MIT/LCS/TM-371

NATURAL RANDOM NUMBERS

David K. Gifford

September 1988

MIT/LCS/TM-371

Natural Random Numbers

David K. Gifford

August 1988

Abstract

We present a method for generating random numbers from natural noise sources that is able to produce random numbers to any desired level of perfection. The method works by transducing a physical noise source to generate a stream of biased natural bits, and then applying an unbiasing algorithm. The Wiener-Kinchine relation is used to derive the autocorrelation present in the stream of biased bits and to define safe sampling rates. Experimental results from an implementation of our method support our analysis. One consequence of our analysis is that a broad class of natural random number generators, including ours, can not generate absolutely perfect random numbers.

Categories and Subject Descriptions: G.3 [**Mathematics of Computing**]–Probability and Statistics: *Random Number Generation*

General Terms: Design, Experimentation, Theory

Additional Key Words and Phrases: natural random number, perfect random number, pseudo-random number

1 Natural Random Bits

Random numbers are useful for a wide variety of computer applications, such as cryptographic keys and simulation. The common intuition of a random number is that it is impossible to predict its value, even with complete knowledge of all values that have been previously produced by its source. We will slightly refine this intuitive notion with a few definitions, and then we will discuss how one can generate random numbers.

We define a *random number* to be a discrete value that is a single trial of a *random process.* A random process is a series of trials, with each trial using identical *process parameters.* A process' parameters completely characterize the possible outcomes of a trial, the respective probabilities of the possible outcomes, and how trials depend on one another. For example, the series of head-tail values that results from flipping a fair coin is a random process. However, if an unfair coin is used for some flips then the series is not a random process because a process parameter (the probability of heads) is not identical for all of the trials. The parameters of a given random process may or may not be known, but they can be estimated by observing the process for a number of trials.

We will call a process *random* if and only if its parameters do not completely determine the values of its trials. A related approach can be found in Kolmogorov Complexity [Kol65] which measures the randomness of a string by its shortest computational description.

In order to simplify our presentation we will focus on binary random processes. We can do this without any loss of generality because integer valued random numbers can be easily and efficiently generated from random bits [Gill72].

We classify random bits into three categories based on their generating processes:

- *Perfect random bits* are generated by an unbiased Bernoulli process where trials are completely independent [Drake67]. Perfect random bits represent a theoretical ideal that we would like a random bit generator to achieve in practice. Consumers of random bits – algorithms and systems – almost always assume a ready supply of perfect random bits.

- *Pseudo-random bits* are generated by an algorithmic process that is completely determined by its *seed.* Thus an algorithmic process is not random, and pseudo-random bits are not random. An observer with unlimited computational resources can compute the "secret" seed of a pseudo-random bit generator and thus know all of its past and future values. However, an observer with bounded computational resources may not be able to tell the difference between pseudo-random bits and perfectly random bits. For example, the *cryptographically strong pseudo-random number generator* described by Blum and Micali [Blum84a] will appear to produce perfect random bits to an observer that is limited to a probabilistic polynomial-time observation algorithm, if the discrete logarithm problem can not be solved in probabilistic polynomial-time.

- *Natural random bits* are generated by transducing a natural random process such as shot noise, radioactive decay, or coin flips. Natural random bits are required by pseudo-random bit generators to serve as seeds. Thus a source of natural random bits is an essential part of any random bit generator.

We describe a technique for generating natural random bits where the generated bits approach perfect random bits with an increase in the time between observations. In the remainder of this paper we review previous work (Section 2), present our method for random number generation (Section 3), discuss experimental results from an implementation of our method (Section 4), and conclude with some implications of our analysis for a broad class of natural random number generators (Section 5).

2 Previous work

The difference between our work and previous work lies in the way that we sample and unbias bits from a natural noise source. As we will explain further in Section 3, existing random number generators use unbiasing algorithms that increase the dependence between output bits. The following is a brief survey of some of these previous random number generation techniques:

- RAND periodically samples a counter that is incremented by a free running oscillator [Rand55]. The original version of the machine had to be modified before it produced statistically acceptable random numbers.

