.7659e+06 \\ -6.5631e-07 \\ 0.15604 \\ 1.8204e-06 \\ 1.347e-06 \\ -2.5081e-07 \\ -10.556 \\ 9.9579 \\ 5.7963 \\ -5.7542 \\ 2.3104 \\ 1.4786 \end{bmatrix}, \hat{x}_{0,Batch} = \begin{bmatrix} -0.036303 \\ -0.27411 \\ -0.18088 \\ 0.040935 \\ 0.032748 \\ -0.014753 \\ -9.4634e+06 \\ -6.5736e-07 \\ 0.14755 \\ 1.8632e-06 \\ 1.3787e-06 \\ -2.5382e-07 \\ -10.564 \\ 9.9834 \\ 5.7943 \\ -5.7819 \\ 2.3444 \\ 1.5125 \end{bmatrix}, \hat{x}_{0,Kalman,Joseph} = \begin{bmatrix} -0.0099614 \\ -0.30223 \\ -0.15338 \\ 0.040912 \\ 0.03278 \\ -0.01474 \\ -8.935e+06 \\ -6.5659e-07 \\ 0.15548 \\ 1.8321e-06 \\ 1.3557e-06 \\ -2.5867e-07 \\ -10.559 \\ 9.9589 \\ 5.8016 \\ -5.7604 \\ 2.3074 \\ 1.503 \end{bmatrix}$$

All three solutions give a very similar solution. The Kalman and the Kalman + Joseph formulation are the closest solutions, as is to be expected. They both are very similar to the Batch method. Since all three should yield identical results on an infinitely precise machine, this is a good thing. Their differences are more pronounced in the elements where the *a-priori* covariance values are larger, and differences are more minute where the covariance matrix is tight. All of this gives evidence that the filters are all working correctly. Since they vary significantly (on this level) with the integration tolerances used, it is impossible to tell which method is the *best* at this time.

1.4 Problem 4 - Plot Trace of Position Elements of Covariance (Kalman)

See Figure 1 below for a plot of the trace of the position elements (first 3x3) of the covariance matrix, shown after the first iteration through the Kalman filter.

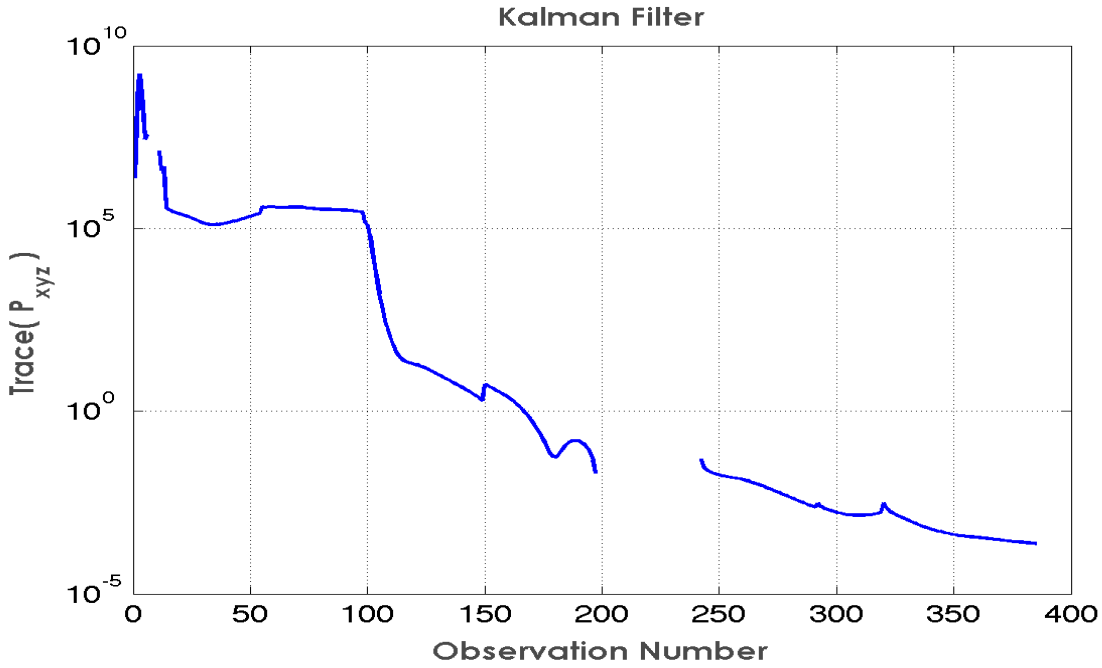

Figure 1: Kalman Covariance Trace

The plot is shown on a semilogy scale for clarity. At a few points there are negative trace values (hence the gaps in the plot), but overall there is a definite decreasing in the trace magnitude. This is expected from the filter, and indicates that the filter is indeed working correctly.

