1) **Code in cuda-integrators folder**

a)

```
Hello from thread 89!
Hello from thread 90!
Hello from thread 91!
Hello from thread 92!
Hello from thread 93!
Hello from thread 94!
Hello from thread 95!
dgin@cscigpu:~/assignments/assignment4/hello_cuda$ ./cuda_hello 1025
dgin@cscigpu:~/assignments/assignment4/hello_cuda$
```

For both versions of cuda_hello, the thread count is hard limited to 1024 per block. Going any further results in the program terminating as seen above. Because cuda_hello.cu never modifies the block count, I assume this means that it is running with only 1 block, and therefore subject to 1024 threads. With cuda_hello1, we can specify the number of blocks, and each of these blocks can have up to the 1024 limit. Naturally, this means we can scale much bigger with blocks and threads than with only one block's threads.

```
dgin@cscigpu:~/assignments/assignment4/hello_cuda$ ./cuda_hello1 10 2
Hello from thread 0 in block 9
Hello from thread 1 in block 9
Hello from thread 0 in block 8
Hello from thread 1 in block 8
Hello from thread 0 in block 2
Hello from thread 1 in block 2
Hello from thread 0 in block 6
Hello from thread 1 in block 6
Hello from thread 0 in block 1
Hello from thread 1 in block 1
Hello from thread 0 in block 7
Hello from thread 1 in block 7
Hello from thread 0 in block 5
Hello from thread 1 in block 5
Hello from thread 0 in block 4
Hello from thread 1 in block 4
Hello from thread 0 in block 0
Hello from thread 1 in block 0
Hello from thread 0 in block 3
Hello from thread 1 in block 3
```

b)

The default command line arguments for cuda_trap1 are: **n(step size), a(start point), b(end point), blk_ct(number of blocks), th_per_blk(threads per block)**.

An important note is that the total number of threads must be at least as much as the step size or higher.

```
dgin@cscigpu:~/assignments/assignment4/cuda-integrators$ ./cuda_trap1 1000000 100
The area as computed by cuda is: 2.000908e+00
The area as computed by cpu is: 2.001454e+00
Device times:  min = 2.408028e-03, max = 2.807856e-03, avg = 2.468004e-03
  Host times:  min = 1.307201e-02, max = 1.435399e-02, avg = 1.321247e-02
```

Modified cuda_trap1 run^

```
dgin@cscigpu:~/assignments/assignment4/cuda-integrators$ time ./trap
Enter a, b, and n
0.0 3.14159265
1000000
With n = 1000000 trapezoids, our estimate
of the integral from 0.000000 to 3.141593 = 2.001454

real    0m9.253s
user    0m0.015s
sys     0m0.001s
dgin@cscigpu:~/assignments/assignment4/cuda-integrators$ time ./cuda_trap1 1000000 1000
The area as computed by cuda is: 2.000961e+00
The area as computed by cpu is: 2.001454e+00
Device times:  min = 2.421141e-03, max = 2.976179e-03, avg = 2.480645e-03
  Host times:  min = 1.307607e-02, max = 1.458192e-02, avg = 1.330097e-02

real    0m1.037s
user    0m0.795s
sys     0m0.240s
```

