Lecture 13 Extended Kalman filter: Tracking a Time-Varying Sinusoid

2. The Time-Varying Sinusoid-Plus-Noise Model and Related State Equations

Consider what one might call a 'real-world' sinusoid:

$$s(t) = A(t)\sin[\theta(t)]. \tag{1}$$

In this work, both the amplitude and frequency are subject to small amounts of random jitter about known (deterministic) amplitude, A_0, and frequency, Ω_0. The goal here is to use the *extended Kalman filter* (EKF) to estimate the time-varying amplitude and frequency. We assume here that (1) is corrupted by a second order Gauss-Markov noise process, $\{n(t)\}$, so that the sampled measurement process is:

$$z(k\Delta) = s(k\Delta) + n(k\Delta). \tag{2}$$

The following seven-dimensional state variable will be used:

$$\mathbf{x}(t) = \left[\omega(t) \quad \Omega(t) \quad \theta(t) \quad a(t) \quad A(t) \quad n(t) \quad n(t-\Delta)\right]^{tr}$$

where

- $\omega(t)$ is the frequency jitter (rad/sec)
- $\Omega(t) = \omega(t) + \Omega_0$ is the time-varying frequency (rad/sec)
- $a(t)$ is the amplitude jitter
- $A(t) = a(t) + A_0$ is the time-varying amplitude
- $\theta(t) = \int_0^t \Omega(\tau)\, d\tau$ is the phase (rad) associated with the time-varying frequency
- $n(t)$ is a random variable associated with a second order Gauss-Markov additive noise process $\{n(t)\}$
- $n(t-\Delta)$ is delayed an amount Δ from $n(t)$.

2.1 The Sinusoid State Equations-

The frequency jitter model- The random frequency jitter, $\omega(t)$, will be modeled as:

$$\dot{\omega}(t) + \beta_\omega \omega(t) = e_\omega(t) \tag{3}$$

with power (i.e. variance) σ_ω^2, and with bandwidth parameter β_ω (rad/sec). A common special case of (3) is when $\beta_\omega = 0$. In this case $\omega(t)$ is termed a random walk, or a Brownian motion. It is a model that is often used in relation to rate gyro position noise [*]. However, it is also a model that presumes that the uncertainty associated with $\omega(t)$ increases over time. In this work we will use $\beta_\omega > 0$, reflecting the assumption that the frequency jitter is centered about the nominal frequency Ω_0, and has limited uncertainty, σ_ω about this value. It also reflects the assumption that the jitter has a limited bandwidth (in fact, a bandwidth that is small relative to Ω_0). The model (3) considered in this work is called a

first order Gauss-Markov (*GM*) model. It is intended to model slow frequency variations caused by road grade and head wind in relation to the use of automotive cruise control.

The amplitude jitter model- The amplitude jitter, $a(t)$, is also assumed to be a first order Gauss-Markov process associated with the stochastic differential equation

$$\dot{a}(t) + \beta_a a(t) = e_a(t) \tag{4}$$

with power σ_a^2, and with bandwidth parameter β_a (rad/sec) where $\beta_a > 0$ is also small relative to Ω_0. Furthermore, in this work the processes (3 and (4) are assumed to be mutually independent. In a cruise control setting, this assumption may or may not be justified, depending on the source of the sinusoid.

The sampled processes- Because all processes are sampled using a sampling interval Δ (chosen so as to avoid measurable aliasing), the sampled versions of (3) and (4) (using impulse-invariant sampling) are, respectively,

$$\omega_k = \alpha_\omega \omega_{k-1} + u_k^{(\omega)} \tag{5}$$

and

$$a_k = \alpha_a a_{k-1} + u_k^{(a)}. \tag{6}$$

In (5) and (6), and henceforth in this work, we will, for notational convenience, use the subscript notation for the time indices, as opposed to the notation in (2). The relation between the model parameters in (3-4) and (5-6) are, respectively,

$$\alpha_\omega = e^{-\beta_\omega \Delta} \quad ; \quad \sigma_{u^{(\omega)}}^2 = \sigma_\omega^2 \left[1 - (\alpha_\omega)^2\right] \tag{7}$$

and

$$\alpha_a = e^{-\beta_a \Delta} \quad ; \quad \sigma_{u^{(a)}}^2 = \sigma_\omega^2 \left[1 - (\alpha_a)^2\right]. \tag{8}$$