- ERNIE [Thomson59] periodically samples a counter that is incremented by random pulses produced by neon discharge tubes. ERNIE was used to produce Britan's Premium Savings Bond prize numbers.

- The TX-2 computer included a natural random number generator that is based on radioactive decay. This random number generator was subjected to extensive empirical tests [Kleinrock60].

- Manelis [Manelis61] generates a random telegraph wave by triggering a flip-flop with the output of a scintillation detector. The output is designed to be used in conjunction with an analog computer.

- Vincent [Vincent70] periodically samples a one-bit counter that is incremented random sequence of bits derived from a natural noise source. Vincent [Vincent71] notes that the "dead time" effects of a white noise quantizer can introduce undesirable correlation. A single card version of this random number generator was produced [Maddocks72].

- Borbas et al [Borbas76] periodically sample a free running counter. The output of the counter is used to produce a random digit with a prespecified maximum value.

- Castanie [Castanie78] periodically samples a one-bit counter that is incremented with the XOR of a sequence of random bits and a sequence of alternating one and zero bits. The XOR combining step is an unbiasing process; because the fixed pattern is unbiased the result of the XOR will also be unbiased. Bias is also reduced by regulating the noise quantizer threshold level with feedback.

- Simmons [Simmons80] periodically samples a one-bit counter that is incremented with pulses produced by a physical noise source. Simmons also describes how unbiased numbers can be processed to have a specified bias.

- The AT&T T7001 [ATT85] periodically samples a one-bit counter that is incremented with pulses produced by a physical noise source. The bits are further processed by XORing output bits with previous bits.

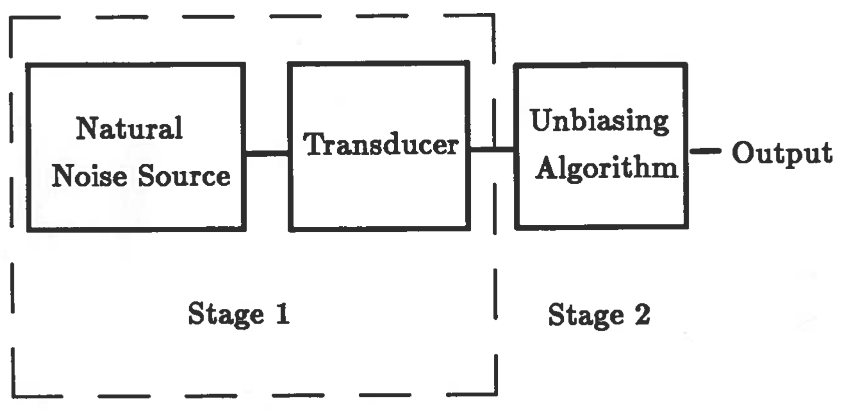


Figure 1: Random Number Generator Block Diagram

3 Natural Random Bit Generation

Our natural random bit generator (Figure 1) operates into two stages:

- Stage 1 transduces a physical noise source to generate a stream of biased natural bits. The sampling time between observations is chosen such that the observed bits are practically independent (see below). Thus the output of Stage 1 is a binary random process with practically independent trails and unknown bias. We assume that the parameters of the natural random process we are observing do not change, and that the threshold level of our transducer drifts very slowly.

  Stage 1 could use multiple physical noise sources. Vazirani [Vazirani87] shows how to combine the output of two independent imperfect random sources to create a random sequence that has better properties than either of the two input sources.

- Stage 2 unbiases the stream of biased and practically independent bits from Stage 1. The first and simplest unbiasing algorithm is due to von Neumann [vonNeumann51]. This algorithm consumes two input bits at a time, and produces output values according to the function

$$00 \Rightarrow \phi$$

$$\begin{aligned} 01 &\Rightarrow 0 \\ 10 &\Rightarrow 1 \\ 11 &\Rightarrow \phi \end{aligned}$$

where ϕ represents no output value. If we assume that the input bits are drawn from a random process (the bias of the input bits is fixed) and the input bits are from independent trials, then it is easy to see that the output of Stage 2 will be perfect random bits.

