First of all I compiled the unoptimized file and i runned it without any optimization flag in compiling with the makefile.

I tried with a lot of different sizes of the matrix in input.

After I used the makefile attached in the folder, that generates 4 different executable file (opt0, opt1, opt2,opt3) which respectively use as flags: -O0 -O1 -O2 -O3.

The result is the following : ( UNOPTIMIZED is with the original makefile) (the first value is the time, the second is the Gflop/s rate)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Matrix size | UNOPTIMIZED | opt0 | opt1 | opt2 | opt3 |
|  |  |  |  |  |  |
| 5\*5 | 0, inf | 0,inf | 1,0.05 G/s | 1,0.05 G/s | 0,inf |
| 15\*15 | 1,0.45 G/s | 1,0.45 G/s | 1,0.45 G/s | 0, inf | 0, inf |
| 150\*15 | 7,6.43 G/s | 7,6.43 G/s | 2,22.5 G/s | 1,45 G/s | 1,45 G/s |
| 150\*150 | 66,0.68 G/s | 66,0.68 G/s | 14,3.2 G/s | 14,3.2 G/s | 7,6,4 G/s |
| 1k\*1k | 4292, 0.46 G/s | 4487, 0.44 G/s | 1135, 1.76 G/s | 1122, 1.78 G/s | 900, 2.22 G/s |
| 10k\*10k | 957835, 0.2 G/s | 957835, 0.2 G/s | 900377, 0.22 G/s | 895584, 0.22 G/s | 950511, 0.21 G/S |
| 25k\*25k | 17 M, 0.073 G/s | |  |  | 22.7 M,0.05 |

Then I used various techinques to modify the matVecMult (such as Strasser’s algorithm and others, that are visible in the optimized.c file and commented). The most efficient was the Loop Unrolling techinque. For this techinque i changed the number of elements calculated in each iteration, and the most efficiency one was 16. This is the graph that compares the times of the unoptimized function and the function implemented by Loop Unrolling with 16 elements:

As visible, with the growht of the Matrix size, the gap is really evident.

At the end I tried to use the Intel Intrinsics, and that was the best efficient method by far.

The function utilizes SIMD instructions by performing array operations on four elements at each iteration. The results are shown in the following graph:

**As said in the optimized.c file, all the functions used are still present in the file itself, and all are commented except for the best one in terms of performance, which is precisely the SIMD implementation.**