As mentioned above, we assume here that both β_ω and β_a are small relative to Ω_0. We will assume that Ω_0 is in the lower half of the frequency analysis range $[0, \pi/\Delta]$. Hence, the bandwidth parameters α_ω and α_a will both be close to 1.0. To quantify this closeness, we begin by noting that the sampled frequency jitter scaled autocorrelation function is

$\rho_\omega(m) = (e^{-\beta\Delta})^m = \alpha_\omega^m$. Then, for a vehicle traveling at 70 miles per with an engine crankshaft frequency $\Omega_o = 2\pi(50)\ rad/s.$ (3000 RPM) we would need to have a sampling frequency $f_{samp} \cong 5(50) = 250 Hz$ in order to have Ω_0 positioned at 40% of the analysis bandwidth of $125 Hz$. The vehicle speed of 70 mph corresponds to ~103 fps. Hence, if we assume road grade changes and head wind correlation length of ~100 ft, then only well after 1 second will the jitter process $\omega(t)$ have small $\rho_\omega(m)$. If we assume that $\rho_\omega(m_1) \cong 0.1$ for $m_1\Delta \cong 1\text{sec.}$ then $m_1 = 250$ samples. In this case, we then have $0.1 = \alpha_\omega^{m_1} = \alpha_\omega^{250}$, or $\alpha_\omega \cong 0.99$.

To obtain the difference equation for the phase, write:

$$\theta_k = \int_0^{k\Delta} \Omega(\tau)\, d\tau = \int_0^{(k-1)\Delta} \Omega(\tau)\, d\tau + \int_{(k-1)\Delta}^{k\Delta} \Omega(\tau)\, d\tau \cong \theta_{k-1} + \overline{\Omega}_k \Delta \tag{9}$$

where

where the rightmost approximate equality assumes that the sampling interval Δ is sufficiently small that for $(k-1)\Delta < \tau < k\Delta$ we have $\Omega(\tau) \cong \Omega[(k-1)\Delta]$.

The final two sinusoid discrete-time state equations are

$$\Omega_k = \Omega_0 + \omega_k \Delta \tag{10}$$

and

$$A_k = A_0 + a_k \,. \tag{11}$$

Combining (5), (6), (9), (10) and (11) gives

$$\mathbf{x}_k \overset{\Delta}{=} \begin{bmatrix} \omega_k \\ \Omega_k \\ \theta_k \\ a_k \\ A_k \end{bmatrix} = \begin{bmatrix} \alpha^{(\omega)} & 0 & 0 & 0 & 0 \\ \alpha^{(\omega)} & 0 & 0 & 0 & 0 \\ 0 & \Delta & 1 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{(a)} & 0 \\ 0 & 0 & 0 & \alpha^{(a)} & 0 \end{bmatrix} \begin{bmatrix} \omega_{k-1} \\ \Omega_{k-1} \\ \theta_{k-1} \\ a_{k-1} \\ A_{k-1} \end{bmatrix} + \begin{bmatrix} u_k^{(\omega)} \\ u_k^{(\omega)} \\ 0 \\ u_k^{(a)} \\ u_k^{(a)} \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega_0 \\ 0 \\ 0 \\ A_0 \end{bmatrix} \overset{\Delta}{=} \mathbf{F}^{(s)} \mathbf{x}_{k-1}^{(s)} + \mathbf{u}_{k-1}^{(s)} + \mathbf{d}_k^{(s)} . \tag{12}$$

2.2 The Noise State Equations-

As mentioned above, in this work we assume the additive noise corresponds to a second order Gauss-Markov process, with stochastic differential equation

$$\ddot{n} + \beta_1 \dot{n} + \beta_2 n = e_n(t) \,. \tag{12}$$

Furthermore, we will assume that the characteristic polynomial associated with (12) has the form

$$s^2 + \beta_1 s + \beta_2 = s^2 + (2\zeta\omega_n)s + \omega_n^2 \tag{13}$$

with complex roots. The parameter ζ is the damping ratio, and the parameter ω_n is the undamped natural frequency. The discrete-time version of (12) is then

$$n_{k+1} = -b_1 n_k - b_2 n_{k-1} + e_{k+1}^{(n)} \tag{14}$$

where $b_1 = -2e^{-\zeta\omega_n\Delta}\cos(\omega_d\Delta)$ and $b_2 = e^{-2\zeta\omega_n\Delta}$. Equation (14) can be written as:

$$\mathbf{x}_k^{(n)} \overset{\Delta}{=} \begin{bmatrix} n_k \\ n_{k-1} \end{bmatrix} = \begin{bmatrix} -b_1 & -b_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} n_{k-1} \\ n_{k-2} \end{bmatrix} + \begin{bmatrix} e_k^{(n)} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \overset{\Delta}{=} \mathbf{F}^{(n)} \mathbf{x}_{k-1}^{(n)} + \mathbf{u}_{k-1}^{(n)} + \mathbf{d}_k^{(n)} . \tag{15}$$

