KAERI/TR-2762/2004

技術報告書

Johnson Noise 온도계의 디지털 신호처리
- Johnson Noise 의 시계열 분석

(Digital Signal Processing for the Johnson Noise Thermometry
- A Time Series Analysis of the Johnson Noise)

한 국 원 자 력 연 구 소

제 출 문

한국원자력연구소장 귀하

이 보고서를 "Johnson Noise 온도계의 디지털 신호처리 - Johnson Noise 의 시계열 분석" 보고서로 제출합니다.

2003 년 6 월

작 성 자 한국원자력연구소
문 병 수
황 인 구
정 종 은
권 기 춘
Oak Ridge National Laboratory
Roger A. Kisner
David E. Holcomb

Contents

표 목 차

그 림 목 차

요 약 문

이 보고서에서는 첫째, 진폭, 주기, 및 위상이 각각 난수인 단일주기 신호들의 합으로부터 생성된 Random 신호는 Continuous Markov Process 를 형성하고 따라서 Gaussian White Noise 가 됨을 보였다. 이 Random 신호를 사용한 Simulation 을 통하여 Johnson Noise 온도계를 원자로 냉각재 온도측정에 사용할 경우 측정값의 상대오차가 0.14%미만인 매우 정확한 온도 값 측정이 가능함을 확인하였다. 물론, FPGA 등을 사용함으로써 충분히 빠른 시간 내에 Johnson Noise 의 통계처리가 가능하다는 전제에서이다. 둘째, Band-Pass Filter 를 거친 Johnson Noise 의 지정된 Filter Band 에 대한 최적의 Sampling Rate 결정에 관한 내용을 기술하였다. 또한, Johnson Noise 의 Linearity 문제와 온도에 따라 상대오차가 어떻게 변화하는지에 대한 분석 결과를 기술하였다.

셋째, 아주 단순한 전기회로로부터 장시간 동안 취득한 일련의 Johnson Noise 신호에 대한 분석 결과를 기술하였다. 취득한 데이터에는 약간의 Channel Noise 가 포함되어 있으며 이같이 Channel Noise 가 포함된 상태에서도 Continuous Markov Process 또는 Gaussian White Noise 의 성질은 유지가 됨을 확인하였다. 넷째, 온도의 값이 변화하고 있을 때 Long-term Average 또는 Moving Average 사용으로 인하여 발생하는 Time Lag 문제 해결을 위한 알고리즘에 대하여 기술하였다. 이 알고리즘은 Haar Wavelet 을 기반으로 하여 작성된 것으로 Long-Term Average 대신 Wavelet 을 사용한 근사값을 취함으로써 Time Lag 을 현격히 줄인 것이다.

SUMMARY

In this report, we first proved that a random signal obtained by taking the sum of a set of single frequency signals with random amplitude, random frequency, and random phase, generates a continuous Markov process and hence it is a Gaussian white noise. We used this random signal to simulate the Johnson noise and verified that the Johnson noise thermometry can be used to improve the measurements of the reactor coolant temperature within an accuracy of below 0.14%, provided that the necessary computation speed can be achieved by using FPGA's. Secondly, by using this random signal we determined the optimal sampling rate when the frequency band of the Johnson noise signal is given. Also the results of our examination on how good the linearity of the Johnson noise is and how large the relative error of the temperature could become when the temperature increases are described.

Thirdly, the results of our analysis on a set of the Johnson noise signal blocks taken from a simple electric circuit are described. We showed that the properties of the continuous Markov process are satisfied even when some channel noises are present. Finally, we describe the algorithm we devised to handle the problem of the time lag in the long-term average or the moving average in a transient state. The algorithm is based on the Haar wavelet and is to estimate the transient temperature that has much smaller time delay. We have shown that the algorithm can track the transient temperature successfully.

Chapter 1. Introduction

Johnson noise thermometry is one of the unconventional methods used for an accurate temperature measurement [1], by which one can establish thermodynamic temperature scale up to 1000°C with an accuracy of 0.2%. During the past 30years, there have been many studies and experimental implementations for application of the noise thermometers for hostile environments [2] such as space applications and high temperature reactors. Temperature measurements in space nuclear reactors [3] require an accuracy of 1 to 2% at temperatures up to about 1400K for about 10 years and the Johnson noise thermometry is believed to be able to provide this performance with measurement uncertainty reduced to 0.2% [4]. For high temperature gas reactors, there have been studies on Johnson noise thermometry for reliable in-core temperature measurements [5]. For nuclear power plant applications, ORNL reports that they performed tests in two operating reactors: Diablo Canyon and Sequoyah to obtain inaccuracies of less than 0.1% for ideal situations and 0.5-1% at the ends of long extension cables [6].

In this report, we describe the results of a statistical analysis performed as a part of the design work for a digital signal processing system to be used for a Johnson noise thermometry. The Johnson noise [7] is a synonym of the "thermal noise" generated by thermal agitation of electrons in a conductor and is known to be a result of the Brownian motion of ionized molecules within a resistance [8]. The noise power P in watts is given by $P = kT\Delta f$, where k is Boltzman's constant in joules per kelvin, T is the conductor temperature in kelvins, and Δf is the bandwidth in hertz. The thermal noise power has the property that it is equal throughout the frequency spectrum, depending only on k and T and hence it can be used to measure the temperature.

A random noise with its autocorrelation function zero everywhere but at 0, is called a white noise and the Johnson noise has the same property so that 'white noise' is another synonym of the Johnson noise, and we use this fact to study the Johnson noise statistically by generating random signals. We also examined a large set of Johnson noise signal blocks taken from a simplest electric circuit to see whether the signal blocks satisfy the properties of the Johnson noise even when some channel noise is present. Note that the data blocks taken from experiments must contain some channel noise and there is no way that we can pick out the Johnson noise alone.

There are a few different methods in implementing the Johnson noise thermometry. We will be using the correlation voltage method that is based on the correlation of signals from two different channels. The sensor signals will pick up noises in the long cables from the amplifiers to the A/D converter. If we take the expected value or the average of the correlation of the same signal through two different channels, however, the contribution by the noise will average out (in long term averages). Note that the correlation of these two signals is the same as the power of the sensor signal and that it is the autocorrelation of the sensor signal itself. Recall that the average of the signal power is a multiple of the temperature and by the Parseval's theorem, the average of the signal power is equal to the average of the cross product for the Fourier transforms of the two channel signals.

We will be using this cross power spectral density to remove the amplifier noises and other EMI noises picked up earlier and will apply a nonlinear gain correction by using the calibration signal. Finally, we will apply a moving average algorithm to improve the accuracy of the temperature for the steady state case and will apply a Haar wavelet approximation to determine the temperature when the temperature is in a transient state.

1. Correlation Voltage Method for Johnson Noise Thermometry

If a sensor resistor is connected directly across the inputs of two high-input-impedance voltage amplifiers, the output of each of the two amplifiers will consist of the sum of two parts, a noise voltage caused by the sensor and a noise voltage caused by that amplifier (and no contribution from the other amplifier). Thus the outputs of the two amplifiers are partially correlated because they both amplify the sensor voltage noise and partially uncorrelated because each independently generates its own noise. The outputs of two amplifiers can be written in the forms,

$$v_{01}(t) = A_1[v_s(t) + v_{a1}(t)], \quad v_{02}(t) = A_2[v_s(t) + v_{a2}(t)]$$

where A_1 and A_2 are the voltage gains of the two amplifiers, v_{a1} and v_{a2}, are the equivalent input noise voltage of the two amplifiers and v_s is the noise voltage of the sensor resistor. The product of the two amplifier output voltages is then

$$v_{01}(t)v_{02}(t) = A_1A_2[v_s(t) + v_{a1}(t)][v_s(t) + v_{a2}(t)]$$

or equivalently,

$$v_{01}(t)v_{02}(t) = A_1A_2[v_s^2(t) + v_s(t)v_{a1}(t) + v_s(t)v_{a2}(t) + v_{a1}(t)v_{a2}(t)] \quad \text{------------} \quad (1)$$

Since the amplifier noise voltages are uncorrelated with zero mean, the expected value (in practical terms, the long-term time average) of their product will be zero. Also the sensor noise voltage and the amplifier noise voltages are uncorrelated with zero mean and the expected value of their product is zero. Therefore,

$$E[v_{01}(t)v_{02}(t)] = A_1A_2E[v_s^2(t)]$$

where "$E(\cdot)$ " mean "the expected value of". Since the temperature of the sensor is proportional to the mean-squared noise voltage (which is the same as the expected value of the square of the noise voltage), the temperature can be computed from the Nyquist equation,

$$\overline{v}_s^2 = 4\kappa TR\Delta f = E[v_s^2(t)] = \frac{E[v_{01}(t)v_{02}(t)]}{A_1 A_2}$$

or

$$T = \frac{E[v_{01}(t)v_{02}(t)]}{4A_1 A_2 \kappa R\Delta f}$$

where *T* is the temperature of the sensor in kelvins, *k* is Boltzmann's constant, $1.38\times10^{-23}\ \frac{J}{K}$, *R* is the resistance of the sensor in ohms and Δf is the equivalent noise bandwidth of the measurement system.

The main advantage of the correlated voltage method is that the noise contributions of the two amplifiers do not have to be known to eliminate their effect. However, any external noise signal that gets into both amplifiers will be correlated at the output of the amplifiers and will cause a positive error in the measured temperature. If the external noise signal is narrowband (which is almost always the case) then digital signal processing can identify this type of interference and eliminate it.

A disadvantage of the correlated voltage method is that a separate measurement of the sensor resistance is needed. This measurement is however easy to make with high accuracy if it is done at DC using the 4-wire technique. It should be pointed out here that the *R* in the Nyquist equation is, strictly speaking, not the DC resistance of the sensor but the real part of the sensor impedance in whatever frequency band the measurement is made. If the measurement of the sensor noise is made in a bandwidth that is centered at a high frequency, the real part of the sensor impedance in that bandwidth may not be the same as the DC resistance. (For typical platinum resistance thermometers the difference between the DC resistance and the real part of the sensor impedance is negligible from dc to beyond 1 MHz.) At higher frequencies care must be taken to ensure that the resistance measured is the same as the real part of the impedance in the frequency range used.

2. Statistical Error Estimate of the Johnson Noise Thermometry

The material in this section is taken from a preprint material written by Professor Mike Roberts of University of Tennessee in May 2002. In this section, we derive an estimate for the uncertainty of the measurement of temperature based solely on the statistical properties of the resistor noise and ignoring the effects of amplifier noise. Of course the amplifier has significant noise that cannot simply be neglected. But this analysis will serve as a starting point for the more general analysis which includes the effects of amplifier noise.

Digital measurement of signal characteristics begins with sampling the signal. If the white noise of the resistor could be band limited and sampled at the Nyquist rate without noise, its mean-squared value could be directly estimated from the samples and knowledge of the bandwidth. (Actually, the signal could be sampled below the Nyquist rate. All that is important in the following statistical analysis is that the samples be independent. However if the signal is sampled below the Nyquist rate the noise and narrowband interference will both be aliased and that complicates some of the later analysis involving the CPSD.) If *N* independent samples are taken, an unbiased estimate of the mean-squared resistor noise voltage is

$$\overline{v}_R^2 = \frac{1}{N}\sum_{n=0}^{N} v_R^2[n]$$

where $v_R[n]$ is the "*n*th" sample of the resistor's open-circuit noise voltage. (The divisor is *N* instead of *N*-1 in this case because the mean value is known apriori to be zero and therefore a degree of freedom is not lost in trying to estimate the mean from the set of *N* data.) This can be rewritten as

$$\overline{v}_R^2 = \frac{E(v_R^2)}{N}\sum_{n=1}^{N}\frac{v_R^2[n]}{E(v_R^2)} = \frac{E(v_R^2)}{N}\chi^2(N) \qquad \text{------------------------------} \qquad (2)$$

where $\chi^2(N)$ is the "chi-square" random variable defined by

$$\chi^2(N) = \sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^2$$

where the x_i's are Gaussian distributed and independent with a mean of μ and a standard deviation of σ . Therefore, $\frac{N\overline{v}_R^2}{E(v_R^2)}$ is chi-squared. The mean and variance of $\chi^2(N)$ are

$$E[\chi^2(N)] = N \text{ and } Var[\chi^2(N)] = 2N$$

from which we have

$$E\left[\frac{N\overline{v}_R^2}{E(v_R^2)}\right] = N \text{ and } Var\left[\frac{N\overline{v}_R^2}{E(v_R^2)}\right] = 2N$$

It is immediately evident from the left-hand equation that $E[\overline{v}_R^2] = E[v_R^2]$, that is, estimate of the mean-squared voltage is unbiased. Using the property of random variables that $Var[ax] = a^2 Var(x\}$, we have

$$Var[\overline{v}_R^2] = 2\frac{\left(E[v_R^2]\right)^2}{N}$$

The variance, $Var[\overline{v}_R^2]$ of the estimate $\overline{v}_R^2$, the mean-squared noise voltage can be used to estimate the standard deviation of the estimate of the mean-squared voltage, $\sigma_{\overline{v}_R^2}$. The standard deviation of the estimate of the mean-squared voltage is

$$\sigma_{\overline{v}_R^2} = \sqrt{Var[\overline{v}_R^2]} = \sqrt{2\frac{\left(E[v_R^2]\right)^2}{N}} = \sqrt{\frac{2}{N}}E[v_R^2]$$

and, dividing both sides by the estimate of the mean-squared voltage,

$$\frac{\sigma_{\overline{v}_R^2}}{E[v_R^2]} = \frac{\sqrt{\frac{2}{N}}E[v_R^2]}{E[\overline{v}_R^2]} \cong \sqrt{\frac{2}{N}} \qquad \text{------------------------} \qquad (3)$$

(The approximation is good for large values of *N*.) This final result simply says that (for large *N*) the fractional standard deviation of the estimate of the mean-squared, open-circuit noise voltage is simply $\sqrt{\frac{2}{N}}$ where *N* is the number of independent samples from which the mean-squared, open-circuit noise voltage is estimated.

In the following sections, we prove that the above are indeed true for experimental data sets taken from a very simple electric circuit and also for the random signals are of the form $\sum_\alpha a_\alpha Sin(2\pi f_\alpha t + \omega_\alpha)$ with a_α being a random number in [0,1], f_α a random integer in $[3\times 2^{15}, 1.2\times 2^{20}]$, and w_α a random number in $[0, 2\pi]$.

Chapter 2. Determination of the Sampling Time

In this chapter, we describe how an optimal sampling rate is determined for the bandpass filtered Johnson noise in a fixed frequency band. The Johnson noise is filtered so that the low frequency noise such as the AC noise and the high frequency noise such as the EMI noise are eliminated. To study the relation between the frequency band and the corresponding optimal sampling time, we used a random signal obtained by summing single frequency signals whose amplitudes, frequencies and the phase angles are random. We start with proving that such a random signal is a Gaussian white noise.

1. A Gaussian White Noise Generator

In this section, we show that the bandpass filtered signals of the form $\sum_{\alpha} a_{\alpha} \mathrm{Sin}(2\pi f_{\alpha} t + b_{\alpha})$ where a_{α} being a random number in [0,1], f_{α} a random integer in a given frequency band, and b_{α} a random number in $[0,2\pi]$, generate Gaussian white noise signals and hence they are adequate for simulating continuous Markov processes. We apply the results to the fluctuation-dissipation formula for the Johnson noise and show that the probability distribution for the long term average of the power of the Johnson noise is a χ^2 distribution and that the relative error of the long-term average is $\sqrt{2/N}$ where N is the number of blocks used in the average.

The fluctuation-dissipation formula is given by

$$V(t) = (2kTR)^{\frac{1}{2}} \Gamma(t) \qquad \text{-------------------------------------} \qquad (1)$$

where V(t) is the fluctuating Johnson electromotive force, T is the absolute temperature, R is the resistance, and $\Gamma(t)$ is the Gaussian white noise. It can be derived from the Ornstein-Uhlenbeck Langevin equation [10]

$$-RI(t) + V(t) - L\frac{dI(t)}{dt} = 0 \qquad \text{----------------------------} \qquad (2)$$

The general form of the Langevin equation is used not only for the Johnson noise, but also for the general continuous Markov processes including the Brownian motion.

It is well known that the fluctuation-dissipation formula (1) can be used to determine a highly accurate temperature (2) be a statistical analysis of the Gaussian white noise $\Gamma(t)$. There are various Monte-Carlo simulation codes [11] including the MCNP4B code [12] and the EGS4 code [13] to simulate the behavior of electrons but they do not generate time sequence outputs. By using these codes, one can compute the sums of energy deposits and hence the number of electrons produced while they pass through the material but not the time sequence outputs from a bulk of input particles passing through the material simultaneously.

We show in the following how to generate a time sequence of a Gaussian white noise and describe how these sequences can be used to satisfy the statistical properties of the Johnson noise which is the expectation value of $V^2(t)$ in the relation (1).

A random signal is a Gaussian white noise by definition if it is normally distributed with a zero mean and if the samples at two different times are uncorrelated. In practical applications, the signal is given as a finite number of discrete sampled points and hence its Fourier transform must be finite, i.e. only a finite number of points are used. Note that if K is the number of sampled points, then there can be at most $K/2$ frequencies with nonzero amplitudes by the Nyquist theorem. Thus, the Gaussian white noise that we are concerned here must be in a frequency band. A typical example of the frequency band is 3×2^{15} cycles/sec to 1.2×2^{20} cycles/sec.

We will show in the following that the random signal $\Gamma(t) = \sum_{k=1}^{N} a_k \mathrm{Sin}(2\pi f_k t + b_k)$ where a_k being a random number in [0,1], f_k a random integer in $[f_L, f_U]$ for some positive integers f_L, f_U, and b_k a random number in $[0, 2\pi]$, generates a Gaussian white noise, i.e. it satisfies $E(\Gamma(t)) = 0$ and $E(\Gamma(t)\Gamma(t')) = \delta(t - t')$.

Let $\Omega = \{\Gamma(t_k \mid k = 1,2,...M\}$ where $t_k = k/M$ and let X_n be a random sample of the N points from Ω. Then we have the following.

Lemma 1. Let X_n be as defined above, then the random variable $v = \sum_{k=1}^{N} X_k$ is normally distributed.

Proof. Let μ and σ be the mean and the standard deviation of Ω. Then from the definition of Ω, it is clear that μ and σ are finite. Since X_n 's are samples of a fixed size N from Ω, the central limit theorem can be applied to conclude that v is normally distributed. Q.E.D.

Lemma 2. Let X be a random variable with $X = r\mathrm{Sin}\theta$ where $r \in [0,1]$, $\theta \in [0, 2\pi]$ are random numbers, i.e. both r and θ are uniformly distributed. The the expectation value of X is zero.

Proof. Let $r_n = n\Delta r$, $\theta_n = n\Delta\theta$, $X_n = r_n \mathrm{Sin}\theta_n$ with $\Delta r = 1/N$ and $\Delta\theta = 2\pi/N$ and let $A_n = \{(r,\theta) \mid r_n < r \le r_n + \Delta r, \theta_n < \theta \le \theta_n + \Delta\theta\}$. Then we have $p((r,\theta) \in A_n) = p(r_n < r \le r_n + \Delta r) \times p(\theta_n < \theta \le \theta_n + \Delta\theta) = \Delta r \times \frac{\Delta\theta}{2\pi}$. Thus, we have $E(X) = \mathrm{Lim}_{N\to\infty} \sum_{n=1}^{N} p((r,\theta) \in A_n) X_n$ which is equal to $\mathrm{Lim}_{N\to\infty} \sum_{n=1}^{N} r_n \mathrm{Sin}\left(\theta_n \frac{\Delta r \Delta\theta}{2\pi}\right) = \frac{1}{2\pi} \int_0^1 \int_0^{2\pi} r \mathrm{Sin}\theta \, dr d\theta$ =0. Q.E.D.

Lemma 3. If a_k and b_k are random numbers in $[0,1]$ and $[0,2\pi]$ respectively, then for any $\varepsilon > 0$, we have $\lim_{N\to\infty} p\left(\left|\frac{1}{N}\sum_{k=1}^{N} a_k \mathrm{Sin}(b_k)\right| > \varepsilon\right) = 0$.

Proof. Let $X = a\mathrm{Sin}(b)$ and $X_k = a_k \mathrm{Sin}(b_k)$ for k=1,2,…N. Then by Lemma 2, we have E(X)=0 and hence using the weak law of large numbers, we have the $p\left(\left|\frac{1}{N}\sum_{k=1}^{N} a_k \mathrm{Sin}(b_k)\right| > \varepsilon\right)$ approaches to 0. Q.E.D.

Lemma 4. If f is an integer, $t_k = kh$ with $h = \frac{1}{M}$, then we have $\sum_{k=0}^{M-1} \mathrm{Sin}(2\pi f t_k + b) = \mathrm{Sin}(b)$.

Proof. Let $z_0 = e^{ib}$, $w = e^{i2\pi fh}$, and $z_k = e^{i(2\pi f t_k)}$. Then we have $z_k = z_0 w^k$ and hence $\sum_{k=0}^{M-1} \mathrm{Sin}(2\pi f t_k + b) = \mathrm{Im}\left(\sum_{k=0}^{M-1} z_k\right)$ which is equal to $\mathrm{Im}\left(z_0 \sum_{k=0}^{M-1} w^k\right) = \mathrm{Im}\left(\frac{z_0(1-w^M)}{1-w}\right)$. Now, note that $w^M = e^{12\pi fhM} = e^{i2\pi f} = w$ and hence the sum equals $\mathrm{Im}(z_0) = \mathrm{Sin}(b)$. Q.E.D.

Theorem 1. Let $\Gamma(t)$ be as defined above and let $t_k = \frac{k}{M}$. Then for any $\varepsilon > 0$, we have

$\lim_{N\to\infty} p\left(\left|\frac{1}{N}\sum_{k=0}^{N-1} \Gamma(t_k)\right| > \varepsilon\right) = 0$ and hence $E(\Gamma(t)) = 0$.

Proof. Recall that we have $\sum_{k=0}^{N-1} \Gamma(t_k) = \sum_{k=0}^{N-1} \left\{\sum_{j=1}^{M} a_j \mathrm{Sin}(2\pi f_j t_k + b_j)\right\}$ and the sum is equal to

$\sum_{j=1}^{M}\left\{\sum_{k=0}^{N-1}\mathrm{Sin}(2\pi f_j t_k + b_j)\right\}$. Now, by Lemma 4, we have $\sum_{k=0}^{N-1}\mathrm{Sin}(2\pi f_j t_k + b_j) = \mathrm{Sin}(b)$ and hence $\sum_{k=0}^{N-1}\Gamma(t_k) = \sum_{j=1}^{M}\mathrm{Sin}(b_j)$. Now, the first part of the theorem follows from Lemma 3 and the second part is trivial. Q.E.D.

To verify the above theorem, we computed a random signal $\Gamma(t)$, which is a sum of a 2^{11} single frequency signals for 2^{18} blocks of 1,024 points. When the mean and standard deviations are computed for 2^6, 2^{10}, 2^{14}, and 2^{18} blocks, we obtain the results shown in Table 2-1. Fig.2-1 and fig.2-2 show how the signal values change as the number of blocks increases with the averages.

