



[ARTICLES](#)

[READING LIST](#)

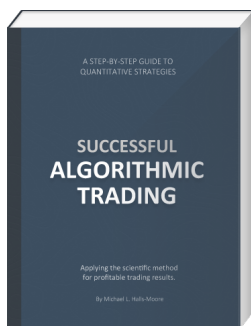
[C++ EBOOK](#)

[QUANT TRADING EBOOK](#)

[A](#)

[QUESTIONS](#)

SUCCESSFUL ALGORITHMIC TRADING



Check out my **new ebook on quant trading** where I teach you how to build **profitable** systematic trading strategies with Python tools.

Find Out More »



ABOUT ME



Hi! My name is Mike and I'm the guy behind QuantStart.com. I used to work in a hedge fund as a quantitative trading developer in London.

Now I research, develop, backtest and implement my own intraday algorithmic trading strategies using C++ and Python.

[MORE »](#)

RANDOM NUMBER GENERATION
CONGRUENTIAL GENERATORS

Share this:

Tweet 1

Like 4

In this article we are going to construct classes to generate random numbers. Random number generators (RNGs) are necessary for Monte Carlo simulation pricing techniques. Other articles on QuantStart.com have discussed random variables distributed as a standard Gaussian.

We will now show how to construct a random number generator to separate the generation of random numbers from their use. It helps us reduce the amount of code and increases extensibility by allowing easy creation and reuse, making the code more maintainable.

There are further reasons to write our own random number generator.

It allows us to make use of **pseudo-random** numbers that possess the correct statistical properties. The use of pseudo-random numbers to improve the convergence rates of Monte Carlo simulation is a common technique. The specific classes. In particular we can implement *anti-thetic sampling* in this manner.

Relying on the `rand` function provided with Python is a `rand` implementation specific, because it varies across platforms. We are unaware of the efficiency of each implementation for cross-platform testing as we cannot guarantee consistent behaviour. We are able to provide **multiple separate** components for different parts of our running program. The seed for the random variable and hence will affect all component behaviours. By implementing our own RNG we

