Generating Quasi-Random Sequences from Slightly-Random Sources.

(Extended Abstract)

Miklos Santha[*]
Umesh V. Vazirani[**]

University of California
Berkeley, CA 94720.

Abstract: Several applications require truly random bit sequences, whereas physical sources of randomness are at best imperfect. We consider a general model for these *slightly−random* sources (e.g. zener diodes), and show how to convert their output into 'random looking' sequences, which we call *quasi−random*. We show that quasi-random sequences are indistinguishable from truly random ones in a strong sense. This enables us to prove that quasi-random sequences can be used in place of truly random ones for applications such as seeds for pseudo-random number generators, randomizing algorithms, and stochastic simulation experiments.

1. Introduction.

The existence of a source of fair coin flips has been extensively assumed for applications such as randomizing algorithms [Ra], cryptographic protocols [Bl1, GM] and stochastic simulation experiments [Sc, KG]. Unfortunately, the available sources of randomness (e.g. zener diodes) are imperfect. The simplest model of an imperfect source of randomness is a coin whose bias is unknown, but fixed. Von Neumann [vN] proposed a very simple real time algorithm to extract unbiased flips from such a source. More recently, Blum [Bl2] considers the question when the imperfect random source is a deterministic finite state Markov process. This source may be regarded as flipping a coin whose bias depends on the current state of the Markov process, and therefore depends on the sequence of bits previously output. Blum shows that the obvious generalization of the von Neumann procedure does not work. He shows by an elegant proof that, surprisingly enough, changing the order in which the bits are output yields independent, unbiased flips.

We consider an extremely general model of an imperfect source of randomness. We shall assume that the previous bits output by the source can condition the next bit in an arbitrarily bad way. Accordingly, the model is that the next bit is output by the flip of a coin whose bias is fixed by an *adversary* who has *complete knowledge* of the history of the process. To make sure that the source does generate some randomness, the adversary is limited to picking a bias greater than δ and smaller than $1-\delta$, for some positive fraction $0< \delta < 1$. This models the known practical sources of randomness such as the zener diode, in which the frequency of 0's and 1's "drifts" over a period of time [Mu]. We shall call such an adversary source a **slightly-random source.**

It can be shown that no algorithm can extract a sequence of absolutely unbiased coin flips from such an adversary source. Instead, we consider a different approach: we introduce the notion of *quasi−random* sequences. These sequences may not be truly independent or unbiased, but will be provably indistinguishable from truly random sequences in a very strong sense (even stronger than that of Yao [Ya]). As a consequence of this indistinguishability, it will

[*]Supported by NSF Grant MCS 82-04506, and by the Pompeo Fellowship.
[**]Supported by NSF Grant MCS 82-04506, and by the IBM Doctoral Fellowship.

0272-5428/84/0000/0434$01.00 © 1984 IEEE

follow that truly random sequences can be replaced by quasi-random sequences in all the usual computational applications of random sequences. As an example, we shall prove that quasi-random sequences can be used instead of truly random coin flips to generate random variates for stochastic simulation experiments. The advantage of considering quasi-random sequences is that they can be generated by slightly-random sources of the type described above, which closely model actual physical devices.

We show how to extract n bit quasi-random sequences from $O(\log n \log^* n)$ such semi-random sources operating in parallel: the algorithm is efficient and uses no storage. Moreover, it is a real-time algorithm in the sense that it generates one quasi-random bit at each step. We also prove that our method of generation achieves optimal compression factor.

Why is it necessary to consider (imperfect) physical sources of randomness in light of the theory of perfect (cryptographically secure) pseudo-random number generators [Sh, BM, Ya]. Blum [Bl] points out that there is a fundamental problem that this theory leaves unsolved: that is the source for the random seed. Using a fair source to generate this seed may be crucial because of the danger that the pseudo-random number generator might amplify any dependence or bias in the bits of the seed. As another example of the versatility of quasi-random sequences, we shall prove that they can be used as seeds for perfect pseudo-random number generators, without weakening the cryptographic security of the generator.

2. Quasi-Randomness.

Definition: A *functional statistical test* is any function f: $\{0,1\}^* \to [0,1]$, where [0,1] denotes the unit interval.

