Chapter 10

BERNOULLI TRIALS: EXAMPLES

The following examples illustrate ideas in Chapter 5. Our running queueing example is continued in the second, following the setup in the first. The third and fourth examples, independent of the first two, show how the transitions of a continuous-state Markov chain can be structured so that our RQMC techniques for Bernoulli trials apply. Section 10.3 shows, in greater generality than we have seen previously, how extreme skewness arises naturally when applying change of measure and/or Russian roulette at the successive states visited of a Markov chain. It also gives the first modern treatment of filtering when splitting and Russian roulette are used as well as change of measure. Example 10.3.1 in that section deals with weight windows, which depend on filtering and attenuate skewness. Except for the subsubsection on tailoring weight windows for RQMC, Section 10.3 can be read independently of the rest of this book. The sixth (and last) example treats network reliability; there is no reference to it elsewhere in this book, so it can be skimmed at first reading.

10.1 Linearity in trial indicators

EXAMPLE 10.1.1 To see the effect of $q$-blocks in a particular case, where the linearity condition in Section 5.4 holds, directly relevant to
Example 10.1.2 below but having wider scope, consider the effect on

$$E \left( \text{Var} \left[ \sum_i \sum_j Z(i, j) \mid X \right] \right)$$

with inhomogeneous indicators $Z(i, j)$ defined as in Section 5.4, $i$ indexing runs in a block, and $j$ indexing sets of trials across runs in a block. Decomposing $Z(i, j)$ as in Section 5.3.3,

$$Z(i, j) = I(i, j) + (1 - I(i, j))R(i, j)$$

clearly, the grand sum of the $I(i, j)$'s is a constant if the overall success total $M(S)$ is initialized to its expectation (when the latter is an integer) or if — for statistical purposes — it and the grand sum are replaced by their common expectation after generating the individual indicators.

The most obvious way to compute the $q$-block output sums the $Z(i, j)$'s over $j$ and then averages those sums. Instead, we sum the $Z(i, j)$'s over $i$ and then average those sums. The point is that the $Z(-, j)$'s are independent, but the $Z(i, \cdot)$'s are not.

So far, we have not seen any benefit from generating the column totals as in Section 5.4. We are about to see some.

Here is an extreme instance where conditioning on the run totals provably reduces variance, by a large amount. Suppose that, for fixed $j$, the $R(i, j)$'s are not only independent but also identically distributed. We then have a sum of iid terms:

$$\sum_{i=1}^{q} I'(i, j))R(i, j) \quad \text{with} \quad I'(i, j) \overset{\text{def}}{=} 1 - I(i, j) ,$$

where the number of non-zero summands equals the column-$j$ success total $M'(F_{kj})$ relative to the $I'(i, j)$'s. The $k$ here refers to the number of stages of the third tree-like algorithm to compute the $r$ column totals given the grand sum of the $I'(i, j)$'s. Clearly,

$$\text{Var} \left[ \sum_i I'(i, j)R(i, j) \right] = \text{Var} [M'(F_{kj})] E^2[R(1, j)] + E [M'(F_{kj})] \text{Var} [R(1, j)]$$