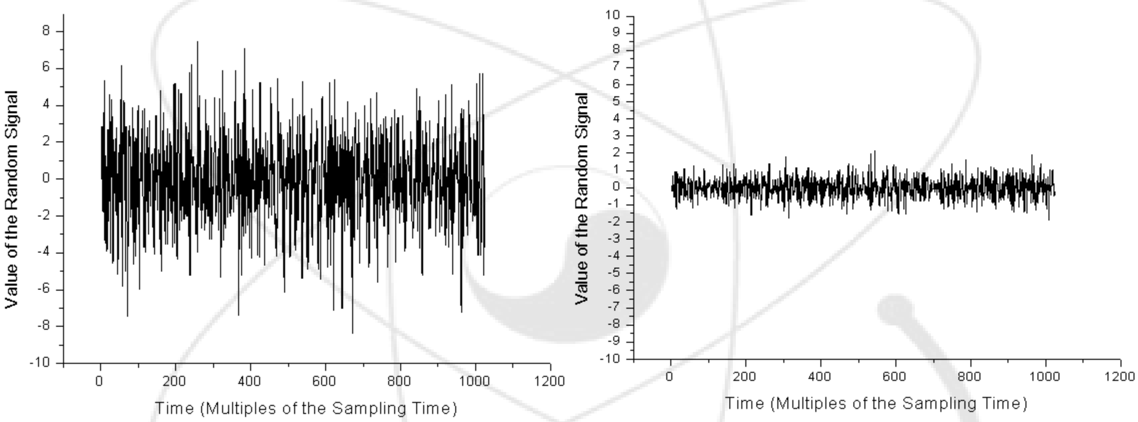

fig.2-1. Sample random Signal (Avg. of 2^6 and 2^{10} blocks)

Next, we consider the correlation coefficient between v_N and w_N where v_N and w_N are samples of size N taken at different times. Recall that the temperature is assumed to be constant so that the squared lengths of the vectors v_N and w_N must be the same and hence if the correlation coefficient approaches zero, then so will the cross product sum.

Table 2-1. Mean and Sigma vs Number of Blocks Used (Sum of 2^{11} Frequencies)

No of Blocks	Mean	Sigma	Ratio of Mean	Ratio of Sigma
64	0.004266	2.407210	1.00000	1.00000
1,024	0.001196	0.597664	0.28036	0.24828
16,384	0.000362	0.141919	0.08486	0.05896
262,144	-0.000014	0.003166	-0.00328	0.00132

Lemma 5. Let N=2^n for some positive integer $n \geq 3$, $h = \frac{1}{N}$, $t_k = kh$ and let f be a random integer in [1, 2^{n-1}]. Then we have $\frac{1}{N}\sum_{k=1}^{N} \mathrm{Cos}(4\pi f t_k) = 0$ and $\frac{1}{N}\sum_{k=1}^{N} \mathrm{Sin}(4\pi f t_k) = 0$.

Proof. First, consider the case of f=1, where we have $\theta_k = \frac{4\pi k}{2^n}$ for k=1,2,…,2^{n-1}. These θ_k's can be identified as points on the unit circle in equal angular intervals and they will fill the whole circle. Therefore, we must have $\sum_{k=1}^{2^{n-1}} \mathrm{Cos}(4\pi f t_k) = 0$ and hence we have $\sum_{k=1}^{2^n} \mathrm{Cos}(4\pi f t_k) = 0$. Next, we consider the case of $f \neq 1$. Note that if f is an odd integer, then $\theta_k = \frac{4\pi f k}{2^n}$, k=1,2,…,$2^{n-1}$ covers all of the points corresponding to $\frac{4\pi k}{2^n}$, k=1,2,…, 2^{n-1}. In general, we may write $f = 2^m f_1$ where f_1 is an odd integer. Note that we have $m \leq n-2$, and hence the same holds in this case also. An exactly the same proof for $\frac{1}{N}\sum_{k=1}^{N} \mathrm{Sin}(4\pi f t_k) = 0$ is omitted. Q.E.D.

Theorem 2. Let a_j, f_j, and b_j be random numbers in the intervals defined above and let $\Gamma(t) = \sum_{j=1}^{M} a_j \mathrm{Sin}(4\pi f_j t + b_j)$. If $t' = t + \tau$ for some $\tau > 0$, then $\lim_{N\to\infty} p\left(\frac{1}{N}\sum_{k=1}^{N} \Gamma(t_k)\Gamma(t_k + \tau) > \varepsilon\right)$=0.

Proof. Let $c_j = 2\pi f_j \tau + b_j$, then we have $\Gamma(t+\tau) = \frac{1}{N}\sum_{j=1}^{N} a_j \mathrm{Sin}(2\pi f_j t + c_j)$. By exchanging the order of sums, the sum $\frac{1}{N}\sum_{k=1}^{N} \Gamma(t_k)\Gamma(t_k + \tau)$ becomes

$$\sum_{i,j=1}^{M} a_i a_j \left\{ \frac{1}{N}\sum_{k=1}^{N} \mathrm{Sin}(2\pi f_i t_k + b_i)\mathrm{Sin}(2\pi f_j t_k + c_j) \right\}$$

Now, the product of the two sine functions inside the sum can be expanded to obtain the following four terms;

(1) $\mathrm{Cos}(b_i)\mathrm{Cos}(c_j)\frac{1}{N}\sum_{k=1}^{N} \mathrm{Sin}(2\pi f_i t_k)\mathrm{Sin}(2\pi f_j t_k)$,

(2) $\mathrm{Cos}(b_i)\mathrm{Sin}(c_j)\frac{1}{N}\sum_{k=1}^{N} \mathrm{Sin}(2\pi f_i t_k)\mathrm{Cos}(2\pi f_j t_k)$,

(3) $\mathrm{Sin}(b_i)\mathrm{Cos}(c_j)\frac{1}{N}\sum_{k=1}^{N} \mathrm{Cos}(2\pi f_i t_k)\mathrm{Sin}(2\pi f_j t_k)$,

(4) $\mathrm{Sin}(b_i)\mathrm{Sin}(c_j)\frac{1}{N}\sum_{k=1}^{N}\mathrm{Cos}(2\pi f_i t_k)\mathrm{Cos}(2\pi f_j t_k)$

Note that as $N \to \infty$, the sum in (1) approaches to $\int_0^1 \{\mathrm{Cos}(2\pi(f_i - f_j)t) - \mathrm{Cos}(2\pi(f_i + f_j)t)\}dt$ which is zero unless i=j. The same holds for all of the other three sums. Thus, the first term becomes $\sum_{i=1}^{M} a_i^2 \mathrm{Cos}(b_i)\mathrm{Cos}(c_i)\frac{1}{N}\sum_{k=1}^{N}\mathrm{Sin}^2(2\pi f_i t_k) = \sum_{i=1}^{M} a_i^2 \mathrm{Cos}(b_i)\mathrm{Cos}(c_i)\frac{1}{2N}\sum_{k=1}^{N}(1 - \mathrm{Cos}(4\pi f_i t_k))$. Now, the sum of the Cosine terms becomes zero by Lemma 5, and hence the sum becomes $\frac{1}{2}\sum_{i=1}^{M} a_i^2 \mathrm{Cos}(b_i)\mathrm{Cos}(c_i)$. Applying the same procedure, one can easily verify that the sums (2) and (3) become zero, and the fourth term becomes $\frac{1}{2}\sum_{i=1}^{M} a_i^2 \mathrm{Sin}(b_i)\mathrm{Sin}(c_i)$. Therefore, by adding the two nonzero terms and using the relation $c_i = b_i + 2\pi f_i \tau$, we obtain $\frac{1}{2}\sum_{i=1}^{M} a_i^2 \mathrm{Cos}(2\pi f_i \tau)$. Finally, one can prove that $\lim_{N\to\infty} p\left(\left|\frac{1}{N}\sum_{i=1}^{N} a_i^2 \mathrm{Cos}(2\pi f_i \tau)\right| > \varepsilon\right) = 0$ by applying an argument similar to the one in the proof of Lemma 3. Q.E.D.

Sample calculation results are shown in Table 2-2 and in fig.2-2.

Table 2-2. Correlation Between Different Sample Blocks (Sum of 2^{11} Single Freq. Signals)

No of Bloc	ρ	Relative ρ	$\sqrt{64/N}$
64	0.02838	1.0000	1.0000
4,096	0.00045	0.0159	0.1250
262,144	0.00004	0.0015	0.0156

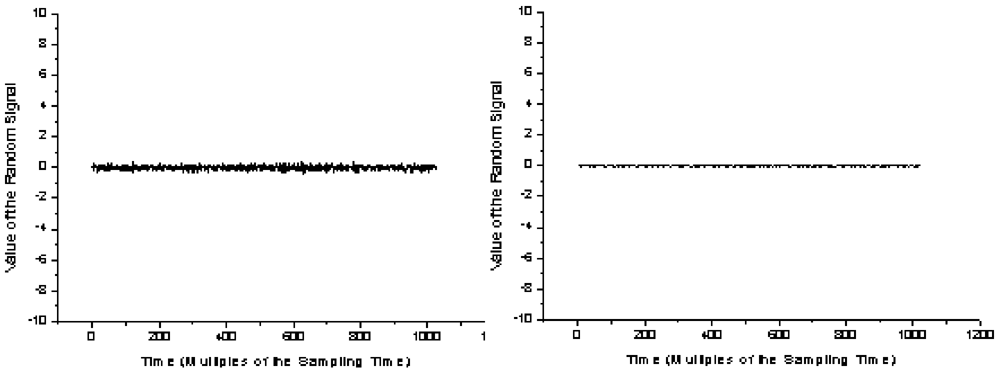

fig.2-2. Sample Random Signal (Avg. of 2^{14} and 2^{18} Blocks)

2. Statistical Analysis on the Gaussian White Noise Signal

In this section, we assume that the temperature T is constant and will consider N blocks of samples each consisting of the K sample points. Let $v_n = (v_{n1}, v_{n2}, ..., v_{nK})$ be the nth block of points sampled from the Johnson noise and let

$$w^2 = \frac{1}{N}\sum_{n=1}^{N} v_n^2 \qquad \text{-------------------------------------} \qquad (3)$$

Note that we have $E[v_n] = 0$ for all n=1,2,…,N since v_n is sampled from a Gaussian white noise and that $E[v_n^2] = E[v_m^2]$ for all n, m. This follows from the assumption that the temperature is constant.

Lemma 6. Let μ be the expectation value of v_n^2, i.e. $E[v_n^2] = \mu$, then the variance of v_n is $\frac{K\mu}{K-1}$ and hence $\sigma(v_n) = \sqrt{\frac{K\mu}{K-1}}$.

Proof. Note that $\mathrm{Var}(v_n) = \frac{1}{K-1}\sum_{k=1}^{K}(v_{nk} - E[v_n])^2 = \frac{1}{K-1}\sum_{k=1}^{K} v_{nk}^2$ which is equal to $\frac{K}{K-1}E[v_n^2] = \frac{K}{K-1}\mu$. Q.E.D.

Lemma 7. Let $\psi = \frac{1}{\nu}\sum_{n=1}^{N} v_n^2$ with $\nu^2 = \mathrm{Var}(v_n)$, then the probability distribution of ψ is Chi-square and $\psi = \frac{(K-1)Nw^2}{K\mu}$.

Proof. Recall that we have $E[v_n] = E[v_m]$ for all n, m and hence we have $\mathrm{Var}(v_n) = \mathrm{Var}(v_m)$ which follows from $\mathrm{Var}(v_n) = \frac{1}{K-1}\sum_{k=1}^{M}(v_n - 0)^2$. Since v_n's are normally distributed and ψ is a sum of the squares of the standard normally distributed random variables, the probability distribution of ψ must be a Chi-square. Note that $w^2 = \frac{1}{N}\sum_{n=1}^{N} v_n^2$ can be written as $\frac{\nu}{N}\psi$ and hence we have $\psi = \frac{N}{\nu} w^2$ from which the result follows using Lemma 5. Q.E.D.

Theorem 3. Let N be a fixed positive integer and let w^2 be as defined above. Then we have $\frac{\sigma(w^2)}{E(w^2)} = \sqrt{\frac{2K}{N(K-1)}}$, an approximation of which is $\sqrt{\frac{2}{N}}$ for large K's.

Proof. First, note that $E(\psi) = \frac{(K-1)N}{K\mu} E[w^2] = \frac{(K-1)N}{K}$. Since ψ is a Chi-square distribution, we have $\mathrm{Var}(\psi) = 2E[\psi]$. Now, using the property that $\mathrm{Var}(au) = a^2 \mathrm{Var}(u)$ for any constant a, we have $\mathrm{Var}(\psi) = \left(\frac{(K-1)N}{K\mu}\right)^2 \mathrm{Var}(w^2)$ which must be equal to $\frac{2(K-1)N}{K}$. Therefore, we have $\mathrm{Var}(w^2) = \frac{2K}{N(K-1)}\mu^2$ from which we obtain $\frac{\mathrm{Var}(w^2)}{(E[w^2])^2} = \frac{2K}{N(K-1)}$. Q.E.D.

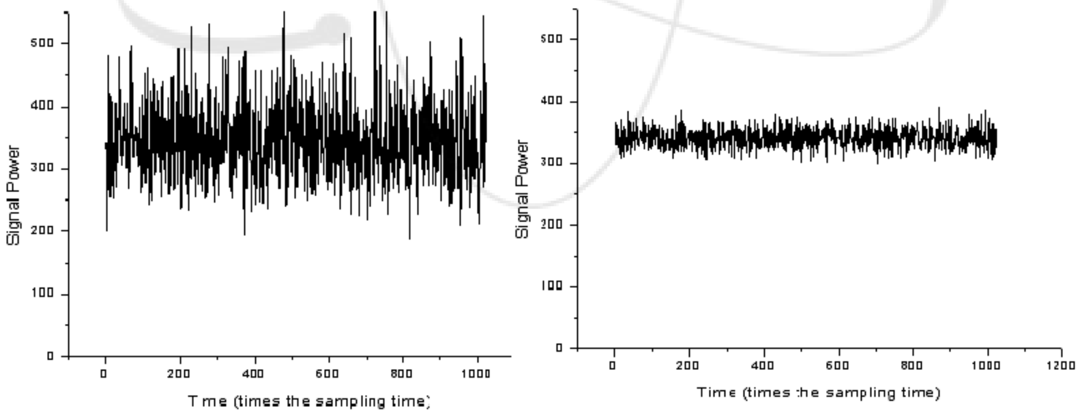

Fig.2-3. Signal Power of Sample Random Signals (Avg. of 2^6 and 2^{10} Blocks)

To verify that the above is true for the bandpass filtered random signals, we computed the signals, each of which is a sum of the 2^{11} random single frequency signals for $N=2^8, 2^{10}, 2^{12}, 2^{14}$. A summary of the results is shown in Table 2-3 and Table 2-4. The column with 'Relative Error' shows the ratio of the sigma over the mean and it can be seen that as the number of blocks increases by a factor of 16, the relative error decreases by a factor of 4 which is what is expected from Theorem 2. Table 2-3 shows the case of the sums of 2^{12} single frequency random signals.

Table 2-3. Relative Error vs Number of Blocks Used for Average
(Sum of 2^{11} Single Freq. Signals)

No of Blocks	Mean	Sigma	Relative Error
64	340.802	58.702	0.17225
1,024	341.129	15.005	0.04399
16,384	341.309	3.7120	0.01088
262,144	341.298	0.7850	0.00230

Table 2-4. Relative Error vs Number of Blocks Used for Average
(Sum of 2^{12} Single Freq. Signals)

No of Blocks	Mean	Sigma	Relative Error
64	685.194	124.93	0.18233
1,024	682.232	30.456	0.04664
16,384	682.104	7.2090	0.01057
262,144	682.708	1.7760	0.00260

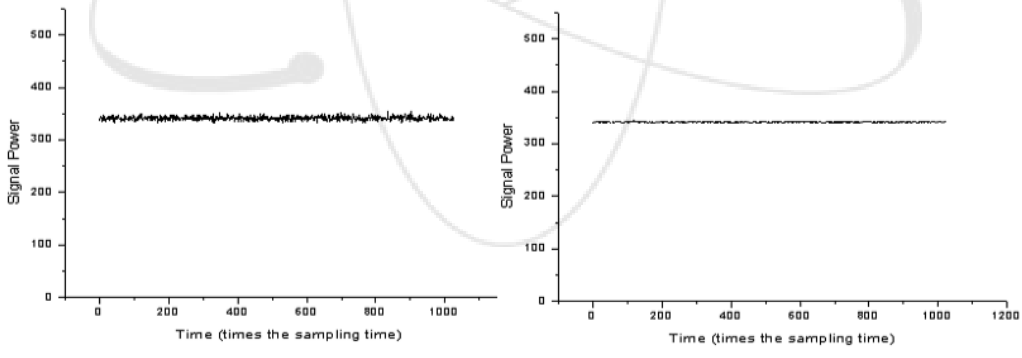

fig.2-4. Signal Power of Sample Random Signal (Avg. of 2^{14} and 2^{15} Blocks)

3. Sampling Time and the Frequency Band

In this section, we describe how we determine an optimal sampling rate when a bandpass filtered random noise is given. We assume that the signal blocks are of size 1,024 points and this will be the size we will be using in our digital signal processing algorithm. Note that we need to use the Fourier transformation of the signal blocks so that not only the EMI noises are eliminated but also the power spectral density of the signal needs can be computed. Recall that a constant multiple of an average of the power spectral density will be used as our temperature.

First, we consider how the frequency band for filtering the noise data and the necessary sampling time are determined. Ideally, the thermometer should have as wide a bandwidth as possible not only to keep the measurement times short but also to increase the accuracy. If we take a shorter sampling time, then we will have more accurate temperature during the same time interval since our temperature value will be computed by taking the average of long-term fluctuating values. But there are practical limitations. The upper bound of the frequency range is required to eliminate the high frequency EMI associated with AM radio transmissions. The lower bound is set to reduce the EMI due to low frequency magnetic fields, particularly associated with the power supplies. A survey by D. white et al. [8] shows that the typical frequency range used is from a few kilohertz to one megahertz. The size of the frequency band and the number of points in the sampling block determine the resolution of the Fourier transformation.

We assume that the number of sample points in a block of samples is 1024 points. This limitation comes from the fact that the FFT algorithm we will be using for a set of 1024 points requires 6M gates while the FPGA board we designed initially has only 15M gates. First, we consider the case where the sampling time is 16MHz, i.e. $h = \frac{1}{2^{24}}$ sec, which is practically the fastest rate currently, considering the fact that 12 Bit A/D conversion must be used. Note that there must be more than two points, i.e. three or more points in one period of a signal so that the amplitude of the signal after the Fourier transformation is properly reflected in the frequency domain. Thus, if we require four points to be the minimum, then the signal with $4h = \frac{1}{2^{22}}$ sec as one cycle time will have the minimum period and hence the maximum frequency of 2^{22} (4MHz). In a set of 1024 sample data, there will be 256 cycles for these signals of frequency 2^{22} (4MHz). Therefore, the amplitudes after FFT of frequencies corresponding to 257 through 512 are wasted. For the lower limit, note that input signals of frequency 2^{17} (128kHz) will be transformed to 8 since there are 8 cycles of such signals in 1024h second time period.

However, the signals we get from the preamplifiers fabricated by Oak ridge National Laboratory initially have frequencies below 1.5MHz and the amplitudes of the frequencies in the range 100kHz to 1.2MHz can be read properly after nonlinear gain correction. Thus, we decided to lower the sampling period to $h = \frac{1}{3\times 2^{20}}$ sec. Note that for signals of frequency 1.2MHz, one period will be $\frac{1}{1.2\times 2^{20}}$ sec, which is 2.5h. Therefore, two or more points will be sampled from each cycle. Now, consider signals of frequency 3kHz=3×2^{10} cycles/sec whose period will be $\frac{1}{3\times 2^{10}}$ sec. One block of 1024 sample points will correspond to a time interval of length 2^{10} h=$\frac{1}{3\times 2^{10}}$ sec, which is the same as the one cycle length of 3kHz signals, and hence 3kHz signals will be transformed to 1 by the Fourier transformation.

Thus, if we take 3×2^5 kHz =96kHz be the lower bound of the filter, then the lower bound 96kHz will be transformed to 32 while the upper bound 1.2MHz will be transformed to about 410. Therefore, we will be using the frequency range [32, 410]. Fig.2-5 shows the power spectral density of a sample Johnson noise signal filtered in the range $[3\times 2^{15}, 1.2\times 2^{20}]$. Note that the values of the spectral density near 32 and 410 are smaller than those at interior points. In the next section, we consider the energy leakage problem at the boundary frequencies 32 and 410.

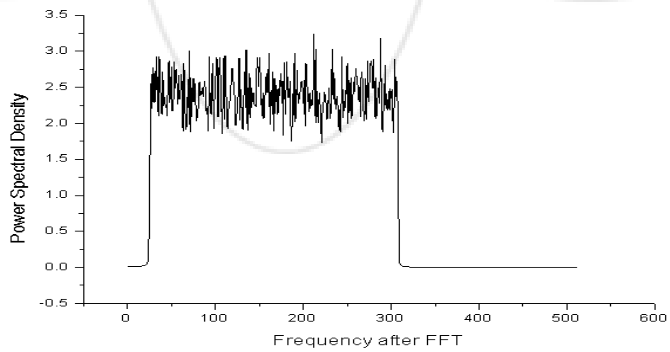

fig.2-5. Power Spectral Density of a Sample Johnson Noise

4. Energy Leakage Problem at Boundary Frequencies

Since the signals are windowed by a frequency band, the power of the original sequence $\{x(i)\}$ concentrated at a single frequency will spread by the window into the entire frequency range, which is called energy leakage. The left hand side graph of fig.2-6 shows an example of the Fourier transform of $\{x(i) = \mathrm{Sin}(2\pi f t_i) | i = 1,2,...1024\}$ with $f = 3\times 2^{16} - 3\times 2^{9}$ and $t_i = \frac{i-1}{3\times 2^{20}}$. Recall that the signals with frequency 3×2^{10} will be transformed to 1 and hence signals with frequency 3×2^{16} will be transformed to 64 and the Fourier transformation of the signal with single frequency $3\times 2^{16} - 3\times 2^{9}$ will have the mean at 63.5 as shown in fig.2-6.

As described earlier, the temperature will be computed by a constant multiple of the average of the cross power density spectrum in the frequency domain. Since only the frequencies in a fixed range of frequencies will pass through the band pass filter described above and the Fourier transformation of the single frequency signals in the band will mostly spread, there will be "energy leakages". To determine how much of the power will be "leaked out" at the boundary frequencies, we computed the following:

(1) Take a signal $f(t) = \mathrm{Sin}(2\pi\omega t)$ with the frequency ω in the interval $[3\times 2^{15}, 1.2\times 2^{20}]$ and Compute $f_i = f(t_i), i = 1,2,...1024$ with $t_i = \frac{i}{3\times 2^{20}}$

(2) Compute the Fourier transformation of $\{f_i | i = 1,2,...1024\}$ to obtain $\{g_i | i = 1,2,...1024\}$ where g_i are the magnitude of the complex numbers

(3) Considering $\{g_i^2 | i = 1,2,...1024\}$ as a probability distribution, compute the mean and sigma of this distribution

(4) Repeat the above process for all the frequencies ranging from 3×2^{15} to 1.2×2^{20} and plot how the means are distributed.