3. The Kalman Filter Equations

3.1 ***The Kalman Filter State Equations***

The Kalman filter state equation is:

$$\mathbf{x}_{k+1} = \mathbf{F}\,\mathbf{x}_k + \mathbf{u}_k + \mathbf{d}_{k+1} \tag{16}$$

where, from (12) and (15), we have $\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_k^{(s)} & \mathbf{x}_k^{(n)}\end{bmatrix}^{tr}$, $\mathbf{u}_k = \begin{bmatrix}\mathbf{u}_k^{(s)} & \mathbf{u}_k^{(n)}\end{bmatrix}^{tr}$, $\mathbf{d}_k = \begin{bmatrix}\mathbf{d}_k^{(s)} & \mathbf{d}_k^{(n)}\end{bmatrix}^{tr}$, and $\mathbf{F} = diag\begin{bmatrix}\mathbf{F}^{(s)} & \mathbf{F}^{(n)}\end{bmatrix}$. The state driving white noise, $\mathbf{d}_k$, has the following covariance matrix:

$$\mathbf{Q} = diag\begin{bmatrix}\mathbf{Q}^{(s)} & \mathbf{Q}^{(n)}\end{bmatrix} \tag{17a}$$

where

$$Q^{(s)} = \begin{bmatrix} \sigma^2_{u^{(\omega)}} & \sigma^2_{u^{(\omega)}} & 0 & 0 & 0 \\ \sigma^2_{u^{(\omega)}} & \sigma^2_{u^{(\omega)}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{u^{(a)}} & \sigma^2_{u^{(a)}} \\ 0 & 0 & 0 & \sigma^2_{u^{(a)}} & \sigma^2_{u^{(a)}} \end{bmatrix} \quad \text{and} \quad Q^{(n)} = \begin{bmatrix} \sigma^2_{u^{(n)}} & 0 \\ 0 & 0 \end{bmatrix}. \tag{17b}$$

For the chosen initial condition estimate $\mathbf{x}_{-1} = \begin{bmatrix}0 & W_0 & 0 & 0 & A_0 & 0 & 0\end{bmatrix}^{tr}$,

the mean-squared error (*mse*) of the estimator, $\widehat{\mathbf{x}}_0^- = \mathbf{x}_{-1}$ is simply

$$\mathbf{P}_0^- \overset{\Delta}{=} E[(\mathbf{x}_0 - \widehat{\mathbf{x}}_0^-)(\mathbf{x}_0 - \widehat{\mathbf{x}}_0^-)^{tr}] = E(\mathbf{x}_0\mathbf{x}_0^{tr}) = \begin{bmatrix} \sigma^2_{\omega} & \sigma^2_{\omega} & 0 & 0 & 0 & 0 & 0 \\ \sigma^2_{\omega} & \sigma^2_{\omega} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\theta_0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{a} & \sigma^2_{a} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{a} & \sigma^2_{a} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_n(0) & R_n(1) \\ 0 & 0 & 0 & 0 & 0 & R_n(1) & R_n(0) \end{bmatrix} \tag{18}$$

where σ^2_{ω} and σ^2_{a} are the variances of the jitter processes $\omega(t)$ and $a(t)$, respectively, and where $\sigma^2_{\theta_0} = \pi^2/3$ assumes that θ_0 is uniformly distributed over the interval $[-\pi, \pi)$.