1.5 Problem 5 - Plot Trace of Position Elements of Covariance (Joseph)

Now, the covariance matrix is computed with the Joseph formulation. Recall that the Joseph formulation is an alternate method to calculate the covariance matrix, which guarantees a symmetric and positive semidefinite covariance matrix. The trace of position elements is plotted in Figure 2

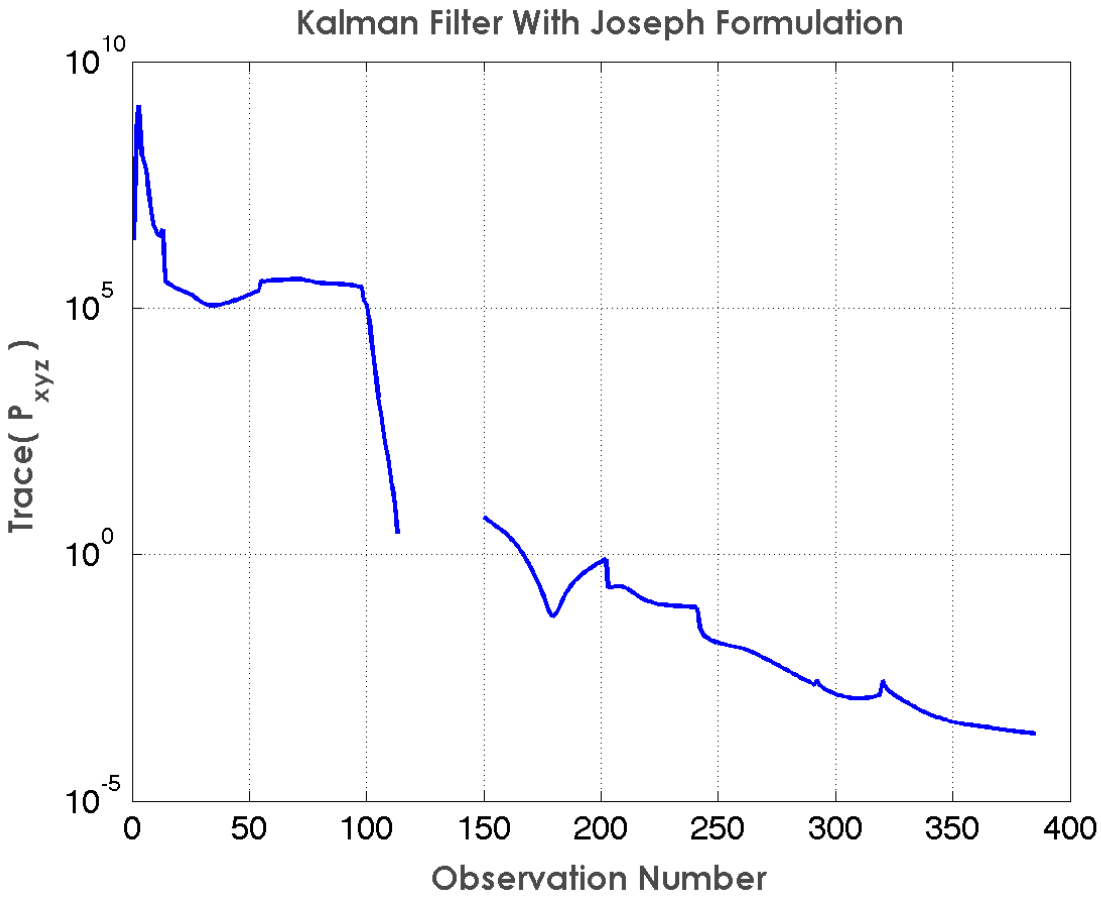

Figure 2: Joseph Covariance Trace

The trace of the covariance matrix calculated with the Joseph formulation looks similar throughout the orbit pass to the one calculated conventionally. That shows that there weren't serious problems with the original matrix becoming asymmetric or non-invertible. In this case, its not really evident that one method is any better than the other.