The maximum efficiency of von Neumann's algorithm is 4 input bits to 1 output bit on average. Elias [Elias72] and Gill [Gill72] present more efficient unbiasing algorithms and show that the efficiency of unbiasing can approach the entropy of the bits to be unbiased.

Blum [Blum84a] presents an algorithm that will unbias bits from a Markov process with no hidden states. Unlike other unbiasing algorithms, Blum's algorithm does not assume independent trails. However, it does make assumptions about the structure of the random process it is observing.

The primary way that our two stage approach differers from previous work is that we clearly separate the production of natural random bits (Stage 1) from unbiasing (Stage 2). Existing random number generators have not clearly separated these two processes, and thus perform unbiasing in a manner that always increases output bit dependence:

- The most popular unbiasing algorithm is based on incrementing a counter with with a natural noise source. In such a scheme output bit n is the mod 2 sum of input samples intervals 0 through n. We assume here without loss of generality that the input sampling rate is greater than the Nyquist rate of the noise source. This unbiasing algorithm is entropy preserving because the input string can be computed from the output string. Thus bits in the output string must be more dependent than bits in the input string because the entropies of the two strings are equal and the output string is less biased than the input string.

- The technique due to Castanie [Castanie78] unbiases a string of input bits by XORing it with a string that consists of alternating 0 and 1 bits. This technique converts all of the bias in the input bit string into dependence in the output string, as the output string is unbiased.

The unbiasing technique we use does not necessarily increase the dependence of output bits. However, it does depend on having practically independent input bits. We will now determine how the rate at which Stage 1 samples its physical noise source will influence the independence of its output bits. At an intuitive level we must wait long enough between samples to give our our noise source time to change, and thus the time we must wait is directly related to how fast the noise source changes value. We will make this reasoning about sampling rate bounds precise by relating the power spectral density of a natural noise source to its autocorrelation.

Let X be the random process at the output of Stage 1's transducer, assuming a perfect transducer. The power spectral density of X is defined by

$$S_X(\omega) = \lim_{T\to\infty} \frac{E[|F_{XT}(j\omega)^2|]}{2T} \tag{1}$$

where $F_{XT}(j\omega)$ is the fourier transform of X over samples from the period $-T$ to T. From an intuitive perspective, the spectral density $S_X(\omega)$ is the average power of X within a bandwidth of one hertz centered at ω in units of volts^2 per hertz.

$S_X(\omega)$ can be estimated by computing Equation 1 over an observation period of a few seconds with a spectrum analyzer. $S_X(\omega)$ can also be estimated from the power spectral density information provided by the manufacturer of the natural noise source and transducer in use. The estimated spectrum must be reduced by the noise introduced by the transducer in order for the spectral estimate to correspond to the natural noise source.

The autocorrelation function $R_X(\tau)$ of X is defined to be

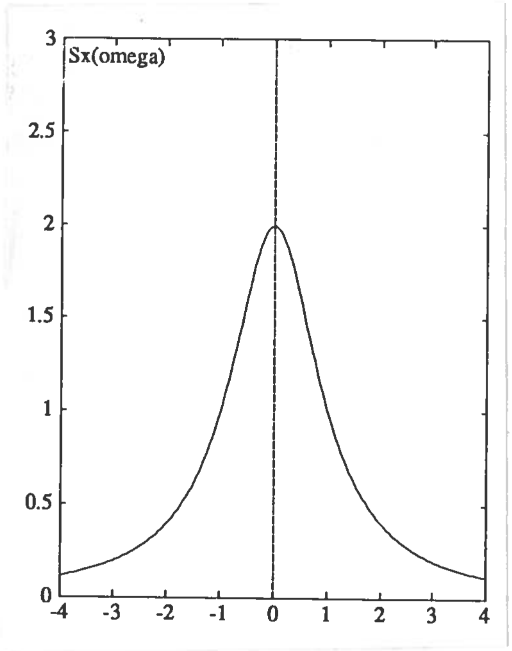
$$R_X(\tau) = E[X(t)X(t+\tau)] \tag{2}$$

If the expected value of X is zero, then $R_X(\tau)$ must be zero for trials taken τ seconds apart to be completely independent.

