A Directly Programmed Solution to ARGESIM Comparison C 8 'Canal-and-Lock System' using Java

Gerhard Höfinger, Felix Breitenecker, Vienna Univ. of Technology, gerhard.hoefinger@gmx.at

Simulator: Java (version 1.4.2) is an object oriented programming language and was used here without any additional packages for simulation. So we had to do without the advantages of simulators (e. g. graphical modelling, libraries, experiment evaluations), but the complex logic could be formulated quite easy and simulation time is reduced to a few seconds.

Model. The system, which had to be simulated, is a canal and lock system through which barges can move in two directions, east and west. The special properties of the system lead to a quite complex logic: The canals, which lead to the lock, are narrow, so only one direction of movement is possible at a time. The lock needs a certain time to raise or lower its water level. When a barge arrives, it has to be raised or lowered respectively, and the lock can operate foresightedly if the barge is still on the way. Only one barge can be in the lock at a time.

Task a: Modelling assignment. Natural objects and therefore classes are barges and the lock itself. Another class, called lock system, contains vectors, which represent the places where barges sojourn during their journey and methods, which model time steps (and that advance the barges, that have to be moved from one vector to the next) and decide to which direction barges may go. A class called simulation adds barges to the system and calls the methods of lock system after every time step. As graphics were omitted, simulation results are printed to the console and written to a file, which can be read by Microsoft Excel (version 2000, later used for statistical computations).

Task b: Model validation with deterministic data. The logic of the model was simulated using the given data sets, and correct results, as given in the definition, were returned.

Task c: Variance reduction experiments. For task c, before the simulation started, a list with arrival times for the barges was generated, using the random generator provided by Java. This returns uniformly or normally distributed pseudo random numbers. With the inverse transform method exponentially distributed interarrival times were obtained. λ (the mean of Exp(x)) was set to 75. Although queues sometimes got long, they did not grow steadily until the end of the simulation. Without using variance reduction methods, the following results in three independent experiments were obtained for mean and 90% confidence intervals (CI):

	Mean	CI	σ^2
Run 1	531,0	± 39,8	242,2
Run 2	538,4	± 41,6	253,0
Run 3	524,3	± 42,0	255,1

Using the method of antithetic variates, the length of the 90% confidence interval could be significantly decreased:

	Mean	CI	σ^2	Reduction of Interval length
Run 1	526,1	± 25,2	108,2	36,8%
Run 2	526,0	± 27,9	119,8	33,0%
Run 3	524,7	± 26,6	114,5	36,5%

The number of barges that may pass the lock while barges heading to the other direction are waiting (called eastmax and westmax respectively) was in the experiments performed earlier each set to 5. Now should be investigated, how the system reacts if eastmax and westmax are raised to 6. Therefore a 90% confidence interval for the difference of the mean barge waiting time resulting from these two strategies should be formed. First, independent experiments were performed, giving the following results:

	Mean	CI	σ^2
Run 1	-77,6	± 64,9	279,1
Run 2	-85,4	± 68,8	295,6
Run 3	-110,8	± 69,5	298,9

The difference was computed by subtracting the mean of runs where (east-/west-)max was 6 from those where it was 5. So, the negative means show that setting eastmax and westmax to 6 resulted in a lower waiting time. The difference is even so high, that it is significant, and the probability of a type I error is less than 0,05. But, to be sure, also here a variance reduction method, the Common Random Number methodology, was used. The success was even better than in the first problem, as can be seen in the following table:

	Mean	CI	σ^2	Reduction of Interval length
Run 1	-62,0	± 7,6	32,6	88,3%
Run 2	-65,8	± 7,5	32,3	89,1%
Run 3	-64,1	± 7,0	30,3	89,9%

The interval lengths could be decreased considerably (the simulation effort had not to be raised).

C8 Classification: Directly Programmed

Simulator: Java 1.4.2