1 Results

1.1 Gauss-Legendre

Solving our integral with Legendre polynomials gives unstable results for $\lambda = 5$ as seen in table 1. Though with a careful choice of $N = 27$ and integration limits $\lambda = 2.9$ our results are precise with 4 leading digits after the decimal point. The error is calculated as the absolute error for both Laguerre and Legendre. The results from our Legendre integration program are found in table 1.

,

Legendre		
N	Approximate integral	Error
11	0.297447	0.104681
13	0.318350	0.125584
15	0.315863	0.123098
17	0.302268	0.109502
19	0.285064	0.092298
21	0.268075	0.075310
25	0.240135	0.047370
27	0.229623	0.036858
27*	0.192725	0.000039

Table 1: Values of the integral for different N's, calculated with Gauss-Legendre. Integration limits are $x \in [-5, 5]$. *: Special case with integration limits $x \in [-2.9, 2.9]$

1.2 Gauss-Laguerre

Improving our algorithm using Legendre polynomials for angles and Laguerre polynomials for radial parts improved accuracy of our results. An increase in N from N = 11 to N = 15 gives an increase in precision, though for any higher increase the accuracy decreases slightly, which is shown in table 2.

Laguerre		
N	Approximate integral	absolutt error
11	0.183021	0.009743
13	0.190217	0.002548
15	0.193285	0.000520
17	0.194396	0.001630
19	0.194732	0.001965
21	0.194807	0.002050
25	0.194804	0.002030
27	0.194795	0.002029

Table 2: Values of the integral for different N's, calculated with Gauss-Laguerre. Integration limits are $x \in [-5, 5]$.

1.3 Monte Carlo

1.3.1 Naïve approach

The results from our Monte Carlo integration program (main.exe), are listed in table 3.

Naïve Monte Carlo			
N	Approximate integral	Standard deviation	Error
10^5	0.21953065	0.154683	0.026764935
10^6	0.14149215	0.0368397	0.051273556
10^7	0.16704012	0.023165	0.025725592
10^8	0.17903453	0.00936631	0.013731177
10^9	0.19105511	0.0041004	0.0017106036

Table 3: Results from running Monte Carlo with cartesian coordinates and integration limits $x \in [-5, 5]$ - our approximation of infinity.

For higher N's, the approximated integral get closer to the actual value and the standard deviation decreases. The error ($|\text{Exact} - \text{Approximated}|$) does however not match up with the standard deviation, and oscillates a bit up and down, despite having a decreasing trend.

1.3.2 Importance sampling

The results from our Monte Carlo integration program (main.exe), are listed in table 4.

Improved Monte Carlo			
N	Approximate integral	Standard deviation	Error
10^5	0.13773907	0.284624	0.055026645
10^6	0.19068327	0.405372	0.0020824368
10^7	0.2075781	0.381901	0.014812393
10^8	0.19459392	0.092418	0.001828214
10^9	0.20918288	0.0646068	0.016417166

Table 4: Results from running Monte Carlo with importance sampling along the exponential distribution and using spherical coordinates.

The improved Monte Carlo integration gets within a small error margin for smaller N's than the naïve, However, it over- and undershoots randomly. The trend is that the standard deviation decreases, but does not match up with the error ($|\text{Exact} - \text{Approximated}|$).

1.4 Paralellization

Our paralellization results was achieved using a quad core Intel Core i5-8250U processor with 6MB cache at 1.6GHz base clock, which boosted to 3.4GHz during testing. Thermal throttling was avoided. The memory was 4GB 2133MHz LPDDR3 soldered on board. See table 5

We also ran this test on an octa-core processor with memory of 8GB 2400MHz (12.5% faster), and achieved an additional speedup compared to the abovementioned computer. See table 6

For runtime imputs the number of samples was set to 10^8, with an approximation of infity of $\lambda = 5$.

Runtime with different optimizations				
Compile flags	-O3 -fopenMP	-O3	-fopenmp	No optimization
Naive MC	12s	31s	71s	173s
Improved MC	15s	38s	79s	200s

Table 5: Shows the time spent on the same calculations with different compile parameters on a quad core processor.($N = 10^8, \lambda = 5$)

Runtime with optimization on octa-core		
Compile flags	-O3 -fopenMP	% faster than the quad-core
Naive MC	8s	50%
Improved MC	11s	36%

Table 6: Shows the time spent on the Monte-Carlo calculations on an octa-core system.($N = 10^8, \lambda = 5$)