1 Discussion

1.1 Monte Carlo

Looking at the results, the non-deterministic nature of Monte Carlo integration shines through. They are not consistent across runs and seem to fluctuate randomly (which they of course do). However, looking at the standard deviation and error, the trend is that the accuracy increases - which is a good sign. The approximations also come "quite" close (meaning order of 10^{-3}...). From equation (**??**) in section **??** Theory, its also worth noting that the standard deviation decreases by order of $\frac{1}{\sqrt{N}}$, regardless of how many dimensions you integrate. Compared to the Gaussian-Quadrature methods, this makes Monte Carlo integration way more viable for multi-dimensional integral solving.

As a note to

1.2 Parallelization

From table **??** it is easy to understand the impact of correct optimization. Not only was the paralellization of the code a big time-saver but also the vectorization flag (-O3) made a really dramatic impact.

Both from no optimization, to parallelization, and from vectorization to vectorization and parallelization, the time spent is halved. However, this was parallelized over four cores, so shouldn't the time be one fourth of the original? The bottleneck is probably memory speed, as we ran the same calculations on a octacore processor with more capacity, but same frequency RAM, and achieved the same results.

This means that further improvements on the parallelization can be done by using faster memory, or changing the code to access memory less frequently.