We are given a source which for every length n, generates n-length strings $x \in \{0,1\}^n$ with some probability $p_n(x)$.

Let $\mu_f(n) = 1/2^n \sum_{|x|=n} f(x)$, be the average value of f on random n-length strings.
Let $\mu_f^*(n) = \sum_{|x|=n} p_n(x) f(x)$, be the average value of f on n-length strings generated by the source.

Definition: A *quasi-random generator* is a source such that for every $t > 0$, for n sufficiently large, and for every functional statistical test f: $|\mu_f(n) - \mu_f^*(n)| < 1/n^t$.

The notion of a functional statistical test is a strengthening of the concept of probabilistic polynomial time statistical test, introduced by Yao [Ya]. Instead of evaluating a pseudo-random number generator on a few statistical tests (as was done in practice), Yao proposed that a pseudo-random number generator is "perfect" if it passes all probabilistic polynomial time statistical tests. [1] An obvious difference between Yao's statistical tests, and our functional statistical tests is that the function f need not be efficiently computable; it need not be computable at all. A more fundamental difference between these two notions arises from the fact that whereas the probabilistic polynomial statistical test is a complexity theoretic notion, the functional statistical test is an information theoretic notion. For this reason, Yao's definition of a perfect pseudo-random number generator is **not uniform:** a pseudo-random number generator is perfect if for any specified level of security $1/n^t$ and any probabilistic polynomial time statistical test there is some seed length n which ensures this security. In general this value n depends on the statistical test, and no finite value n will ensure this level of security with respect to all statistical tests. On the other hand quasi-randomness is a uniform concept: for any desired level of security, a length n can be picked so that the n-length quasi random sequences achieve this security relative to **every** functional statistical test.

The following two theorems illustrate how the strong properties of quasi-random generators allow them to replace truly random sequences. The first theorem shows that quasi-random sequences are just as "good" as truly random sequences when fed as seeds to pseudo-random number generators.

A probabilistic polynomial time statistical test is a function from $\{0,1\}^*$ to $\{0,1\}$, which is computed by a probabilistic polynomial time Turing machine. A pseudo-random number generator passes a probabilistic polynomial time statistical test if for every $t>0$, for n sufficiently large, the average value of the test (function) on n-length pseudo-random strings differs by no more than $1/n^t$ from the average value of the test on truly random strings.

Theorem 1: Let G: $\{0,1\}^* \rightarrow \{0,1\}^*$ be a perfect pseudo-random number generator. Then G with seeds generated by a quasi-random source is also perfect (passes all probabilistic polynomial time statistical tests).

Proof: The basic idea of the proof is: suppose to the contrary that the generator when fed quasi-random seeds fails a probabilistic polynomial time statistical test T; then the quasi-random number generator fails the functional statistical test obtained by composing the pseudo-random number generator with the test T.

More formally: Let T: $\{0,1\}^* \rightarrow \{0,1\}$ be any probabilistic polynomial time statistical test, and t>0 fixed. Suppose that G on n-length seeds generates poly(n)-length sequences for some polynomial poly(n).

Recall that $\mu_{TG}(n) = 1/2^n \sum_{|x|=n} T(G(x))$.

Now, $|\mu_T(poly(n)) - \mu_{TG}(n)| = O(1/n^t)$ because G is perfect, and $|\mu_{TG}(n) - \mu^*_{TG}(n)| = O(1/n^t)$ because our source is quasi-random.

It follows that
$|\mu_T(poly(n)) - \mu^*_{TG}(n)| = O(1/n^t)$.
Since this is true for each test T, and every t > 0, it follows that the pseudo-random number generator with quasi-random seeds is perfect. Q.E.D.

Next, we show that quasi-random sequences can be used in place of sequences of coin flips by any procedure that generates (random variates of) a desired distribution τ from a sequence of coin flips. Let f : $\{0,1\}^* \rightarrow$ Z, the set of integers. Given a probability distribution Pr on $\{0,1\}^n$, f induces a probability distribution on Z in a natural way as follows: $p_n(y) = \sum_{|x|=n, f(x)=y} Pr(x)$. Let τ be the desired distribution. As a measure of the closeness with which τ is approximated by the distribution induced by the function f, let $\alpha_n = \sum_{y \in Z} |\tau(y) - p_n(y)|$. Intuitively, α_n measures the area between the density plots of the two distributions.