1.6 Problem 6 - Error Ellipsoid

The plot of the error ellipsoid for the covariance matrix after one pass is below. I plotted it with the provided *plotEllipsoid.m* function included in the assignment.

```
tmp = load('P.mat');
P_Batch = tmp.P;
P = P_Batch(1:3,1:3);
[evecs,evals] = eig(P);
semi(1) = sqrt(evals(1,1));
semi(2) = sqrt(evals(2,2));
semi(3) = sqrt(evals(3,3));
semi = sort(semi);
semi = semi([3,2,1]);
plotEllipsoid(evecs,semi)
figure_awesome('save')
```

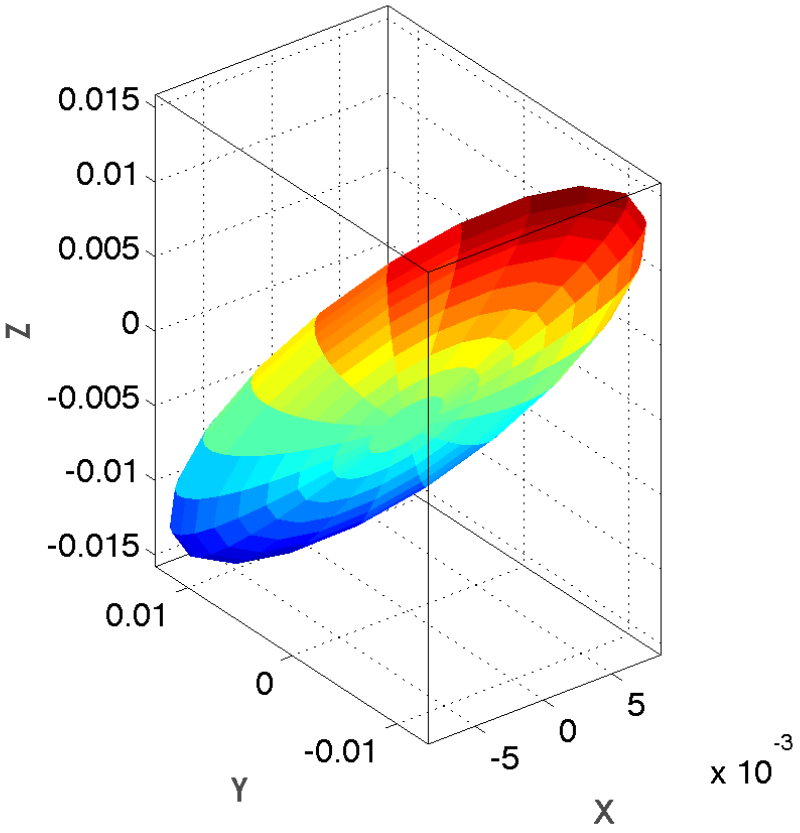

Figure 3: Covariance Ellipsoid

In Figure 3, you can see that the ellipsoid is tightest around X, then looser around Y then Z. Where tightness describes the size of the semi-major axis, and the *confidence* of our state predictions thereof. From the probability ellipsoid, it is evident that the Kalman filter is most accurate in the X and Y directions. This is a good tool to use in filter analysis. If I wanted to hit a target with the tightest tolerances in Z, I'd recognize this as a potential problem.

1.7 Problem 7 - Book Problem

I solved book problem 4.41 from the book using *Matlab*, with the provided dataset. The code is included in Appendix A. While initially confusing in how it was stated, this was actually a really useful example of a DMC, in a simple setting. It is a fairly complicated process, and seeing it here with the simple example was really helpful. The problem statement and outline left a lot to be inferred, but overall it was a really cool example. Figure 4 and Figure 5 show the result from this experiment.