The result of calculation is shown in the right hand side graph of fig.2-6, where the sigma values ranges from 0.3 to 1.0 with the mean at 0.669. Note that we may take 0.669+3$\times$0.234=1.339 as the maximum span of the "leakage" and hence if any of the frequencies in the range [34,408] will not have any 'leakage'. Note that the leaked out energy at 34 will be at 33 or below and hence the average of the power spectral density at interior points will be proportional to the temperature. This fact will be reflected in our digital signal processing algorithm.

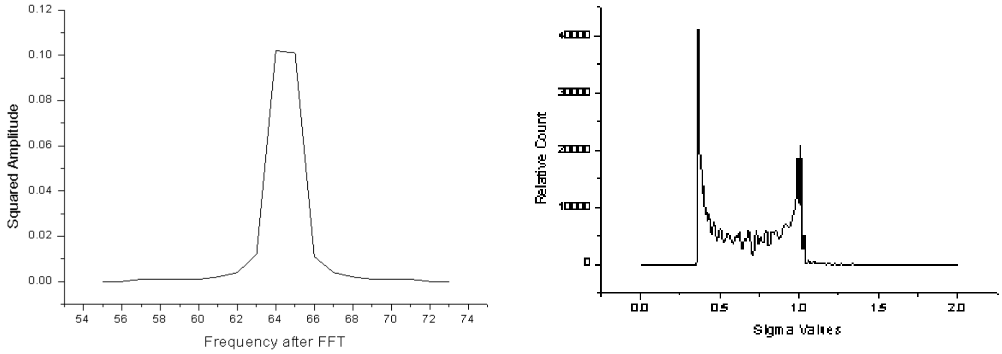

fig.2-6. Transformed Freq. Distribution for a Single Freq. Input (Left) and Sigma Values for Transformed Single Frequency Signals (Right)

Chapter 3. Linearity of the Noise Thermometry

In this chapter, we will show that the Johnson noise and the channel noise picked up in the long signal cables are uncorrelated, i.e. the correlation coefficient approaches to zero as the number of blocks used in the average increases. We also show that any pair of blocks of Johnson noise samples are uncorrelated, and hence the temperature or equivalently, the average of the power spectral density is linear. Let v_1 and v_2 be the Johnson noise signals taken at the same temperature T and at different times. Suppose that we have $v_1 + v_2$ as a Johnson noise signal at some other time, then we would expect that the signal corresponds to the temperature 2T. This follows from

$$E[(v_1 + v_2)^2] = E[v_1^2] + E[v_2^2] + 2E[v_1 v_2] \approx E[v_1^2] + E[v_2^2]$$

1. Correlation between the Channel Noise and the Johnson Noise

In this section, we consider the effect of the noise picked up in the long channel cables from the amplifier to the A/D converter where the temperature is calculated. As described in Chapter 2, if we let the signals from the two channels be $\phi = \{\phi_i \mid i = 1,2,...N\}$ and $\varphi = \{\varphi_i \mid i = 1,2,...N\}$, and write $\phi_i = x_i + \varepsilon_{1i}$ and $\varphi_i = x_i + \varepsilon_{2i}$, where $x = \{x_i \mid i = 1,2,...N\}$ is the sensor signal without the channel noise and $\varepsilon_{1i}, \varepsilon_{2i}$ being the channel noise signals, then the expectation value of the product of two signals ϕ and φ can be written

$$E(\phi\varphi) = E((x + \varepsilon_1)(x + \varepsilon_2)) = E(x^2) + E(x\varepsilon_2) + E(\varepsilon_1 x) + E(\varepsilon_1 \varepsilon_2) \quad \text{------------------------} \quad (4)$$

In the following, we will examine how large are the values $E(x\varepsilon_2)$, $E(\varepsilon_1 x)$, and show that $E(\varepsilon_1 \varepsilon_2)$ approach to zero faster than the rate $\sqrt{\frac{2}{N}}$ which is the relative error for $E(x^2)$. We will assume that the channel noises are of the same form as the Johnson noise except that the channel noise is not filtered and that they are sums of much less number of single random frequency signals. For the signal x, we used signals of the form $x(t) = \sum a \times \mathrm{Sin}(2\pi ft + b)$ with the amplitude a being a random number in [0,1], the phase angle b also a random number in $[0, 2\pi]$, and the frequency f is a random integer in $[3 \times 2^{15}, 1.2 \times 2^{20}]$. For the sampling time, we use $h = \frac{1}{3 \times 2^{20}}$ so that $x_i = f\left(\frac{i}{3 \times 2^{20}}\right)$, for i=1,2,…,1024. As described earlier in Chapter 2, the number of terms in the sum of f(t) determines the temperature so that smaller number of terms will correspond to a lower temperature. The left hand side graph of fig.3-1 shows the power of a signal

$\{x_i \mid i = 1,2,...,1024\}$ obtained from sums of 4096 random single frequency signals after taking the average of 64 blocks. For the channel noise signals $\{y_i \mid i = 1,2,...,1024\}$, we used the similar function f(t) with the number of added single frequency signals range from 1 to 64, and 1000 to 1200. The right hand side graph of fig.3-1 shows the power of a noise signal obtained by averaging 64 noise signals each with a single frequency, a sum of two single frequencies, a sum of three single frequencies, and so forth. When the correlation coefficient c between the two sequences $\{x_i\}$ and $\{y_i\}$ are computed,

$$c = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}$$

we find that c ranges from 0.028 to 0.038 while mean of the signal powers range from 675 to 686. When average of 4096 blocks (see fig.3-3) is taken, the correlation coefficient decreases to 0.00004 while the signal power remains to be the same. When the noise is a sum of 2^{15} random single frequency signals, the correlation coefficient is found to be around 0.0006 that is still far below the desired relative error for the signal power, i.e. $\sqrt{\frac{2}{4096}} = \frac{\sqrt{2}}{64} = 0.022$. Therefore, we may consider the random noise signals and the sensor signals are uncorrelated.

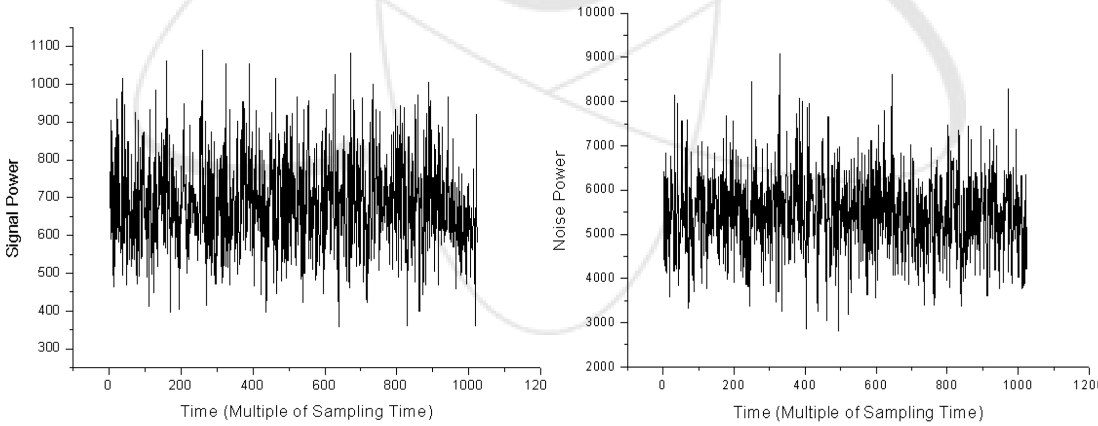

fig.3-1. Signal Power of a Sample Voltage Signal and Power of a Sample Noise Signal (Avg. of 64Blocks)

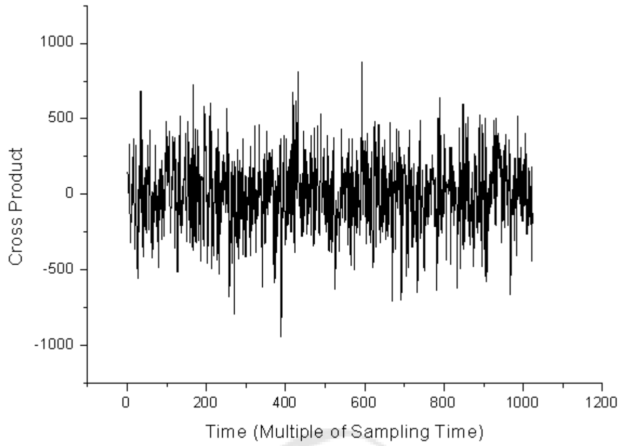

fig.3-2. Cross Product of a Sample Noise and the Sensor Signal
(Avg. of 64Blocks – Correlation Coefficient is 0.03)

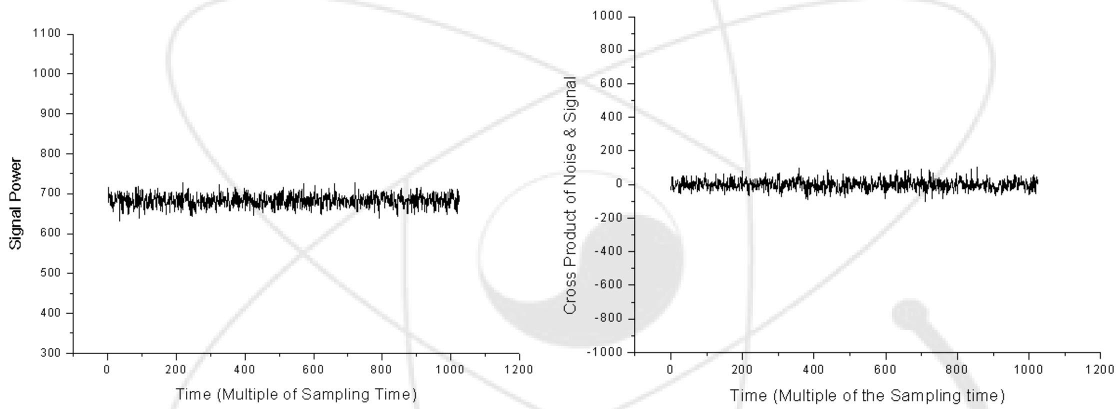

fig.3-3. Signal Power (left) and Cross Product of Noise and the Signal (right)
(Average of 4096Blocks – Corr. Coeff. is 0.0006)

2. Correlation Between Different Sets of Sensor Signals

In this section we consider the correlation coefficients between the different pairs of sensor signal blocks sampled at different times. We restrict our attention to only random signals corresponding to the same temperature, i.e. the sums of the same number of single frequency random signals. Let $x = \{x_i \mid i = 1,2,...N\}$ and $y = \{y_i \mid i = 1,2,...N\}$ be two different sets of sampled signals with N=1024. We would like to estimate $E[(x+y)^2] - (E[x^2] + E[y^2]) = 2E[xy]$ so that $E[(x+y)^2] \approx (E[x^2] + E[y^2])$. The graphs shown in fig.3-4, fig.3-5, fig.3-6 show the computed results for random signals obtained by taking sums of 2048 single frequency random signals.

As can be seen in the graphs, the correlation coefficient

$$\gamma = \frac{\sum_{i=1}^{M} x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}} = \frac{M \times E[xy]}{M \times \sqrt{E[x^2]E[y^2]}} = \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}} \approx \frac{E[xy]}{\sqrt{E[x^2]}}$$

gets reduced from 0.02838 for 64 block averages to 0.00045 for 64×64 block averages, and to 0.000043 for $64 \times 64 \times 64$ block averages. Note that the ratio of this reduction is smaller than the ratio of the square root of the number of blocks and hence we may consider that two different pairs of sample signal blocks are uncorrelated.

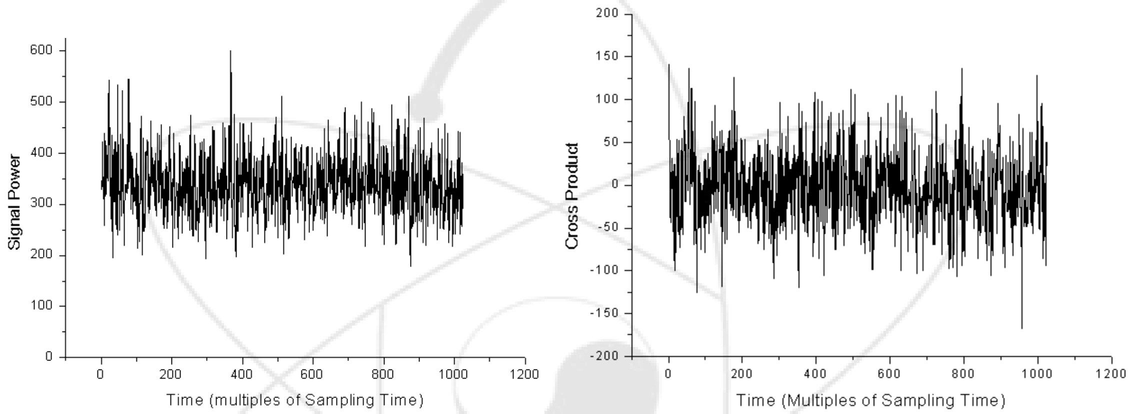

fig.3-4. Signal Power (left) and Cross Product of two Signals (right)

(Average of 64 Blocks – Corr. Coeff. is 0.02838)

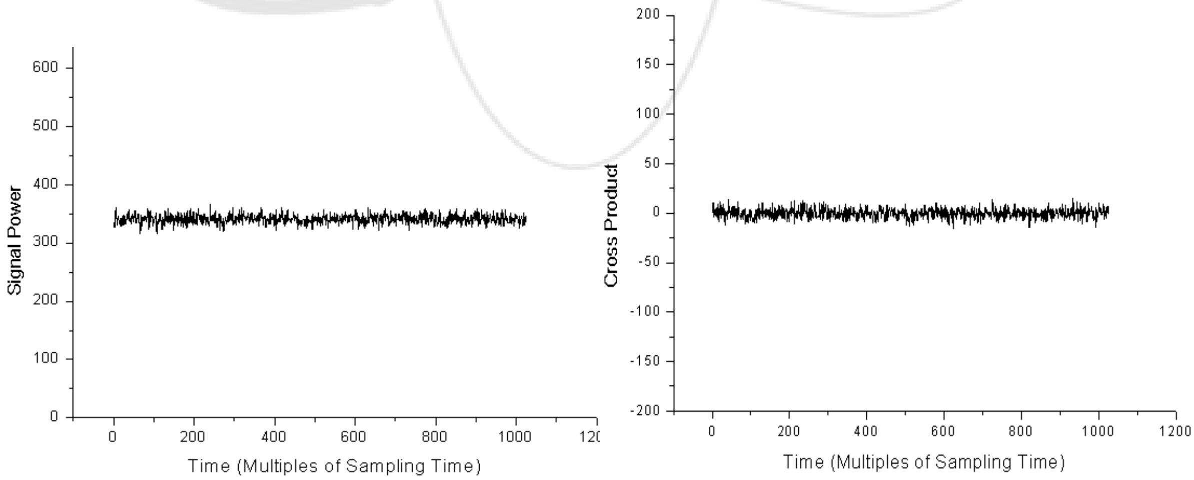

fig.3-5. Signal Power (left) and Cross Product of two Signals (right)

(Average of 4096 Blocks – Corr. Coeff. is 0.00045)

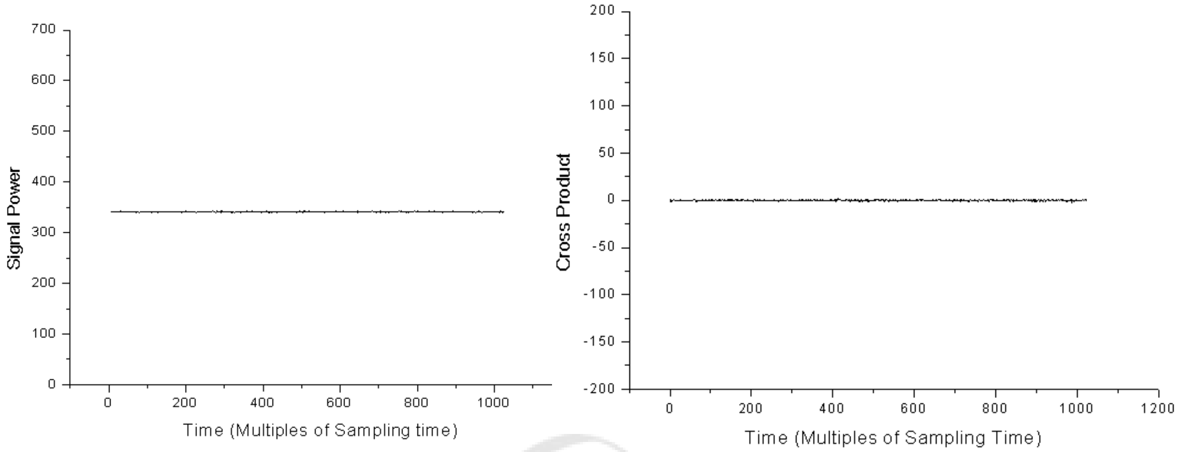

fig.3-6. Signal Power (left) and Cross Product of two Signals (right)
(Average of 262,144 Blocks – Corr. Coeff. is 0.00004324)

3. Linearity of the Processed Signal Power

In this section, we study how the signal power or equivalently the power spectral density increases as the number of random signals summed to form the Johnson noise increases. We performed calculations for the functions obtained by summing 2^{11}, $2^{12},\ldots,2^{18}$ single frequency random signals with the frequencies in the range $[3\times2^{15},1.2\times2^{20}]$. Table 1 shows a summary of the results where the first two rows are the means and the sigmas of the averaged signal power while the remaining two rows are for the power spectral density.

Table 3-1. Average of Signal Power and Power Spectrum Density

No of Signals	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}
Sigma(SP)	341.129	682.232	1365.417	2732.603	5468.062	10914.70	21800.38	43663.86
Sigma(SP)	15.005	30.456	58.799	122.948	233.322	445.624	889.649	1797.51
Mean(PSD)	0.45116	0.90241	1.80599	3.61388	7.23178	14.43632	28.83659	57.74675
Sigma(PSD)	0.01536	0.03250	0.06013	0.12422	0.22321	0.46009	0.95255	2.00390

The computed results in Table 3-1 are drawn by graphs shown in fig.3-7. The left hand side graph of fig.3-7 shows the means of the signal power in the middle curve that is almost a straight line. The two other curves surrounding the middle curve are drawn so that they reflect the one-sigma band. Similarly, the right hand side graph of fig.3-7 is drawn for the power spectral density in Table 3-1. Fig.3-8 is another schematic illustration of the linearity shown in Table 3-1 with the Gaussian functions center at the means and with the sigmas as the standard deviations.

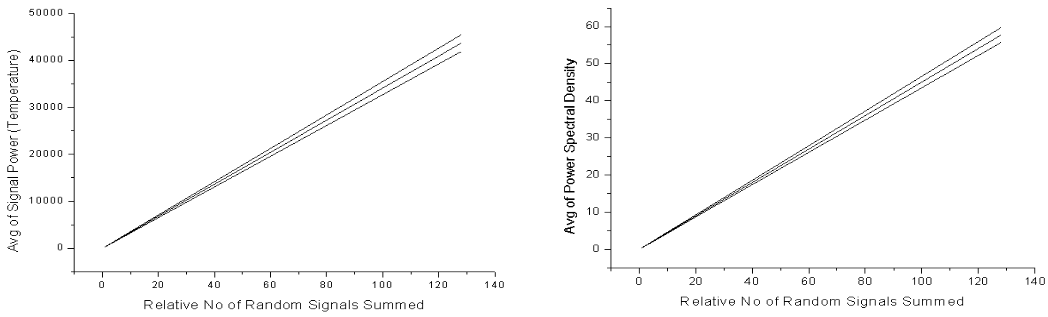

Fig.3-7. Linearity of Johnson Noise Signals with One Sigma Band -1

(Average of 1024 Blocks)

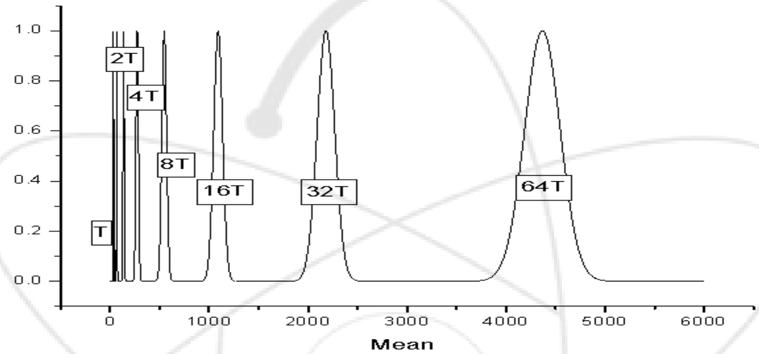

Fig.3-8. Linearity of Johnson Noise Signals with One Sigma Band –2

4. Analysis of Experimental Data Sets

In this section, we describe the results of our studies on the statistical properties of thermal noise, or equivalently of a continuous stochastic random process, using a series of sample data sets taken from an electric circuit. We describe the results of our analyses on how the theoretical error estimate in terms of the number of blocks used in taking the average compared with the one from the actual measured noise containing some channel noise and how accurate the Johnson noise thermometry can be with the given set of electronic equipments that are being used

Fig.3-9 shows an example of the power of the noise signal obtained by taking the average of 8 blocks and fig.3-10 shows its equivalent in the frequency domain. Similarly, fig.3-11 shows an example of 64 block averages and its Fourier transform is in shown in fig.3-12. The mean and sigma for the average of 8 blocks are 1.465 and 0.604 respectively so that the ratio is 0.410, while for the average of 64 blocks in fig.3; they are 1.402 and 0.202 with the ratio of 0.144. Hence, the relative error reduces from 41.0% to 14.4% as the number of blocks increases from 8 to 64. For the corresponding power spectral density shown in fig.3-10, the average and sigma are 1.478 and 0.622 so that the ratio becomes 0.420 for the average of 8 blocks. For the average of 64 blocks shown in fig.3-12, they are 1.410 and 0.198 so that the ratio is 0.141.

Figures fig.3-13 through fig.3-21 show examples of the averaged signal power and their Fourier transforms, where the number of blocks in taking the averages increases from 256 to 65,536. As can be seen from these figures, the relative error $\frac{\mu}{\sigma}$ decreases as the number of blocks increases. Table 3-2 shows a summary of the computed results of the minimum and maximum of the relative errors for the different set of blocks. The results are compared with $\sqrt{\frac{2}{N}}$ in (3) of Chapter 1. Note that the experimental results are very close to the theoretical estimates. A summary of the computed results is shown in fig.3-23.