Theorem 2: Let f be any function as above; let p_n be the probability density induced by f when all n length strings are picked with uniform probability, and let q_n be the probability density induced by f when the n length strings are generated by a quasi-random generator. Let $\alpha^*_n = \sum_{y \in Z} |\tau(y) - q_n(y)|$. Then $|\alpha_n - \alpha^*_n| < 1/n^t$, for every t, for sufficiently large n.

Comment: The above theorem can be used to get effective bounds: given a bound one can compute n so that the error (the area between the two density plots) introduced by substituting quasi-random sequences for truly random ones is less than the bound. This value of n, is guaranteed to work regardless of the algorithm used for generating the random variates, because of the uniformity property. The fact that functional statistical tests in the definition of quasi-randomness are not required to be polynomial time tests is also very important in this theorem. This is because the running time of algorithms for producing some distributions has not been analyzed, and may well be super-polynomial.

Proof: Let $f(x) = \begin{cases} 1 \text{ if } p_n(f(x)) > q_n(f(x)) \\ 0 \text{ otherwise} \end{cases}$

Let g(x) = 1 - f(x).

Then $\sum_{y \in Z} |p_n(y) - q_n(y)|$

$= \sum_{y \in Z,\, p_n(y) > q_n(y)} (p_n(y) - q_n(y)) + \sum_{y \in Z,\, q_n(y) \geq p_n(y)} (q_n(y) - p_n(y))$

$= |\mu_f(n) - \mu^*_f(n)| + |\mu_g(n) - \mu^*_g(n)|$

$= O(1/n^t) + O(1/n^t)$

$= O(1/n^t)$ for every t > 0.

Finally $|\alpha_n - \alpha^*_n|$

$= |\sum_{y \in Z} |\tau(y) - p_n(y)| - \sum_{y \in Z} |\tau(y) - q_n(y)||$

$\leq \sum_{y \in Z} ||\tau(y) - p_n(y)| - |\tau(y) - q_n(y)||$

$\leq \sum_{y \in Z} |p_n(y) - q_n(y)|$

$= O(1/n^t)$ for every t > 0.

Q.E.D.

3. Extracting Quasi-random Sequences from Semi-random Sources.

Recall that a *slightly-random source* is a process which generates sequences of 0's and 1's from the flips of a biased coin, where the bias of the coin is determined by an adversary.

We now describe the basic principle underlying the generation of quasi-random strings despite the presence of an adversary. Consider the following "high-quality" source: The sequences of length n are generated by a coin C_n whose bias can be slightly changed after each flip, with the constraint that the bias must be greater than $1/2-\varepsilon(n)$ and less than $1/2+\varepsilon(n)$. Before each flip of this coin, an adversary, who has knowledge of the history of the coin flips sets the bias of the coin. The purpose of the adversary is to create as much dependence in the distribution of flip sequences as possible. We would like to show that if $\varepsilon(n)$ is a sufficiently small function of n, then this source is quasi-random.

Theorem 3: If for every t and for all sufficiently large n $\varepsilon(n) < 1/n^t$, then the source defined above is quasi random.

Proof: For any n-length string $x=x_1 \cdots x_n$, for every i, $1 \le i \le n$, let $\Pr(x_i | x_1 \cdots x_{i-1})$ denote the conditional probability that the i^{th} coin flip is x_i, when the result of the first i-1 coin flips is $x_1 \cdots x_{i-1}$. Then the probability that the source generates x is:

$$p_n(x) = Pr(x_n | x_1..x_{n-1}) \times \cdots \times Pr(x_2 | x_1) Pr(x_1).$$

Since each successive bit is generated by the flip of a coin whose bias is between $1/2-\varepsilon(n)$ and $1/2+\varepsilon(n)$:

$$(1/2-\varepsilon(n))^n \le p_n(x) \le (1/2+\varepsilon(n))^n.$$

For any functional statistical test f:

$$\begin{aligned} |\mu_f(n) - \mu_f^*(n)| &= \Big| \sum_{|x|=n} (1/2^n - p_n(x)) f(x) \Big| \\ &\le \sum_{|x|=n} |1/2^n - p_n(x)| \\ &\le 2^n ((1/2+\varepsilon(n))^n - 1/2^n) \\ &= (1+2\varepsilon(n))^n - 1. \end{aligned}$$

