天猫店铺网址:https://telesky.tmall.com/

DT0064
Design tip

Noise analysis and identification in MEMS sensors, Allan, Time, Hadamard, Overlapping, Modified, Total variance

By Andrea Vitali

Main components	
LSM6DS3H	iNEMO inertial module: 3D accelerometer and 3D gyroscope
LSM6DSM/LSM6DSL	iNEMO inertial module: 3D accelerometer and 3D gyroscope
STEVAL-STLKT01V1	SensorTile development kit

Purpose and benefits

This design tip explains how to analyze and identify noise in MEMS sensors. Allan and Hadamard variance are explained, together with their variations (Overlapping, Modified and Total). Theoretical variance #1 (Theo1) is also mentioned.

Benefits:

- How to characterize MEMS sensors by means of Allan and other variances

The signal and the noise

The basic assumption is that the signal of interest is constant and flat during the measurement. The sensor output however is the sum of the signal of interest and the noise. Roughly speaking, the noise should average to zero in the long term.

Many samples are taken during the measurement. Analysis and identification of noise can help in determining how many samples can be averaged to minimize the variance of the sensor output.

The problem with the standard variance is that it is not well behaved for increasing length of the data run. To solve this problem, the Allan variance was developed. The Allan variance σ^2 is computed as the average of the squared difference between consecutive “samples” (2-sample variance). “Samples” are computed by averaging over m-samples in an time interval $\tau = m*Ts$, where $Ts=1/Fs$ is the sampling interval, and Fs is sampling frequency.

Allan variance and more

The Allan deviation $\sigma(\tau)$ is the square root of the Allan variance $\sigma^2(\tau)$. The slope of the log-log plot depends on the noise type. It is the slope which enables noise type identification. See the table below.

天猫店铺网址:https://telesky.tmall.com/

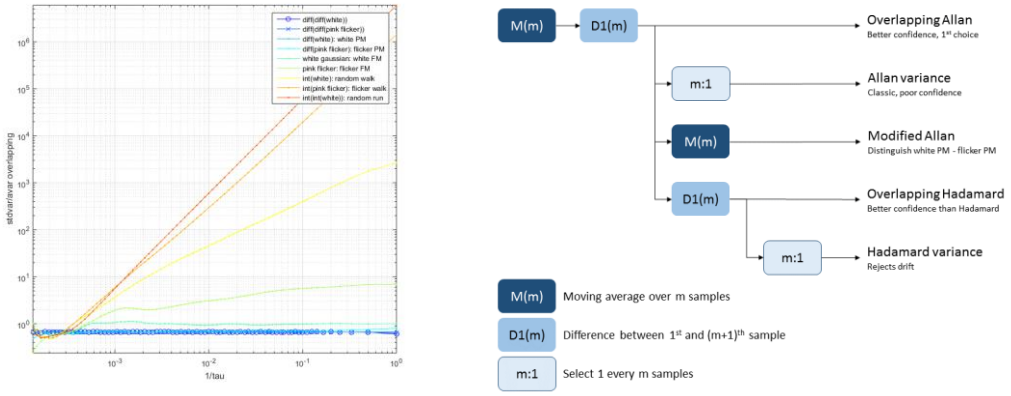
天猫店铺网址:https://telesky.tmall.com/

Allan variance (non-overlapping AVAR, overlapping OAVAR, modified MAVAR) and Hadamard variance (non-overlapping HVAR and overlapping OHVAR) can be computed using a chain of digital filters:

- M(m) a moving average over m-samples,
- D1(m) first difference between sample n^{th} and sample $(n+m+1)^{th}$,
- m:1 downsampling by selection of 1 sample out of m

Transients must not appear in the output. The variance is computed by squaring and averaging the output samples. The deviation is the square root of the variance. The confidence of the deviation can be estimated as the deviation itself divided by square root of the number of averaged output samples.

Figure 1. Left: ratio of var/AVAR; Right: block diagram of processing chain



Some important notes:

- Overlapping Allan (OAVAR) and overlapping Hadamard variance (OHVAR) have better confidence and should be used up to m = 10% of data run length.

- Time variance (TVAR) is a scaled version of Modified Allan variance (MAVAR), the scaling factor is $\tau^2/3$. It is optimal for white phase noise (this means: derivative of white Gaussian noise, which is known as white frequency noise).

- Allan Total, Modified Total and Hadamard Total variance have better confidence up to m = 30-50% of data run length. Total variance is computed by extending the data run length by reflection on both sides. The number of sample to be reflected is 2m for Allan, 3m for Modified Allan and Hadamard.