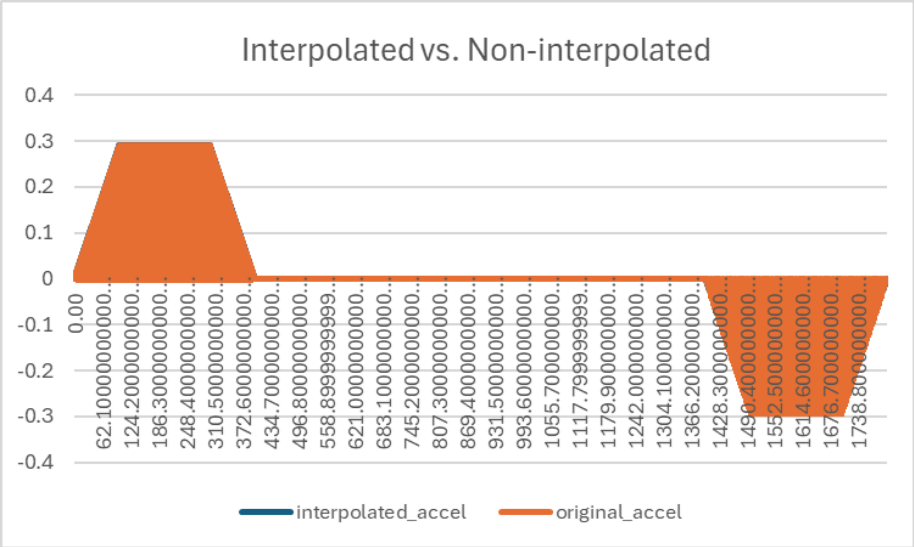
Timed run of both trap.c and cuda_trap1.cu^

I actually ended up getting a slower runtime with CUDA. Assuming that I didn't mess something up on my end, I think that this implies that the problem isn't large enough and that the runtime suffers during resynchronization.

SU = STime/PTime => .015/.795 = **~1.886%** speedup

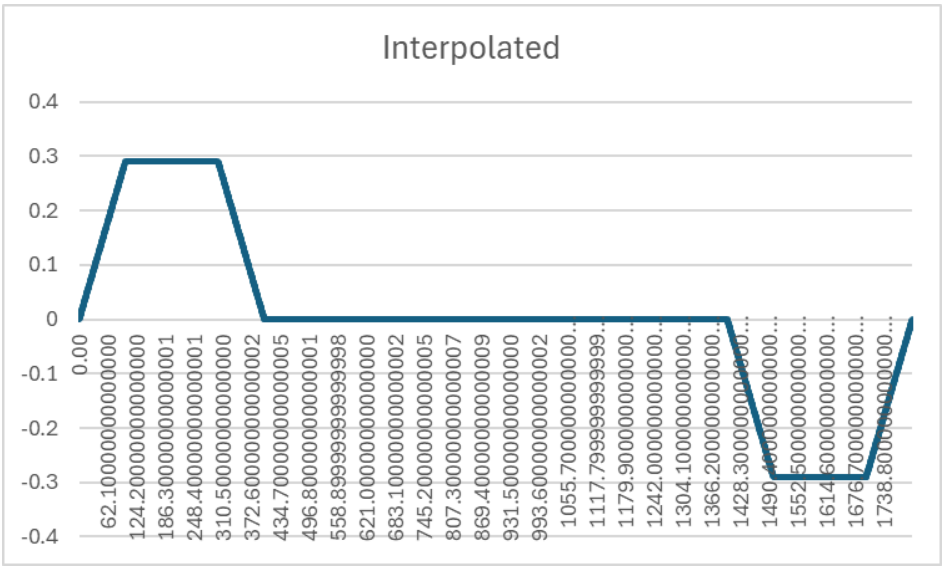
P = C(1-1/SU)/C-1 => 16(1-1/0.01886)/16-1 = **-55.4904** <- This can't be right, but I like the idea of a program with -5549.04% parallel portion

2) **Code in functiongen folder**



Non-interpolated acceleration (m/s/s)

For starters, the new file has 10 times the number of entries, but the linear nature of the graphs leaves little visible difference. There were a few time segments that were off by less than a billionth of a second which caused the time scale to display really grossly long decimal numbers..



Interpolated acceleration (m/s/s)

I couldn't figure out how to make the interpolated line more visible than the non-interpolated line, so I isolated the interpolated line, and as described previously, there is very little noticeable difference due to the linear nature of the graph.