3.2 *The Kalman Filter Measurement Equations*

In view of the results of the last section, the measurement equation (1) can be written as

$$z_k \;=\; A_k \sin(\theta_k) + n_k \;=\; h^{(s)}(\mathbf{x}_k^{(s)}) \;+\; \begin{bmatrix}\mathbf{0} & 1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}_k^{(s)} \\ \mathbf{x}_k^{(n)}\end{bmatrix}. \tag{19}$$

where $h(\mathbf{x}_k) = A_k \sin(\theta_k)$. Because the Kalman filter is a linear filter, it is necessary to linearize $h(\mathbf{x}_k)$. To this end, a first order Taylor series expansion about the sinusoid state estimate $\hat{x}_k^{(s)-} = \left[\hat{\omega}_k^- \quad \hat{W}_k^- \quad \hat{\theta}_k^- \quad \hat{a}_k^- \quad \hat{A}_k^-\right]^{tr}$ is used. Specifically,

$$h^{(s)}(\hat{\mathbf{x}}_k^-) \cong \hat{A}_k^- \sin(\hat{\theta}_k^-) + [\hat{A}_k^- \cos(\hat{\theta}_k^-)](\theta_k - \hat{\theta}_k^-) + [\sin(\hat{\theta}_k^-)](A_k - \hat{A}_k^-). \tag{20}$$

This, in turn, can be rewritten as

$$h^{(s)}(\hat{\mathbf{x}}_k^-) \cong [\hat{A}_k^- \cos(\hat{\theta}_k^-)]\theta_k + [\sin(\hat{\theta}_k^-)]A_k \quad - [\hat{A}_k^- \cos(\hat{\theta}_k^-)]\hat{\theta}_k^- \quad \stackrel{\Delta}{=} \quad h_3^{(s)}\theta_k + h_5^{(s)}A_k \quad - \; d_k. \tag{21}$$

From (21) it follows that (19) may be written as

$$z_k \quad = \quad A_k \sin(\theta_k) + n_k \quad \cong \quad \left[0 \quad 0 \quad h_3^{(s)} \quad 0 \quad h_5^{(s)} \quad 1 \quad 0\right]\mathbf{x}_k \; - \; d_k \stackrel{\Delta}{=} \mathbf{H}_k\mathbf{x}_k \; - \; d_k. \tag{22}$$

where the matrix $\mathbf{H}_k = \left[0 \quad 0 \quad h_3^{(s)} \quad 0 \quad h_5^{(s)} \quad 1 \quad 0\right]$ is used in the computation of the Kalman filter gain

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^{tr}\,(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^{tr})^{-1}. \tag{23}$$

It should be noted that the rightmost term in (22) is not used, since a more accurate estimate of the measurement z_k is simply

$$\hat{z}_k \quad = \quad \hat{A}_k^- \sin(\hat{\theta}_k^-) + \hat{n}_k^-. \tag{24}$$

The value of the exercise (20-22) is that it identified a structure for $\mathbf{H}_k$, which is needed to compute $\mathbf{K}_k$ in (23).

4. Examples

The following examples relate to the time-varying sinusoid-plus process described in Table 1. The sampling interval is $\Delta = 1\,\text{sec}.$

TABLE 1. Time-Varying Sinusoid and Colored Noise Parameter Values ($\Delta = 1\,\text{sec}.$)

Sinusoid:

Frequency: $F_0 = \Omega_0/2\pi = 0.2Hz \quad ; \quad \sigma_f = \sigma_\omega/2\pi = .005\,Hz \quad ; \quad \beta_f \cong .16\,Hz \quad ; \quad \alpha_\omega = 0.996$

Amplitude: $A_0 = 1.0 \quad ; \quad \sigma_a = .023 \quad ; \quad \beta_a \cong .16 \quad ; \quad \alpha_a = 0.996$

Noise: $\sigma_n = .707 \quad ; \quad f_n = \omega_n/2\pi = 0.106\,Hz \quad ; \quad \zeta = 0.1$

From Table 1 we see that the 3σ value of both the frequency and amplitude jitter is approximately 7% of the nominal values. To arrive at the value for the frequency jitter parameter $\alpha_\omega = 0.996$, first note that the nominal frequency $F_0 = \Omega_0/2\pi = 0.2Hz$ is a normalized frequency for an analysis range [0 , 0.5] *Hz*. Assuming that this normalized frequency corresponds to an engine frequency of 50 *Hz*, then the real time sampling frequency is 250 *Hz*, which corresponds to a sampling period $\Delta = .004\,\text{sec}.$ To mimic the influence of changing road grade on the fluctuations in

