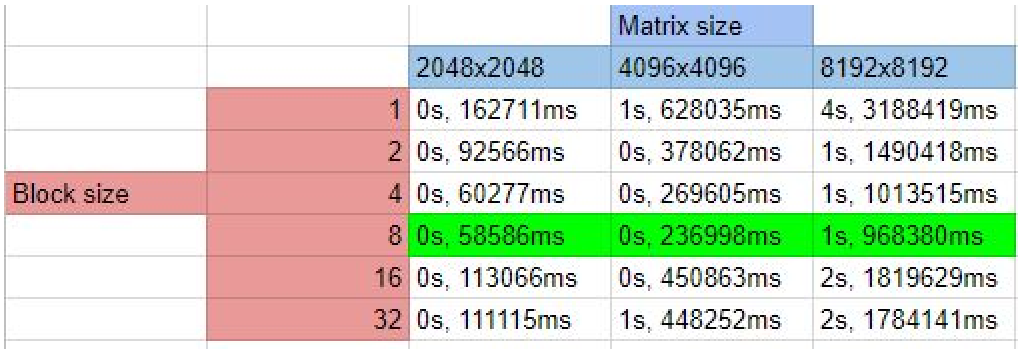
Problem 1 Timing Results
Nick Parisik



			Matrix size	
		2048x2048	4096x4096	8192x8192
	1	0s, 162711ms	1s, 628035ms	4s, 3188419ms
	2	0s, 92566ms	0s, 378062ms	1s, 1490418ms
Block size	4	0s, 60277ms	0s, 269605ms	1s, 1013515ms
	8	0s, 58586ms	0s, 236998ms	1s, 968380ms
	16	0s, 113066ms	0s, 450863ms	2s, 1819629ms
	32	0s, 111115ms	1s, 448252ms	2s, 1784141ms

As shown above, a block size of 8 seems to be the quickest consistently for processing matrices with the given dimensions.

The code becomes faster due to cache hits because of the way we are accessing the elements in our 1d arrays. When we access a specific element in a matrix, elements near that one are also store in a cache line, so accessing them becomes quicker (spatial locality).