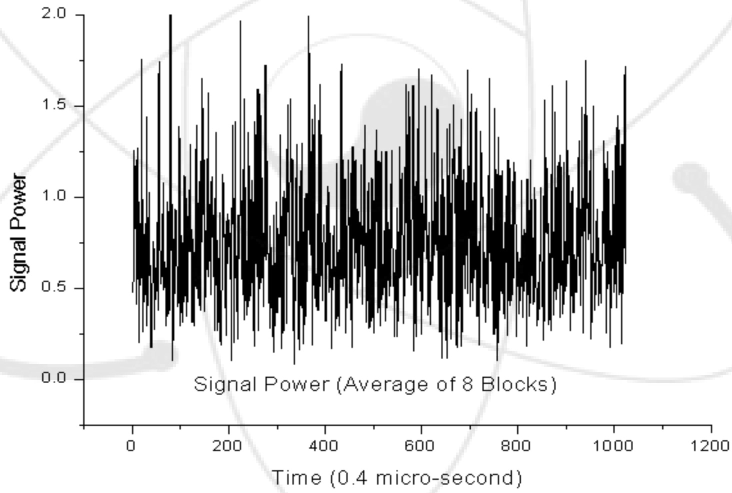

fig.3-9. Power of Johnson noise plus channel noise (average of 8 blocks)

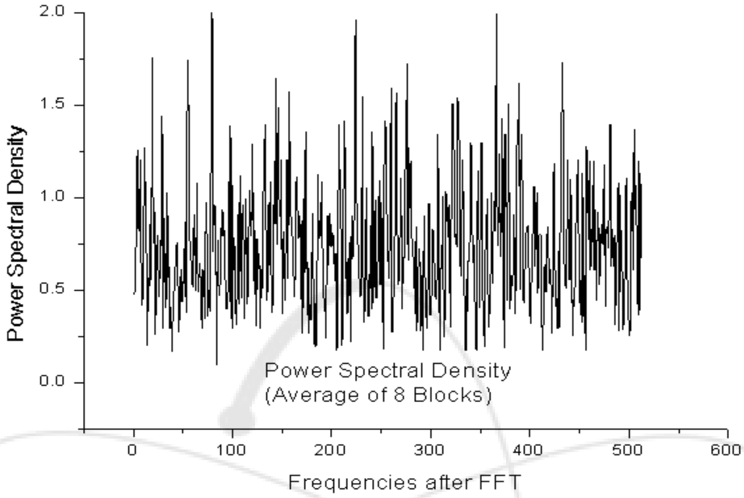

fig.3-10. Power spectral density of Johnson noise plus channel noise (average of 8 blocks)

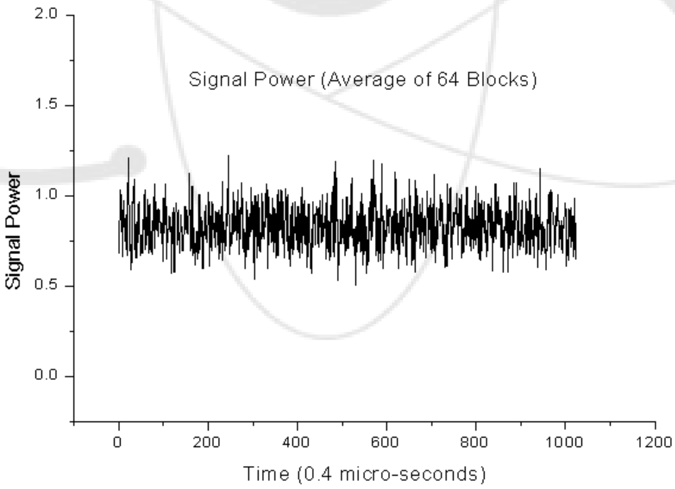

fig.3-11. Power of Johnson noise plus channel noise (average of 64 blocks)

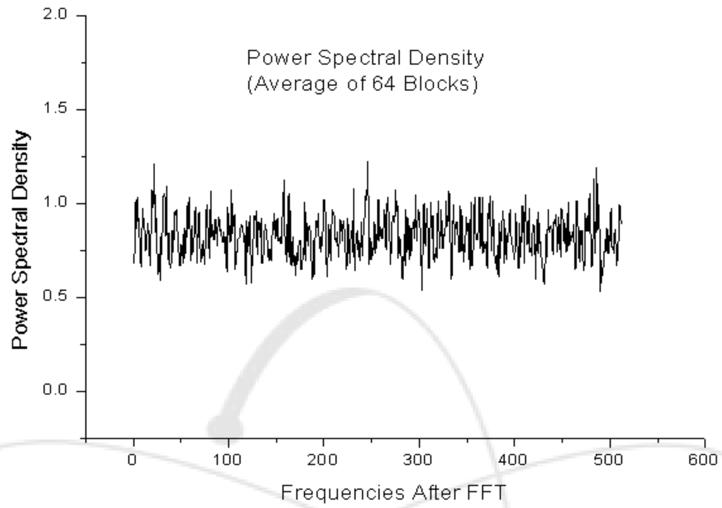

fig.3-12. Spectral power density of Johnson noise plus channel noise (average of 64 blocks)

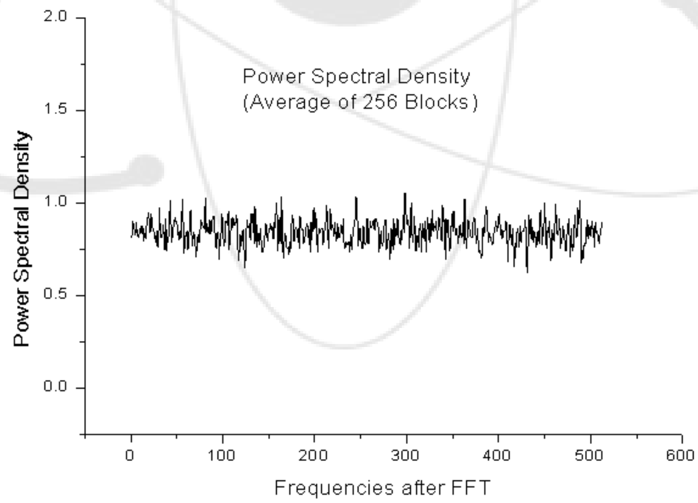

fig.3-13. Spectral power density of Johnson noise plus channel noise (average of 256 blocks)

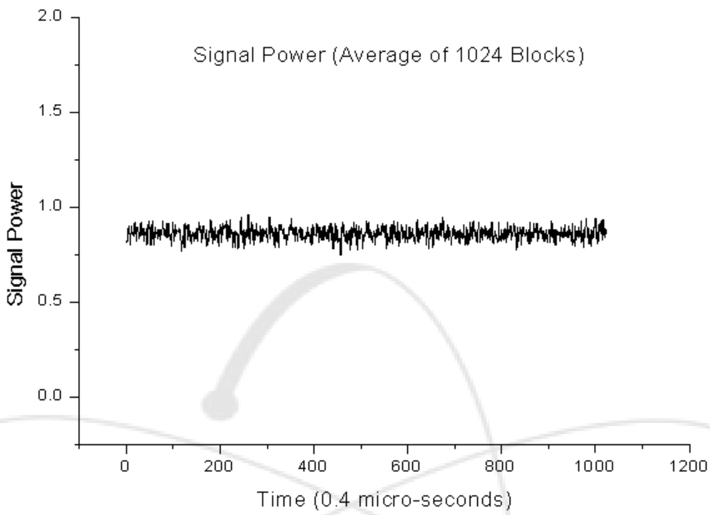

fig.3-14. Power of Johnson noise plus channel noise (average of 1024 blocks)

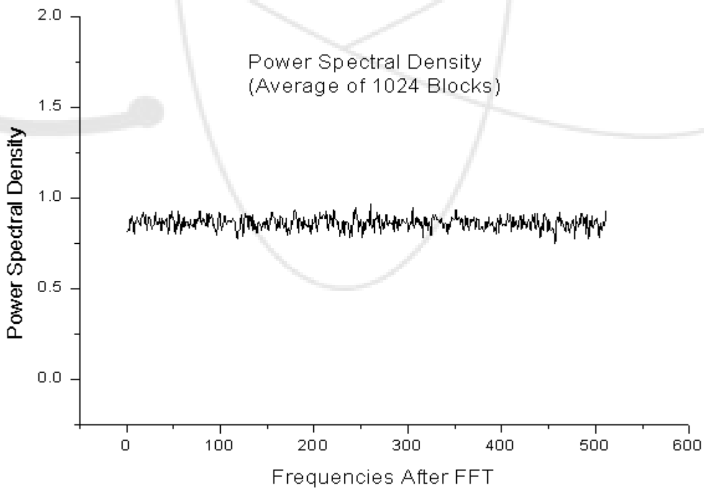

fig.3-15. Spectral power density of Johnson noise plus channel noise (average of 1024 blocks)

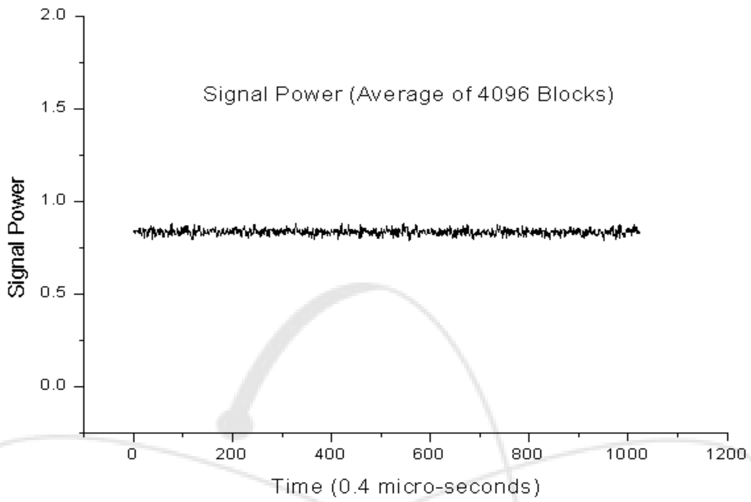

fig.3-16. Power of Johnson noise plus channel noise
(average of 4096 blocks)

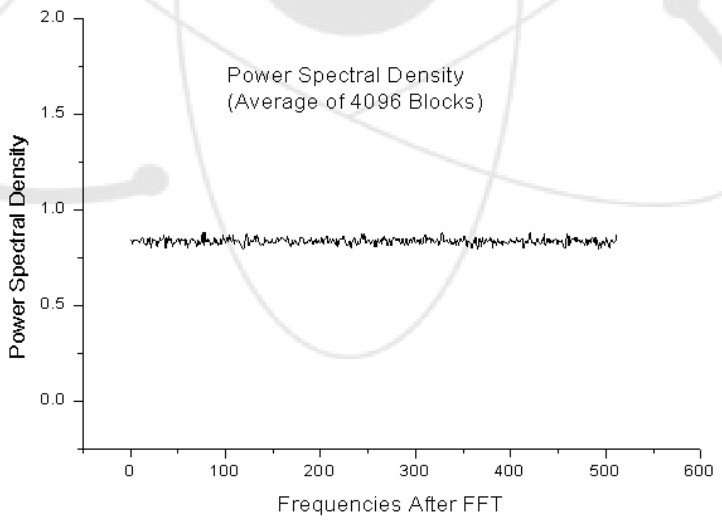

fig.3-17. Spectral power density of Johnson noise plus channel noise
(average of 4096 blocks)

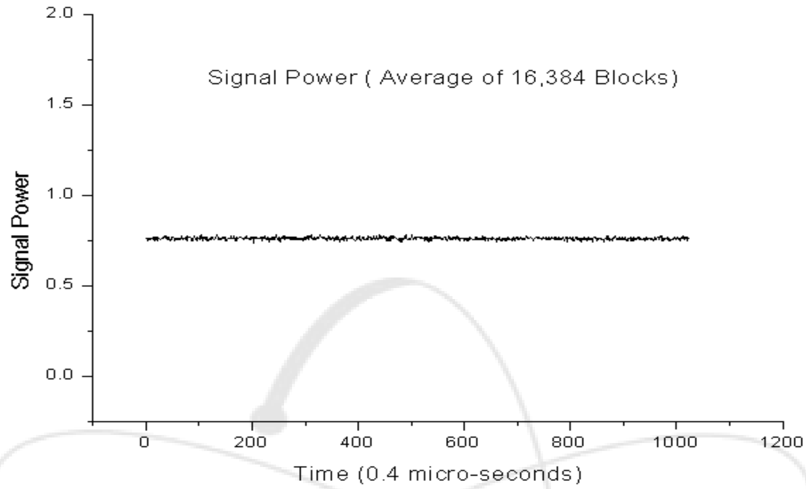

fig.3-18. Power of Johnson noise plus channel noise (average of 16,384 blocks)

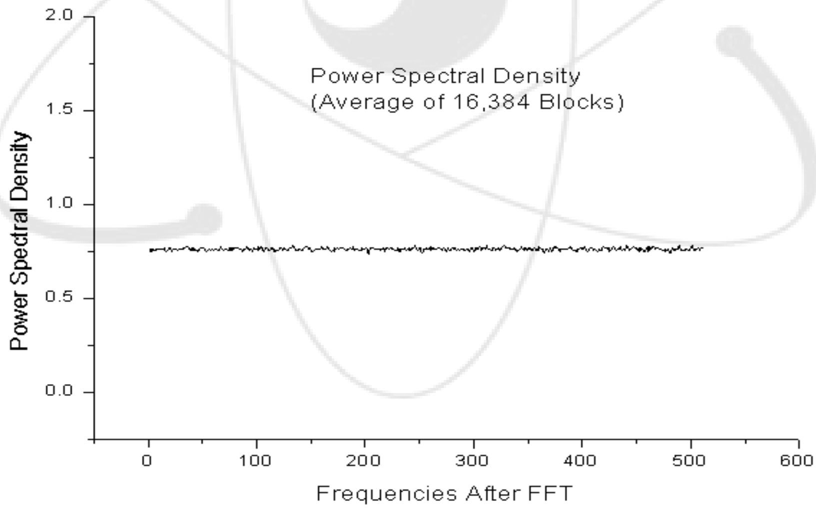

fig.3-19. Spectral power density of Johnson noise plus channel noise (average of 16,384 blocks)

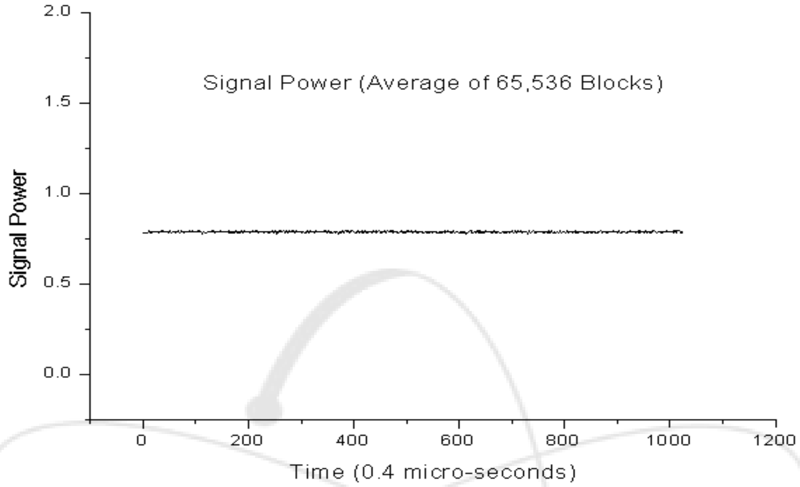

fig.3-20. Power of Johnson noise plus channel noise
(average of 65,536 blocks)

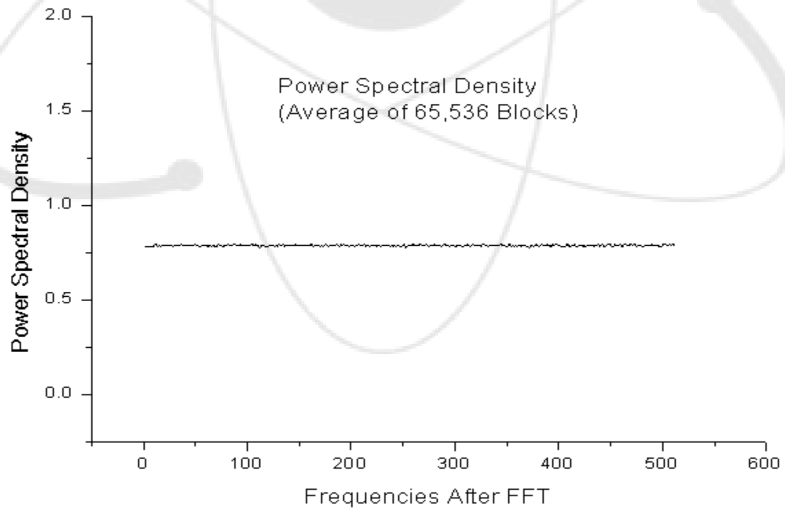

fig.3-21. Spectral power density of Johnson noise plus channel noise
(average of 65,536 blocks)

Table 3-2. Relative Error

No of Blocks	64	256	1024	4096	16384	65536
Min.	.1273	.0716	.0374	.0184	.0099	.0053
Max.	.1622	.0845	.0438	.0228	.0106	.0053
Avg.	.1448	.0781	.0406	.0206	.0103	.0053
Estim.	.1768	.0884	.0442	.0221	.0110	.0055

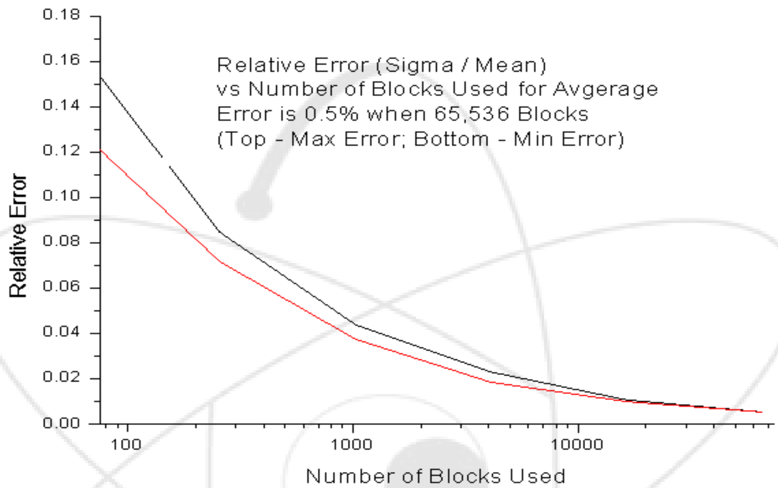

fig.3-22. Relative error of Johnson noise as the number of blocks increases
(Top curve – maximum error, bottom curve – minimum error)

As can be seen in the last column of Table3-2 and in fig.3-20 and fig.3-21, the relative error for the average of 65,536 blocks is 0.53% which is nearly the same as the theoretical estimate of 0.55%. By comparing the numbers in the last two rows, we conclude that if we had used enough data blocks so that we could have taken an average of $4\times 65{,}536$ blocks, then we should be able to get 0.28%. We also conclude that the Johnson noise signal blocks satisfy the properties of the continuous Markov process even in the presence of some channel noise.

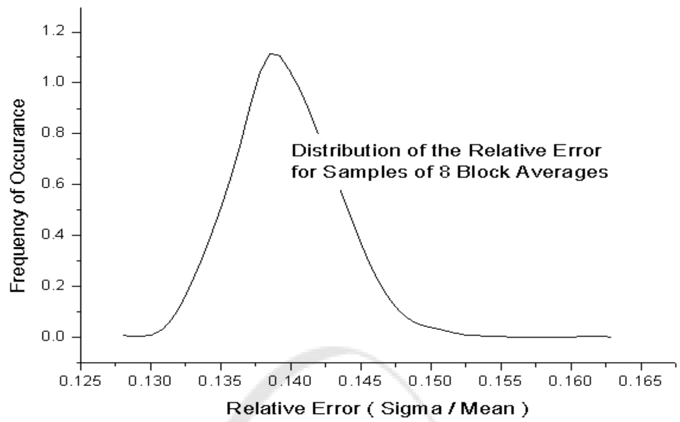

fig.3-23. Distribution of the relative error for the averages of 64 blocks

Chapter 4. Handling the Transient Temperature

1. Measurement Uncertainty

In this section, we describe the results of calculations performed to verify the error estimates (3) described in Chapter 1 and the results of an analysis to determine the amount of changes expected by replacing the averages by the moving averages. The two graphs in Fig.4-1 and the two graphs in fig.4-3 show an example on how the signal power changes as the number of blocks used in taking the averages increases. All of the two graphs in fig.4-1 and the two graphs in fig.4-3 show a random signal obtained by taking a sum of 1,024 single frequency signals with frequencies in the range $[3\times 2^{15}, 1.2\times 2^{20}]$. The first graph in fig.4-1 is obtained by taking an average of 64 blocks, the second is an average of 1,024 blocks, the first in fig.4-3 is an average of 2^{14} blocks, and the second in fig.10 is an average of 2^{18} blocks. Each of the blocks consists of 1,024 sampled points and each point is sampled at $\frac{1}{3\times 2^{20}}$ sec intervals. The two figures fig.4-2 and fig.4-3 show the corresponding power spectral density curves.

Table 4-1. Relative Error vs the Number of Blocks used for Averages (Sum of 2^{11} random signals)

	Average of Signal Power			Average of Spectral Power Density		
No of Blocks	Mean	Sigma	Sigma/Mean	Mean	Sigma	Sigma/Mean
64	340.802	58.702	0.17225	0.45070	0.06156	0.13658
1024	341.129	15.005	0.04399	0.45150	0.01515	0.03356
16384	341.309	3.7120	0.01088	0.45171	0.00396	0.00867
262144	341.298	0.7850	0.00230	0.45168	0.00122	0.00270

Table 4-2. Relative Error vs the Number of Blocks used for Averages (Sum of 2^{12} random signals)

	Average of Signal Power			Average of Spectral Power Density		
No of Blocks	Mean	Sigma	Sigma/Mean	Mean	Sigma	Sigma/Mean
64	685.194	124.93	0.18233	0.90625	0.12115	0.13367
1024	682.232	30.456	0.04664	0.90335	0.03092	0.03428
16384	682.104	7.209	0.01057	0.90268	0.00822	0.00911
262144	682.703	1.776	0.00260	0.90348	0.00316	0.00350

Table 4-1 shows a summary of the calculated results for random noise signals that are sums of 2,048 single frequency signals. A total of 262,144 blocks of 1,024 points each are generated and the means and sigmas for 64 blocks, 16×64 blocks, $16\times 16\times 64$ blocks, $16\times 16\times 16\times 64$ blocks are shown. The relative errors defined as the ratio of sigma over the mean are shown in column 4 for the signal power, and in column 7 for the power spectral density. One can draw a conclusion from these ratios that the relation (3) in Chapter 1 is true. It is apparent from Table 4-2 that the same holds for the case when the noise signals are sums of 4,096 single frequency signals.