We are done by the following elementary calculation:

$$\begin{aligned} (1+2\varepsilon(n))^n - 1 &= 2n\varepsilon(n) + o(n\varepsilon(n)) \\ &= O(1/n^t) \text{ for every } t > 0. \end{aligned}$$

Q.E.D.

Now the task of extracting quasi-random sequences from $O(\log n \log^* n)$ slightly-random sources is reduced to constructing the high-quality source described above. Each of the $O(\log n \log^* n)$ slightly-random sources has its own adversary who picks the bias of the next flip. Once again each adversary has complete knowledge of the previous coin flips. We shall use the "unbiasing" property of the parity (xor) function. This property has been very effectively exploited in the past by Yao [Ya] to construct a perfect pseudo-random number generator, from any one way function. The algorithm QGEN below converts slightly-random sources into a high quality source. In the next section, we shall prove that the choice of the parity function in the algorithm QGEN is optimal.

Algorithm QGEN:
Input: m sequences of bits, each of length n.

$$x_{11}, \ldots, x_{1n},$$
$$x_{k1}, \ldots, x_{kn},$$

$$x_{m1}, \ldots, x_{mn}.$$

Output: $y = y_1, \ldots, y_n \in \{0,1\}^n$.
Begin:
for i=1 **to** n **do:**

$$y_i := parity(x_{1i} + \cdots + x_{mi})$$

endfor
End.

Consider the following source: For each n, n-length sequences are generated by feeding n-bit outputs of $\delta^{-1} \log n \log^* n$ slightly-random sources to QGEN. QGEN converts these inputs into an n-length sequence.

Theorem 4: The source defined above is quasi-random.

Proof: We show that this source is "high quality" and therefore by Theorem 3 it is quasi-random. More precisely we show that if m slightly-random sources were used, then each bit output by QGEN has bias in the range $[1/2-(1-2\delta)^m, 1/2+(1-2\delta)^m]$; i.e.
$|Pr(y_i=0 | y_1, \ldots, y_{i-1}=u)$
$\quad - Pr(y_i=1 | y_1, \ldots, y_{i-1}=u)| < (1-2delta)^m$.
We introduce the following notation, for $1 \le k \le \log n \log^* n$:

$$r_{k,i} = Pr(x_{1i} + \ldots + x_{ki} = 0 | y_1 \cdots y_{i-1} = u),$$

$$s_{k,i} = Pr(x_{1i} + \ldots + x_{ki} = 1 | y_1 \cdots y_{i-1} = u).$$

We prove by induction on k, that

$$|r_{k,i} - s_{k,i}| < (1-2\delta)^k.$$

The basis step of the induction follows directly from the definition of a semi-random source. To prove the induction step, let

$$Pr(x_{k+1i}=0 \mid y_1 \cdots y_{i-1}=u) = p \ ,$$

$$Pr(x_{k+1i}=1 \mid y_1 \cdots y_{i-1}=u) = q \ .$$

$\delta < p,q < 1-\delta$, because the bias of the $k+1^{th}$ coin is between δ and $1-\delta$. Moreover x_{k+1i} is independent from $x_{1i}, \ldots, x_{ki}$, because the input sequences are generated by parallel sources. A simple consideration shows that

$$r_{k+1,i} = r_{k,i} \cdot p + s_{k,i} \cdot q \ ,$$

$$s_{k+1,i} = s_{k,i} \cdot p + r_{k,i} \cdot q \ .$$

Then

$$\begin{aligned} |r_{k+1,i} - s_{k+1,i}| &= |(r_{k,i} - s_{k,i})(p-q)| \\ &< (1-2\delta)^k (1-2\delta) \\ &= (1-2\delta)^{k+1} \ . \end{aligned}$$

This completes the induction.