- Theoretical #1 variance (Theo1) has better confidence up to m = 75% of data run length. For all other variances, the stride time is the same as the averaging time used to compute the “samples”, var(τ=m*Ts). For Theo1, stride time is m*Ts minus averaging time; the averaging time goes from m/2 down to 1; stride time goes from m/2 up to m-1; the average stride time is 0.75*m*Ts: Theo1(τ=0.75*m*Ts). Theo1 is biased, Theo1BR (bias removed) is unbiased, Theo1H is Allan for m<10% and is Theo1BR for m>10%.

天猫店铺网址:https://telesky.tmall.com/

天猫店铺网址:https://telesky.tmall.com/

MatLab code to compute variance

This is the reference MatLab code to compute the aforementioned variance as a chain of digital filters.

```
% Allan as a digital filter
n % averaging factor, averaging n samples
Fs % sampling frequency
Ts=1/Ts;
tau=n*Ts;

%---- STDVAR and OSTDVAR
Mn=ones(1,n)/n; % averaging filter
dataM=filter(Mn,1,data); dataM=dataM(n:end); % filter and remove transient
ostdvar()=var(dataM);
stdvar()=var(dataM(1:n:end));

%---- AVAR and OAVAR
D1n=zeros(1,n+1); D1n(1)=+1; D1n(end)=-1; % differentiating filter
dataMD=filter(D1n,1,dataM); dataMD=dataMD(n+1:end); % filter and remove transient
L=length(dataMD);           oavar()=0.5*sum(dataMD.^2)/(L);
L=length(dataMD(1:n:end));  avar()=0.5*sum(dataMD(1:n:end).^2)/(L);

%---- MAVAR
dataMDM=filter(Mn,1,dataMD);
dataMDM=dataMDM(n:end);
L=length(dataMDM);          mavar()=0.5*sum(dataMDM.^2)/(L);

%---- HVAR and OHVAR
dataMDD=filter(D1n,1,dataMD); dataMDD=dataMDD(n+1:end); % filter and remove transient
L=length(dataMDD);          ohvar()=0.5*sum(dataMDD.^2)/(L);
L=length(dataMDD(1:n:end)); hvar()=0.5*sum(dataMDD(1:n:end).^2)/(L);
```

The slowest line in the above code is the moving average filter. Execution is much faster if a running sum is maintained where the new term is added and the old term is discarded.

```
% OAVAR optimized
n2=n*n;
acc(1)=sum(data(1:n)); % init running sum
for i=1:N-n, acc(i+1)=acc(i)-data(i)+data(i+n); end; % running sum
oavar()=0.5*sum( (acc(1:N-2*n+1) - acc(1+n:N-n+1)).^2 )/(N-2*n+1)/n2;

% AVAR
diffL=fix((N-2*n+1 -1)/n)+1;
avar()=0.5*sum( (acc(1:n:N-2*n+1)-acc(1+n:n:N-n+1)).^2 )/(diffL)/n2;

% STDVAR and OSTDVAR
stdvar()=var(acc(1:n:N-n+1))/n2;
ostdvar()=var(acc(1:N-n+1))/n2;

% MAVAR optimized
n4=n2*n2;
acc2(1)=sum(acc(1:n)); % init running sum
for i=1:N-2*n+1, acc2(i+1)=acc2(i)-acc(i)+acc(i+n); end; % running sum
mavar()=0.5*sum( (acc2(1:N-3*n+2) - acc2(1+n:N-2*n+2)).^2 )/(N-3*n+2)/n4;

% OHVAR
ohvar()=0.5*sum( ( (acc(1  :N-3*n+1) - acc(1+  n:N-2*n+1)) - ...
                   (acc(1+n:N-2*n+1) - acc(1+2*n:N-  n+1)) ).^2 )/(N-3*n+1)/n2;

% HVAR
diffL=fix((N-3*n+1 -1)/n)+1;
hvar()=0.5*sum( ( (acc(1  :N-3*n+1) - acc(1+  n:n:N-2*n+1)) - ...
                  (acc(1+n:n:N-2*n+1) - acc(1+2*n:n:N-  n+1)) ).^2 )/(diffL)/n2;
```

The code can be further optimized. As an example, for an optimized C-code, there is the possibility to compute the variance in one-pass (while usually two passes are required: the first for the mean, and the second for the actual variance).

