Weierstrass Institute for
Applied Analysis and Stochastics

Leibniz
Association

Computational finance

Christian Bayer

Let $(\Omega, \mathcal{F}, P)$ denote a probability space supporting a random variable S modelling an asset price. Often, S might take values in $C([0,T];\mathbb{R}^d)$ or $D([0,T];\mathbb{R}^d)$. Some examples of European options include:

Example

A European *call* option on S^1 has payoff of the form

$$f(S) = (S_T^1 - K)^+.$$

Example

A *lookback* option might have a payoff

$$f(S) = (S_T^1 - \min_{t\in[0,T]} S_t^1)^+.$$

Example

A (down-and-out) *barrier* option has a payoff

$$f(S) = (S_T^1 - K)^+ \mathbf{1}_{\min_{t\in[0,T]} S_t^1 > B}.$$

Assuming that we have already chosen a risk-neutral measure P and payoffs are already discounted, the Europen option pricing problem is of the form:

Compute $E[f(S)]$.

Let $X := f(S)$ and assume $X \in L^1(\Omega, \mathcal{F}, P)$. We try to solve the integration problem $E[X]$.

- Classical numerical integration usually not suitable, as the integration problems are often high dimensional and the integrands non-smooth.

Monte Carlo simulation

(X_i) i.i.d. sequence of copies of X, $M \in \mathbb{N}$. Then

$$\overline{X}_M := \frac{1}{M} \sum_{i=1}^{M} X_i \xrightarrow[M \to \infty]{\text{a.s.}} E[X], \quad E\left[\left(E[X] - \overline{X}_M\right)^2\right] = \frac{\operatorname{var} X}{M}.$$

- How to generate random numbers on a computer?
- How to simulate from a given, complicated (asset price) distribution?
- Speeding up MC simulation by variance reduction.
- Deterministic integration using low discrepancy sequences.

Suppose that $S_t = X_t^1$, where $X_t \in \mathbb{R}^d$ solves a stochastic differential equation (SDE)

$$dX_t = V(X_t)dt + \sum_{i=1}^{d} V_i(X_t)dW_t^i, \quad X_0 = x \in \mathbb{R}^d,$$

where $W^1, \ldots, W^d$ denote independent Brownian motions (or Levy processes).

Example (Stochastic volatility models)

$X_t = (S_t, V_t)$, where V_t is the stochastic variance. E.g., the Heston model is defined by

$$dS_t = \sqrt{V_t} S_t (\rho dW_t + \sqrt{1-\rho^2} dW_t^{\perp}),$$
$$dV_t = \kappa(\theta - V_t)dt + \eta \sqrt{V_t} dW_t.$$

How to simulate from the distribution of X?

Convention

For a vector field $V : \mathbb{R}^d \to \mathbb{R}^d$ and a function $h : \mathbb{R}^d \to \mathbb{R}$ we set $Vh(x) := \nabla h(x) \cdot V(x)$.

Let $u(t, x) := E[g(X_T) \mid X_t = x]$. Then

$$\partial_t u(t, x) + Lu(t, x) = 0, \quad u(T, x) = g(x), \quad t \in [0, T], \quad x \in \mathbb{R}^d,$$

where

$$Lh(x) := V_0 h(x) + \frac{1}{2} \sum_{i=1}^{d} V_i^2 h(x), \quad V_0(x) := V(x) - \frac{1}{2} \sum_{i=1}^{d} DV_i(x) \cdot V_i(x), \quad x \in \mathbb{R}^d.$$

Compute the option price $u(0, x)$ by solving the PDE, using the finite element method or by Fourier methods.

American options prices are solutions to optimal stopping problems.

$$\sup_{\tau \in \mathcal{T}_{0,T}} E[f(S_\tau)], \quad \mathcal{T}_{0,T} := \{0 \le \tau \le T, \ \tau \text{ stopping time}\}.$$

- Important class of options.
- Simple example of a truly high-dimensional problem.
- Stochastic optimal control problem.

