5615 CUDA—Assignment 3
Exponential Integral Calculation

Hugh Delaney

May 25, 2021

1 CUDA Implementation

Please see `gpu_funcs.cu`.

Advanced techniques used:

- Constant/Shared memory—constant memory was used in the first implementations, but it was markedly slower than initializing shared memory with hard coded values. The final implementation uses shared memory as a fast alternative to constant memory.
- Multiple Cards/Streams—Multiple cards are used if you provide the `-s` option. This gives a real boost to performance, especially due to using `cudaMemcpyAsync()`, which allows memory transfer on one card to overlap with computation on the other card. `make run2` will run the code on two cards and report individual timings for each card. Since streams are understood to be launched simultaneously, we can take the overall run time as the greater of the two individual timings.

Advanced techniques considered:

- Texture memory—Texture memory was considered so that we could index between a and b with floating point indexes. However this was decided against since the overhead of creating and initializing texture memory outweighed the cost of creating x values in kernel, which requires no cudaMemcpys before the kernel can start.
- Dynamic Parallelism—Potential approach of launching one thread per n value, and precomputing `psi` (if `a<=1.0`), before launching enough threads/blocks to compute the `numberOfSamples` x values from `a` to `b` for that given value of `n`.

2 Performance