The autocorrelation function $R_X(\tau)$ can be shown to be the inverse Fourier transform of $S_X(\omega)$:

$$R_X(\tau) = F^{-1}\{S_X(\omega)\} \tag{3}$$

which is known as the Wiener-Kinchine relation.



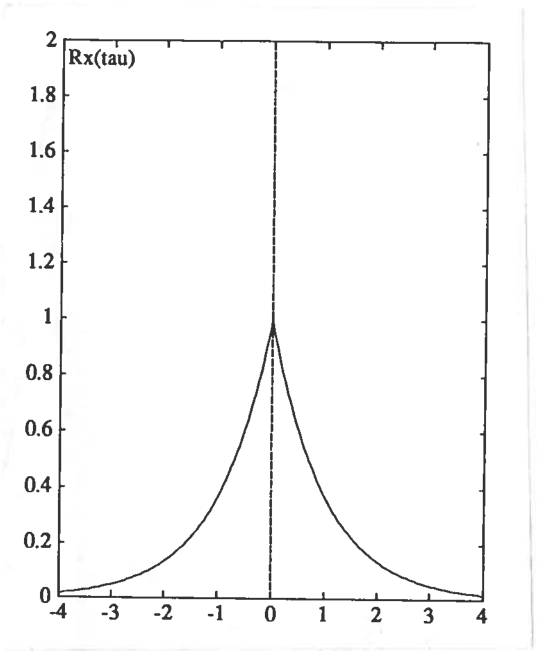


Figure 2: $S_X(\omega)$ and $R_X(\tau)$. The $S_X(\omega)$ x-axis is in units of β and the y-axis is in units of A/β. The $R_X(\tau)$ x-axis is in units of β and the y-axis is in units of A.

If we assume that our random process X has a power spectral density function of

$$S_X(\omega) = \frac{2A\beta}{\omega^2 + \beta^2} \qquad A > 0, \beta > 0 \tag{4}$$

then its autocorrelation function will be

$$R_X(\tau) = Ae^{-\beta|\tau|} \tag{5}$$

The plot of $S_X(\omega)$ and $R_X(\tau)$ in Figure 2 shows that $R_X(\tau)$ decreases exponentially with time and only reaches 0 in the limit. Equation 5 provides a way of bounding the autocorrelation at the output of our transducer after time τ given a noise source of bandwidth β.

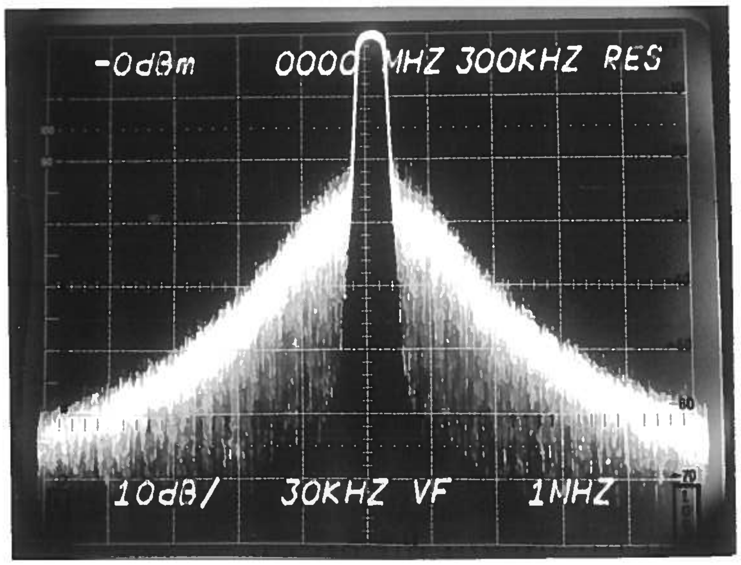


Figure 3: Power spectrum at the output of the CA3140 amplifier. Each vertical division is 10db of power, and each horizontal division is 1 MHz. DC is in the center of the figure.

4 Experimental Results

Stage 1 of our natural random number generator consists of a KN1201 noise diode and supporting circuitry [KSW], followed by a CA3140 amplifier, and an AMD686 comparator. The comparator samples the output of the CA3140 every 100ns to see if it is above zero and latches the resulting value.