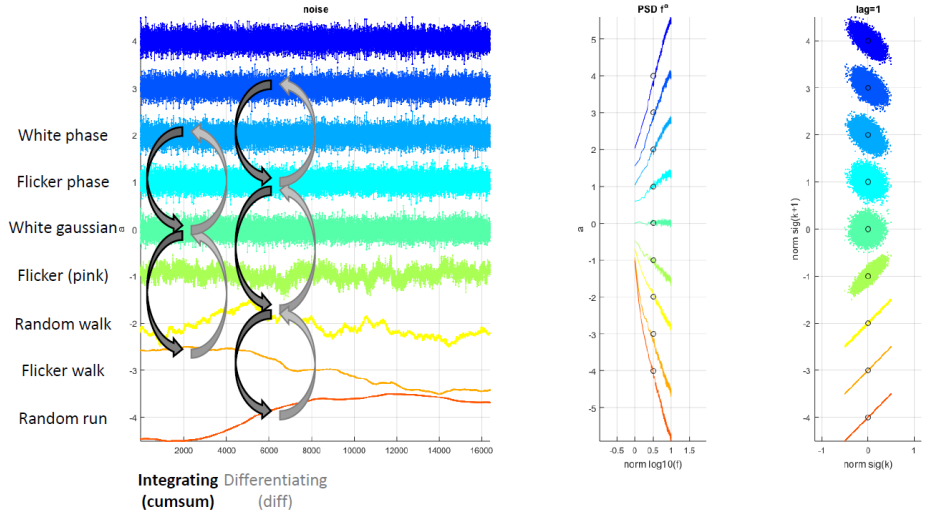
Noise models

White gaussian (WH) noise is the first basic type of noise. By integration (cumsum) one gets **Random Walk** (RW). By another integration one gets **Random Run** (RR). By taking the derivative (diff) one moves from White Frequency (WHFM) to White Phase noise (WHPM).

天猫店铺网址:https://telesky.tmall.com/

天猫店铺网址:https://telesky.tmall.com/

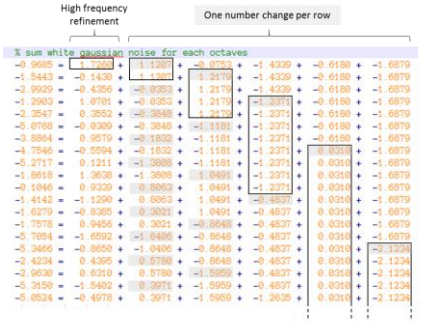
Figure 2. Noise plot, power spectral density (PSD), lag plot (x_k vs x_{k+1})



Flicker 1/f noise, also known as **pink** noise, is the other basic type of noise. It is equivalent to noise generated by solving dx(t)/dt = n1*x(t) + n2*w(t), where w(t) is white. It can be generated by summing different white noise generators in different octaves. By integration (cumsum), one gets **Flicker Walk** (FW). By taking the derivative (diff) one moves from Flicker Frequency (FLFM) to Flicker Phase (FLPM).

```
% generator for 1/f noise (flicker, pink)
  N % length of noise vector
  sd; % standard deviation for noise generators
  n=zeros(1,N); % init noise vector
  imax=floor(log2(N)); ngen=randn(1,imax+1)*sd; % init noise gen
  ngensum=sum(ngen); % init running sum
  for i=1:N,
    % find number of trailing zeros in the counter
    tlz=0; t=i; while mod(t,2)==0, tlz=tlz+1; t=t/2; end;

    % update running sum
    ngensum=ngensum-ngen(tlz+1); % remove old value
    ngen(tlz+1)=randn(1)*sd; % update noise generator
    ngensum=ngensum+ngen(tlz+1); % add new value
    n(i)=ngensum+(rand(1)*sd); % sum + noise for high freq
  end;
```



Noise identification

For noise identification, one must look at the slope in sigma-tau plots (deviation, square root of variance) and sigma^2-tau plot (variance). Modified Allan or Time deviation/variance is needed to distinguish between White PM and Flicker PM noise.

