

Paper review

Motivation and general structure

The paper and software bundle introduce a library to compute the weight distribution of a random linear code, primitive that is both necessary and hard to compute in polynomial time, and useful in a number of scenarios. The author focuses the optimization steps on the algorithmic side, and also on introducing techniques to leverage vector (SIMD) capabilities of modern processors. The paper is fairly balanced and it is relatively easy to follow.

In general, the revised version of the manuscript addresses many of the comments and suggestions from the first review. The software bundle now includes more detailed test drivers and a comprehensive documentation via Doxygen, which greatly improves the quality of the package. In the introduction, please remove the last two bullet points regarding features. They are out of the scope of the paper.

The main comments for this second review still fall into the vectorization part and the companion experimental results section:

Optimizations and experimental results

This reviewer appreciates the effort in extending the vectorization target to NEON-capable ARM architectures. The effort in generalizing the vector instructions in the algorithms and pseudo-codes is also positive.

The analysis of results (mainly tables 3 and 4) is still the weakest part of the paper, for this reviewer. Specifically:

- It is not clear at all the target architecture used for the experimental part. The authors introduce two different platforms in section 5:
 - Intel Core i5, with no more detail.
 - Intel Xeon Gold 5118, with no more detail.
 - Intel Xeon E3-1220, with no more detail.

It is not clear which platform has been used for the evaluation of the Intel part, and why these three platforms are introduced. Tables 3 and 4 seem to refer to platform 1 (Intel Core i5). This may be a weird Intel Core processor, as Core generations do not support AVX512. If that is the case, how can it be possible to report results on AVX512 in table 3?

In general, the experimental setup description needs to be fully rewritten, clarified (with models and specific characteristics of each processor), prior to continuing with the section.

- Regarding the analysis of results (tables 3 and 4). The scaling between 128-bit, 256-bit and 512-bits vectors is still confusing. For many cases, the speedup is minimal comparing both approaches. Comments on this are

mandatory. As suggested in the first review, profiling the codes becomes mandatory to help explaining the differences between data.

- Authors have added the scalar version results in table 3. The speedup for the vector implementations is excessive. How can it be explained? Again, code profiling needs to be in place in order to give extended insights on this type of observations.
- Table 4 does not include results for AVX-512 or NEON. Is there a reason for this?
- As stated in the first review, please provide suggestions or a qualitative analysis of the optimal vector width selection depending on the target n , k , q values. The answer and modifications in this regard is not satisfactory.
- A detailed analysis/profiling of the code, that illustrate the main parts of the implementation that are affected by different kind of optimizations, is also mandatory. In general, a way of elucidating the amount of speedup gathered from algorithmic modifications in your codes compared with other approaches, and from vector/thread-level parallelization, is also necessary to enrich the paper.

Code review

The provided code does not compile for an ARM architecture (GCC 10.5.0 on an NVIDIA Jetson Orin AGX). Maybe this error does not appear with clang-15, but others also appear with clang-12. Please correct or clearly state the compiler and version required to build the package in ARM targets:

```
$ make
./src/lib_neon.cpp: In function 'long long unsigned int weight_GF4_128(int)':
./src/lib_neon.cpp:2786:15: error: invalid conversion from 'long long unsigned int*' to 'uint64_t'
2786 |         vst1q_u64(res64, result);
      |         ^~~~~~
      |         |
      |         long long unsigned int*
In file included from ./src/lib_neon.cpp:19:
/usr/lib/gcc/aarch64-linux-gnu/10/include/arm_neon.h:27923:22: note:   initializing argument 2 of 'uint64_t vst1q_u64(uint64_t*, uint64x2_t)'
27923 | vst1q_u64 (uint64_t *__a, uint64x2_t __b)
      |         ~~~~~~^~~
./src/lib_neon.cpp: In function 'long long unsigned int weight_CF2(int)':
./src/lib_neon.cpp:2961:19: error: invalid conversion from 'long long unsigned int*' to 'uint64_t'
2961 |         vst1q_u64(res64, result);
      |         ^~~~~~
      |         |
      |         long long unsigned int*
In file included from ./src/lib_neon.cpp:19:
```

```

/usr/lib/gcc/aarch64-linux-gnu/10/include/arm_neon.h:27923:22: note:   initializing argument
27923 | vst1q_u64 (uint64_t *__a, uint64x2_t __b)
      |               ~~~~~^~~
./src/lib_neon.cpp: In function 'void linear_combinations_CF2_SSE_less_than(int, int)':

```

Besides this detail, that needs to be addressed in a further review so that NEON capabilities can be assessed, the new version of the software bundle greatly improves the previous package in both functionality, reproducibility and quality of the documentation.

Typo in the refman.pdf: Title of Section 2.1.3: Makefiels -> Makefiles. Typo in page 15 of the manuscript: depends -> depend.