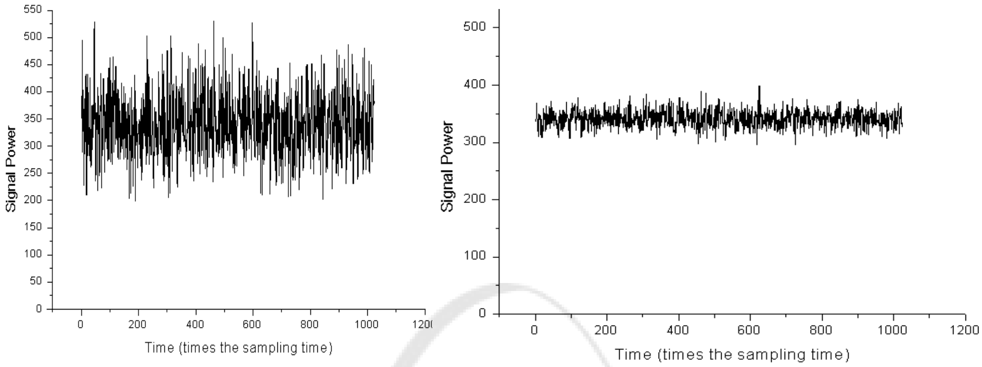

Fig.4-1. Signal Power (Left - Average of 2^6 Blocks ; $\mu = 340.802, \sigma = 58.702$
Right –Average of 2^{10} Blocks; μ=341.129,σ=15.005)

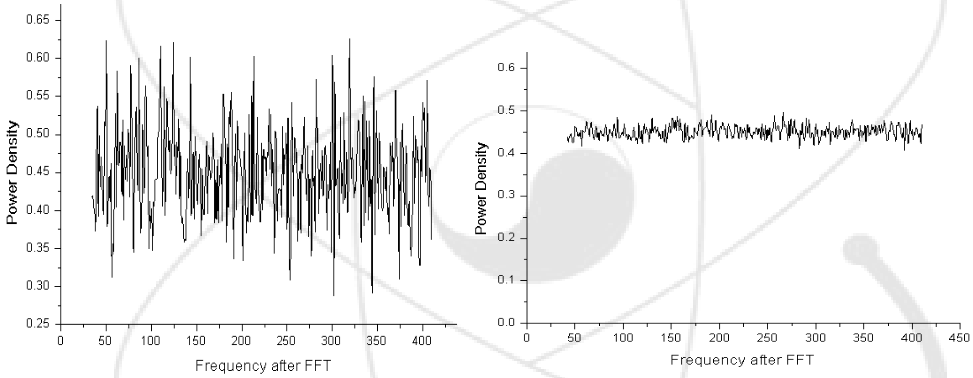

Fig.4-2. Power Density Spectrum (Left – Avg. of 2^6 Blocks ; $\mu = 0.45070, \sigma = 0.06156$
Right –Avg. of 2^{10} Blocks; $\mu = 0.45116, \sigma = 0.01536$)

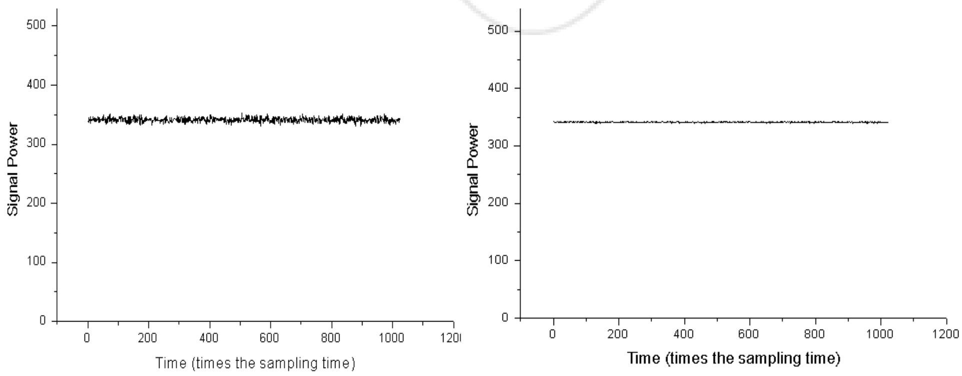

Fig.4-3. Signal Power (Left - Average of 2^{14} Blocks; $\mu = 341.309, \sigma = 3.712$
Right – Average of 2^{18} Blocks; $\mu = 341.298, \sigma = 0.785$)

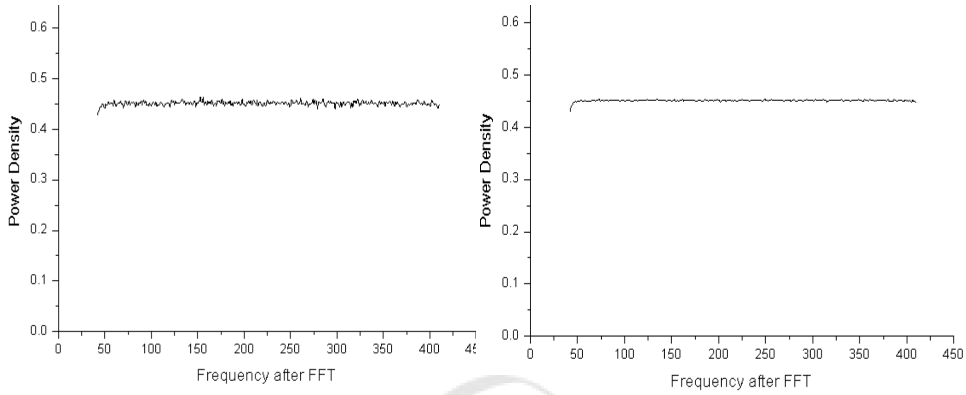

Fig.4-4. Power Density Spectrum (Left – Avg. of 2^{14} Blocks ; $\mu = 0.45144, \sigma = 0.00458$
Right –Avg. of 2^{18} Blocks; $\mu = 0.45141, \sigma = 0.00271$)

The following four figures from fig.4-5 through fig.4-8 show another example of a random signal obtained by taking a sum of 2,048 single frequency signals with frequencies in the range $[3\times 2^{15}, 1.2\times 2^{20}]$. It is a sum of twice the number of single frequency signals used in the example shown in fig.4-1 through fig.4-4. The first graph in fig.4-5 is obtained by taking an average of 64 blocks, the second is an average of 1,024 blocks, the first in fig.4-7 is an average of 2^{14} blocks, and the second in fig.4-7 is an average of 2^{18} blocks. The two figures fig.4-6 and fig.4-8 show the corresponding power spectral density curves. Once again, we see that the relation (3) in Chapter 1 works well with this example.

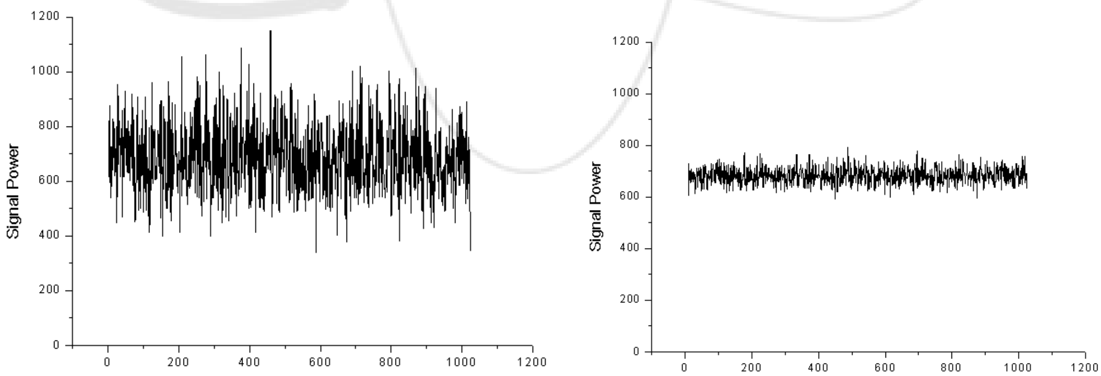

Fig.4-5. Signal Power (Left - Average of 2^{6} Blocks ; $\mu = 685.194, \sigma = 124.93$
Right –Average of 2^{10} Blocks; $\mu = 682.232, \sigma = 30.456$)

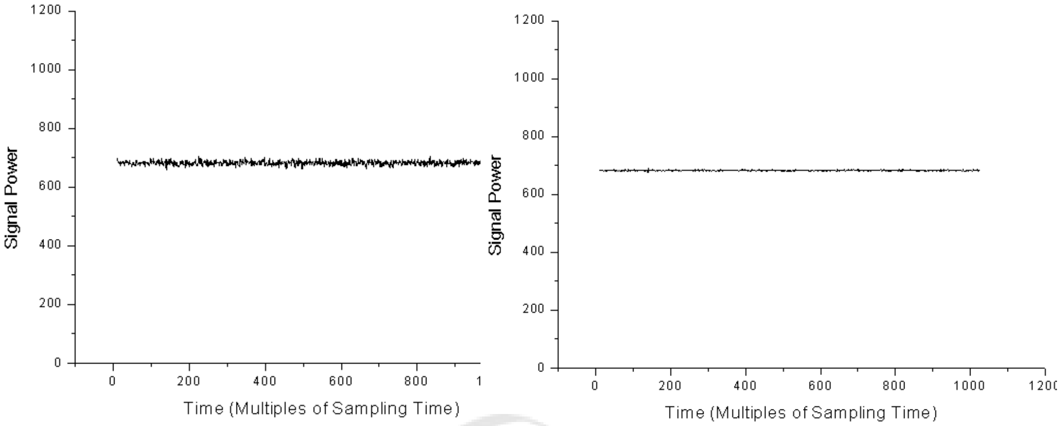

Fig.4-6. Signal Power (Left – Avg. of 2^{14} Blocks ; $\mu = 682.104, \sigma = 7.209$
Right –Avg. of 2^{18} Blocks; $\mu = 682.703, \sigma = 1.776$)

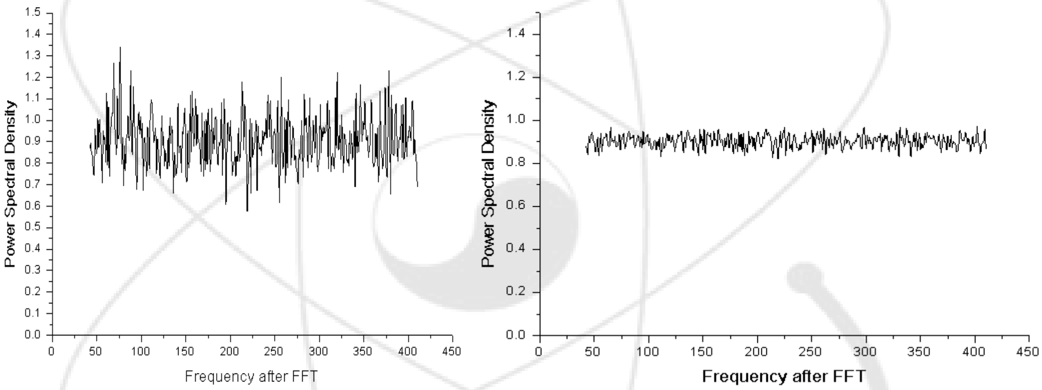

Fig.4-7. Power Density Spectrum (Left – Avg. of 2^{8} Blocks ; $\mu = 0.90625, \sigma = 0.12115$
Right –Avg. of 2^{10} Blocks; $\mu = 0.90335, \sigma = 0.03092$)

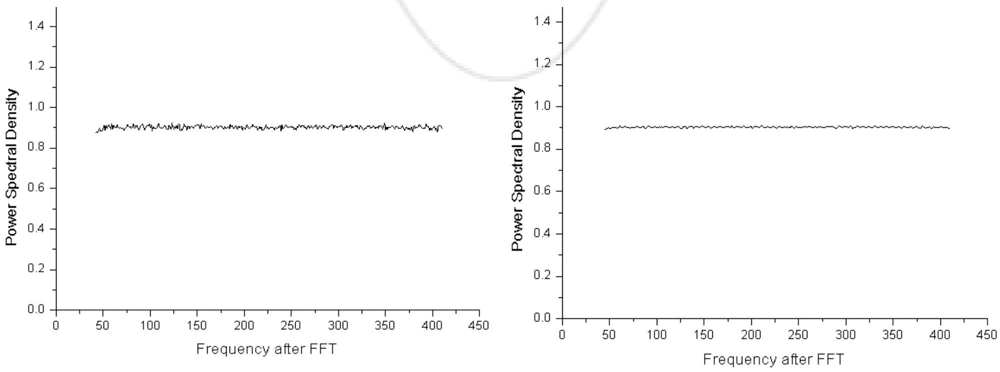

Fig.4-8. Power Density Spectrum (Left – Avg. of 2^{14} Blocks ; $\mu = 0.90268, \sigma = 0.00822$
Right –Avg. of 2^{18} Blocks; $\mu = 0.90348, \sigma = 0.00316$)

2. Moving Averages

We consider the problem of replacing the averages by the moving averages. As seen above, the high accuracy of the Johnson noise thermometer is achieved through taking a long average of the signal power. To obtain an accuracy of 0.28%, however, one quarter of a million blocks of samples are needed to be averaged so that a storage of $250{,}000 \times 1{,}024$ floating numbers (1GB) are needed to store the power spectral density obtained though FFT. To avoid this large storage area and to avoid the necessary I/O time, one must use a moving average algorithm.

Assume that we want to take an average of N terms in a time sequence $\{x_n\}$. Let a_{m-1} be the moving average using the values up through x_{m-1}. Then with the value x_m, the new average a_m will be computed by $a_m = (1-h)a_{m-1} + hx_m$, where $h = \frac{1}{N}$. Repeating the same, the next average is computed as $a_{m+1} = (1-h)^2 a_{m-1} + h(1-h)a_m + hx_{m+1}$. In general, we have

$$a_N = a_0 + h\sum_{i=1}^{N}(1-h)^{i-1}x_{N+1-i}$$

To compute how large is the contribution by $\{x_{-i} \mid i = 0,1,2,...\}$, we assume that the temperature is steady, i.e. $x_i = x_j$ for all i and j. Then the multiplier in the sum of the last N terms will be $h\sum_{i=1}^{N}\gamma^{i-1} = 1-\gamma^N$, where $\gamma = 1-h$.

Therefore, the remaining sum will be $\gamma^N = \left(1-\frac{1}{N}\right)^N$ whose limit as $N \to \infty$ is $e^{-1} = 0.36787944$. The results of sample calculations for N=2^{10}, 2^{16}, 2^{20} are; when $N = 2^{10}$ we have $\gamma^N = 0.3677$, $\gamma^{5N} = 0.00672$, $\gamma^{6N} = 0.000247$, and when $N = 2^{16}$, we have $\gamma^N = 0.3679$, $\gamma^{5N} = 0.00674$, $\gamma^{6N} = 0.000248$, and they are almost the same for $N = 2^{20}$. The coefficients of x_N, $x_{N-2^{13}}$, $x_{N-2\times2^{13}}$, $x_{N-7\times2^{13}}$, $x_{N-8\times2^{13}}$ are found to be 0.152588E-4, 0.118835E-4, 0.104872E-4, 0.925489E-5, 0.816740E-5, 0.720770E-5, 0.636077E-5, and 0.561472E-5 respectively. The ratio of the 65,536th coefficient relative to the most recent is 0.3678822, i.e. about 36%. Thus, we conclude that if we use four to five times larger number of blocks, then we can achieve nearly the same accuracy by using the moving averages provided that the temperature is steady.

A sample calculated result is shown in fig.4-9. The left hand side graph shows the signal power for a sum of 2^{11} random single frequency signals and it corresponds to the right hand side graph of fig.4-1. We can say that the means and sigmas are the same and the slight difference comes from the fact that the temperature is measured at different times. The right hand side graph of fig.4-9 corresponds to the right side graph of fig.4-3. Note that the sigma in fig.4-3 is 0.785 while the sigma in fig.4-9 is 1.645 about twice of the one in fig.4-3. Therefore, we need to take moving averages about 4 times more blocks than the case of ordinary averages, which is what we have claimed above.

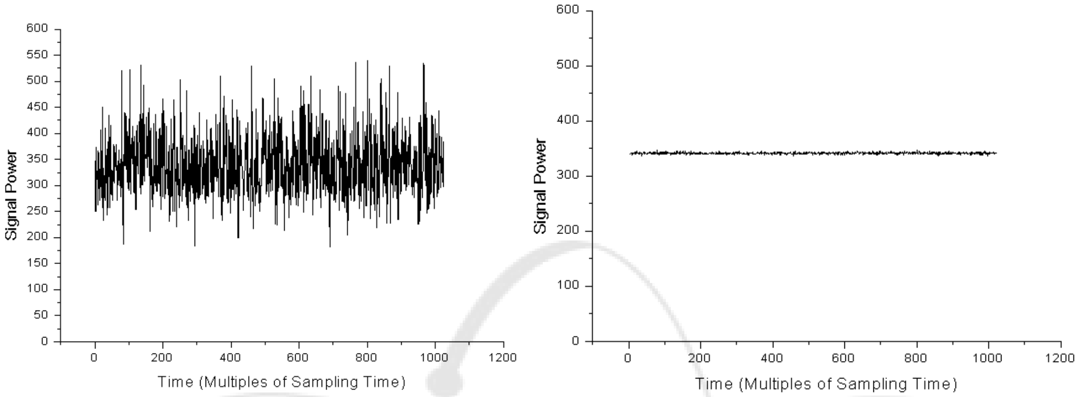

Fig.4-9. Signal Power (Left - Average of 2^6 Blocks ; $\mu = 341.346, \sigma = 59.138$

Right –Moving Avg. of 2^{18} Blocks; $\mu = 341.349, \sigma = 1.645$)

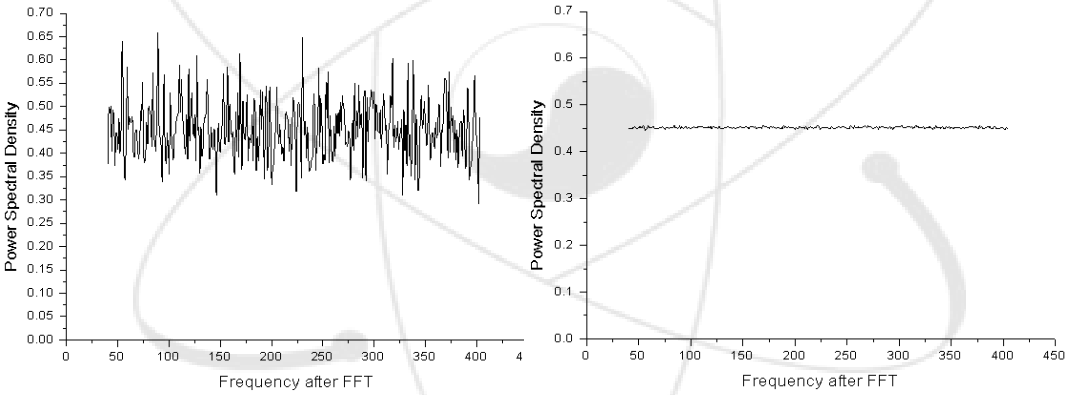

Fig.4-10. Power Density Spectrum (Left – Avg. of 2^6 Blocks ; $\mu = 0.451840, \sigma = 0.06324$

Right –Moving Avg. of 2^{18} Blocks; $\mu = 0.45181, \sigma = 0.00191$)

3. Handling the Transient Temperature by the Haar Wavelet

In this section, we describe how the multi-resolution analysis using the Haar wavelet can be used to compute the transient temperature where the long-term average is not appropriate due to the increase or decrease of the temperature value. We show that the approximation based on the Haar wavelet picks up a trend curve when it is applied to a band pass filtered sinusoidal signal. Hence, the temperature trend can be determined from the Haar approximation of the power of the Johnson noise.

In the following, we consider the multi-resolution analysis (MRA) [14,15] of a function to be a successive approximation of the function, and similarly we will consider the discrete wavelet transformation (DWT) of a function to be a successive approximation of the function based on the wavelet. The Haar wavelet decomposition of a continuous function f(t) on [0,1] is defined[17] as follows; Let $x_i = ih$, I=0,1,2,…, 2^n where $h = 2^{-n}$ and let

$$f_k(t) = 2^k \int_{2^{-k}\ell}^{2^{-k}(\ell+1)} f(\tau)d\tau\,, \quad 2^{-k}\ell \le t \le 2^{-k}(\ell+1) \qquad \text{------------------} \quad (1)$$

for $\ell = 0,1,2,...,2^k - 1$. Note that the function $f_k(t)$ is constant on every subinterval of length 2^{-k} starting from 0. Let the scaling function of the Haar wavelet be

$$\varphi(t) = \begin{cases} 1, & 0 \le t \le 1 \\ 0, & \text{otherwise} \end{cases} \qquad \text{--------------------------} \quad (2)$$

If we define

$$c(k,\ell) = 2^k \int_{2^{-k}\ell}^{2^{-k}(\ell+1)} f(t)dt \qquad \text{--------------} \quad (3)$$

Then the function $f_k(t)$ can be written as

$$f_k(t) = \sum_{\ell=0}^{2^k} c(k,\ell)\varphi(2^k t - \ell) \qquad \text{------------------} \quad (4)$$

and it is routine to check that the multi-resolution relation [16] becomes

$$c(k-1,\ell) = \frac{1}{2}(c(k,2\ell) + c(k,2\ell+1)) \qquad \text{----------} \quad (5)$$

For the wavelet transformation, we define

$$g_{k-1}(t) = f_{k-1}(t) - f_k(t) \qquad \text{--------------} \quad (6)$$

and the Haar wavelet $\psi(t)$ as $\psi(t) = \phi(2t) - \varphi(2t-1)$. Then we can write

$$g_{k-1}(t) = \sum_{\ell=0}^{2^k} d(k-1,\ell)\psi(2^k t - \ell) \qquad \text{------------------} \quad (7)$$

where the coefficients satisfy

$$d(k-1,\ell)=\frac{1}{2}(c(k,2\ell+1)-c(k,2\ell)) \qquad \text{------------} \qquad (8)$$

In the following, we prove that the approximation of a sinusoidal signal in a fixed frequency band by the 4^{th} step of the multi-resolution analysis using the Haar wavelet reduces the amplitude of the oscillation to less than 15% of the original.

Lemma 8. Let m and N be positive integers such that N is divisible by 256, i.e. $N=256N_1$. If $\frac{3}{16}\le\frac{m}{N}\le\frac{1}{2}$ and if $m=m_0N_1+m_1$ where $0\le m_1<N_1$, then we have $48\le m_0\le 127$.

Proof. From $\frac{3}{16}\le\frac{m_0N_1+m_1}{2^8N_1}<\frac{1}{2}$, we have $3\times 2^4N_1\le m_0N_1+m_1<2^8N_1$. Hence, by dividing the above by N_1, we obtain $3\times 2^4\le m_0+\frac{m_1}{N_1}<2^8$. Now, note that $0\le\frac{m_1}{N_1}<1$ and hence $3\times 2^4\le m_0+\frac{m_1}{N_1}$ if and only if $48\le m_0$. Q.E.D.

Next, we define $S(\ell)=\frac{1}{16}\sum_{k=1}^{16}k^\ell \operatorname{Sin}\left(\frac{mk\pi}{128}\right)$ and $C(\ell)=\frac{1}{16}\sum_{k=1}^{16}k^\ell \operatorname{Cos}\left(\frac{mk\pi}{128}\right)$, for $\ell=0,1,2...$, then by direct calculations, we have the following.

Lemma 9. If $S(\ell)$ and $C(\ell)$ are as defined above, then for all integers m with $48\le m\le 127$, we have $|S(0)|,|C(0)|\le 0.1$, $|S(1)|,|C(1)|\le 1.0$, $|S(2)|,|C(2)|\le 15$, $|S(3)|,|C(3)|\le 250.$, and $|S(4)|,|C(4)|\le 4000.$

Lemma 10. If $0\le x<0.5$, then $\sum_{k=n}^{\infty}\frac{x^k}{k!}\le\frac{x^n}{n!}(1+3x)$.