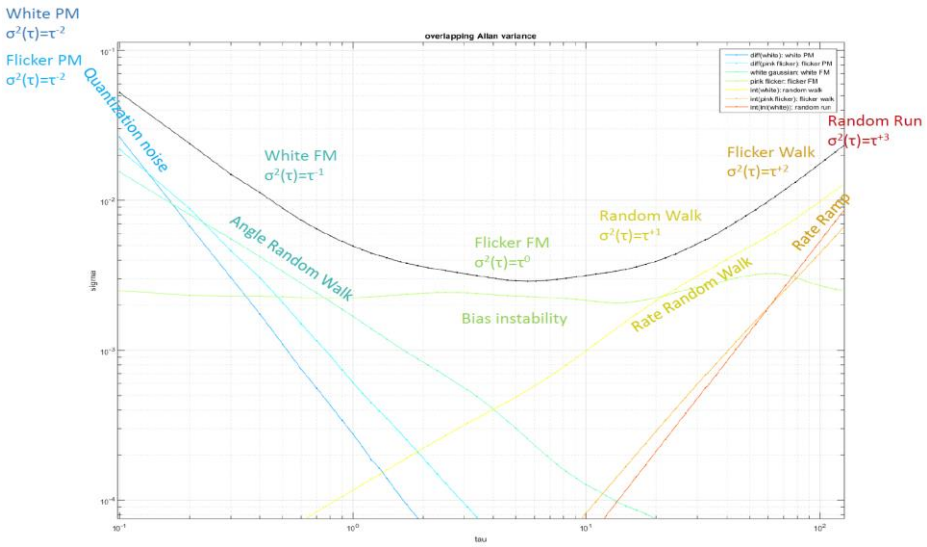
Noise type	Matlab code	Spectrum f^x	ADEV $\sigma(\tau)$	AVAR $\sigma^2(\tau)$	MADEV mod.$\sigma(\tau)$	MAVAR mod.$\sigma^2(\tau)$	TDEV $\sigma_T(\tau)$	TVAR $\sigma^2_T(\tau)$
White PM	Diff(WhiteFM)	f^{+2}	τ^{-1}	τ^{-2}	$\tau^{-3/2}$	τ^{-3}	$\tau^{-1/2}$	τ^{-1}
Flicker PM	Diff(FlickerFM)	f^{+1}	τ^{-1}	τ^{-2}	τ^{-1}	τ^{-2}	τ^{0}	τ^{0}
White FM	Randn(1)	f^{0}	$\tau^{-1/2}$	τ^{-1}	$\tau^{-1/2}$	τ^{-1}	$\tau^{+1/2}$	τ^{+1}
Flicker FM	See above	f^{-1}	τ^{0}	τ^{0}	τ^{0}	τ^{0}	τ^{+1}	τ^{+2}

天猫店铺网址:https://telesky.tmall.com/

天猫店铺网址:https://telesky.tmall.com/

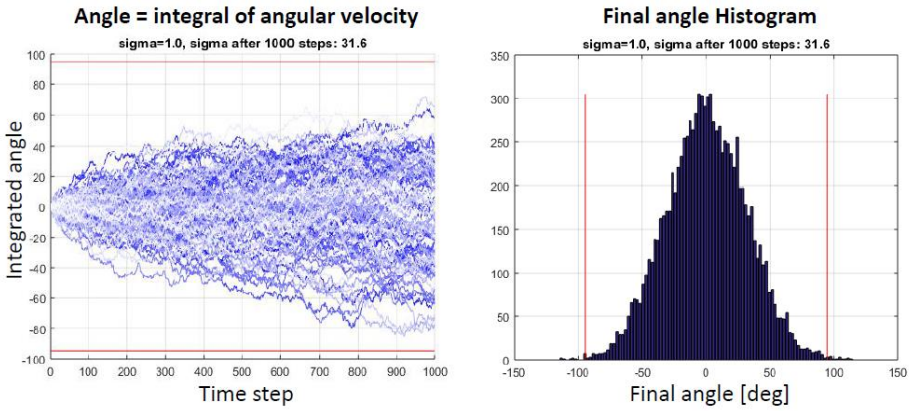
Random Walk	Cumsum(WhiteFM)	f^{-2}	$\tau^{+1/2}$	τ^{+1}	$\tau^{+1/2}$	τ^{+1}	$\tau^{+3/2}$	τ^{+3}
Flicker Walk	Cumsum(FlickerFM)	f^{-3}	τ^{+1}	τ^{+2}	τ^{+1}	τ^{+2}		
Random Run	Cumsum(RandomWalk	f^{-4}	$\tau^{+3/2}$	τ^{+3}	$\tau^{+3/2}$	τ^{+3}		

Figure 3. Allan variance plot, $\sigma^2(\tau)$ vs τ, for different types of noise with specific slopes.



Noise identification for Gyroscope

The gyroscope output is an angular velocity affected by white noise. Angular position is obtained by integration. When the gyroscope is not rotating, the output is not zero as it should be; instead it is white noise with zero mean and given standard deviation. The integration will lead to a non-zero final angle. This is the Angular Random Walk (ARW). Final angle error RMS = ARW*sqrt(time). Example: ARW 1deg/sqrt(s) * sqrt(1000s) = 31.6deg RMS.



Angular random walk can be found in the datasheet: **ARW in deg/sqrt(s) = noise density deg/s/sqrt(Hz)**. ARW can also be found in Allan plots: it is the intercept of Allan deviation segment with slope $\tau^{-1/2}$ at τ=1s, or Allan variance segment with slope τ^{-1} at τ=1s.

天猫店铺网址:https://telesky.tmall.com/

天猫店铺网址:https://telesky.tmall.com/

Related design support material
STEVAL-STLKT01V1, SensorTile development kit
Documentation
LSM6DS3H, iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
LSM6DSM, iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
LSM6DSL, iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope
AN4844, Application Note, LSM6DS3H: always on 3D accelerometer and 3D gyroscope

Revision history

Date	Version	Changes
13-Jul-2016	1	Initial release

天猫店铺网址:https://telesky.tmall.com/