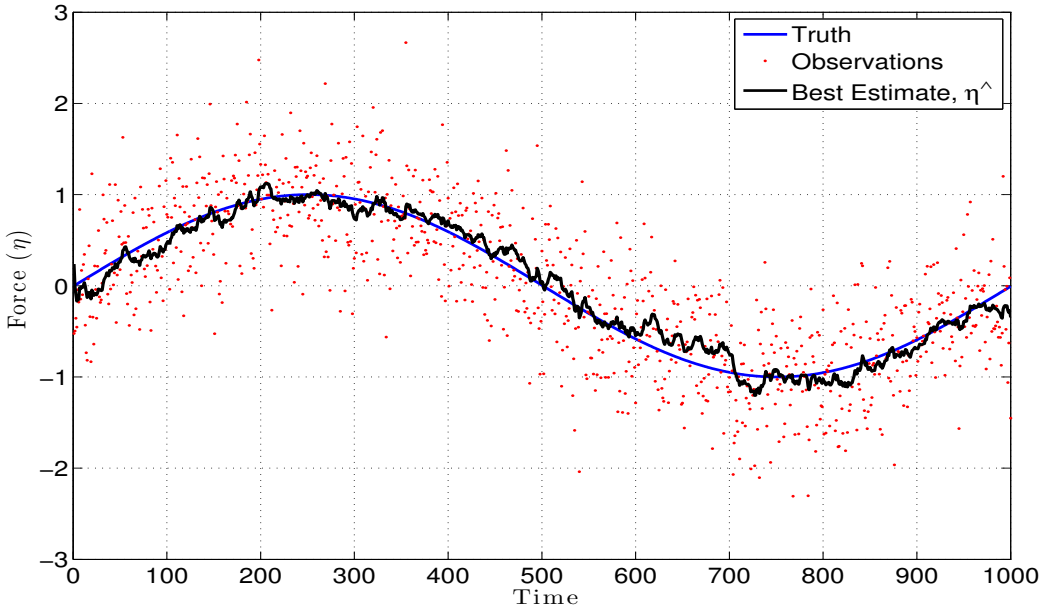

Figure 4: DMC Example

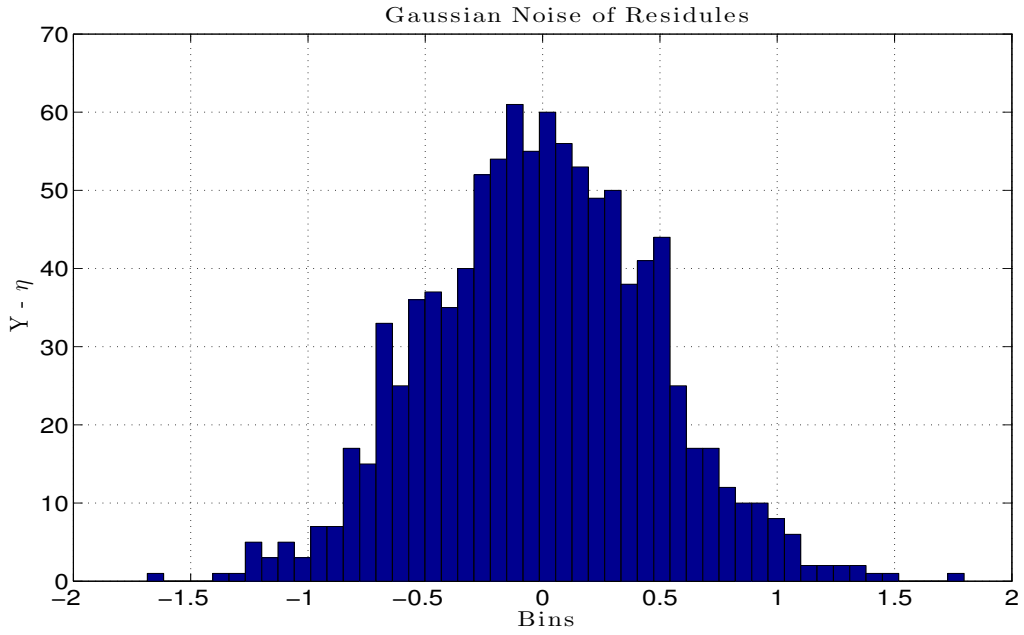

Figure 5: DMC Histogram of Residules

Even with a fairly large amount of noise on top of a sinusoidal (un-modeled) observation, the DMC method has backed out a fairly concise estimation of the mis-modeled forces. η holds the force data, and its path is close to the true un-modeled force.

Just for the sake of experimenting, I used *Matlab*'s curve fitting utils to turn the force estimate, $\hat{\eta}$ into an executable function. That function, evaluated for the times given is plotted against the true force acting in Figure 6.