Theorem 3. If m and N are positive integers with $N=256N_1$ and if $\frac{3}{16}\le\frac{m}{N}\le\frac{1}{2}$, then we have $\left|\frac{1}{16}\sum_{k=1}^{16}\operatorname{Sin}\left(\frac{2\pi mk}{N}\right)\right|\le 0.15$.

Proof. Let $m=m_0N_1+m_1$ and write $\operatorname{Sin}\left(\frac{2\pi mk}{N}\right)$ as $\operatorname{Sin}\left(\frac{\pi m_0k}{128}+\frac{\pi m_1k}{128N_1}\right)$. Note that we have $48\le m_0\le 127$ by Lemma 8. Let $\alpha=\frac{\pi m_0}{128}$ and $\beta=\frac{\pi m_1}{128N_1}$, then $\beta\le\frac{1}{40}$. Now, using

$\text{Sin}(\alpha k + \beta k) = \text{Sin}(\alpha k)\text{Cos}(\beta k) + \text{Cos}(\alpha k)\text{Sin}(\beta k)$ which is equal to

$$\text{Sin}(\alpha k)\sum_{\ell=0}^{\infty}(-1)^{\ell}\frac{(\beta k)^{2\ell}}{(2\ell)!} + \text{Cos}(\beta k)\sum_{\ell=0}^{\infty}(-1)^{\ell}\frac{(\beta k)^{2\ell+1}}{(2\ell+1)!}.$$

Therefore, we can write

$$\left|\frac{1}{16}\sum_{k=1}^{16}\text{Sin}\left(\frac{2\pi mk}{N}\right)\right| \le |C(0)| + |S(1)| + \frac{\beta^2}{2!}|C(2)| + \frac{\beta^3}{3!}|S(3)| + \frac{\beta^4}{4!}|C(4)| + \left|\sum_{\ell=5}^{\infty}\frac{(\beta k)^{\ell}}{(2\ell)!}\right|.$$

Now, by using Lemma 9 and Lemma 10, it is routine to check that the last sum is less than or equal to 0.15. Q.E.D.

Corollary 1. If $f(t)$ is a signal with mean zero, sampled at a rate of N samples per second, band pass filtered with lower limit greater than $\frac{3}{16}N$ and upper limit frequency below $\frac{N}{2}$, then the maximum amplitude of the 4th step Haar approximation of $f(t)$ is below 15% of the original amplitude.

Fig.18 shows an example of a stationary Johnson noise signal power and its approximation, where the oscillating curve with the larger amplitude is the original and the curve in the middle is the approximated. Fig.19 shows an example where the temperature is increasing.

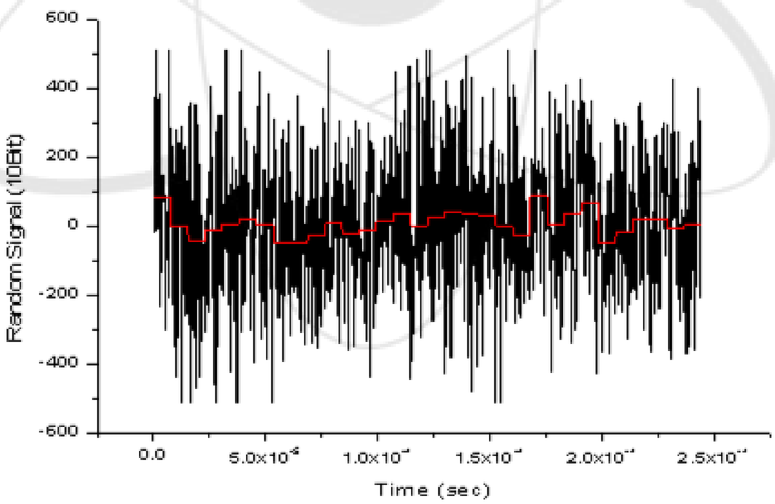

fig.4-11. Stationary Johnson Noise Signal Power with an Haar Approximation

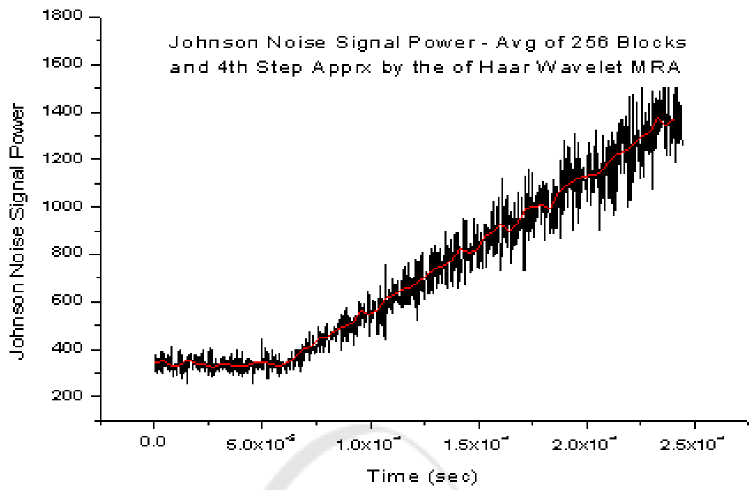

fig.4-12. Transient Johnson Noise Signal Power with an Haar Approximation

4. Accuracy vs the Processing Time

In this section, we consider how much time is needed to compute the long-term averages of the cross- power spectral density. Assume that we need an accuracy of 0.28% that is equal to $\sqrt{\frac{2}{250000}}$ so that the number of blocks required is one quarter of a million by (3). We further assume that the sampling rate is $\frac{1}{2^{24}}$ sec, which is practically the fastest A/D conversion time currently assuming that 12bit accuracy is used. The largest number of blocks that can be sampled in 1 second would then be 2^{14} so that sampling 2^{18} blocks (one quarter of a million blocks) would require 16seconds. On the other hand, for the process time whose major component is the FFT algorithm, we need to compute the time required to do one FFT algorithm. To do an FFT of 1024 points, one can count easily that there are 75,796 multiplications or divisions, 95,282 additions or subtractions, and 20,480 evaluations of the Cosine or Sine function. With a 2.8GHz Pentium-IV processor using FORTRAN, one can perform 10^8 multiplications, $4\,4\times10^8$ additions, about 10^7 evaluations of the Sine or Cosine function separately with no type conversions of variables such as from integer to real is used. Therefore, using these one can compute that an FFT takes $20{,}480\times10^{-7}+75{,}796\times10^{-8}+95{,}282\times10^{-8}/4.4\ =0.00302$ sec, so that about 330 FFT's can be performed in 1 second. In our development, we will be using Xilinx gate arrays so that more FFT's can be performed in 1 second. How many FFT's can be done in 1 second will depend not only on the number of gates used but also on the type of gate arrays. Note that no matter how fast the

FPGA's would do the FFT, the limit of 16 seconds sampling time would still be there. However, for nuclear power plant applications, the maximum RCS temperature change during the normal operation is 27°C/hr (0.008°C/sec=0.48°C/Min) due to the operation limit specified in the technical specification. Note that 0.48°C is below 0.28% of 310°C and hence the change in 1 minute is within the accuracy range of our Johnson noise thermometer.

Chapter 5. Conclusion

We have shown that a random signal of the form $\sum_{\alpha} a_{\alpha} \mathrm{Sin}(2\pi f_{\alpha} t + b_{\alpha})$ where a_{α} being a random number in [0,1], f_{α} a random integer in a given frequency band, and b_{α} a random number in $[0,2\pi]$, generates a continuous Markov process and hence it is a Gaussian white noise so that the Johnson noise can be simulated using the random signal. Through a statistical analysis on this continuous Markov process, we verified that the Johnson noise thermometry can be used to improve the measurements of the reactor coolant temperature within an accuracy of below 0.14%, provided that the necessary computation speed can be achieved by using FPGA's. Using this random signal, we were able to determine the optimal sampling rate assuming the frequency band of the Johnson noise signal is given. Also we were able to see how good the linearity of the Johnson noise is and how large the relative error of the temperature would become when the temperature increases.

We have also verified through experiments that the Johnson noise satisfies the properties of the continuous Markov process even in the presence of some channel noise. Thus, we do not have to eliminate the channel noise. Instead, the channel noise must be lowered to a level of the desired accuracy, i.e. the ratio between the channel noise power and the signal power must be maintained within the desired accuracy of say 0.14%.

Finally, we examined how to handle the problem of the long-term average or the moving average in a transient state. Note that the long-term average of the Johnson noise power will make the measured temperature change very slowly. Assuming that we sample 4 million points per second, we need 64 seconds to take 250,000 blocks of 1,024 points which is equivalent to 0.28% accuracy. Thus, we devised an algorithm based on the Haar wavelet to estimate the transient temperature that has much smaller time delay and have shown that the algorithm will track the ambient temperature successfully.

References

1. Herman A. Tasman, Joseph Richter, “Unconventional methods for measuring hight temperatures”, High Temperatures – High Pressure 11 (1979) 87-101.
2. Dimitrios C. Agouridis, “Noise Thermometers for Hostile Environments: A Theoretical Evaluation”, IEEE Trans. On Instrumentation and Measurements, 39(5) (1990) 780-784.
3. M. J. Roberts, T. V. Blalock, and R. L. Shepard, “Application of Johnson Noise Thermometry to Space Nuclear Reactors”, Space Nuclear Power Systems (1989) 87-93.
4. R. L. Shepard and J. M. Weiss, “Use of SP-100 Thermometry Technology to Improve Operation of Electric Power Plants”, Proc. 31st Intersociety Energy Conversion Eng. Conf. (1996) 114-118.
5. H. Brixy, R. Hecker, J. Jehmen, P. Barbonus, R. Hans, ”Temperature Measurement: Development Work on Noise Thermometry and Improvement of Conventional Thermocouples for Applications in Nuclear Process Heat”, Proc. Specialist Meeting on Gas Cooled Reactor Core and High Temperature Instrumentation, Windermere, England (IAEA-TC-389/6-7).
6. R. L. Shepard, et al., “In Situ Calibration of Nuclear Plant Platinum Resistance Thermometers Using Johnson Noise Methods”, EPRI NP-3113, Project 1440-1 (1983).
7. http://glossary.its.bldrdoc.gov/fs-1037/dir-037/_5417.htm
8. D.R. White, R. Galleano, A. Actis, et al., “The status of Johnson noise thermometry”, Metrologia 33(4) (1996) 325-335.
9. John G. Proakis and Dimitris G. Manolakis, “Digital Signal Processing, Principles, Algorithms, and Applications”, MacMillan Publishing Co. (1992).
10. Daniel T. Gilespie, *The mathematics of Brownian motion and Johnson noise*, Am. J. Phys. 64(3) (1996) 225-240.
11. C. Jacobi and L. Reggiani, *The Monte Carlo method for the solution of charge transport in semiconductors and application to covalent materials*, Review of Modern Physics 55(3) (1983) 645-705.
12. R. H. Olster and D. T. Seagraves, *Practical MCNP for the Health Physicists, Rad Engineer & Medical Physicist*, Lawrence Livermore National Lab. (2002).
13. W. R. Nelson, H. Hirayama, D. W. Rogers, *The EGS4 code system*, Stanford Linear Accelerator Center, Stanford University (1985).
14. Dimitrios C. Agouridis, *Noise Thermometers for Hostile Environments: A Theoretical Evaluation*, IEEE Trans. On Instrumentation and Measurements, 39(5) (1990) 780-784.
15. John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing, Principles, Algorithms, and Applications*, MacMillan Publishing Co. (1992).
16. Raghuveer M. Rao and Ajit S. Bopardikar, “Wavelet Transforms; Introduction to Theory and Applications”, Addison-Wesley Longman (1998).

17. Charles K. Chui, “An Introduction to Wavelets”, Academic Press, Inc. (1992).
18. D. T. Gilespie, *A Mathematical Comparison of Simple Models of Johnson Noise and shot Noise,* J. Phys., Condens. Matter. 12, 4195-4205 (2000).

Attachment 1. Program for Generating Noise Signals

```
C
C        WRITTEN BY B.S.MOON, NOV.30, 2002
C
                     PROGRAM RANDFN
                     COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                     COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                     DIMENSION XREAL(1024), AMPL(1024), XIMAG(1024)
                     DIMENSION CUMVAL2(1024,16,16,16), CUMFRQ2(377,16,16,16)
                 DIMENSION VALINT3(1024,16,16), FRQINT3(377,16,16)
                 DIMENSION VALINT2(1024,16), FRQINT2(377,16)
                 DIMENSION VALINT1(1024), FRQINT1(1024)
               CHARACTER*8 BUF
                 DATA   PI / 3.141592653589 /

C
C      THE ABOVE DIMENSION MUST BE GREATER THAN or EQUAL THE VALUE OF 'N'
C        N IS THE NUMBER OF DATA POINTS; MUST BE OF 2**NU
C        NU IS LOG ( BASE 2 ) OF N = GAMMA IN BRIGHAM'S TEXT BOOK
C
               OPEN( 6,FILE='V1801A.DAT')
               OPEN( 7,FILE='F1801A.DAT')
               OPEN( 8,FILE='S1801A.DAT')

               OPEN(16,FILE='V1802A.DAT')
               OPEN(17,FILE='F1802A.DAT')

             OPEN(26,FILE='V1803A.DAT')
               OPEN(27,FILE='F1803A.DAT')

             OPEN(36,FILE='V1804A.DAT')
               OPEN(37,FILE='F1804A.DAT')

               OPEN(99,FILE='DEBUG.DAT')

                     NU      = 10                        ! USED TO BE 10
                     STIME = 1./ (3* 2.**20)       ! SAMPLING TIME
                     TIMES = 0.                        ! START TIME

                     N        = 2.**NU
                ITEMPR = 2**18              ! DETERMINES TEMPERATURE


               ILOW   =   3 * 2**15          ! LOWER BOUND FREQ
               IUPP   = 1.2* 2**20          ! UPPER BOUND FREQ

                  DO 100 J = 1, 1024
                XJ(J) = (J-1) * STIME
100             CONTINUE
C
C   GENERATE THE SEED FOR RANDOM NUMBERS
C


                    CALL TIME ( BUF )
                WRITE(99,10) BUF
 10           FORMAT( A8 )
                    REWIND 99
```

```
C
              READ(99,20) IHOUR, MIN, ISEC
 20         FORMAT(I2,1X,I2,1X,I2)

              ISTART = IHOUR*3600 +  MIN*60 +  ISEC
                   CALL SEED( ISTART )
C
                   DO 9900 LOOPL = 1,   1          !16
                   DO 9000 LOOPM = 1,   1          ! 16
                   DO 8000 LOOPS = 1, 16
                                DO 7000   INDX = 1, 64
C
C       INITIALIZE THE ARRAY
C
                     DO 200 J = 1, N
                  XREAL(J) = 0
                       XIMAG(J) = 0
200           CONTINUE
C
C     GENERATE THE RANDOM SIGNAL DATA AND COMPUTE fORURIER TRANSFORM
C          VMAG2 COMPUTED BY GENFUN
C       FMAG2 COMPUTED BY FFT
C
                   CALL GENFUN ( ITEMPR, XREAL )
           CALL     FFT    ( XREAL )
             PRINT *, LOOPL, LOOPM, LOOPS, INDX
C
C     COMPUTE MEAN AND SIGMA OF CROSS POWER SPECTRAL DENSITY
C
             SUM2 = 0
           DO 500 K = 1, 1024
                     SUM2 = SUM2 +  VMAG2(K)
500         CONTINUE
             VMEAN2 = SUM2 / 1024

                     SUM4 = 0
                   DO 550 K = 1, 1024
                     SUM4 = SUM4 +  ( VMAG2(K) - VMEAN2) **2
550         CONTINUE
                     VSIGMA = SQRT( SUM4 / 1023 )

             SUM2 = 0
           DO 600 K = 34, 410
                     SUM2 = SUM2 +  FMAG2(K)          ! FMAG2 IS COMPUTED IN FFT
600         CONTINUE
             FMEAN2 = SUM2 / 377
C
                     SUM4 = 0
                   DO 650 K = 34, 410                ! 1, 512
                     SUM4 = SUM4 +  ( FMAG2(K) - FMEAN2) **2
650         CONTINUE
                     FSIGMA = SQRT( SUM4 / 376 )



C                    WRITE(6,50)   LOOPL,LOOPM,LOOPS,INDX,VMEAN2,VSIGMA
C                    WRITE(7,50)   LOOPL,LOOPM,LOOPS,INDX,FMEAN2,FSIGMA

 50           FORMAT( 4I4, 2E15.8 )
```

```
C
C   SAVE THE COMPUTED RESULTS FOR LONG AVERAGES
C
              DO 700 K = 1, 1024
                        CUMVAL2(K,LOOPS,LOOPM,LOOPL) =
          1                   CUMVAL2(K,LOOPS,LOOPM,LOOPL) +  VMAG2(K) / 64
700          CONTINUE

             DO 750 K = 1, 377
                        CUMFRQ2(K,LOOPS,LOOPM,LOOPL) =
          1             CUMFRQ2(K,LOOPS,LOOPM,LOOPL) +  FMAG2(K+ 33) / 64
750          CONTINUE

7000          CONTINUE
C
C    SAVE THE RESULTS & COMPUTE MEAN AND SIGMA OF 1024 BLOCKS
C
                  IF ( LOOPS .EQ. 1 .AND. LOOPM .EQ. 1 .AND.
          1                          LOOPL .EQ. 1) THEN
                       WRITE(6,60)   ( CUMVAL2(K,1,1,1), K=1,1024 )
                       WRITE(7,65)   ( CUMFRQ2(K,1,1,1), K=1, 377 )
                  ENDIF
 60             FORMAT( 8F9.3 )
 65             FORMAT( 8F9.5 )

         CALL STAT(CUMVAL2(1,LOOPS,LOOPM,LOOPL),1024,VMEAN2,VSIGMA)
C                      SUM2V = 0
C                      DO 1200 K = 1, 1024
C                      SUM2V = SUM2V +  CUMVAL2(K,LOOPS,LOOPM,LOOPL)
C1200         CONTINUE
C                      VMEAN2 = SUM2V / 1024

C                      SUM4V = 0
C                    DO 1250 K = 1, 1024
C               SUM4V = SUM4V+ (CUMVAL2(K,LOOPS,LOOPM,LOOPL)-VMEAN2 )**2
C1250         CONTINUE
C                      VSIGMA = SQRT( SUM4V / 1023 )
C
C        FREQUENCY DOMAIN
C
         CALL STAT(CUMFRQ2(1,LOOPS,LOOPM,LOOPL),377,FMEAN2,FSIGMA)
C                      SUM2F = 0
C                      DO 1300 K = 1, 377
C                      SUM2F = SUM2F +  CUMFRQ2(K,LOOPS,LOOPM,LOOPL)
C1300         CONTINUE
C                      FMEAN2 = SUM2F / 377

C                      SUM4F = 0
C                    DO 1350 K = 1, 377
C               SUM4F = SUM4F+ (CUMFRQ2(K,LOOPS,LOOPM,LOOPL)-FMEAN2)**2
C1350         CONTINUE
C                      FSIGMA = SQRT( SUM4F / 376 )

                       WRITE(8,90)   LOOPS, VMEAN2, VSIGMA, FMEAN2, FSIGMA
 90          FORMAT( I5, 4E15.8 )
8000                   CONTINUE            ! END OF LOOPS
C
C      COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
```

```
                 DO 2100 K = 1, 1024
                         VALINT3(K,LOOPM,LOOPL) = 0
2100            CONTINUE

              DO 2500 L = 1,    16                ! L=LOOPS
              DO 2200 K = 1, 1024
                 VALINT3(K,LOOPM,LOOPL) = VALINT3(K,LOOPM,LOOPL) +
          1                                CUMVAL2(K,L,LOOPM,LOOPL) / 16
2200            CONTINUE
2500          CONTINUE
C
                 DO 2600 K = 1, 377
                         FRQINT3(K,LOOPM,LOOPL) = 0
2600            CONTINUE

              DO 2800 L = 1,    16                ! L=LOOPS
              DO 2700 K = 1, 377
                 FRQINT3(K,LOOPM,LOOPL) = FRQINT3(K,LOOPM,LOOPL) +
          1                                CUMFRQ2(K,L,LOOPM,LOOPL) / 16
2700            CONTINUE
2800          CONTINUE

                 IF (LOOPM .EQ. 1 .AND. LOOPL .EQ. 1) THEN
              CALL STAT ( VALINT3, 1024, VMEAN, VSIGMA )
              CALL STAT ( FRQINT3,   377, FMEAN, FSIGMA )
                 WRITE(16,60)       VMEAN, VSIGMA
                 WRITE(16,60)   ( VALINT3(K,1,1), K=1,1024 )
                 WRITE(17,65)       FMEAN, FSIGMA
                       WRITE(17,65)   ( FRQINT3(K,1,1), K=1, 377 )
                 ENDIF

9000              CONTINUE      ! END OF LOOPM
C
C        COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                 DO 3100 J = 1, 1024
                         VALINT2(J,LOOPL) = 0
3100            CONTINUE

              DO 3500 K = 1,    16                ! L=LOOPS
                 DO 3200 J = 1, 1024
                 VALINT2(J,LOOPL) = VALINT2(J,LOOPL) +
          1                                       VALINT3(J,K,LOOPL) / 16
3200            CONTINUE
3500          CONTINUE
C
                 DO 3600 K = 1, 377
                         FRQINT2(K,LOOPL) = 0
3600            CONTINUE

              DO 3800 K = 1,    16                ! L=LOOPS
              DO 3700 J = 1, 377
                 FRQINT2(J,LOOPL) = FRQINT2(J,LOOPL) +
          1                                FRQINT3(J,K,LOOPL) / 16
3700            CONTINUE
3800          CONTINUE

               IF ( LOOPL .EQ. 1 )    THEN
             CALL STAT ( VALINT2, 1024, VMEAN, VSIGMA )
```

```
          CALL STAT ( FRQINT2,   377, FMEAN, FSIGMA )
             WRITE(26,60)      VMEAN, VSIGMA
             WRITE(26,60)   ( VALINT2(K,1), K=1,1024 )
             WRITE(27,65)      FMEAN, FSIGMA
             WRITE(27,65)   ( FRQINT2(K,1), K=1, 377 )
            ENDIF

9900                 CONTINUE             ! END OF LOOPL
C
C     COMPUTE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
              DO 4100 J = 1, 1024
                     VALINT1(J) = 0
4100          CONTINUE

           DO 4500 K = 1,   16
              DO 4200 J = 1, 1024
             VALINT1(J) = VALINT1(J) +  VALINT2(J,K) / 16
4200          CONTINUE
4500        CONTINUE
C
              DO 4600 K = 1, 377
                     FRQINT1(K) = 0
4600          CONTINUE

           DO 4800 K = 1,   16
           DO 4700 J = 1, 377
             FRQINT1(J) = FRQINT1(J) +  FRQINT2(J,K) / 16
4700          CONTINUE
4800        CONTINUE

          CALL STAT ( VALINT1, 1024, VMEAN, VSIGMA )
          CALL STAT ( FRQINT1,   377, FMEAN, FSIGMA )
             WRITE(36,60)      VMEAN, VSIGMA
             WRITE(36,60)   ( VALINT1(K), K=1,1024 )
             WRITE(37,65)      FMEAN, FSIGMA
             WRITE(37,65)   ( FRQINT1(K), K=1, 377 )

             STOP
                END

          SUBROUTINE FFT( XREAL )
                  COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                  COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                  DIMENSION XREAL(1), AMPL(1024), XIMAG(1024)
             DATA   PI / 3.141592653589 /

C
C     BOX #2 IN FIG. 8.6
C
                  N2   = N / 2
                  NU1 = NU - 1
                  K     = 0
C
C     BOX #4, #5, #6
C
                  DO 300 L = 1, NU

100            CONTINUE
```

```
          DO 200 I = 1, N2
             DIVIDE = 2.**NU1
             J    = K / DIVIDE
             ARG = 2*PI * IBITR(J,NU) / N
             C     = COS(ARG)
             S     = SIN(ARG)
             K1   = K +  1
            K1N2 = K1 +  N2
            TREAL = XREAL(K1N2) * C +  XIMAG(K1N2) * S
            TIMAG = XIMAG(K1N2) * C - XREAL(K1N2) * S
            XREAL(K1N2) = XREAL(K1) - TREAL
            XIMAG(K1N2) = XIMAG(K1) - TIMAG
            XREAL(   K1) = XREAL(K1) +  TREAL
            XIMAG(   K1) = XIMAG(K1) +  TIMAG
             K     = K +  1
200       CONTINUE
C
             K     = K +  N2
           IF (K .LT. N ) GO TO 100
             K     = 0
             NU1 = NU1 - 1
             N2   = N2 / 2
300      CONTINUE
C
C
C
         DO 500 K = 1, N
           J = K - 1
           I = IBITR(J,NU) +  1
           IF ( I .LE. K ) GO TO 500
           TREAL = XREAL(K)
           TIMAG = XIMAG(K)
             XREAL(K) = XREAL(I) / DIVIDE
             XIMAG(K) = XIMAG(I) / DIVIDE
             XREAL(I) = TREAL      / DIVIDE
             XIMAG(I) = TIMAG      / DIVIDE
500       CONTINUE
C
C   DIVIDE BY 2**NU TO OBTAIN PROPER AMPLITUDE
C
           DIVIDE = 2.**NU
             SUM2 = 0
         DO 600 K = 1, N
             XREAL(K) = XREAL(K) / DIVIDE
             XIMAG(K) = XIMAG(K) / DIVIDE
             FMAG2(K) = XREAL(K)**2 +  XIMAG(K)**2

600       CONTINUE

          RETURN
           END




C
C     TO GENERATE DATA FOR TESTING FFT ALGORITHM
C
```

```
                    SUBROUTINE GENFUN ( ITEMPR, XREAL )

                    COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                    COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                    DIMENSION XREAL(1)
               REAL RAND


                    DATA   TWOPI / 6.283185307 /
                    F(X)   = AMPL * SIN( WN1*X +  ALPH )
C

                             DO 1000 LOOP = 1, ITEMPR

                    RAND1 = RAND (   )
                    RAND2 = RAND (   )
                    RAND3 = RAND (   )

                    AMPL    = RAND1
                    MEGA    = RAND2 * (IUPP - ILOW) +  ILOW
                    WN1     = TWOPI * MEGA
                    ALPH    = RAND3 * TWOPI

C
C    COMPUTE THE XREAL DATA ARRAY
C
                    DO 200 J = 1, N
                      XJJ = XJ(J)
                      XREAL(J) = XREAL(J) +  F( XJJ )
200        CONTINUE
C
1000         CONTINUE
C
C    COMPUTE THE CROSS PRODUCT OF XREAL WITH ITSELF
C
                      DO 2000 J = 1, N
                      VMAG2(J) = XREAL(J) * XREAL(J)
2000         CONTINUE

                             RETURN
                              END
```

```
C
C    REVERSE THE BIT STRING
C
```

```
                  FUNCTION IBITR( J, NU )
                  DATA JSAVE / 0 /
C
C   INITIALIZE VARIABLES
C
                  J1 = J
                  IBITR = 0
C
                     DO 100 I = 1, NU
                  J2 = J1 / 2
                  IBITR = IBITR * 2   +   (J1 - 2*J2 )
                  J1 = J2
100         CONTINUE

                   RETURN
                   END
C
            SUBROUTINE STAT ( ARRAY, LENGTH, AMEAN, SIGMA )
            DIMENSION ARRAY(1)
C
C         COMPUTE THE MEAN
C
                     SUM1 = 0
                   DO 200 J = 1, LENGTH
                     SUM1 = SUM1 +  ARRAY(J)
200         CONTINUE
              AMEAN = SUM1 / LENGTH
C
C    COMPUTE THE STANDARD DEVIATION
C
                     SUM2 = 0
                   DO 300 J = 1, LENGTH
                     SUM2 = SUM2 +  ( ARRAY(J) - AMEAN )**2
300         CONTINUE
              SIGMA = SQRT ( SUM2 / (LENGTH-1) )

                     RETURN
                     END
```