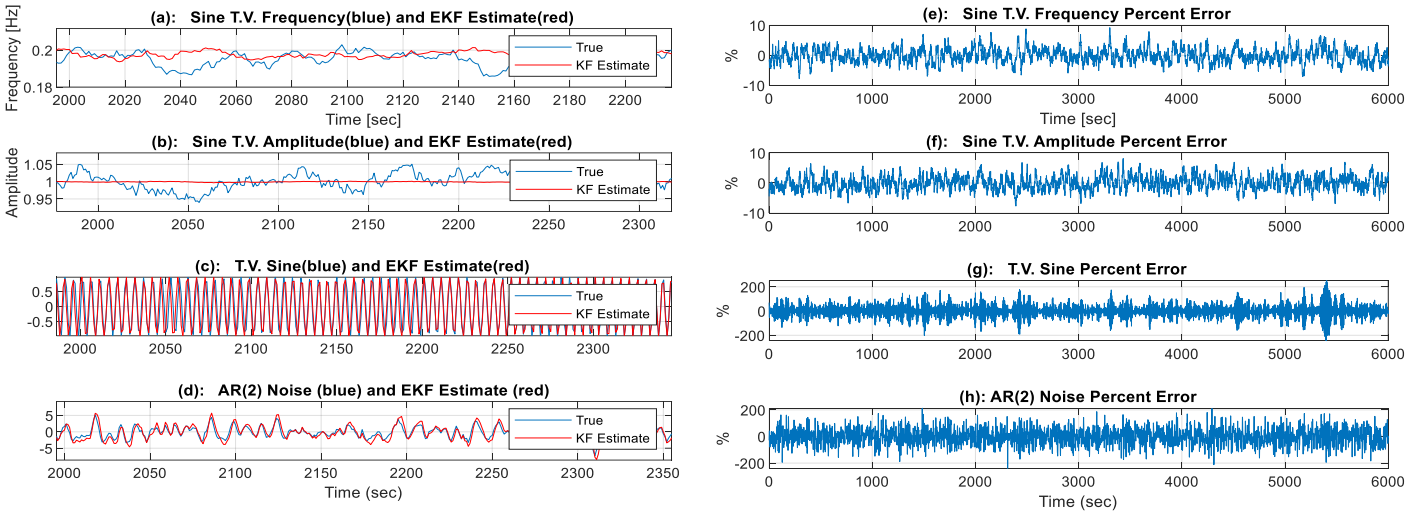
cruise control speed, it was assumed that the frequency jitter $4\tau_\omega$ de-correlation time was approximately 4 seconds. Hence, the jitter real time bandwidth is $\beta_\omega \cong 1\, rad/\sec$ [or $\beta_f \cong .16\, Hz$]. The result is that $\alpha_\omega = e^{-\beta_\omega \Delta} = 0.996$. The amplitude jitter was assumed to behave in the manner of the frequency jitter.

COMMENTS:

1. The data generation and KF codes are included in the Appendix. The student is encouraged to run it.

2. A more in-depth investigation of this problem is given in the Lectures folder. It is entitled: Extended Kalman Filtering- Tracking a Time-Varying Sinusoid. It is included, in part, to give the student ideas as to the type of investigation one might pursue in the context of a 573 project.

Partial Output:



