Exercise given by Danilo Ardagna

Implement with MPI a parallel function to approximate definite integrals on limited intervals $\Omega \in \mathbb{R}$ with the Monte Carlo method. The required prototype is as follows:

```
std::pair<double, double>
montecarlo (const std::function<double (double)> & f, unsigned long N);
```

where f is the integrand and N is the sample size. Assume N is known only on the master node. For this exercise, consider $\Omega = [-1, +1]$. The pair returned by the function must contain the approximation of the integral as first element, while the second one is its estimated variance. This applies to all the MPI processes.

Monte Carlo method Recall that the Monte Carlo quadrature method draws N observations from a random variable $X \sim U(\Omega)$ and estimates the integral with:

$$Q\_N = |\Omega|\overline{Y_N}$$

where$Y = f(X)$. Since $|\Omega| = \int_\Omega dx$, by the law of large numbers:

$$\lim_{N \to +\infty} Q_N = I = \int_\Omega f(x)dx$$

Let $\sigma^2 = \mathrm{Var}(Y)$, then the variance of the quadrature formula is:

$$Var(Q_N) = \frac{|\Omega|^2}{N}\sigma^2$$

whence, exploiting the unbiased variance estimator S_N^2 for σ^2, it is possible to derive an estimator of the quadrature variance:

$$S_{Q_N}^2 = \frac{|\Omega|^2}{N}S_N^2$$

Random numbers

The <random> header provides implementations of pseudo-random number generators and statistical distributions. See the following example and recall that every process should seed differently its random engine. Initialization:

```
std::default_random_engine engine (289);
std::uniform_real_distribution distro (-1., 1.);
```

Pseudo-random number generation:

```
const double x = distro (engine);
```