Attachment 2. Program for Testing the Moving Average Algorithm

```
C
C          TO TEST THE MOVING AVERAGE ALGORITHM
C          WRITTEN BY B.S.MOON, APR.16, 2003
C
                         PROGRAM MAVG
                         COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                         COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                         DIMENSION XREAL(1024), AMPL(1024), XIMAG(1024)
                         DIMENSION CUMVAL2(1024), CUMFRQ2(365)

                           CHARACTER*8 BUF
                         DATA    PI / 3.141592653589 /

C
C      THE ABOVE DIMENSION MUST BE GREATER THAN or EQUAL THE VALUE OF 'N'
C          N IS THE NUMBER OF DATA POINTS; MUST BE OF 2**NU
C          NU IS LOG ( BASE 2 ) OF N = GAMMA IN BRIGHAM'S TEXT BOOK
C
                OPEN( 6,FILE='V1101M.DAT')
                OPEN( 7,FILE='F1101M.DAT')
                OPEN( 8,FILE='S1101M.DAT')

                OPEN(16,FILE='V1102M.DAT')
                OPEN(17,FILE='F1102M.DAT')

                OPEN(26,FILE='V1103M.DAT')
                OPEN(27,FILE='F1103M.DAT')

                OPEN(36,FILE='V1104M.DAT')
                OPEN(37,FILE='F1104M.DAT')

                OPEN(99,FILE='DEBUG.DAT')

                         NU       = 10                              ! USED TO BE 10
                         STIME = 1./ (3.* 2.**20)         ! SAMPLING TIME
                         TIMES = 0.                              ! START TIME

                         N         = 2.**NU
                       ITEMPR = 2**11              ! DETERMINES TEMPERATURE
                  FACTR    = 1. / 2**16      ! MOVING AVERAGE

                ILOW    =    3 * 2**15           ! LOWER BOUND FREQ
                IUPP    = 1.2* 2**20           ! UPPER BOUND FREQ

                           DO 100 J = 1, 1024
                       XJ(J) = (J-1) * STIME
100            CONTINUE
C
C   GENERATE THE SEED FOR RANDOM NUMBERS
```

```
C
                    CALL TIME ( BUF )
                    WRITE(99,10) BUF
 10         FORMAT( A8 )
                    REWIND 99
C
                    READ(99,20) IHOUR, MIN, ISEC
 20         FORMAT(I2,1X,I2,1X,I2)

                    ISTART = IHOUR*3600 + MIN*60 + ISEC
                    CALL SEED( ISTART )
C
                                        DO 9900 LOOPL = 1, 16
                                        DO 9000 LOOPM = 1, 16
                                        DO 8000 LOOPS = 1, 16
                                DO 7000    INDX = 1, 64

C
C      INITIALIZE THE ARRAY
C
                      DO 200 J = 1, N
                              XREAL(J) = 0
                              XIMAG(J) = 0
200           CONTINUE
C
C    GENERATE THE RANDOM SIGNAL DATA AND COMPUTE fORURIER TRANSFORM
C    VMAG2 COMPUTED BY GENFUN
C      FMAG2 COMPUTED BY FFT
C
                    CALL GENFUN ( ITEMPR, XREAL )
                      CALL      FFT    ( XREAL )
                              PRINT *, LOOPL, LOOPM, LOOPS, INDX
C
C    COMPUTE MEAN AND SIGMA OF CROSS POWER SPECTRAL DENSITY
C
                              SUM2 = 0
                      DO 500 K = 1, 1024
                      SUM2 = SUM2 + VMAG2(K)
500         CONTINUE
                              VMEAN2 = SUM2 / 1024

                      SUM4 = 0
                    DO 550 K = 1, 1024
                      SUM4 = SUM4 + ( VMAG2(K) - VMEAN2) **2
550         CONTINUE
                      VSIGMA = SQRT( SUM4 / 1023 )

                              SUM2 = 0
                      DO 600 K = 40, 404
                      SUM2 = SUM2 + FMAG2(K)            ! FMAG2 IS COMPUTED IN FFT
600         CONTINUE
```

```
                  FMEAN2 = SUM2 / 365
C
              SUM4 = 0
             DO 650 K = 40, 404             ! 1, 512
              SUM4 = SUM4 + ( FMAG2(K) - FMEAN2) **2
650       CONTINUE
              FSIGMA = SQRT( SUM4 / 364 )

C             WRITE(6,50)   LOOPL,LOOPM,LOOPS,INDX,VMEAN2,VSIGMA
C             WRITE(7,50)   LOOPL,LOOPM,LOOPS,INDX,FMEAN2,FSIGMA
 50        FORMAT( 4I4, 2E15.8 )
C
C   SAVE THE COMPUTED RESULTS FOR LONG AVERAGES
C
      IF ( LOOPS .EQ. 1 .AND. LOOPM .EQ. 1
       1                    .AND. LOOPL.EQ.1 ) THEN
            DO 710 K = 1, 1024
        CUMVAL2(K) = CUMVAL2(K) + VMAG2(K) / 64
710        CONTINUE
C
            DO 720 K = 1, 365
        CUMFRQ2(K) = CUMFRQ2(K) + FMAG2(K+39) / 64
720        CONTINUE
C
      ELSE

               DO 760 K = 1, 1024
               CUMVAL2(K) =
       1       CUMVAL2(K) * (1-FACTR) + VMAG2(K) * FACTR
760       CONTINUE

              DO 780 K = 1, 365
               CUMFRQ2(K) =
       1       CUMFRQ2(K) * (1-FACTR) + FMAG2(K+39) * FACTR
780       CONTINUE
           ENDIF

7000      CONTINUE
C
C    SAVE THE RESULTS & COMPUTE MEAN AND SIGMA OF 1024 BLOCKS
C
                   IF ( LOOPS .EQ. 1 .AND. LOOPM .EQ. 1 .AND.
       1           LOOPL .EQ. 1) THEN
              WRITE(6,60)   ( CUMVAL2(K), K=1,1024 )
              WRITE(7,65)   ( CUMFRQ2(K), K=1, 365 )
                   ENDIF
 60        FORMAT( 8F9.3 )
 65        FORMAT( 8F9.5 )

           CALL STAT(CUMVAL2,1024,VMEAN2,VSIGMA)
C
```

```
C     FREQUENCY DOMAIN
C
               CALL STAT(CUMFRQ2,365,FMEAN2,FSIGMA)

                      WRITE(8,90)   LOOPS, VMEAN2, VSIGMA, FMEAN2, FSIGMA
 90          FORMAT( I5, 4E15.8 )

8000              CONTINUE            ! END OF LOOPS
C
C      COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                            IF (LOOPM .EQ. 1 .AND. LOOPL .EQ. 1) THEN

                      WRITE(16,60)      VMEAN2, VSIGMA
                      WRITE(16,60)   ( CUMVAL2(K), K=1,1024 )
                      WRITE(17,65)      FMEAN2, FSIGMA
                      WRITE(17,65)   ( CUMFRQ2(K), K=1, 365 )
                            ENDIF

9000           CONTINUE     ! END OF LOOPM
C
C      COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                           IF ( LOOPL .EQ. 1 )   THEN

                     WRITE(26,60)      VMEAN2, VSIGMA
                     WRITE(26,60)   ( CUMVAL2(K), K=1,1024 )
                     WRITE(27,65)      FMEAN2, FSIGMA
                     WRITE(27,65)   ( CUMFRQ2(K), K=1, 365 )
                           ENDIF

9900              CONTINUE             ! END OF LOOPL
C
C      COMPUTE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                     WRITE(36,60)      VMEAN2, VSIGMA
                     WRITE(36,60)   ( CUMVAL2(K), K=1,1024 )
                     WRITE(37,65)      FMEAN2, FSIGMA
                     WRITE(37,65)   ( CUMFRQ2(K), K=1, 365 )

                          STOP
                       END

                      SUBROUTINE FFT( XREAL )
                     COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                     COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                     DIMENSION XREAL(1), AMPL(1024), XIMAG(1024)
                     DATA   PI / 3.141592653589 /

C
C      BOX #2 IN FIG. 8.6
```

```
C
                    N2    = N / 2
                    NU1 = NU - 1
                    K      = 0
C
C        BOX #4, #5, #6
C
                    DO 300 L = 1, NU

100              CONTINUE

                      DO 200 I = 1, N2
                                DIVIDE = 2.**NU1
                                J      = K / DIVIDE
                                ARG = 2*PI * IBITR(J,NU) / N
                                C      = COS(ARG)
                                S      = SIN(ARG)
                                K1    = K + 1
                               K1N2 = K1 + N2
                               TREAL = XREAL(K1N2) * C + XIMAG(K1N2) * S
                               TIMAG = XIMAG(K1N2) * C - XREAL(K1N2) * S
                               XREAL(K1N2) = XREAL(K1) - TREAL
                               XIMAG(K1N2) = XIMAG(K1) - TIMAG
                               XREAL(    K1) = XREAL(K1) + TREAL
                               XIMAG(    K1) = XIMAG(K1) + TIMAG
                                K      = K + 1
200            CONTINUE
C
                                K      = K + N2
                              IF (K .LT. N ) GO TO 100
                                K      = 0
                                NU1 = NU1 - 1
                                N2    = N2 / 2
300          CONTINUE
C
C
C
                     DO 500 K = 1, N
                              J = K - 1
                              I = IBITR(J,NU) + 1
                              IF ( I .LE. K ) GO TO 500
                              TREAL = XREAL(K)
                              TIMAG = XIMAG(K)
                                XREAL(K) = XREAL(I) / DIVIDE
                                XIMAG(K) = XIMAG(I) / DIVIDE
                                XREAL(I) = TREAL        / DIVIDE
                                XIMAG(I) = TIMAG        / DIVIDE
500            CONTINUE
C
C     DIVIDE BY 2**NU TO OBTAIN PROPER AMPLITUDE
C
```

```
                    DIVIDE = 2.**NU
                      SUM2 = 0
             DO 600 K = 1, N
                      XREAL(K) = XREAL(K) / DIVIDE
                      XIMAG(K) = XIMAG(K) / DIVIDE
                      FMAG2(K) = XREAL(K)**2 + XIMAG(K)**2

600       CONTINUE

              RETURN
                    END

C
C      TO GENERATE DATA FOR TESTING FFT ALGORITHM
C
           SUBROUTINE GENFUN ( ITEMPR, XREAL )

           COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
           COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
           DIMENSION XREAL(1)
           REAL RAND


           DATA   TWOPI / 6.283185307 /
           F(X)    = AMPL * SIN( WN1*X + ALPH )
C

                   DO 1000 LOOP = 1, ITEMPR

           RAND1 = RAND (    )
           RAND2 = RAND (    )
           RAND3 = RAND (    )

           AMPL    = RAND1
           MEGA    = RAND2 * (IUPP - ILOW) + ILOW
           WN1      = TWOPI * MEGA
           ALPH    = RAND3 * TWOPI

C
C    COMPUTE THE XREAL DATA ARRAY
C
           DO 200 J = 1, N
             XJJ = XJ(J)
             XREAL(J) = XREAL(J) + F( XJJ )
200      CONTINUE
C
1000      CONTINUE
C
C    COMPUTE THE CROSS PRODUCT OF XREAL WITH ITSELF
C
             DO 2000 J = 1, N
```

```
                VMAG2(J) = XREAL(J) * XREAL(J)
2000      CONTINUE

                       RETURN
                        END
C
C    REVERSE THE BIT STRING
C
              FUNCTION IBITR( J, NU )
              DATA JSAVE / 0 /
C
C   INITIALIZE VARIABLES
C
              J1 = J
              IBITR = 0
C
                DO 100 I = 1, NU
              J2 = J1 / 2
              IBITR = IBITR * 2    +    (J1 - 2*J2 )
              J1 = J2
100        CONTINUE

               RETURN
               END
C
                SUBROUTINE STAT ( ARRAY, LENGTH, AMEAN, SIGMA )
                DIMENSION ARRAY(1)
C
C    COMPUTE THE MEAN
C
                SUM1 = 0
               DO 200 J = 1, LENGTH
                SUM1 = SUM1 + ARRAY(J)
200        CONTINUE
                        AMEAN = SUM1 / LENGTH
C
C    COMPUTE THE STANDARD DEVIATION
C
                SUM2 = 0
               DO 300 J = 1, LENGTH
                SUM2 = SUM2 + ( ARRAY(J) - AMEAN )**2
300        CONTINUE
                        SIGMA = SQRT ( SUM2 / (LENGTH-1) )

                RETURN
                END
```

Attachment 3. Program for Long-Term Averages of Measured Data

```
C
C     TO STUDY THE CHANNEL NOISES ARE ELIMINATED BY CPSD
C          WRITTEN BY B.S.MOON, APR.2, 2003
C
                              PROGRAM NOISE
                              COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                              COMMON / SAVE / ILOW, IUPP, STIME, TIMES, PMAG(1024)
                              DIMENSION XREAL(1024), YREAL(1024)
                              DIMENSION CUMVAL2(1024,10000),CUMVALF(1024,10000)
                              DIMENSION CUMPRD2(1024,10000)
                              DIMENSION VALINT3(1024), VALINTF(1024), PRDINT3(1024)
                              DIMENSION VTEMP(1024,64), FTEMP(1024,64), PTEMP(1024,64)
                              DIMENSION VALINT1(1024), PRDINT1(1024), CORR(64)
                              DIMENSION VT(1024), FT(1024), PT(1024)
                                CHARACTER*8 BUF
                              DATA    PI / 3.141592653589 /

C
C      THE ABOVE DIMENSION MUST BE GREATER THAN or EQUAL THE VALUE OF 'N'
C          N IS THE NUMBER OF DATA POINTS; MUST BE OF 2**NU
C          NU IS LOG ( BASE 2 ) OF N = GAMMA IN BRIGHAM'S TEXT BOOK
C
                    OPEN( 6,FILE='V12X2N.DAT')
                    OPEN( 7,FILE='P12X2N.DAT')

                    OPEN(16,FILE='V12Y2N.DAT')
                    OPEN(17,FILE='P12Y2N.DAT')
                    OPEN(18,FILE='S12Y2N.DAT')

                    OPEN(99,FILE='DEBUG.DAT')

                              NU       = 10                              ! USED TO BE 10
                              STIME = 1./ (3* 2.**19)       ! SAMPLING TIME - 1.5M
                              TIMES = 0.                             ! START TIME

                              N         = 2.**NU
                            ITEMPR = 2**12                ! DETERMINES TEMPERATURE


                    ILOW    =    3 * 2**15          ! LOWER BOUND FREQ
                    IUPP    = 1.2* 2**20          ! UPPER BOUND FREQ
                    NLOOPS = 64 * 16              ! @@@@@@@@@@@@@@@@@@

                                DO 100 J = 1, 1024
                              XJ(J) = (J-1) * STIME
100             CONTINUE
C
C   GENERATE THE SEED FOR RANDOM NUMBERS
C
```

```
                CALL TIME ( BUF )
                WRITE(99,10) BUF
 10       FORMAT( A8 )
                REWIND 99
C
                READ(99,20) IHOUR, MIN, ISEC
 20       FORMAT(I2,1X,I2,1X,I2)

                ISTART = IHOUR*3600 + MIN*60 + ISEC
                CALL SEED( ISTART )
C
                          DO 8000 LOOPS = 1, NLOOPS
                      DO 7000    INDX = 1, 64
C
C      INITIALIZE THE ARRAY
C
                 DO 200 J = 1, N
                     XREAL(J) = 0
                     YREAL(J) = 0
200        CONTINUE
C
C    GENERATE THE RANDOM SIGNAL DATA AND COMPUTE fORURIER TRANSFORM
C    VMAG2 COMPUTED BY GENFUN
C      FMAG2 COMPUTED BY FFT
C
                CALL GENFUN ( ITEMPR, XREAL, 1 )

                NOISEF = LOOPS + 2**15                    !CHANNEL NOISE
                  CALL GENFUN ( NOISEF, YREAL, 2 )    !GENRATE FMAG2
C
C     COMPUTE THE PRODUCT
C
                     DO 400 J = 1, 1024
                     PMAG(J) = XREAL(J) * YREAL(J)
400        CONTINUE

                     PRINT *, LOOPS, INDX
C
C    COMPUTE MEAN AND SIGMA OF CROSS POWER SPECTRAL DENSITY
C            FOR XREAL ARRAY FIRST
C
                 CALL STAT ( VMAG2, 1024, VMEANX, VSIGX )
                 CALL STAT ( FMAG2, 1024, VMEANY, VSIGY )
                 CALL STAT (    PMAG, 1024, PMEAN, PSIGMA )

                 DIVIDE = SQRT( VMEANX * VMEANY )
                CORRC    = ABS( PMEAN / DIVIDE )

C      WRITE(8,50)    LOOPS,INDX,VMEANX,VSIGX,
C      1                     VMEANY, VSIGY, PMEAN, CORRC
 50        FORMAT( 2I4, 6E12.6)
```

```
                    CORR(INDX) = CORRC
C
C   SAVE THE COMPUTED RESULTS FOR LONG AVERAGES
C
                     DO 700 K = 1, 1024
                    VTEMP(K,INDX) = VMAG2(K)
                    FTEMP(K,INDX) = FMAG2(K)
                    PTEMP(K,INDX) = PMAG (K)
700         CONTINUE

7000         CONTINUE
C
C    SAVE THE RESULTS & COMPUTE MEAN AND SIGMA OF 1024 BLOCKS
C
                     DO 750 K = 1, 64
                     DO 750 J = 1, 1024
                     CUMVAL2(J,LOOPS) = CUMVAL2(J,LOOPS) + VTEMP(J,K)
                   CUMVALF(J,LOOPS) = CUMVALF(J,LOOPS) + FTEMP(J,K)
                     CUMPRD2(J,LOOPS) = CUMPRD2(J,LOOPS) + PTEMP(J,K)
750          CONTINUE

                   DO 760 J = 1, 1024
                            CUMVAL2(J,LOOPS) = CUMVAL2(J,LOOPS) / 64
                     CUMVALF(J,LOOPS) = CUMVALF(J,LOOPS) / 64
                            CUMPRD2(J,LOOPS) = CUMPRD2(J,LOOPS) / 64
760          CONTINUE

                              IF ( LOOPS .EQ. 1 ) THEN
                    WRITE(6,60)   ( CUMVAL2(K,1), K=1,1024 )
                    WRITE(7,60)   ( CUMVALF(K,1), K=1,1024 )
                    WRITE(7,65)   ( CUMPRD2(K,1), K=1,1024 )
                              ENDIF
 60           FORMAT( 5E15.8 )
 65           FORMAT( 5E15.8 )
C
C    COMPUTE AVERAGE OF THE CORRELATION COEFFICIENTS
C
                           SUM = 0
                           DO 7100 J = 1, 64
                           SUM = SUM + CORR(J)
7100            CONTINUE
C
                           DO 7200 J = 1, 1024
                           VT(J) = CUMVAL2(J,LOOPS)
                           FT(J) = CUMVALF(J,LOOPS)
                           PT(J) = CUMPRD2(J,LOOPS)
7200          CONTINUE
C
                           RATIO = SUM / 64
            CALL STAT ( VT, 1024, VMEAN2, VSIGMA )
```

```
          CALL STAT ( FT, 1024, FMEAN2, FSIGMA )

C
C     FREQUENCY DOMAIN
C
                CALL STAT( PT, 1024, PMEAN, PSIGMA )

          WRITE(18,90) LOOPS,VMEAN2,VSIGMA,FMEAN2,FSIGMA,PMEAN,RATIO
 90       FORMAT( I5, 6E12.5 )

8000            CONTINUE         ! END OF LOOPS
C
C       COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                 DO 2100 K = 1, 1024
                     VALINT3(K) = 0
                     VALINTF(K) = 0
                     PRDINT3(K) = 0
2100        CONTINUE

                     DO 2500 K = 1, 1024
                     DO 2200 L = 1, NLOOPS            ! L=LOOPS
                        VALINT3(K) = VALINT3(K) + CUMVAL2(K,L)
                     VALINTF(K) = VALINTF(K) + CUMVALF(K,L)
                     PRDINT3(K) = PRDINT3(K) + CUMPRD2(K,L)
2200        CONTINUE
2500      CONTINUE
C
                     DO 2700 K = 1, 1024
                     VALINT3(K) = VALINT3(K) / NLOOPS
                     VALINTF(K) = VALINTF(K) / NLOOPS
                        PRDINT3(K) = PRDINT3(K) / NLOOPS
2700        CONTINUE

                     CALL STAT ( VALINT3, 1024, VMEAN, VSIGMA )
                 CALL STAT ( VALINTF, 1024, FMEAN, FSIGMA )
                     CALL STAT ( PRDINT3, 1024, PMEAN, PSIGMA )
                 WRITE(16,60)     VMEAN, VSIGMA
                 WRITE(16,60)   ( VALINT3(K), K=1,1024 )
          RATIO = ABS( PMEAN ) / SQRT( VMEAN*FMEAN )
                 WRITE(17,65)     FMEAN, FSIGMA, PMEAN, RATIO
                 WRITE(17,65)   ( PRDINT3(K), K=1,1024 )

9000         CONTINUE    ! END OF LOOPM
C
C       COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                        STOP
                     END
C
```

```
C         TO GENERATE DATA FOR TESTING FFT ALGORITHM
C
                SUBROUTINE GENFUN ( ITEMPR, XREAL, INDIC )

                COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                DIMENSION XREAL(1)
                REAL RAND


                DATA    TWOPI / 6.283185307 /
                F(X)    = AMPL * SIN( WN1*X + ALPH )
C
                        DO 100 J = 1, 1024
                  XREAL(J) = 0
100             CONTINUE
C
                       DO 1000 LOOP = 1, ITEMPR

                RAND1 = RAND (    )
                RAND2 = RAND (    )
                RAND3 = RAND (    )

                AMPL    = RAND1
                MEGA    = RAND2 * (IUPP - ILOW) + ILOW
                WN1     = TWOPI * MEGA
                ALPH    = RAND3 * TWOPI

C
C     COMPUTE THE XREAL DATA ARRAY
C
                DO 200 J = 1, N
                  XJJ = XJ(J)
                  XREAL(J) = XREAL(J) + F( XJJ )
200        CONTINUE
C
1000         CONTINUE
C
C     COMPUTE THE CROSS PRODUCT OF XREAL WITH ITSELF
C
                               IF ( INDIC .EQ. 1 ) THEN
                  DO 2000 J = 1, N
                  VMAG2(J) = XREAL(J) * XREAL(J)
2000         CONTINUE
                         ELSE
                  DO 3000 J = 1, N
                  FMAG2(J) = XREAL(J) * XREAL(J)
3000         CONTINUE
                         ENDIF
                       RETURN
                        END
```

```
C
C     REVERSE THE BIT STRING
C
                    FUNCTION IBITR( J, NU )
                    DATA JSAVE / 0 /
C
C    INITIALIZE VARIABLES
C
                    J1 = J
                    IBITR = 0
C
                       DO 100 I = 1, NU
                    J2 = J1 / 2
                    IBITR = IBITR * 2    +    (J1 - 2*J2 )
                    J1 = J2
100          CONTINUE

                     RETURN
                     END
C
                       SUBROUTINE STAT ( ARRAY, LENGTH, AMEAN, SIGMA )
                       DIMENSION ARRAY(1)
C
C     COMPUTE THE MEAN
C
                       SUM1 = 0
                     DO 200 J = 1, LENGTH
                       SUM1 = SUM1 + ARRAY(J)
200          CONTINUE
                              AMEAN = SUM1 / LENGTH
C
C     COMPUTE THE STANDARD DEVIATION
C
                       SUM2 = 0
                     DO 300 J = 1, LENGTH
                       SUM2 = SUM2 + ( ARRAY(J) - AMEAN )**2
300          CONTINUE
                              SIGMA = SQRT ( SUM2 / (LENGTH-1) )

                       RETURN
                       END
```