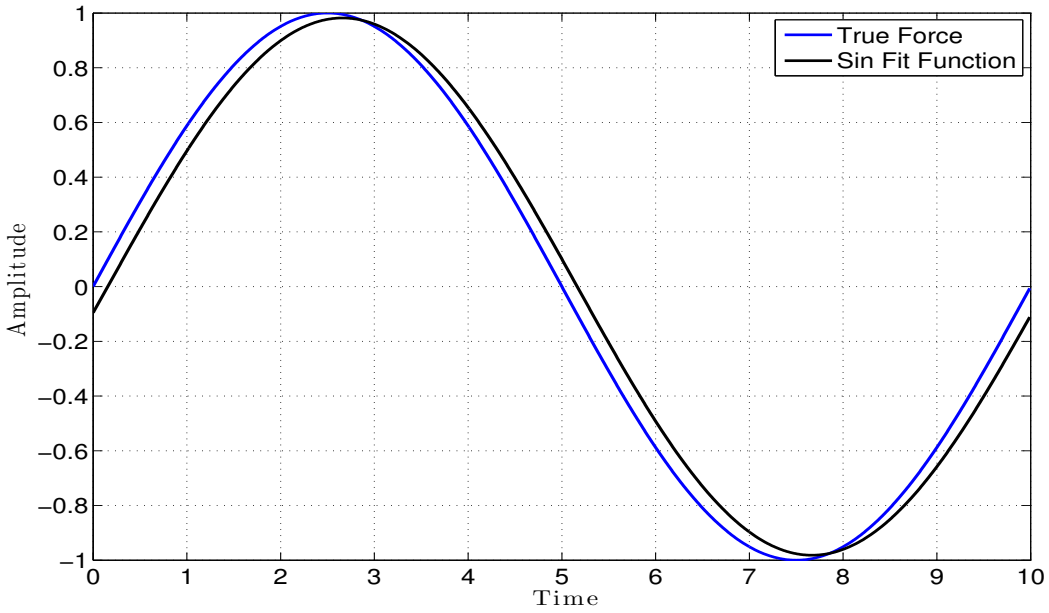

Figure 6: Truth and Computed Forces

With this additional model for the previously unknown force, we can conceivably augment our system to take it into account. As a basic example, the resulting plot of the particle motion is shown in Figure 7 with our estimated force subtracted from the observations. What is left is what appears to be a zero resulting force, indicating that we have successfully accounted for the unknown force. While this is far from a final fool-proof example, it serves to illustrate the concept.

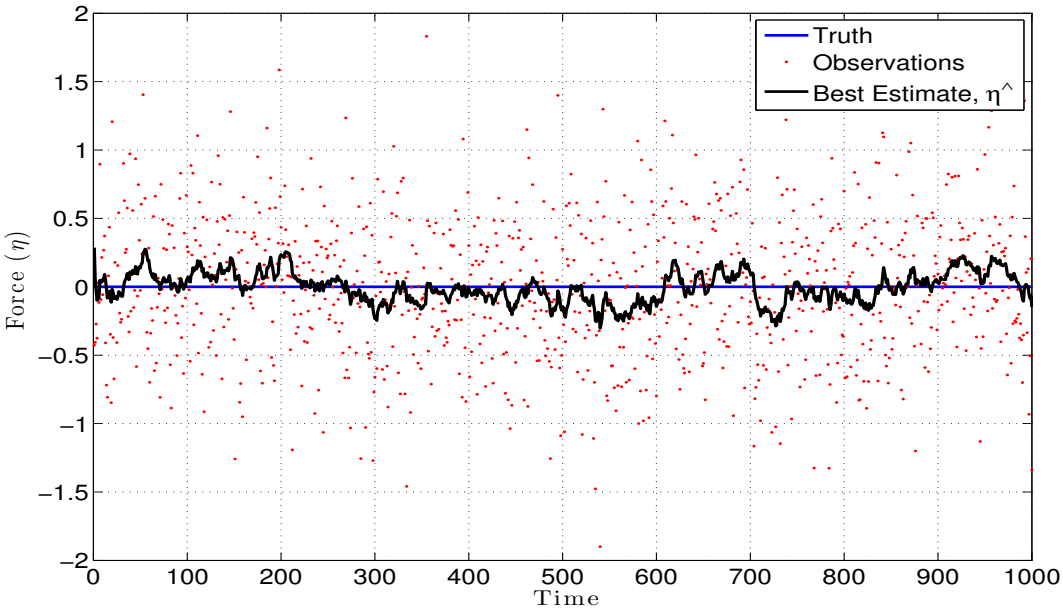

Figure 7: Motion with Force Estimate Subtracted

In a practical applicationI would try to do this live. I would likely perform a run through the Kalman filter to get a DMC force estimation. Then I would have my filter try various fits to find a function that matched the DMC results, and add that function into my dynamical model for integration. In actuality, this process is likely to be really difficult, but the idea is that the DMC results could be used to augment an erroneous dynamical model.

A Appendix A- MATLAB Generation Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Zach Dischner—12/4/2012
%
% ASEN 5070—Statistical Orbit Determination
%
% Homework 11
%
% Inputs    : None
%
% Outputs   : Plots for homework 11
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;clear all;close all; format compact;format long g;tic

%% Plot the Ellipsoid
tmp = load('P.mat');
P_Batch = tmp.P;
P = P_Batch(1:3,1:3);
[evecs,evals] = eig(P);
semi(1) = sqrt(evals(1,1));
semi(2) = sqrt(evals(2,2));
semi(3) = sqrt(evals(3,3));
semi = sort(semi);
semi = semi([3,2,1]);
plotEllipsoid(evecs,semi)


%% Problem 4—41, call function
Problem4_41

figure_awesome('save')



    % Problem 4—41 in StatoD book

clc;clear all;close all
set(0,'defaulttextinterpreter','latex')

obs     =load('hw11.dat');
time    =obs(:,1);
Y       =obs(:,2);

Beta    = 0.02;
sigma   = 0.67;

Eta(1) = Y(1);

Truth = sin(2*pi*time/10);

for ii = 1:length(Y)