The random number generator was included on a cryptography board for the Xerox D0 computer, and care was taken to provide the random number generator with an isolated power supply. Each section of the random number generator was further decoupled with RC components in order to reduce unwanted correlation.

Figure 3 shows the power spectral density of our natural noise source as measured by a spectrum analyzer. Figure 4 shows autocorrelation of the natural random bit stream at the output of the comparator. The estimates

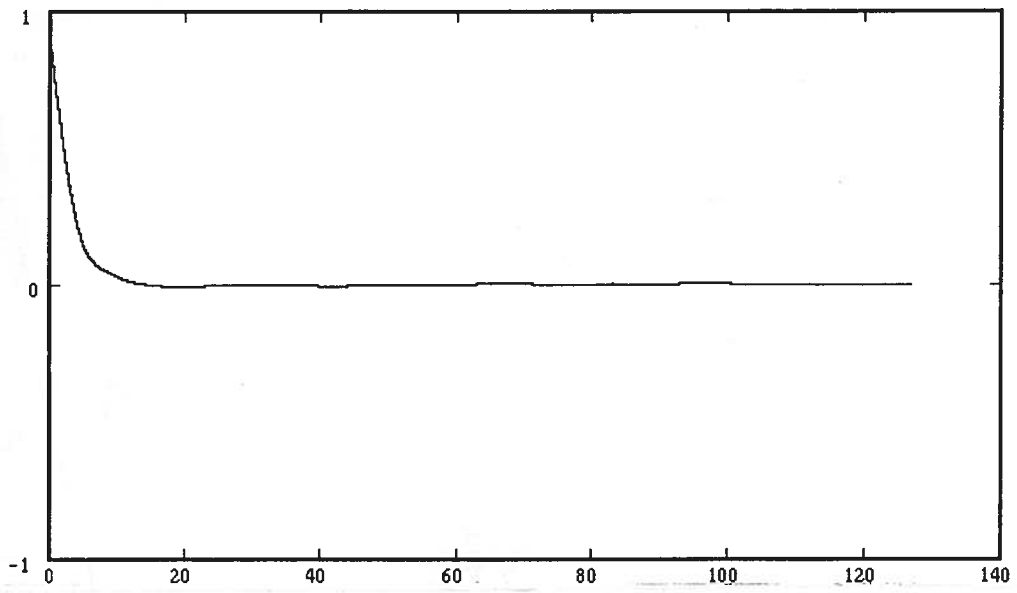


Figure 4: Autocorrelation of the comparator output bit stream that was computed from a sample of 1048576 bits observed 100 nanoseconds apart over 104 milliseconds. X-axis units are 10^{-7} seconds.

of β derived from Figures 3 and 4 are about a factor of five different. Figure 3 interpreted with respect to Equation 4 suggests that β is about 500 Khz. Figure 4 interpreted with respect to Equation 5 suggests that β is 2.3 Mhz. We are unsure why these two figures are different, but one possibility is that Figure 3 is not an accurate reflection of the power spectral density as seen by the AMD686 comparator. This could result if the measurement was performed at an improper impedance.

Stage 2 is implemented by software that performs the von Neumann unbiasing algorithm. We ensured that the time between observations of the Stage 1 output value were always greater than 25 microseconds. Thus by Equation 5 and the observed β of 500 KHz (Figure 3) we expect that the autocorrelation of the bit stream that is input to Stage 2 is less than e^{-12}. In retrospect we would lower the sampling rate in order to further reduce autocorrelation.

We subjected a Stage 2 output sequence of 262144 bits to frequency tests on individual bits, pairs of bits, and bytes. The Chi-Square statistics were 0.41 for the bias test (52% confidence), 5.59 for the pair test (13% confidence), and 244.25 for the byte test (46% confidence). We also computed the lengths of sequences of zeros between ones from 0 to 24. The Chi-Square statistic for the gap test was 19.3 (72% confidence). These statistics are not good enough to accept the null hypothesis that the Stage 2 output sequence is in fact random. Unfortunately we no longer have access to the random number generator to conduct further tests.