Appendix Matlab Code

```
% PROGRAM NAME: sinegen_AND_sinekfdelt2018.m
%PURPOSE: generate a realization of a t.v. sinusoid+noise
%=======================
npts=6000;   % LENGTH OF REALIZATION
ntot=npts + 500;
%================================================================
% GENERATION OF TIME-VARYING SINUSOID with AR(1) Ampl. & Freq.
% ================================================================
% NOMINAL AMPLITUDE AND FREQUENCY:
a0=1.0; %nominal amplitude
w0=2*pi*0.2; % nominal frequency
%======================
sea=0.01;      % amplitude driving noise sigma
sew=0.01;      % angular frequency driving noise sigma
sea2=sea^2;
sew2=sew^2;
%======================
% Generation of white processes
eda=sea*randn(1,ntot);
edw=sew*randn(1,ntot);
% Compute variances of a(t) and w(t)for use in EKF
aa=0.9; aw=0.95;
sa2=sea2/(1-0.9^2);
sw2=sew2/(1-0.95^2);
%====================
% Specify Initial Conditions
da(1)=eda(1);
dw(1)=edw(1);
theta(1)=0;
a(1)=a0;
w(1)=w0;
% Generate SIGNAL Realization
for t=2:ntot
da(t)=aa*da(t-1)+eda(t);
a(t)=a0 + da(t);
dw(t)=aw*dw(t-1)+edw(t);
w(t) = w0 + dw(t);
theta(t) = theta(t-1) + w(t);
end
%====================
%Plot True Freq. and Ampl.
ftrue=w(501:ntot)/(2*pi);
atrue=a(501:ntot);
figure(1)                  % FIGURE 1
plot(ftrue)
title('Actual T.V. SINUSOID Frequency')
xlabel('Time [sec]')
ylabel('Frequency [Hz]')
figure(2)                 % FIGURE 2
plot(atrue)
title('Actual T.V.SINUSOID Amplitude')
xlabel('Time [sec]')
ylabel('Amplitude')
% Compute and plot T.V. Sinusoid Realization
s=a.*sin(theta);
s=s(501:ntot);
figure(3)                % FIGURE 3
plot(s)
title('T.V. Sinusoid s(t) with A=1 & w=0.2')
% ======================================================
%         ADDITIVE NOISE REALIZATION AND PSD
% ======================================================
% Generate AR(2) Noise Process
a1=-1.6*cos(pi/5); a2=0.64;
Rn0=1.0; %Noise variance
```

```
%Compute Rn1 and sn2:
AA=[-a1 -a2 1;-(1+a2) 0 0;-a1 -1 0]; BB=Rn0*[1;a1;a2];
RR=AA^-1*BB;
Rn1=RR(1); Rn2=RR(2);
su2=BB(3); % additive AR noise sigma of white noise input
su=su2^.5;
u=su*randn(1,ntot);
% Compute and Plot Noise PSD
delf=1/2048;
fvec=0:delf:0.5-delf;
den=fft([1 a1 a2],2048);
den=den(1:1024);
ARspec=su2*(abs(den)).^-2;
ARspecdB=10*log10(ARspec);
figure(4)                    % FIGURE 4
plot(fvec,ARspecdB)
ylabel('dB')
xlabel('Frequency (Hz)')
title('AR(2) Noise PSD')
nar=zeros(1,ntot);
nar(1)=u(1);
nar(2)=u(2);
for t=3:ntot
   nar(t)=-a1*nar(t-1)-a2*nar(t-2)+u(t);
end
n=nar(501:ntot);
% ========================================================
%    CONSTRUCTION OF MEASUREMENT PROCESS REALIZATION
% ========================================================
z=s + n;
figure(5)                   % FIGURE 5
nvec=1:npts;
plot(nvec,z, 'b')
hold on
plot(nvec,s,'m')
hold off
title('T.V. Sinusoid with & without AR(2) Noise')
[1 a1 a2]
%*****************************************************
%*****************************************************
%*****************************************************
%*****************************************************
% PROGRAM NAME: sinekf_7D_ver3.m contained in 573 2013 folder
% PURPOSE: Track a t.v. sinusoid corresponding to sinegenver3.m
% with 7-D state
% REQUIRED INPUT:
%   z-array from program sinegen.m
%=====================================
% Model Matrices for xk = [wk Wk thk ak Ak nk nk-1]
% for deterministic input d = [0 w0 0 0 a0 0 0]':
Phi = [ aw 0 0   0 0    0   0; % wkm1 <--wn sew2
        aw 0 0   0 0    0   0; % Wkm1
        0  1 1   0 0    0   0; % akm1 <--wn sea2
        0  0 0 aa 0     0   0; % thkm1
