

Numerical Finance Reading Course

Sheet 2 - Sample Solution

Exercise 1: Discrepancy

Prove Proposition 2.3.3. In part (b), consider only $m = 1$.

Solution:

a) $0 \leq D_N \leq 1$: It holds

$$\begin{aligned} D_N(X) &= \sup_{Q \in \mathcal{Q}} \left| \frac{\#\{x_i \in X : x_i \in Q\}}{\#X} - \text{vol}(Q) \right| \\ &\leq \sup_{Q \in \mathcal{Q}} \left| \max \left\{ \frac{\#\{x_i \in X : x_i \in Q\}}{\#X}, \text{vol}(Q) \right\} \right| \leq 1. \end{aligned}$$

$D_N \geq 0$ is obvious.

b) $D_N^* \leq D_N \leq 2^m D_N^*$:

As $Q^* \subset Q$, the first inequality is obvious.

Let $m = 1$. Then it holds for all $Q \in \mathcal{Q}$: $Q = [a, b)$ for some $a, b \in [0, 1]$, $a < b$ and $\text{vol}(Q) = b - a$. Similarly, $Q^* \supset Q^* = [0, c)$ for some $c \in [0, 1]$ and $\text{vol}(Q^*) = c$. Hence,

$$\begin{aligned} D_N(X) &= \sup_{Q \in \mathcal{Q}} \left| \frac{\#\{x_i \in X : x_i \in Q\}}{\#X} - \text{vol}(Q) \right| \\ &= \sup_{a, b \in [0, 1], a < b} \left| \frac{\#\{x_i \in [a, b)\}}{\#X} - (b - a) \right| \\ &= \sup_{a, b \in [0, 1], a < b} \left| \frac{\#\{x_i \in [0, b)\} - \#\{x_i \in [0, a)\}}{\#X} - (b - a) \right| \\ &= \sup_{a, b \in [0, 1], a < b} \left| \left(\frac{\#\{x_i \in [0, b)\}}{\#X} - b \right) - \left(\frac{\#\{x_i \in [0, a)\}}{\#X} - a \right) \right| \\ &\leq \sup_{b \in [0, 1]} \left| \frac{\#\{x_i \in [0, b)\}}{\#X} - b \right| + \sup_{a \in [0, 1]} \left| \frac{\#\{x_i \in [0, a)\}}{\#X} - a \right| \\ &= 2D_N^*. \end{aligned}$$

c) $D_N^* \geq \frac{1}{2N}$ for $m = 1$:

From part (b) follows that $D_N^* \geq \frac{1}{2}D_N$. Consider now D_N . For any $x \in X$ and $\epsilon > 0$, the interval $J = [x, x + \epsilon) \cap [0, 1)$ is in \mathcal{Q} . Hence

$$\begin{aligned} D_N(X) &= \sup_{Q \in \mathcal{Q}} \left| \frac{\#\{x_i \in X : x_i \in Q\}}{\#X} - \text{vol}(Q) \right| \\ &\geq \left| \frac{\#\{x_i \in J\}}{\#X} - \text{vol}(J) \right| \\ &\geq \left| \frac{1}{N} - \epsilon \right|, \quad \text{as } x \in J \\ &\geq \frac{1}{N} - \epsilon. \end{aligned}$$

This concludes the proof.

Exercise 2: Monte Carlo Integration

a) Evaluate the following integral by Monte Carlo integration and show the convergence rate:

$$\int_{-1}^1 \sqrt{1-x^2} dx.$$

b) Antithetic variables use the fact that if $u \sim U[0, 1]$ then also $\tilde{u} := 1 - u \sim U[0, 1]$. Using $u_1, \tilde{u}_1, u_2, \tilde{u}_2, \dots$ in a simulation might reduce the variance σ_F if $\text{Cov}(F(u), F(\tilde{u})) < 0$, as is the case for example for monotone functions F .

Compute the integral

$$\int_0^1 (x-b)^3 dx$$

by Monte Carlo integration for $b = 0.55$ and $b = 0.5$ with and without the use of antithetic variables and compare the error and the convergence rates.

