3 Monte Carlo Integration

The main task in this exercise is to implement a numerical integration for a one-dimensional function $f(x) \to y$ using a Monte Carlo technique.

$$I = \int_a^b f(x)\,dx \quad \approx \quad \frac{1}{N}\sum_{i=1}^{N} f(\epsilon_i) \tag{1}$$

Where ϵ_i $i = 1, 2...N$ consist of random numbers in the range (a, b). An easier way to implement the random numbers is to transform the boundaries of the integral from (a, b) to $(0, 1)$.

3.1 Transformation

The first integral is defined by:

$$I_1 = \int_a^b \sin(x)\,dx \tag{2}$$

with a simple substitution by $u = \frac{x-a}{b-a}$ and $x = u(b-a) + a$ $dx = du(b-a)$we could rewrite (2) into a different boundary.

$$I_1 = (b-a)\int_0^1 \sin(x(b-a)+a)\;dx \approx \frac{b-a}{N}\sum_{i=1}^{N}\sin(\epsilon_i(b-a)+a) \tag{3}$$

The second integral is defined by:

$$I_2 = \int_a^b \sin^2\left(\frac{1}{x}\right)dx \tag{4}$$

and now we could rewrite (4) as described above.

$$I_2 = (b-a)\int_0^1 \sin^2\left(\frac{1}{x(b-a)+a}\right)\;dx \approx \frac{b-a}{N}\sum_{i=1}^{N}\sin^2\left(\frac{1}{\epsilon_i(b-a)+a}\right) \tag{5}$$

The third integral is defined by:

$$I_3 = \int_a^b x^3\,dx \tag{6}$$

and now we could rewrite (4) as described above.

$$I_3 = (b-a)\int_0^1 \Big(x(b-a)+a\Big)^3\;dx \approx \frac{b-a}{N}\sum_{i=1}^{N}\Big(\epsilon_i(b-a)+a\Big)^3 \tag{7}$$

3.2 Random number generator

The sequence to evaluate our random numbers ϵ_i is implemented by creating a MASTER RNG with a fixed seed. This MASTER RNG creates the seeds for the RNGs for the other threads when the program is parallelized. Those RNGs create $N = sampl$ random numbers.

3.3 Runtime analysis

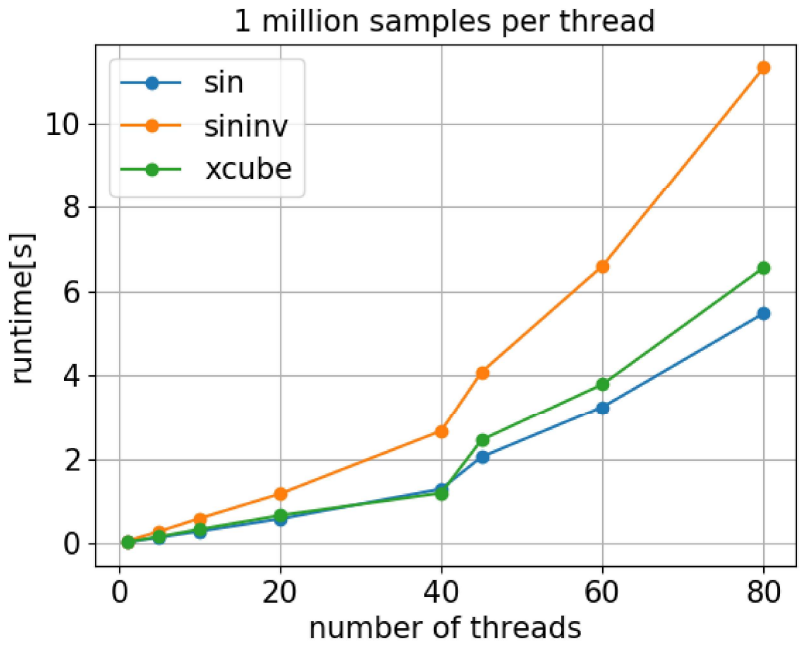

Figure 4: Different runtimes for 1 million samples per thread

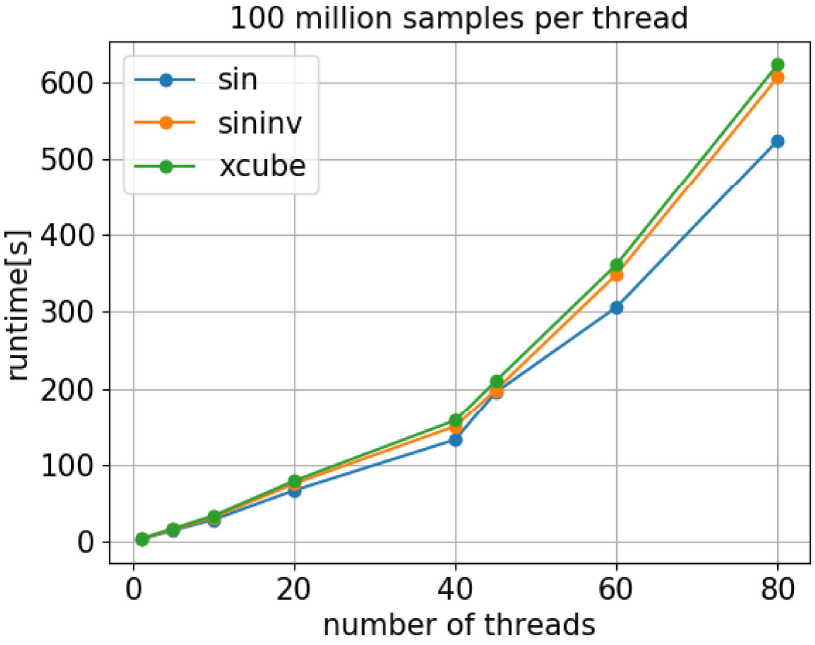

Figure 5: Different runtimes for 100 million samples per thread

The increased runtime per thread is due to the overhead of spawning the threads and the reduction at the end. The sightly increased slope in the runtime when using more than 40 threads is due to

the architecture of the cluster. Every node consists of $2 \times 20 = 40$ physical cores with potential hyper-threading. From thread 1 to 20 we might work on one CPU and then from thread 21 to 40 we probably start using the second CPU as well. After thread 40 we have to make use of hyper-threading. This is the reason why the slope of the runtime increases slightly. To picture this behaviour better we also simulate it for 45 and 60 threads. This behaviour means we only reached a poor weak scaling.