QRA Simulation Techniques for Solving Systems of Stochastic Differential Equations

ISI Technical Note
Inductive Solutions, Inc.

A. Background

Using simulation to solve systems of stochastic differential equations is based on creating a discrete version of the system. In other words, we approximate

$$d\boldsymbol{r} = (\boldsymbol{m} - \boldsymbol{L})\, dt + \boldsymbol{S} \bullet d\boldsymbol{Z}$$

by a discrete version of the stochastic differential equation, e.g. the following *Euler* scheme

$$\Delta\boldsymbol{r} = (\boldsymbol{m} - \boldsymbol{L})\, \Delta t + \boldsymbol{S} \bullet \Delta\boldsymbol{Z}$$

where

$$\Delta\boldsymbol{Z} = \sqrt{\Delta t} < N_1, \ldots, N_k >$$

and N_j denotes a set of independent, identically distributed standard zero-mean unit variance gaussian random variables. Here $\boldsymbol{r}$, $d\boldsymbol{r}$, $\Delta\boldsymbol{r}$, $\boldsymbol{m}$, and $\boldsymbol{L}$ are n-vectors; $\boldsymbol{S}$ is the $n \times k$ *diffusion* matrix (k is the number of *factors* in the model; n is the order of the model), and $\bullet$ denotes the usual vector inner product.

Given an initial value $\boldsymbol{r}(0) = \boldsymbol{r_o}$, the solution evolves as we compute the values of $\boldsymbol{r}$ for $t = 0,\ \Delta t,\ 2\Delta t,\ 3\Delta t \ldots, n\Delta t = T$:

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \Delta\boldsymbol{r}(t)$$

Given these values, a specific function of $\boldsymbol{r}(t)$, can be computed for each path. for example, for a single factor interest rate model (k =1), the bond price is computed by integration

$$B(t;T) = e^{-\int_t^T r(s)ds}$$

This integation can be approximated as a discrete sum by a variety of methods. Given a set of N paths, the expected value of the bond price is approximated by

$$\bar{B}(t;T) = \mathrm{E}[B(t;T) \,|\, r^1(s),\ \ldots\ , r^N(s)] = \tfrac{1}{N} \sum_{r(s)} B(t;T)$$

We can estimate standard deviation of the bond price as well. The accuracy of these estimates depend on the stepsize Δt used in the discrete version of the stochastic

differential equations and the number of scenarios N. The error is approximately o(Δt) and $o(\frac{1}{\sqrt{N}})$ respectively.

The random gaussian vector $\Delta \boldsymbol{Z}$ is generated by a variety of *Monte Carlo* techniques. These techniques typically rely on a method for generating uniformly distributed random numbers and then transforming these numbers into a gaussian distribution. For a single factor model, this amounts to creating an algorithm that "randomly" picks numbers in the interval [0, 1]; for a multi-factor model, the algorithm picks "random" numbers from a *k*-dimensional box.

B. Random Number Generation

QRA incorporates several random number generation techniques, grouped into whether they are *pseudo random* or *quasi random*. (For a detailed discussion, see the monographs by Niederreiter and by Tezuka.)

Pseudo random generators are designed to maximize their period (the time when numbers start repeating) and minimize the serial correlation between sets of numbers. They are usually based on linear congruential algorithms, using integer multiplication and division by a large prime number (i.e., finding remainders *modulo p*).

Quasi random generators are designed simply to fill the space in an interval more uniformly than uncorrelated random points. They are usually based on Laurent series expansions over finite fields. Their advantage is in multi-dimensional spaces: they fill higher dimensional boxes more efficiently than other methods. In a more formal sense, using quasi random generators can reduce the error associated with a simulation from $o(\frac{1}{\sqrt{N}})$ to $o(\frac{1}{N})$.

QRA provides the following pseudo random and quasi random generators:

- L'Ecuyer (pseudo random)
  A long period generator based on linear congruential algorithm that can generate approximately 10^{18} pseudo random scalar numbers.

- Knuth (pseudo random)
  A variant of the linear congruential method, based on a subtractive algorithm.

- Faure-Niederreiter (quasi random)
  Generates low discepancy *p*-vector sequences using monic irreducable prime polynomials over the field of integers modulo *p*. QRA pre-computes the irreducable prime polynomials and uses matrix multiplication modulo *p* to generate the sequences.

- Faure-Niederreiter-Tezuka (quasi random)
  A variation of Faure-Niederreiter that uses a different expression for matrix multiplication.

- Sobol'-Niederreiter (quasi random)
  Generates low discepancy *p*-vector sequences using monic irreducable prime polynomials over the field of integers modulo *2*. (These polynomials were pre-computed and saved up to dimension 5000, to increase efficiency of the generator.)

- Richtmeyer (quasi random)
  Generates low discepancy sequences based on roots of prime numbers. (These roots were pre-computed and saved up to dimension 5000, to increase efficiency of the generator.)

All QRA quasi random generators are configured to generate at most 10^9 quasi random vectors of at most 5,000 dimensions.

The algorithm used to transform these pseudo-random or quasi random numbers into a gaussian distribution is based on a fast series expansion of the inverse of the cumulative normal density function.

B. Error Reduction

Error reduction from the $o(\Delta t)$ requires a scheme that results in stable solutions. For example, Talay has shown that the *Euler* scheme is more stable then other "higher order" schemes (i.e,. those of order $o(\Delta t^2)$ and higher) that involve higher order Taylor series approximations to the differential.

One way to reduce the error is use quasi-random numbers in generating the sets of gaussian random variables.

Another way is in creating a Brownian Bridge (see Morokoff et al) to essentially modify the regions of integration.

QRA supports both of these methods. It supports the four quasi random number generation methods and uses a Brownian Bridge interpolation scheme in integration.