Attachment 4. Program for Testing the Channel Noise after CPDS

```
C
C     TO STUDY THE CHANNEL NOISES ARE ELIMINATED BY CPSD
C          WRITTEN BY B.S.MOON, APR.2, 2003
C
                          PROGRAM NOISE
                          COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                          COMMON / SAVE / ILOW, IUPP, STIME, TIMES, PMAG(1024)
                          DIMENSION XREAL(1024), YREAL(1024)
                          DIMENSION CUMVAL2(1024,10000),CUMVALF(1024,10000)
                          DIMENSION CUMPRD2(1024,10000)
                          DIMENSION VALINT3(1024), VALINTF(1024), PRDINT3(1024)
                          DIMENSION VTEMP(1024,64), FTEMP(1024,64), PTEMP(1024,64)
                          DIMENSION VALINT1(1024), PRDINT1(1024), CORR(64)
                          DIMENSION VT(1024), FT(1024), PT(1024)
                            CHARACTER*8 BUF
                          DATA    PI / 3.141592653589 /

C
C      THE ABOVE DIMENSION MUST BE GREATER THAN or EQUAL THE VALUE OF 'N'
C         N IS THE NUMBER OF DATA POINTS; MUST BE OF 2**NU
C         NU IS LOG ( BASE 2 ) OF N = GAMMA IN BRIGHAM'S TEXT BOOK
C
                 OPEN( 6,FILE='V12X2N.DAT')
                 OPEN( 7,FILE='P12X2N.DAT')

                 OPEN(16,FILE='V12Y2N.DAT')
                 OPEN(17,FILE='P12Y2N.DAT')
                 OPEN(18,FILE='S12Y2N.DAT')

                 OPEN(99,FILE='DEBUG.DAT')

                          NU       = 10                        ! USED TO BE 10
                          STIME = 1./ (3* 2.**19)       ! SAMPLING TIME - 1.5M
                          TIMES = 0.                          ! START TIME

                          N        = 2.**NU
                         ITEMPR = 2**12                ! DETERMINES TEMPERATURE


                 ILOW    =    3 * 2**15           ! LOWER BOUND FREQ
                 IUPP    = 1.2* 2**20           ! UPPER BOUND FREQ
                 NLOOPS = 64 * 16               ! @@@@@@@@@@@@@@@@@@

                           DO 100 J = 1, 1024
                         XJ(J) = (J-1) * STIME
100              CONTINUE
C
C    GENERATE THE SEED FOR RANDOM NUMBERS
C
```

```
                  CALL TIME ( BUF )
                  WRITE(99,10) BUF
 10          FORMAT( A8 )
                  REWIND 99
C
                  READ(99,20) IHOUR, MIN, ISEC
 20          FORMAT(I2,1X,I2,1X,I2)

                  ISTART = IHOUR*3600 + MIN*60 + ISEC
                  CALL SEED( ISTART )
C
                                    DO 8000 LOOPS = 1, NLOOPS
                              DO 7000    INDX = 1, 64
C
C       INITIALIZE THE ARRAY
C
                    DO 200 J = 1, N
                           XREAL(J) = 0
                           YREAL(J) = 0
200            CONTINUE
C
C     GENERATE THE RANDOM SIGNAL DATA AND COMPUTE fORURIER TRANSFORM
C     VMAG2 COMPUTED BY GENFUN
C       FMAG2 COMPUTED BY FFT
C
                  CALL GENFUN ( ITEMPR, XREAL, 1 )

                  NOISEF = LOOPS + 2**15                    !CHANNEL NOISE
                    CALL GENFUN ( NOISEF, YREAL, 2 )    !GENRATE FMAG2
C
C      COMPUTE THE PRODUCT
C
                           DO 400 J = 1, 1024
                           PMAG(J) = XREAL(J) * YREAL(J)
400            CONTINUE

                           PRINT *, LOOPS, INDX
C
C     COMPUTE MEAN AND SIGMA OF CROSS POWER SPECTRAL DENSITY
C                FOR XREAL ARRAY FIRST
C
                    CALL STAT ( VMAG2, 1024, VMEANX, VSIGX )
                    CALL STAT ( FMAG2, 1024, VMEANY, VSIGY )
                    CALL STAT (    PMAG, 1024, PMEAN, PSIGMA )

                    DIVIDE = SQRT( VMEANX * VMEANY )
                  CORRC    = ABS( PMEAN / DIVIDE )

C       WRITE(8,50)    LOOPS,INDX,VMEANX,VSIGX,
C       1                         VMEANY, VSIGY, PMEAN, CORRC
 50            FORMAT( 2I4, 6E12.6)
```

```
                    CORR(INDX) = CORRC
C
C   SAVE THE COMPUTED RESULTS FOR LONG AVERAGES
C
                     DO 700 K = 1, 1024
                    VTEMP(K,INDX) = VMAG2(K)
                    FTEMP(K,INDX) = FMAG2(K)
                    PTEMP(K,INDX) = PMAG (K)
700       CONTINUE

7000       CONTINUE
C
C    SAVE THE RESULTS & COMPUTE MEAN AND SIGMA OF 1024 BLOCKS
C
                     DO 750 K = 1, 64
                     DO 750 J = 1, 1024
                     CUMVAL2(J,LOOPS) = CUMVAL2(J,LOOPS) + VTEMP(J,K)
                   CUMVALF(J,LOOPS) = CUMVALF(J,LOOPS) + FTEMP(J,K)
                     CUMPRD2(J,LOOPS) = CUMPRD2(J,LOOPS) + PTEMP(J,K)
750        CONTINUE

                   DO 760 J = 1, 1024
                            CUMVAL2(J,LOOPS) = CUMVAL2(J,LOOPS) / 64
                     CUMVALF(J,LOOPS) = CUMVALF(J,LOOPS) / 64
                            CUMPRD2(J,LOOPS) = CUMPRD2(J,LOOPS) / 64
760        CONTINUE

                              IF ( LOOPS .EQ. 1 ) THEN
                    WRITE(6,60)    ( CUMVAL2(K,1), K=1,1024 )
                    WRITE(7,60)    ( CUMVALF(K,1), K=1,1024 )
                    WRITE(7,65)    ( CUMPRD2(K,1), K=1,1024 )
                              ENDIF
 60         FORMAT( 5E15.8 )
 65         FORMAT( 5E15.8 )
C
C    COMPUTE AVERAGE OF THE CORRELATION COEFFICIENTS
C
                           SUM = 0
                           DO 7100 J = 1, 64
                           SUM = SUM + CORR(J)
7100          CONTINUE
C
                           DO 7200 J = 1, 1024
                           VT(J) = CUMVAL2(J,LOOPS)
                           FT(J) = CUMVALF(J,LOOPS)
                           PT(J) = CUMPRD2(J,LOOPS)
7200        CONTINUE
C
                           RATIO = SUM / 64
             CALL STAT ( VT, 1024, VMEAN2, VSIGMA )
```

```
              CALL STAT ( FT, 1024, FMEAN2, FSIGMA )


C
C     FREQUENCY DOMAIN
C
                        CALL STAT( PT, 1024, PMEAN, PSIGMA )

               WRITE(18,90) LOOPS,VMEAN2,VSIGMA,FMEAN2,FSIGMA,PMEAN,RATIO
 90           FORMAT( I5, 6E12.5 )

8000                     CONTINUE              ! END OF LOOPS
C
C        COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                         DO 2100 K = 1, 1024
                              VALINT3(K) = 0
                              VALINTF(K) = 0
                              PRDINT3(K) = 0
2100            CONTINUE

                              DO 2500 K = 1, 1024
                              DO 2200 L = 1, NLOOPS                  ! L=LOOPS
                                VALINT3(K) = VALINT3(K) + CUMVAL2(K,L)
                              VALINTF(K) = VALINTF(K) + CUMVALF(K,L)
                              PRDINT3(K) = PRDINT3(K) + CUMPRD2(K,L)
2200            CONTINUE
2500          CONTINUE
C
                              DO 2700 K = 1, 1024
                              VALINT3(K) = VALINT3(K) / NLOOPS
                              VALINTF(K) = VALINTF(K) / NLOOPS
                                PRDINT3(K) = PRDINT3(K) / NLOOPS
2700            CONTINUE

                              CALL STAT ( VALINT3, 1024, VMEAN, VSIGMA )
                         CALL STAT ( VALINTF, 1024, FMEAN, FSIGMA )
                              CALL STAT ( PRDINT3, 1024, PMEAN, PSIGMA )
                         WRITE(16,60)        VMEAN, VSIGMA
                         WRITE(16,60)    ( VALINT3(K), K=1,1024 )
               RATIO = ABS( PMEAN ) / SQRT( VMEAN*FMEAN )
                         WRITE(17,65)        FMEAN, FSIGMA, PMEAN, RATIO
                         WRITE(17,65)    ( PRDINT3(K), K=1,1024 )

9000              CONTINUE      ! END OF LOOPM
C
C        COMPUTE INTERMEDIATE MEAN AND SIGMA & SAVE THE FINAL RESULTS
C
                                STOP
                              END
C
```

```
C        TO GENERATE DATA FOR TESTING FFT ALGORITHM
C
                SUBROUTINE GENFUN ( ITEMPR, XREAL, INDIC )

                COMMON NU, N, NFREQ, XJ(1024), VMAG2(1024), FMAG2(1024)
                COMMON / SAVE / ILOW, IUPP, STIME, TIMES, CUMX(1024)
                DIMENSION XREAL(1)
                REAL RAND


                DATA   TWOPI / 6.283185307 /
                F(X)   = AMPL * SIN( WN1*X + ALPH )
C
                     DO 100 J = 1, 1024
                  XREAL(J) = 0
100            CONTINUE
C
                    DO 1000 LOOP = 1, ITEMPR

                RAND1 = RAND (   )
                RAND2 = RAND (   )
                RAND3 = RAND (   )

                AMPL   = RAND1
                MEGA   = RAND2 * (IUPP - ILOW) + ILOW
                WN1    = TWOPI * MEGA
                ALPH   = RAND3 * TWOPI

C
C    COMPUTE THE XREAL DATA ARRAY
C
                DO 200 J = 1, N
                  XJJ = XJ(J)
                  XREAL(J) = XREAL(J) + F( XJJ )
200       CONTINUE
C
1000       CONTINUE
C
C    COMPUTE THE CROSS PRODUCT OF XREAL WITH ITSELF
C
                          IF ( INDIC .EQ. 1 ) THEN
                  DO 2000 J = 1, N
                  VMAG2(J) = XREAL(J) * XREAL(J)
2000       CONTINUE
                      ELSE
                  DO 3000 J = 1, N
                  FMAG2(J) = XREAL(J) * XREAL(J)
3000       CONTINUE
                      ENDIF
                    RETURN
                     END
```

```
C
C     REVERSE THE BIT STRING
C
                          FUNCTION IBITR( J, NU )
                          DATA JSAVE / 0 /
C
C    INITIALIZE VARIABLES
C
                          J1 = J
                          IBITR = 0
C
                             DO 100 I = 1, NU
                          J2 = J1 / 2
                          IBITR = IBITR * 2    +    (J1 - 2*J2 )
                          J1 = J2
100           CONTINUE

                           RETURN
                           END
C
                             SUBROUTINE STAT ( ARRAY, LENGTH, AMEAN, SIGMA )
                             DIMENSION ARRAY(1)
C
C     COMPUTE THE MEAN
C
                             SUM1 = 0
                           DO 200 J = 1, LENGTH
                             SUM1 = SUM1 + ARRAY(J)
200           CONTINUE
                                    AMEAN = SUM1 / LENGTH
C
C     COMPUTE THE STANDARD DEVIATION
C
                             SUM2 = 0
                           DO 300 J = 1, LENGTH
                             SUM2 = SUM2 + ( ARRAY(J) - AMEAN )**2
300           CONTINUE
                                    SIGMA = SQRT ( SUM2 / (LENGTH-1) )

                             RETURN
                             END
```

서 지 정 보 양 식

수행기관보고서번호	위탁기관보고서번호	표준보고서번호	INIS 주제코드
KAERI/ TR-2762 /2004			

제목 / 부제

Johnson Noise 온도계의 디지털 신호처리 - Johnson Noise의 시계열 분석

연구책임자 및 부서명	문 병 수(계측제어 인간공학 연구부)
연 구 자 및 부 서 명	황인구,정종은, 권기춘(계측제어 인간공학 연구부)

Roger A. Kisner, David E. Holcomb (Oak Ridge national Laboratory)

출 판 지		발행기관	한국원자력연구소	발행 년	2004.6
페 이 지	83 p.	도 표	있음(O), 없음( )	크 기	Cm.
참고사항					

공개여부	공개(O), 비공개( )	보고서종류	기술보고서
비밀여부	대외비 ( ), __ 급비밀		
연구위탁기관	과학기술부	계약 번호	

초록 (15-20줄내외)

이 보고서에서는 첫째, 단일주기 신호들의 합으로부터 생성된 Random 신호가 Continuous Markov Process 를 형성하며 따라서 Gaussian White Noise 가 됨을 보였다. 이 Random 신호를 사용한 Simulation 을 통하여 Johnson Noise 온도계를 원자로 냉각재 온도측정에 사용할 경우 측정값의 상대오차가 0.14%미만인 매우 정확한 온도 값 측정이 가능함을 확인하였다. 둘째, Band-Pass Filter 를 거친 Johnson Noise 의 지정된 Filter Band 에 대한 최적의 Sampling Rate 결정에 관한 내용을 기술하였다. 또한, Johnson Noise 의 Linearity 문제와 온도에 따라 상대오차가 어떻게 변화하는지에 대한 분석 결과를 기술하였다. 셋째, 아주 단순한 전기회로로부터 장시간 동안 취득한 일련의 Johnson Noise 신호에 대한 분석 결과를 기술하였다. 취득한 데이터에는 약간의 Channel Noise 가 포함되어 있으며 이같이 Channel Noise 가 포함된 상태에서도 Continuous Markov Process 또는 Gaussian White Noise 의 성질은 유지가 됨을 확인하였다. 넷째, 온도의 값이 변화하고 있을 때 Long-term Average 또는 Moving Average 사용으로 인하여 발생하는 Time Lag 문제 해결을 위한 알고리즘에 대하여 기술하였다. 이 알고리즘은 Haar Wavelet 을 기반으로 하여 작성된 것으로 Long-Term Average 대신 Wavelet 을 사용한 근사값을 취함으로써 Time Lag 을 현격히 줄인 것이다.

주제명키워드 (10단어내외)

Johnson Noise Thermometry; Random Noise; Channel Noise; Haar Wavelet; Gaussian White Noise; Continuous Markov Process; Moving Average; Accuracy; Cross Power Spectral Density; Correlated Voltage

BIBLIOGRAPHIC INFORMATION SHEET

<table>
<tr><td>Performing Org. Report No.</td><td>Sponsoring Org. Report No.</td><td>Standard Report No.</td><td>INIS Subject Code</td></tr>
<tr><td>KAERI/TR-2762/2004</td><td></td><td></td><td></td></tr>
</table>

<table>
<tr><td>Title / Subtitle</td><td></td></tr>
<tr><td colspan="2">Digital Signal Processing for the Johnson Noise Thermometry
- A Time Series Analysis of the Johnson Noise</td></tr>
</table>

<table>
<tr><td>Project Manager and Department</td><td>Byung Soo Moon
(Instrumentation & Control - Human Factors Department)</td></tr>
<tr><td>Researcher and Department</td><td>In-Koo Hwang, Chong-Eun Chung, Kee Choon Kwon
(Instrumentation & Control - Human Factors)</td></tr>
<tr><td colspan="2">Roger A. Kisner, David E. Holcomb (Oak Ridge national Laboratory)</td></tr>
</table>

<table>
<tr><td>Publication Place</td><td></td><td>Publisher</td><td>Korea Atomic Energy Research Institute</td><td>Publication Date</td><td>June 2004</td></tr>
<tr><td>Page</td><td>83 p.</td><td>Ill. & Tab.</td><td>Yes(O), No ( )</td><td>Size</td><td>Cm.</td></tr>
<tr><td>Note</td><td colspan="5"></td></tr>
</table>

<table>
<tr><td>Open</td><td>Open(O), Closed( )</td><td rowspan="2">Report Type</td><td rowspan="2">Technical Report</td></tr>
<tr><td>Classified</td><td>Restricted( ), ___Class Document</td></tr>
<tr><td>Sponsoring Org.</td><td>Ministry of Science And Technology</td><td>Contract No.</td><td></td></tr>
</table>

Abstract (15-20 Lines)

In this report, we first proved that a random signal obtained by taking the sum of a set of single frequency signals generates a continuous Markov process. We used this random signal to simulate the Johnson noise and verified that the Johnson noise thermometry can be used to improve the measurements of the reactor coolant temperature within an accuracy of below 0.14%. Secondly, by using this random signal we determined the optimal sampling rate when the frequency band of the Johnson noise signal is given. Also the results of our examination on how good the linearity of the Johnson noise is and how large the relative error of the temperature could become when the temperature increases are described. Thirdly, the results of our analysis on a set of the Johnson noise signal blocks taken from a simple electric circuit are described. We showed that the properties of the cntinuous Markov process are satisfied even when some channel noises are present. Finally, we describe the algorithm we devised to handle the problem of the time lag in the long-term average or the moving average in a transient state. The algorithm is based on the Haar wavelet and is to estimate the transient temperature that has much smaller time delay. We have shown that the algorithm can track the transient temperature successfully.

Subject Keywords (About 10 words)

Johnson Noise Thermometry; Random Noise; Channel Noise;Haar Wavelet; Gaussian White Noise; Continuous Markov Process; Moving Average; Accuracy; Cross Power Spectral Density; Correlated Voltage