    Htilde(ii) = 1;
    R(ii) = 1;
    Q(ii) = 1;

    if ii > 1
        m(ii) = exp(—Beta*(time(ii)—time(ii—1)));
        Gamma(ii) = sqrt(sigma^2/(2*Beta)*(1—m(ii)^2));
        PhiStep(ii) = m(ii);
        EtaBar(ii)  = PhiStep(ii)*EtaHat(ii—1);
```

```
        Pbar(ii)    = PhiStep(ii)*P(ii-1)*PhiStep(ii)' + Gamma(ii)*Q(ii-1)*Gamma(ii)';
    else
        m(ii) = exp(-Beta*(time(ii)));
        Gamma(ii) = sqrt(sigma^2/(2*Beta)*(1-m(ii)^2));
        Eta(ii) = m;
        Pbar(ii) = 1;
        EtaBar(ii) = 1;
    end

    K(ii) = Pbar(ii)/(Pbar(ii)+1);
    EtaHat(ii) = EtaBar(ii) + K(ii)*(Y(ii)-EtaBar(ii));
    P(ii) = (eye(size(K(ii)*Htilde(ii))) - K(ii)*Htilde(ii))*Pbar(ii);


end



figure
plot(Truth)
hold on
plot(Y,'r.')
plot(EtaHat,'k')


legend('Truth','Observations','Best Estimate, \eta^{\wedge}')
xlabel('Time')
ylabel('Force ($\eta$)')

figure
hist((Y-EtaHat'),50)
xlabel('Bins')
ylabel('Y - $\eta$')
title('Gaussian Noise of Residules')

% Fit Result to Sin
[fitresult, gof] = createFit(time, EtaHat);
% Eta_fcn = feval(fitresult,time);
%Convert cfit to a function handle
cfit2mfile(fitresult,'eta_force');
% Evaluate 'backed out' force function
T2 = eta_force(time);

figure
plot(time,Truth)
hold on
plot(time,T2,'k')
xlabel('Time')
ylabel('Amplitude')
legend('True Force','Sin Fit Function')


%% Now do it again, correcting for the modeled force
clear y Htilde R Q m Gamma PhiStep Pbar EtaBar Eta Hat K P
Y       =obs(:,2) - eta_force(time);

Beta    = 0.02;
sigma   = 0.67;

Eta(1) = Y(1);

Truth = zeros(length(time),1);%sin(2*pi*time/10);

for ii = 1:length(Y)

    Htilde(ii) = 1;
    R(ii) = 1;
    Q(ii) = 1;

    if ii > 1
        m(ii) = exp(-Beta*(time(ii)-time(ii-1)));
        Gamma(ii) = sqrt(sigma^2/(2*Beta)*(1-m(ii)^2));
```

```
        PhiStep(ii) = m(ii);
        EtaBar(ii)  = PhiStep(ii)*EtaHat(ii-1);
        Pbar(ii)    = PhiStep(ii)*P(ii-1)*PhiStep(ii)' + Gamma(ii)*Q(ii-1)*Gamma(ii)';
    else
        m(ii) = exp(-Beta*(time(ii)));
        Gamma(ii) = sqrt(sigma^2/(2*Beta)*(1-m(ii)^2));
        Eta(ii) = m;
        Pbar(ii) = 1;
        EtaBar(ii) = 1;
    end

    K(ii) = Pbar(ii)/(Pbar(ii)+1);
    EtaHat(ii) = EtaBar(ii) + K(ii)*(Y(ii)-EtaBar(ii));
    P(ii) = (eye(size(K(ii)*Htilde(ii))) - K(ii)*Htilde(ii))*Pbar(ii);


end

figure
plot(Truth)
hold on
plot(Y,'r.')
plot(EtaHat,'k')


legend('Truth','Observations','Best Estimate, \eta^{\wedge}')
xlabel('Time')
ylabel('Force ($\eta$)')

figure
hist((Y-EtaHat'),50)
xlabel('Bins')
ylabel('Y - $\eta$')
title('Gaussian Noise of Residules')

figure_awesome('save')
```

```
    % function Xstar0 = KalmanFilter()
% Compute new xhat
% Looks for functions to return:
%   G
%   Htilde
% Inside of /Filters/KalmanTools.m
%
% Notation:    t0 is a priori, like M_(t_i-1)
%              t1 is current step   M_(ti)

clc; clear all;close all; format compact;tic
warning off MATLAB:nearlySingularMatrix

%% First, Load and Parse Observation Data
load('Observations.mat');
t_obs       = obs(:,1);
station     = obs(:,2);
rho_obs     = obs(:,3);
rhodot_obs  = obs(:,4);

%% Pre-Allocations
x       = zeros(length(obs));
y       = x;
z       = x;
xdot    = x;
ydot    = x;
zdot    = x;
```

