

LinCodeWeightInv: Library for Computing the Weight Distribution of Linear Codes Over Finite Fields

Maria Pashinska-Gadzheva and Iliya Bouyukliev

The authors have developed a C/C++ library for computing the weight distribution of random linear codes over certain finite fields with cardinalities less than 64. Their library offers various instruction set versions, including SSE4.1, AVX2, AVX512, Neon, and No Vectorization.

The achieved speedup compared to MAGMA and GAP is substantial and varies depending on several factors, such as the field type, code length, code dimension, and the version used.

I believe that the article holds considerable merit and has the potential to be published in this journal. However, there are certain issues that should be addressed prior to its acceptance. These include:

1. The condition “if $h = 1$ then $gf = 1$ end if” prevents entry into the for loop on line 13 of Algorithm 2. Consequently, starting with $r = 1$ and $h = 1$ results in these values never being updated to $r + 1$ and $h + 1$. It appears that simply changing $gf = 1$ to $gf = 2$ would resolve this issue. (This correction applies to all other algorithms as well.)
2. In line 14 of Algorithm 2, the variable e is treated as an integer, which may not be suitable if $q = p^r$ is a prime power. To address this, consider representing e as α^{q-e} , where α is a primitive element of the finite field. This adjustment ensures that if $e = 1$, then $\alpha^{q-1} = 1$, otherwise, it iterates through all elements of the finite field.
3. In Section 3.3, when referring to x , is it meant to represent α , a primitive element of \mathbb{F}_q ?
4. In Section 5, it is expected a broader exploration of lengths and dimensions; however, the current presentation offers only two lengths and one dimension for each field. While we acknowledge the constraints on table length, an alternative approach could involve selecting specific finite fields and illustrating the speedup trends graphically across different values of n and k . This graphical representation would offer readers a clear visualization of how the speedup varies with these parameters.

5. In Section 6, there’s mention of the possibility of adapting the algorithms to efficiently compute the minimum distance. However, it’s important to clarify that in the current work, the focus is primarily on computing the weight distribution. While the algorithms may have the potential for adaptation, their current implementation is not optimized for efficient minimum distance computation. Consequently, the efficiency in this regard may not be comparable to specialized methods.

Additionally, comparisons with existing approaches, such as those referenced in [17] and [27], should be carefully contextualized. It’s evident that the performance for computing the minimum distance might lag behind methods explicitly designed for this purpose, like MAGMA. This disparity in performance could be attributed to the algorithms’ primary focus on weight distribution computation rather than minimum distance determination.

This clarification aims to provide a more precise understanding of the algorithms’ intended application and their performance relative to other methods.