Finally, substituting any function that grows faster than logn asymptotically (we choose $\log n \log^* n$) for k, we get

$$\begin{aligned} &|Pr(y_i=0 \mid y_1 \cdots y_{i-1}=u) \\ &\quad - Pr(y_i=1 \mid y_1 \cdots y_{i-1}=u)| < (1-\delta)^{\log n \log^* n} \\ &\quad = O(1/n^t) \text{ for } every\ t>0 \ . \end{aligned}$$

Q.E.D.

4. Lowerbounds, or the Power of the Adversary.

We prove below that the choice of the parity function in the algorithm QGEN of Section 3 is optimal. The model is that any algorithm must look at a fixed size block of slightly-random bits to produce one quasi-random bit. Thus any such algorithm is in general a boolean function mapping m bits into 1 bit. We prove below that m must grow faster than logn asymptotically to achieve quasi-randomness. Clearly, it suffices to consider the (hardest) case when the m slightly-random bits are outputs of m distinct slightly-random sources.

Theorem 5: Let $f:\{0,1\}^m \to \{0,1\}$ be any boolean function. Then there are strategies for m adversaries such that the bias of $f(s_m \cdots s_1)$ differs atleast $(1-2\delta)^m$ from 1/2, where s_i is a bit generated by the i^{th} source.

Proof: By induction on the number of sources. The basis follows from the definition of a slightly-random source. For the inductive step, fix the output of the 1^{st} source to be 0. Then by the inductive assumption, the other m-1 adversaries have strategies which force the bias of f restricted to $s_1 = 0$, to be bounded atleast $(1-2\delta)^{m-1}$ away from 1/2. Let $p_0(0)$ be the probability that f restricted to s_1 is 0, and $p_0(1)$ the probability that it is 1. We assume without loss of generality that the inductively assumed strategies for the m-1 adversaries yield: $p_0(0) - p_0(1) = \Delta \geq (1-2\delta)^{m-1}$. Fix these strategies for the m-1 adversaries.

Case 1: For f resticted to $s_1 = 1$, $p_1(0) + \Delta \geq p_1(1)$. In this case the first adversary chooses bias $1-\delta$ towards 0. Then Pr[0] - Pr[1] $\geq (1-\delta)\Delta - \delta\Delta$

$$\begin{aligned} &= (1-2\delta)\Delta \\ &\geq (1-2\delta)^m. \end{aligned}$$

Case 2: $p_1(0) < p_1(1) + \Delta$. In this case the first adversary chooses bias $1-\delta$ towards 1. Then Pr[1] - Pr[0] $\geq (1-\delta)\Delta - \delta\Delta$

$$\begin{aligned} &= (1-2\delta)\Delta \\ &\geq (1-2\delta)^m. \end{aligned}$$

Q.E.D.

It is somewhat surprising that the bound of Theorem 5 is exactly the same as the bound proved in Theorem 4. This directly yields:

Corollary: the parity function achieves the most efficient conversion of slightly-random source outputs into quasi-random sequences.

Finally, we show that that there is no algorithm to convert the output of a single slightly-random source into quasi-random sequences, no matter how much bit-compression is allowed. Let $f: \{0,1\}^m \to \{0,1\}$ be any boolean function. Intuitively, f tries to compress m bits of the source output into one quasi-random bit. We prove that for every f, there is an adversary strategy so that the bias of the extracted bit is $1 - \delta$ towards 1, thus showing that the extracted bit is just as bad as any bit in the original source output.

The function f, may be represented in a complete binary tree of height m as follows: the two branches from each node are labelled 0 and 1. Each path from root to leaf then corresponds

to a unique binary string of length m. The value of f on a string is assigned to the corresponding leaf in the tree. An adversary strategy for the slightly-random source consists of labelling for each node of the tree: the 1-branch with a bias b between δ and 1 - δ, and the corresponding 0-branch 1 - b. The probability of picking any root-leaf path in the tree is simply the product of the biases on the edges.

