	A	B	C	D	E	F	G	H	I
1	Run	Width	Height	Frame	Parallel Time (sec)	Parallel Float Time (sec)	Serial Time (sec)	Speed Up (Serial / Parallel)	Speed Up (Serial / Float Parallel)
2	1	320	200	10	0.0013	0.0002	0.2498	192.1538462	1249
3	2	640	400	32	0.0073	0.0019	3.1765	435.1369863	1671.842105
4	3	640	400	60	0.0127	0.0033	6.1496	484.2204724	1863.515152
5	4	1024	768	32	0.0204	0.0054	9.6623	473.6421569	1789.314815
6	5	1024	768	60	0.0365	0.0095	17.558	481.0410959	1848.210526
7	6	1920	1080	60	0.0943	0.0249	47.5685	504.4379639	1910.381526
8	7	2560	1440	60	0.1682	0.0437	84.5633	502.754459	1935.086957
9	8	3840	2160	60	0.3794	0.0968	190.3321	501.6660517	1966.240702
10	9	4096	2160	32	0.1134	0.056	108.4349	956.2160494	1936.3375
11	10	4096	2160	60	0.2048	0.1035	201.5322	984.0439453	1947.171014

The graph above shows all my timings for my code that was ran. The serial time was ran on my computer locally with the gpu being rtx 2070. The parallel time is the default time for the code written in double and the parallel float time is with the code variables being change to float. The speedup also follows that pattern.

We can clearly see the speedup of the code with the variables changed to float has significantly improved the speed up. We see roughly twice better speed up compared to than the speed up with the double variables. The plausible reasons for this are probably because float variables are 4 bytes compared to double 8 bytes which requires less competition. We also see a trend in the speedup of increasing problem size we see a more better speed up. Another reason is probably some gpu architectures are faster with float then double. Also with are input size being 4K and around we will see a faster speedup.