1 Discussion

1.1 Gaussian quadrature

From the tables presented in the results section(**??** and **??**) one can simply compare the two methods in accuracy and stability with increasing N. It seems for the Quadtrature methods that a higher number for integration points,N, beyond what discussed in the results(N¿ 11), does not yield better results. Te reason for this might be due to the function beeing really close to zero beyond a = -3, b = 3. The Laguerre and Legandre polynomials might also increse for a higher N. In the special case for N = 27, a = -2.9 and b = 2.9 the results improove. A good improvement of the experiment/study might therefore be to compare results for different integration limits eg. [-3,3] and [-4,4]. Comparing Legandre with Laguerre there is a seginficant improvment in presision, and to some point stability when increasing N.

1.2 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 the large difference between the standard deviation and the error, this is probably due to the high unceirtanty in the integral result. We might get lucky and get a spot on answer - giving us a tiny error - but other times we might not. This is reflected in the standard deviation being quite high. This is an issue, since we don't get quite as accurate answers as we might've hoped, but it is probably due to not efficient enough code and computers that aren't powerful enough.

1.3 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 a mix of memory speed, small cache and low processing power as the memory speed upgrade of 12% on the octa-core PC is not enough to justify the 40% speed increase compared to the quad-core PC.

This means that further improvements on the parallelization can be done by using faster memory, changing the code to access memory less frequently, and to add more processing power.