        0  0 0 aa 0     0   0; % Akm1
        0  0 0   0 0 -a1 -a2;% nkm1  <--wn su2
        0  0 0   0 0    1   0];% nkm2
R=0; %Measurement noise variance - ideally ZERO
%=====================================
% EKF Initial Conditions
xm = [0 w0 0 0 a0 0 0]';
x=xm;
Pm = [sw2     sw2    0     0    0     0   0;
      sw2     sw2    0     0    0     0   0;
       0       0     0     0    0     0   0;
       0       0     0    sa2   sa2   0   0;
```

```
       0       0     0    sa2   sa2   0   0;
       0       0     0     0    0    Rn0 Rn1;
       0       0     0     0    0    Rn1 Rn0];
%====================================
% EKF Loop
I = eye(7);
for k=1:npts
d = [0 w0 0 0 a0 0 0]';
Q = [sew2  sew2  0     0          0       0  0;
     sew2  sew2  0     0          0       0  0;
      0    0     0     0          0       0  0;
      0    0     0   sf*sea2    sf*sea2   0  0;
      0    0     0   sf*sea2    sf*sea2   0  0;
      0    0     0     0          0      su2 0;
      0    0     0     0          0       0  0];
H = [0 0 xm(5)*cos(xm(3))  0   sin(xm(3)) 1 0];
K = Pm*H'*(H*Pm*H' + R)^(-1);
zm = xm(5)*sin(xm(3))-a1*xm(6) - a2*xm(7);
xhatk=xm + K*(z(k) - zm);
x=[x,xhatk];
P=(I - K*H)*Pm;
xm=Phi*xhatk + d;
Pm=Phi*P*Phi' + Q;
end
x=x(:,2:npts+1);
tvec = 2:npts+1;
what = x(2,:);
fhat=(2*pi)^-1 * what;
thetahat = x(3,:);
ahat= x(5,:);
nhat = x(6,:);
%=========================================================================
tvec = 1:npts;
%=========================================================================
%    PLOTS OF STATE AND ESTIMATION ERROR PROCESSES

% STATE PLOTS:
figure(10)
%plot(tvec,f,tvec,fhat,'r');
subplot(4,1,1), plot(tvec,ftrue,tvec,fhat,'r');
legend('True','KF Estimate')
title('(a):   Sine T.V. Frequency(blue) and EKF Estimate(red)')
xlabel('Time [sec]')
ylabel('Frequency [Hz]')
%xlabel('Time (sec)')
grid

%figure(11)
%plot(tvec,atrue,tvec,ahat,'r');
subplot(4,1,2), plot(tvec,atrue,tvec,ahat,'r');
legend('True','KF Estimate')
title('(b):   Sine T.V. Amplitude(blue) and EKF Estimate(red)')
%xlabel('Time [sec]')
ylabel('Amplitude')
%xlabel('Time (sec)')
grid

%=====================
% Reconstruct Signal Estimate
shat=ahat.*sin(thetahat);
%figure(12)
%plot(tvec,s,tvec,shat,'r');
subplot(4,1,3), plot(tvec,s,tvec,shat,'r');
legend('True','KF Estimate')
title('(c):   T.V. Sine(blue) and EKF Estimate(red)')
%xlabel('Time (sec)')
```

```
grid

%figure(13)
%plot(tvec,n,tvec,nhat,'r');
subplot(4,1,4), plot(tvec,n,tvec,nhat,'r')
legend('True','KF Estimate')
title('(d):   AR(2) Noise (blue) and EKF Estimate (red)')
xlabel('Time (sec)')
grid

% ESTIMATION PERCENT ERROR PLOTS:
figure(11)
%plot(tvec,f,tvec,fhat,'r');
ef = 100*(fhat-ftrue)./ftrue;;
subplot(4,1,1), plot(tvec,ef);
title('(e):   Sine T.V. Frequency Percent Error')
xlabel('Time [sec]')
ylabel('%')
grid

ea = 100*(ahat-atrue)./atrue;
subplot(4,1,2), plot(tvec,ea);
title('(f):   Sine T.V. Amplitude Percent Error')
%xlabel('Time [sec]')
ylabel('%')
grid

% Reconstruct Signal Estimate
shat=ahat.*sin(thetahat);
rse_s=sqrt(mean(s.^2));
es = (100/rse_s)*(shat-s);
subplot(4,1,3), plot(tvec,es);
title('(g):   T.V. Sine Percent Error')
ylabel('%')
grid

rse_n=sqrt(mean(n.^2));
en = (100/rse_n)*(nhat-n);
subplot(4,1,4), plot(tvec,en)
title('(h): AR(2) Noise Percent Error')
ylabel('%')
xlabel('Time (sec)')
grid
%======================
% Compute estimated percent mse's:
rmse_f=100*sqrt(mean((ftrue-fhat).^2)/mean(ftrue.^2));
rmse_a=100*sqrt(mean((atrue-ahat).^2)/mean(atrue.^2));
rmse_s=100*sqrt(mean((s-shat).^2)/mean(n.^2));
rmse_n=100*sqrt(mean((n-nhat).^2)/mean(n.^2));

[rmse_f rmse_a rmse_s rmse_n]
```