3) **Code is in train folder**

a)

```
dgin@o251-05:~/Documents/csci551/assignment4/train$ mpiexec -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_3
Enter the number of steps:
1800000
Final velocity:   -0.000000001663111
Final position: 122000.265429368155310 meters
```

To change dt between problem parts, I enter the number of steps in relation to the 1800s timespan. A major issue that I ran into was that attempting to integrate position from the freshly integrated velocity was consistently wrong. The velocity was correctly 0 but position would hover around the low 4000 range.

b)

```
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 1 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
1800000
Final velocity:   -0.000000001663140
Final position: 122000.003999757274869 meters

real    0m3.658s
user    0m0.031s
sys     0m0.019s
```

DT = 0.001^

```
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 1 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
18000000
Final velocity:   -0.000000016954608
Final position: 122000.004021860542707 meters

real    0m28.664s
user    0m0.163s
sys     0m0.022s
```

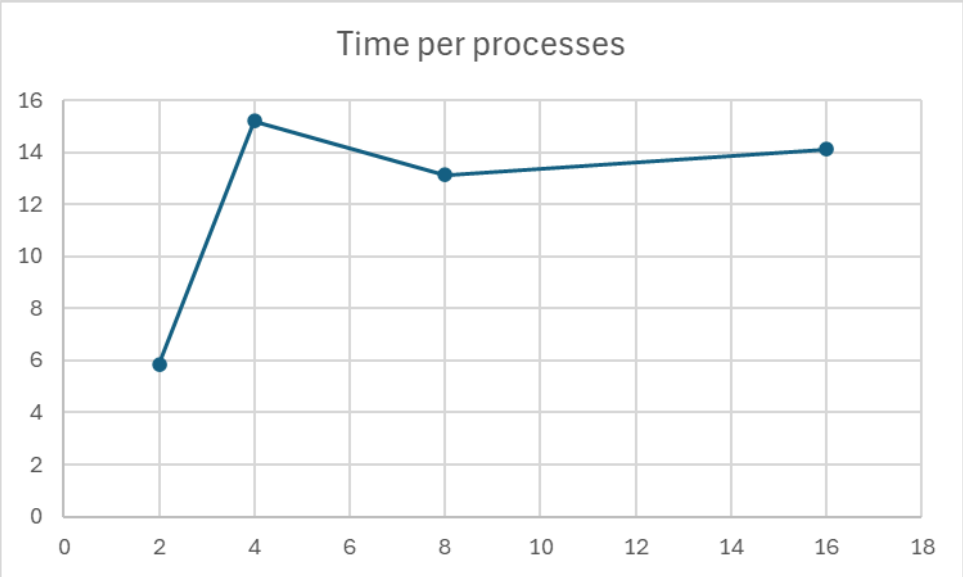
DT = 0.0001^

I already implemented it with MPI, so I just changed the run settings to use 1 process on 1 node. I also changed the number of steps to be 18,000,000 to achieve dt = 0.00001.

c)

```
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 2 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
18000000
Final velocity:   -0.000000016954232
Final position: 122000.012723503736197 meters

real    0m6.086s
user    0m5.861s
sys     0m0.071s
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 4 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
18000000
Final velocity:   -0.000000016954232
Final position: 122000.030143435171340 meters

real    0m5.340s
user    0m15.206s
sys     0m0.119s
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 8 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
18000000
Final velocity:   -0.000029047574003
Final position: 122000.052639328088844 meters

real    0m6.560s
user    0m13.132s
sys     0m11.894s
dgin@o251-05:~/Documents/csci551/assignment4/train$ time mpiexec -n 16 -ppn 1  ./mpi_trap4_3
Enter the number of steps:
18000000
Final velocity:   -0.000058092345725
Final position: 122000.105593413783936 meters

real    0m6.506s
user    0m14.119s
sys     0m10.637s
```

Scaling only increasing process count 2, 4, 8, 16^



Time taken increases beyond 2 probably due to performing synchronization. However, past 4 processes, the time-increase plateaus significantly.