```
Xsite1  = x;
Ysite1  = x;
Zsite1  = x;
Xsite2  = x;
Ysite2  = x;
Zsite2  = x;
Xsite3  = x;
Ysite3  = x;
Zsite3  = x;
y1   = zeros(2,length(obs));
Xsite   = x;
Ysite   = x;
Zsite   = x;
theta   = x;


%% Initialize Variables

% Calculations for rho, rhodot. Put into bigger functiom sometime
%-----------------------------------------------
findrhostar      = @(x,y,z,Xsite,Ysite,Zsite,theta) sqrt(x^2+y^2+z^2+Xsite^2+Ysite^2+Zsite^2-2*(x*Xsite+y*Ysite)
findrhodotstar   = @(x,y,z,xdot,ydot,zdot,Xsite,Ysite,Zsite,theta,theta_dot,rho) (x*xdot + y*ydot + z*zdot - (xd
                   +(xdot*Ysite - ydot*Xsite)*sin(theta) + theta_dot*(x*Ysite - y*Xsite)*cos(theta) - zdot*Zsit
                                                          /rho;
%-----------------------------------------------

% System Constants
%-----------------------------------------------
Phi_Init    = eye(18,18);
tol         = 1e-13;
uE          = 3.986004415e14;        % m^3/s^2
J2          = 1.082626925638815e-3;  % []
Cd          = 2;                     % []
theta_dot   = 7.29211585530066e-5;   % rad/s
time        = t_obs;
%-----------------------------------------------

% Information
%-----------------------------------------------
sigma_rho   = 0.01;                  % rms std
sigma_rhodot= 0.001;                 % rms std
R           = [sigma_rho^2  ,         0        ; ...
                    0       ,    sigma_rhodot^2];
W = inv(R);
Pbar0       = diag([1e6,1e6,1e6,1e6,1e6,1e6,1e20,1e6,1e6,1e-10,1e-10,1e-10,1e6,1e6,1e6,1e6,1e6,1e6]);
%-----------------------------------------------

% Initial Conditions
%-----------------------------------------------
RV_Init     = [757700,5222607.0,4851500.0,2213.21,4678.34,-5371.30];
Station_Init= [-5127510.0 , -3794160.0 , 0.0 ,...               %101
                3860910.0 , 3238490.0  , 3898094.0 , ...        %337
                549505.0  , -1380872.0 , 6182197.0 ];           %394
Const_Init = [uE , J2 , Cd ];
%-----------------------------------------------

% Form Initialization State
%-----------------------------------------------
Xstar0 = [RV_Init , Const_Init , Station_Init , reshape(Phi_Init,1,length(Phi_Init)^2)]';
%-----------------------------------------------

% Initialize Kalman Filter
%-----------------------------------------------
xbar       = zeros(18,1);
xhat       = xbar;

Phi_tk_t0   = Phi_Init;%ones(size(Phi_Init));
%-----------------------------------------------


%% Perform Kalman Loop
num_iterations = 1;
```

```
for ii = 1:num_iterations
    tr=[];
   P(:,:,1)          = Pbar0;

   % Dynamical Integration
   %-----------------------------------------------
   tol_mat     = ones(size(Xstar0)) .* tol;
   options     = odeset('RelTol',tol,'AbsTol',tol,'OutputFcn',@odetpbar);

   [time,StatePhi] = ode45(@StateDeriv_WithPhi,time,Xstar0,options);
   %-----------------------------------------------


   % Reform Phi Matrix
   %-----------------------------------------------
   for jj = 1:length(time)
          Phi(:,:,jj)     = reshape(StatePhi(jj,19:end),size(Phi_Init));
          Xstar    = StatePhi(:,1:18);
          x        = Xstar(:,1);
          y        = Xstar(:,2);
          z        = Xstar(:,3);
          xdot     = Xstar(:,4);
          ydot     = Xstar(:,5);
          zdot     = Xstar(:,6);
          Xsite1   = Xstar(:,10);
          Ysite1   = Xstar(:,11);
          Zsite1   = Xstar(:,12);
          Xsite2   = Xstar(:,13);
          Ysite2   = Xstar(:,14);
          Zsite2   = Xstar(:,15);
          Xsite3   = Xstar(:,16);
          Ysite3   = Xstar(:,17);
          Zsite3   = Xstar(:,18);

