RANDU: Implementations of RANDU are readily available\(^1\) and so I have provided a \texttt{randu} library that works similar to the \texttt{rng} library:

- \texttt{randu()} returns a real-valued random number between 0 and 1;
- \texttt{PutSeed()} accepts a long as a parameter, which will be used as the initial seed for RANDU.

Source for \texttt{randu.h,c} is available on the course web page.

\textbf{Note:} RANDU should be used with an odd initial seed!

1. Implement the uniformity test of randomness (see Algorithm 10.1.1 and details below). As in Example 10.1.1, let the number of calls to \texttt{Random()} be \(n = 10\,000\), the number of histogram bins be \(k = 1000\), the corresponding critical points for 95\% confidence be \(v^*_1 = 913.3\) and \(v^*_2 = 1088.5\), and the chi-square statistic be as defined on page 445 of the text.

(a) Using 12345 as the initial seed to the first replication, perform 256 replications of the uniformity test on sequences returned from \texttt{Random()} (i.e., from the library \texttt{rng}).

One replication of the experiment consists of \(n = 10\,000\) calls to \texttt{Random()}, followed by computing the chi-square statistic, and then comparing that statistic to the corresponding critical values. This means you will be running the uniformity test 256 different times, i.e., generating 256 realizations of the chi-square statistic and comparing each to the critical values. Since you want each replication to use a different sequence of \(n = 10\,000\) random variates, let the initial seed for each subsequent replication be determined by the (then) current state of the generator. In other words, if you do not reset the initial seed after each replication, the generator will automatically use a different sequence of variates.

Provide a graphic similar to that given in Figure 10.1.1. How many of the 256 chi-square statistics so generated fell outside the interval defined by the critical values?

(b) Perform the same test as in (a) but using the \texttt{randu} library. Provide a graphic similar to that given in Figure 10.1.1. How many of the 256 chi-square statistics so generated fell outside the interval defined by the critical values?

2. Implement the runs-up test of randomness (see Algorithm 10.1.3).

(a) Repeat the experiments in part 1, but using the runs-up test of randomness. Let the number of histogram bins be \(k = 6\) (recall that a run-up of length higher than six is very unlikely so we group all such runs-up into the final bin), let the number of runs-up be \(n = 28\,800\), the values for \(p(x)\) be as defined at the top of page 446, the chi-square statistic be as defined at the bottom of page 445, and (as in Example 10.1.3) the critical values corresponding to 95\% confidence be \(v^*_1 = 0.83\) and \(v^*_2 = 12.83\).

Provide a graphic similar to that given in Figure 10.1.4. How many of the 256 chi-square statistics so generated fell outside the interval defined by the critical values?

(b) Perform the same tests as in (a) but using the \texttt{randu} library. Provide a graphic similar to that given in Figure 10.1.4. How many of the 256 chi-square statistics so generated fell outside the interval defined by the critical values?

(c) Perform the same tests as in (a) and (b) but using \(n = 57\,600\), \(n = 115\,200\), and \(n = 230\,400\). In each of these cases, how many of the 256 chi-square statistics so generated fell outside the interval defined by the critical values for each of \texttt{rng} and \texttt{randu}? (Note that the period of our random number generator is approximately \(2^{31}\), and since we are using 256 replications, provided \(n\) is smaller than \(2^{31}/256 > 8\,000\,000\) there should be no cycling.)

\textbf{Submitting:} Include six figures, one for each of \texttt{randu} and \texttt{rng} from: the uniformity test; the runs-up test with \(n = 28\,800\); and the runs-up test with \(n = 230\,400\). Package your figures into a gzipped tarball similar to previous labs, and drop your tarball into the \texttt{lab4} folder in the shared Box folder for this class. Your lab is due by 23:59 on Sun 11 Feb.

\(^1\)Thanks to Dan Cer for the implementation on which this library is based. See \url{http://cer.freeshell.org/renma/Randu/Randu.c}. 

---

**CMSC 326: Simulation**

**Tests of Randomness**