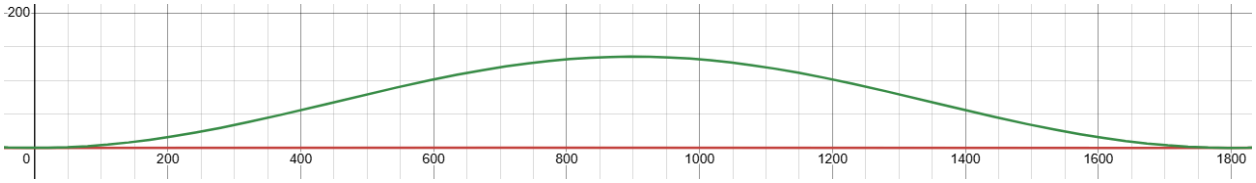
Equations and graphical output verified with Symbolab and Desmos.



$$\int \sin(x)\,dx$$

Solution

$$-\cos(x) + C$$

Relevant code is in the train_integrator folder. Switching output between velocity and distance is a simple comment/uncomment switch in the f() function definition.

Step size 1/100th(s) achieved by setting n=180,000
Step size 1/1000th(s) with n=1,800,000

1) **Left-Riemann code is in mpi_trap4_1.c**

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_1
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 121999.425781250000000
```

Single distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_1
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 122000.110351562500000
```

Scaled distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_1
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000000000000000
```

Single velocity^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_1
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000007629394531
```

Scaled velocity^

I ran into issues of overestimation when trying to scale up and scale out. I believe this may have to do with the float precision addition being spread across processes because the non-scaled output was exactly as expected: slight underestimation due to Left-Riemann sum's gaps under the curve and between rectangles.

2) **Mid-Riemann code in mpi_trap4_2.c**

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_2
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 121999.999999996391125
```

Single distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_2
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 121999.999999997235136
```

Scaled distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_2
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =   -0.000000000000028
```

Single velocity^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_2
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =   -0.000000000000028
```

Scaled velocity^

Mid-Riemann was much more accurate and suffered far less inconsistency between scaled and unscaled outputs. Both distances are very slightly beneath the expected 122,000 meters. The velocities were slightly negative. However, the non-scaled output was negative too, so this leads me to believe that it is moreso the nature of the mid-Riemann sum calculation with double precision that causes this rather than anything to do with MPI.

3) **Float trapezoid code in mpi_trap4_3.c**

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_3
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 121999.421875000000000
```

Single distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_3
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 122000.112304687500000
```

Scaled distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_3
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000000000000000
```

Single velocity^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_3
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000000000000000
```

Scaled velocity^

Float trapezoid's output was a little more inaccurate than I was expecting, but still well within reasonable range of 122,000 meters. The unscaled output was expectedly beneath 122,000. Switching to scaled once again introduced a noticeable difference between the single output though; a seemingly consistent trend with float precision interacting with MPI. The velocity comparison performed far better for the trapezoid though with no difference between scaled and unscaled outputs for the MPI arguments that I have been using for the others.

The output did change when altering the process count.

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 8 -ppn 2 -f c1_hosts ./mpi_trap4_3
Enter a, b, and n
0 1800 180000
With n = 180000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =   -0.000001907348633
```

Scaled velocity w/ increased process count^

With 8 processes set, the final velocity was consistently slightly under the expected value of 0. This is a better display of the float precision issue.

4) **Double trapezoid code in mpi_trap4_4.c**

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_4
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 122000.000000000902219
```

Single distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_4
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 = 121999.999999999039574
```

Scaled distance^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpiexec ./mpi_trap4_4
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000000000000057
```

Single velocity^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 4 -ppn 2 -f c1_hosts ./mpi_trap4_4
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =   -0.000000000000071
```

Scaled velocity^

```
dgin@o251-03:~/Documents/csci551/assignment3/train_integrator$ mpirun -n 8 -ppn 2 -f c1_hosts ./mpi_trap4_4
Enter a, b, and n
0 1800 1800000
With n = 1800000 rectangles, our estimate
of the integral from 0.000000 to 1800.000000 =    0.000000000000021
```

Scaled velocity w/ increased process value^

The unscaled version is unexpectedly slightly over 122,000. To my understanding, trapezoid sum should always be slightly under, which the scaled output instead gave. This could be due to precision double. Between the scaled and unscaled outputs, they are only off of the expected 122,000 by a billionth of a second. Double trapezoid has a very tight range and is expectedly competing with mid-Riemann for most accurate output.
The float implementations are off by between ~0.1 - 0.5 meters of distance between implementations, while the double implementation outputs are off by less than a billionth of a meter if not less.