   %-----------------------------------------------


   % Htilde, yi, Ki
   %-----------------------------------------------
       theta(jj)          = theta_dot*time(jj);
       Htilde             = zeros(2,18);
       % Check Stations
       %-----------------------------------------
       %Station 1
       if station(jj) == 101
           Xsite(jj)   = Xsite1(jj);   Ysite(jj)=Ysite1(jj);   Zsite(jj)=Zsite1(jj);
           % Find H Tilde
           %-----------------------------------------------
           Htilde = FindHtilde(Xsite(jj),Ysite(jj),Zsite(jj),theta(jj),theta_dot,x(jj),xdot(jj),y(jj),ydot(jj),
           %-----------------------------------------------
           Htilde  = [Htilde , zeros(2,6)];

       end

       %Station 2
       if station(jj) == 337
           Xsite(jj)   = Xsite2(jj);   Ysite(jj)=Ysite2(jj);   Zsite(jj)=Zsite2(jj);
           % Find H Tilde
           %-----------------------------------------------
           Htilde = FindHtilde(Xsite(jj),Ysite(jj),Zsite(jj),theta(jj),theta_dot,x(jj),xdot(jj),y(jj),ydot(jj),
           %-----------------------------------------------
           Htilde  = [Htilde(:,1:9) , zeros(2,3), Htilde(:,10:12),zeros(2,3)];
       end

       %Station 3
       if station(jj) == 394
           Xsite(jj)   = Xsite3(jj);   Ysite(jj)=Ysite3(jj);   Zsite(jj)=Zsite3(jj);
           % Find H Tilde
           %-----------------------------------------------
           Htilde = FindHtilde(Xsite(jj),Ysite(jj),Zsite(jj),theta(jj),theta_dot,x(jj),xdot(jj),y(jj),ydot(jj),
           %-----------------------------------------------
           Htilde  = [Htilde(:,1:9),zeros(2,6),Htilde(:,10:12)];
```

```
        end
        %-------------------------------------------------


        % Time Update
        %-------------------------------------------------
        if jj > 1
            Phi_step= Phi(:,:,jj)/Phi(:,:,jj-1);
            xbar(:,:,jj)    = Phi_step*xhat(:,:,jj-1);
            P(:,:,jj)    = Phi_step*P(:,:,jj-1)*Phi_step';

        end
        %-------------------------------------------------

        % Put into FindG
        %-------------------------------------------------
        rhostar    = findrhostar(x(jj),y(jj),z(jj),Xsite(jj),Ysite(jj),Zsite(jj),theta(jj));
        rhodotstar= findrhodotstar(x(jj),y(jj),z(jj),xdot(jj),ydot(jj),zdot(jj),Xsite(jj),Ysite(jj),Zsite(jj),th
        %-------------------------------------------------

        % Find Observation Deviations
        %-------------------------------------------------
        ystar        = [rhostar;rhodotstar];
        y1(:,jj)    = [rho_obs(jj);rhodot_obs(jj)] - ystar;
        %-------------------------------------------------


        % Kalman Gain
        %-------------------------------------------------
        K1           = P(:,:,jj)*Htilde'*inv(Htilde*P(:,:,jj)*Htilde' + R);
        %-------------------------------------------------

        % Measurement Update
        %-------------------------------------------------
        xhat(:,:,jj) = xbar(:,:,jj) + K1*(y1(:,jj) - Htilde*xbar(:,:,jj));
        P(:,:,jj) = (eye(size(K1*Htilde)) - K1*Htilde)*P(:,:,jj)*(eye(size(K1*Htilde))-K1*Htilde)' + K1*R*K1';
%         P(:,:,jj) = (eye(size(K1*Htilde)) - K1*Htilde)*P(:,:,jj);
        %-------------------------------------------------

        tr = [tr,trace(P(1:3,1:3,jj))];

    end   % End observation loop


    % Update Best Guess of Initial Conditions
    %-------------------------------------------------%
     Xstar0 = [Xstar0(1:18) + inv(Phi(:,:,end))*xhat(:,:,end); (reshape(Phi_Init,length(Phi_Init)^2,1))];
     P      = inv(Phi(:,:,end))*P(:,:,end)*inv(Phi(:,:,end))';
     xbar   = xbar(:,:,end-1) - xhat(:,:,end-1);



    %-------------------------------------------------
    fprintf('RMS of rho is :  %3.5f \n',rms(y1(1,:)))
    fprintf('RMS of rhodot is :  %3.5f \n',rms(y1(2,:)))



    figure(1)
    subplot(num_iterations,2,2*ii-1)
    plot(y1(1,:))
    ylabel('rho residules')
    xlabel('Observation Number')

    subplot(num_iterations,2,2*ii)
    plot(y1(2,:))
    ylabel('rho dot residules')
    xlabel('Observation Number')
```

```
    figure(2)
    subplot(num_iterations,1,ii)
    semilogy(tr)
    xlabel('Observation Number')
    ylabel('Trace( P_x_y_z )')
    title('Kalman Filter With Joseph Formulation')

 end
figure_awesome('save')

fprintf('\n\nRunning Time for Kalman Filter : %3.5f\n\n',toc)
```