Problem

How to algorithmically generate realizations $u_1, u_2, \dots$ of i.i.d. random variables $U_1, U_2, \dots$ distributed according to $\mathcal{U}([0,1[)$.

- Equivalent: Generate realizations $i_1, i_2, \dots$ of i.i.d. random variables $I_1, I_2, \dots$ uniformly distributed on $\{0, 1, \dots, m-1\}$ for fixed (large enough) m. Then set $u_\ell := i_\ell / m$.
- Crucial importance for computations. Do use a trusted, well-established RNG!

Definition

A random number generator (RNG) is a structure $(X, x_0, T, G, \{0, 1, \dots, m-1\})$ where X is a finite set (the state space), $x_0 \in X$ is the initial state (the seed), $T : X \to X$ is a transition function, and $G : X \to \{0, \dots, m-1\}$ is the output function. We have

$$x_l := T(x_{l-1}) \quad \text{and} \quad i_l := G(x_l) \quad \text{for} \quad l = 1, 2, \dots .$$

Good RNGs

1

2

3

4

5

6

7

Linear congruential generators

$X = \{0, \ldots, m-1\}$, $G(x) = x$, $T(x) = (ax + c) \mod m$

- Simple to implement, fast, well-understood theoretically.
- Full period m provided that
  - c and a are relatively prime,
  - every prime number dividing m also divides $a - 1$,
  - if m is divisible by 4 then so is $a - 1$.
- E.g., Numerical Recipes suggests: $m = 2^{32}$, $a = 1664525$, $c = 1013904223$
- Common weakness: Fix d and consider $((i_l, i_{l+1}, \ldots, i_{l+d-1}))_{l=1,2,\ldots}$. While $(I_l, \ldots, I_{l+d-1})$ is uniformly distributed on the set $\{0, \ldots, m-1\}^d$, $(i_l, i_{l+1}, \ldots, i_{l+d-1})$ tend to lie on a (possibly) small number of hyperplanes.

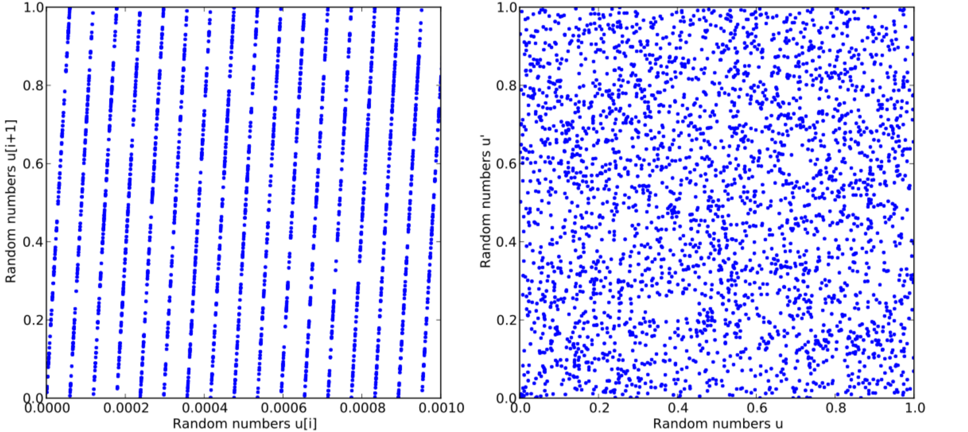

Figure: Hyperplane property for the linear congruential generator with $a = 16807$, $c = 0$, $m = 2^{31} - 1$. On the left, we have plotted $2\,000\,000$ points (u_i, u_{i+1}), on the right 3000 pairs (i.e., 6000 random numbers plotted as pairs).

- Use a common source of randomness for all threads.
- Use different RNGs for different threads.
- Use a single RNG split into equally-spaced blocks.
- Use one RNG with random seeds.