5 In search of perfect random bits

One consequence of Equation 5 is that our random number generation technique can not generate perfect random bits. We can approach perfection by waiting longer and longer between Stage 1 samples, but the autocorrelation in our samples will never reach zero. Thus our output bit stream will always contain some dependence, although we may not be able to detect it. This dependence is caused by the finite bandwidth of our transducer and the natural noise source we are observing.

It appears that in order to produce absolutely perfect random bits an algorithm will have to be developed that can unbias and uncorrelate bits from a random process of unknown structure. The existence of such an algorithm is an interesting open question.

Acknowledgments: I would like to thank Manuel Blum, Peter Elias, John Gill, Shafi Goldwasser, Silvio Micali, Ron Rivest, Larry Stewart, Gerry Sussman, and Chuck Thacker for useful discussions. A portion of the work described here was performed at the Xerox Palo Alto Research Center.

References

[ATT85] AT&T, "Data Sheet, T7001 Random Number Generator", Berkeley Hights, New Jersey, May 1985.

[Blum84a] Blum, M., "Independent Unbiased Coin Flips from a Correlated Biased Source: a Finite State Markov Chain", Proc. 25th ACM Foundations of Computer Science Conference, October 1984, pp. 425-433.

[Blum84b] Blum, M., and Micali, S. "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", SIAM J. Computing 13, 4 (November 1984), pp. 850-864.

[Castanie78] Castanie, F., "Generation of Random Bits with Accurate and Reproducible Statistical Properties", Proceedings of the IEEE, Vol. 66, No. 7, July 1978, pp. 865-870.

[Drake67] Drake, A., *Fundamentals of Applied Probability Theory*, McGraw Hill, New York, 1967.

[Elias72] Elias, P., "The Efficient Construction of an Unbiased Random Source", Annals of Math. Statistics 1972, Vol. 43, No. 3, pp. 865-870.

[Gill72] Gill, J.T., "Probabilistic Turing Machines and Complexity of Computation", Ph.D. Thesis, University of California, Berkeley, 1972.

[Kleinrock60] Kleinrock, L., "A Program for Testing Sequences of Random Numbers", Report 51G-0018, MIT Lincoln Laboratory, October 25, 1960.

[Kol65] Kolmogorov, A., "Three Approaches to the Concept of 'The Amount of Information"', Prob. Inf. Transmission 1, 1 (1965).

[KSW] "Noise Diodes: KN1201, KN1201A, KN1301, KN1401", KSW Electronics Corporation, Burlington, MA.

[Maddocks72] Maddocks, R.S., et al, "A compact and accurate generator for truly random binary digits", Journal of Physics E: Scientific Instruments, Vol. 5, pp. 542-544.

[Manelis61] Manelis, J.B., "Generating random noise", Electronics, September 8, 1961, pp. 66-69.

[Murray70] Murray, H.F., "A General Approach for Generating Natural Random Variables", IEEE Transactions on Computers, December, 1970, pp. 1210-1213.

[Rand55] Rand Corporation, "A Million Random Digits", Glencoe, Illinois: The Free Press, 1955.

[Simmons80] Simmons, R.E., "Random Number Generator", U.S. Patent No. 4,183,088; January 8, 1980.

[Thomson59] Thomson, W.E., "ERNIE – A Mathematical and Statistical Analysis", Journal of the Royal Statistical Society A, Vol. 133, Part 3, pp. 301-333.

[Vazirani87] Vaziranni, U.V., "Strong Communication Complexity or Generating Quasi-Random Sequences from Two Communicating Semi-Random Sources", Combinatorica Vol. 7, No. 4, pp. 375-392.

[Vincent70] Vincent, C.H., "The generation of truly random binary numbers", Journal of Physics E: Scientific Instruments, Vol. 3, pp. 594-598.

[Vincent71] Vincent, C.H., "Precautions for accuracy in the generation of truly random binary numbers", Journal of Physics E: Scientific Instruments, Vol. 4, pp. 825-828.

[vonNeumann51] von Neumann, J., "Various Techniques Used in Connection With Random Digits", Monte Carlo Applied Math Series No. 12, 1951, pp. 36-38.