Richard L. Garwin
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 945-2555

February 2, 1981

FILE

The "imperfect coin" is indispensable to the discussion of probability, and it or its electronic analog is useful in Monte Carlo calculations, and the like. Here I describe an "efficient imperfect coin," which may be new to some who read this. In short, it allows a perfect coin flipped an average of two times to mimic an imperfect coin for which an arbitrary probability of obtaining "heads" has been prescribed.

I shall first describe the algorithm, then give the self-evident proof, and finally mention two applications.

Let the prescribed probability of obtaining "heads" on a single toss of the (conjectured) imperfect coin be M. Write M as a binary fraction-- i.e., a real number beginning "0.xxxx...," where the "x" are either 1 or 0, and the fraction may terminate or not, as the case may be. This representation of M may be truncated to the number of binary places which give the desired accuracy. Thus, the bits of M beginning from the binary point are identified as M(i).

Now flip a perfect coin, assigning binary "1" to a "head" and "0" to a "tail." Let the sequence of heads and tails be represented by the emerging binary fraction F(i). After each flip, i, compare F(i) with M(i). If they disagree, stop. The imperfect coin toss is complete. If they continue to agree, proceed until the first disagreement. The disagreements may be of two types, interpreted as follows: If F is 0 when M is 1, then the imperfect coin is stated to show heads, If F is 1 when M is 0, the imperfect coin is stated to show tails.

The expected number of flips to the first disagreement (for an untruncated, non-terminating M) is evidently the sum over i of i*(0.5)**i. As promised, this sum is 2. So we have prescribed an algorithm which in an average of 2 tosses mimics precisely the results of a single flip of a coin with prescribed imperfection.

The proof is simple. For any prescribed (truncated) accuracy of P binary places, the 2**P possible outcomes of P flips of the true coin are represented by the 2**P points on the number line from 0 to (but not including) 1. The truncated M is represented as a position on the number line. All points on the number line are equally probable to arise in a single sequence of P flips of a true coin. Therefore, the probability of finding an outcome which is strictly less than M equals M itself. If one imagines that one has performed P flips of the true coin, one can subtract F from M and call the simulated imperfect coin H(heads) if M - F > 0, and tails otherwise. But one need not perform the substraction to see which is bigger. One may compare M(i) with F(i) for i = 1, 2..., stopping with the first disagreement. The rule given above obviously is equivalent to subtraction and branch on 0. Q.E.D.

In a computer implementation, of course one may have available a set of random numbers, considered here as a sequence of random bits. Obviously, one need not generate the random bits as they are inspected, but can draw them from a previously-generated and tabulated store of bits.

Two examples may be of interest. The first is a homey one which does not use a computer. Thus, if I owe someone $7.94 for dinner and have only a $20 bill and must leave immediately, never to communicate with that person again, we may agree that it is acceptable for me to play a game with that person, in which he/she has a $7.94/$20 chance of winning. Thus, we require an imperfect coin with precisely those odds of coming up heads; a single flip of the coin would then discharge my debt. Thus, we would convert that ratio into a binary fraction, and apply the above algorithm with the expectation of requiring only two tosses of a real coin.

A second, more serious application is for instance in computer simulation of dense interacting molecular systems. In the pioneering work of Metropolis, Von Neumann, Rosenbluth, et al, they approached the simulation problem from the starting point of generating all possible points in phase space, and then accepting them for the ensemble with the probability exp(-E/kT). The only problem, of course, that that configurations generated at random have such a small chance of being accepted that one could run a computer forever without finding an acceptable configuration (for reasonably dense systems with hard-core interactions). They solved this problem by starting from a more-or-less regular array, with little overlap, and then moving each atom in turn by a small step, accepting the new configuration with probability as given before. The size of the step can be selected to provide a reasonable probability of acceptance. The acceptance may be imagined performed with a single flip of an imperfect coin with the probability of heads as given. The algorithm above can determine that acceptance simply by the inspection of 2 bits (on the average) from a stock of random numbers.

A note on the history of this algorithm. I doubt that I am the first to conceive this algorithm. It occurred to me some twenty years ago by a very complex process of reducing the number of coin flips required to obtain arbitrary odds by a method involving partial binary fractions. The procedure was precisely that given above, but my explanation was so convoluted that eyes glazed before I finished the motivation and the proof. In trying to simplify the presentation, the mapping on the number line came to mind. I have discussed the algorithm in essentially this form since about 1974, and I believe that it appears in publications of Knuth and also of Martin Gardner, who may have obtained it from an independent source. I make no claim to originality, but I believe the result should be more widely known.

RLG:mla:033'FILE:020281FILE