Define the weight of a subtree be the number of 1-leaves in it. Assume without loss of generality that f(x) = 1 for atleast 1/2 fraction of the strings of length m. The following adversary strategy guarantees that the probability of reaching a 1-leaf (i.e. f(x) = 1) is atleast $1-\delta$: for each node, label the branch leading to the heavier subtree with bias 1 - δ. The proof goes by induction on the height, m, of the tree. Consider a subtree of height k, rooted at A. Let a denote the number of 1-leaves in the subtree. Then a is either 2^k or a k bit number $a_k a_{k-1} \cdots a_1$. Let $\sigma(i)$ denote the number of 1's in the prefix $a_k \cdots a_{i+1}$. Then we associate a value, v(a), with the subtree:

$$v(a) = \begin{cases} 1 \text{ if } a = 2^k \\ \sum_{i=1}^{k} a_i \delta^{\sigma(i)}(1-\delta)^{i-\sigma(i)} \end{cases}$$

Intuitively, this value is the probability of reaching a 1-leaf if the adversary follows the above strategy on a tree with all 1-leaves appearing consecutively from left to right.

Theorem 6: For every boolean function f, there is an adversary strategy, such that the probability that the function value has bias atleast $1-\delta$, when the inputs are sequences generated by the adversary source.

Sketch of Proof: We prove by induction, on the height of the subtree, that the above adversary strategy ensures probability atleast v(a) of reaching a 1-leaf when starting from the root of any subtree A. This proves the theorem because we can assume without loss of generality that $|\{x: f(x) = 1\}| \geq 2^{m-1}$, so the value of the whole tree is atleast $1-\delta$.

The idea of the inductive step is to show that if A and B are sons of C in the tree, and $v(a) \geq v(b)$, then $v(c) \leq v(a)(1-\delta) + v(b)\delta$. By the inductive assumption, the adversary can force probability atleast v(a) of reaching a 1-leaf when starting from the root of A, and similarly for B. Thus he can force probability atleast v(c) of reaching a 1-leaf starting from the root of C, by picking the branch leading to A with probability $1-\delta$.

5. Acknowledgements:

We are extremely indebted to Manuel Blum, not only for raising the issues that sparked off this research, but also for his ability to detect very subtle flaws in "proofs". Vijay Vazirani played a critical role in helping us clarify several conceptual points. Sampath Kannan was a catalyst for the lower bound proofs. We wish to thank them and Ashok Chandra, Richard Karp, Dexter Kozen, Steven Rudith, Michael Sipser & Avi Wigderson for some very useful discussions.

6. References.

[Bl1] M. Blum, "Coin Flipping by Telephone," IEEE COMPCON (1982).

[Bl2] M. Blum, "Independent Unbiased Coin Flips From a Correlated Biased Source: a Finite State Markov Chain," to appear.

[BBS] L. Blum, M. Blum and M. Shub, "A Simple Secure Pseudo-Random Number Generator," to appear in SIAM Journal of Computing.

[BM] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," 1982 FOCS.

[GM] S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," 1982 STOC.

[KG] W. Kennedy and J. Gentle, *Statistical Computing*, Marcel Dekker, Inc. New York.

[Kn] D. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA (second edition 1981).

[vN] J. von Neumann, "Various Techniques Used in Connection with Random Digits," Notes by G. E. Forsythe, National Bureau of Standards, Applied Math Series, 1951, Vol 12, 36-38. Reprinted in von Neumann's Collected Works, Vol 5, Pergamon Press (1963), 768-770.

[Ra] M. Rabin, "Probabilistic Algorithms," Algorithms and Complexity, J.Traub, Editor, Academic Press (1976), pp. 21-39.

[Sc] B. Schmeiser, "Random Variate Generation: A Survey," 1980 IEEE. Simulation with Discrete Models: A State-of-the-Art View, T.Oren, C. Shub, P. Roth (eds.).

[Sh] A. Shamir, "On the Generation of Cryptographically Strong Pseudo-Random Sequences," 1981 ICALP.

[Ya] A. Yao, "Theory and Applications of Trapdoor Functions," 1982 FOCS.

[Mu] H. F. Murry, "A general approach for generating natural random variables," IEEE Trans. Comput., vol. C-19, pp. 1210-1213. Dec 1970.