Solution:

a) To use Monte Carlo integration based on $U[0, 1]$ -distributed pseudo random numbers, we have to rewrite the integral using the change of variables $x = -1 + 2y$, $dx = 2dy$:

$$\begin{aligned} \int_{-1}^1 \sqrt{1-x^2} dx &= \int_0^1 \sqrt{1-(2y-1)^2} 2dy \\ &= \int_0^1 4\sqrt{y(1-y)} dy. \end{aligned}$$

Then an implementation might look like the following:

```

int main(){

    /*
     * UNIX RAND Generator
     */
    long long int a = 1103515245LL;
    long long int b = 12345LL;
    long long int M = 2147483648LL;
    long long int seed = 1;
    LCG urand(a, b, M, seed);

    /*
     * Monte Carlo Integration
     */

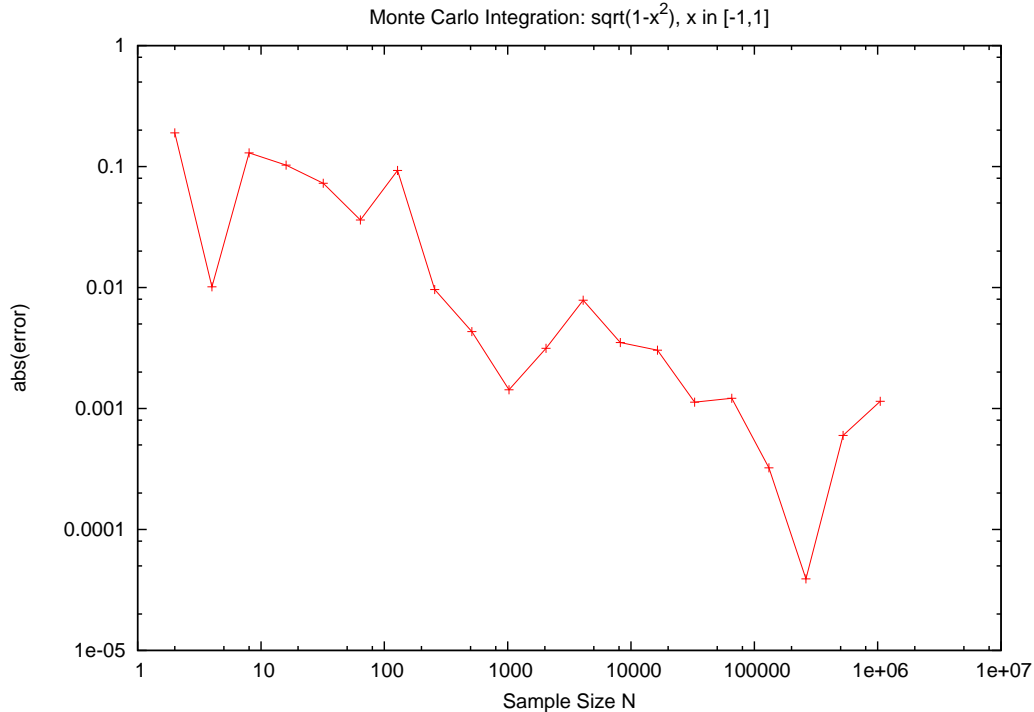
    // Analytic solution
    double sol = 0.5*M_PI;

    // Compute integral for different sample size
    // to observe convergence rate
    for(int N = 10; N <= 10000000; N *= 10){
        double integral = 0;
        for(int j = 0; j < N; j++){
            double u = urand.uni();
            integral += 4.*sqrt(u * (1 - u));
        }
        // Write sample size and (absolute) error to standard out
        cout << N << " " << abs(integral/(double)N - sol) << endl;
    }

    return 0;
}

```

A log-log plot shows that the convergence rate is indeed $O(n^{-\frac{1}{2}})$:



b) A sample implementation is

```

int main(int argc, char* argv[]){

    // Usage
    if(argc != 2){
        cout << "Usage: " << argv[0] << " " << endl;
        return 0;
    }

    /*
     * UNIX RAND Generator
     */
    long long int a = 1103515245LL;
    long long int b = 12345LL;
    long long int M = 2147483648LL;
    long long int seed = 1;
    LCG urand(a, b, M, seed);

    // Define parameter b
    double c1 = atof(argv[1]);
    // Analytic solution
    double sol = 0.25 * ( pow(1.-c1, 4) - pow(c1, 4) ) ;

    /*
     * Standard Monte Carlo Integration
     */
    for(int N = 2; N <= (1<<20); N *= 2){
        double integral = 0;
        for(int j = 0; j < N; j++){
            double u = urand.uni();
            integral += pow(u-c1, 3);
        }
        cout << N << " " << abs(integral/((double)N) - sol) << endl;
    }
    cout << endl << endl;

    /*
     * Monte Carlo Integration with antithetic variables
     */
    // In each loop 2 pseudo random numbers are generated, hence
    // we need only half the number of loops.
    for(int N = 1; N <= (1<<19); N *= 2){
        double integral = 0;
        for(int j = 0; j < N; j++){
            double u1 = urand.uni();
            double u2 = 1. - u1;
            integral += pow(u1-c1, 3);
            integral += pow(u2-c1, 3);
        }
        cout << 2*N << " " << abs(integral/((double)(2*N) - sol) << endl;
    }
    return 0;
}

```

