Last factor optimization

To test if a reduced design matrix (RDM) $(r \times n)$ is rL-minimal, we have to check all possible:

- Row permutations
- Column permutations
- Possible set of new basic factors

To see if any, produces a design that is rL-smaller.

A possible set of new basic factors, is a set of r columns among the n, which are linearly independent of each other. To test if a set is linearly independent, we compute the reciprocal condition inversion number.

However, another condition might be interesting, is that the new set of basic factor must contain the last factor added to the parent design, because if it doesn't, it means that the combination has already been tested previously.

Therefore, we could restrict the search for new sets of basic factors, to sets that:

- Are linearly independent
- Contain the last added factor.

This reports test if optimizing the algorithm, by implementing this restriction, yields any advantage in terms of computing time.

Methodology

To test this restriction, we tested all pemutations of a given design and tested if:

1. The new set of basic factors contained the last added factor
2. The new set of basic factors is linearly independent
3. The produced design is rL-smaller

And compiled into a single vector, which, for a single design gives us:

- True if it is rL-smaller
- Number of discarded new sets of basic factors
- Number of iterations needed to find the rL-smaller design

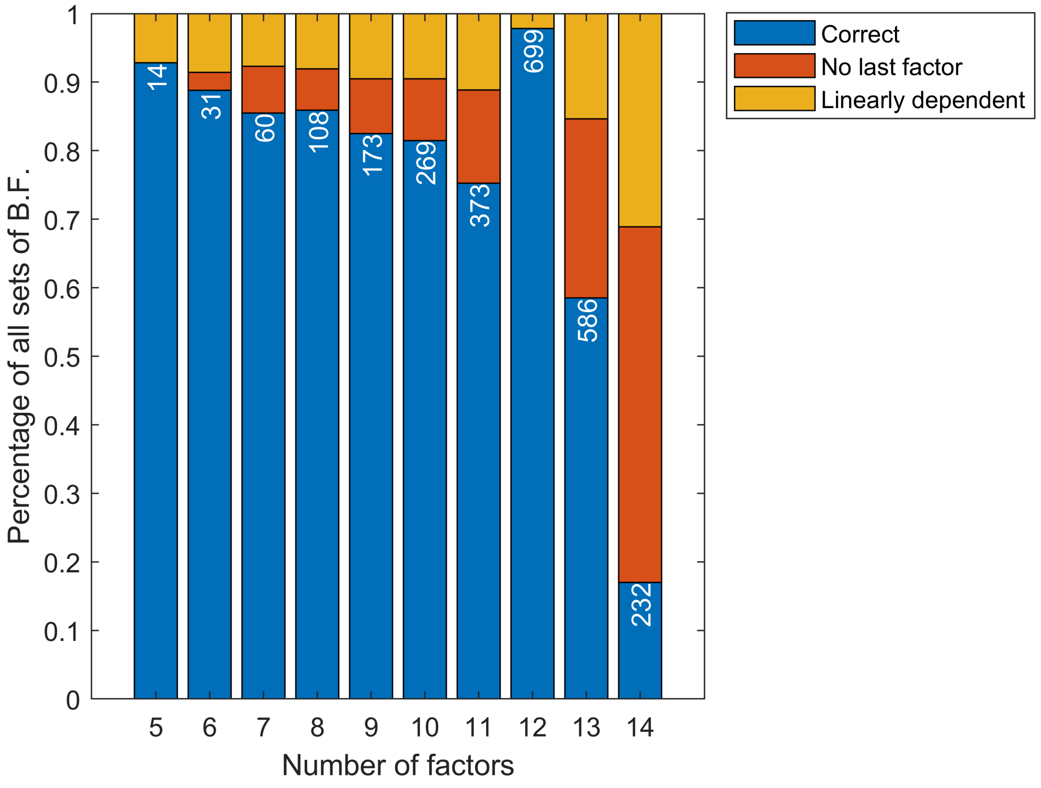
The goal is now to compare if these values are more advantageous when we use the last-factor optimization method (LFopt).

New sets of basic factors

Given a design with n factors, there are $\binom{n}{r}$ ways of picking a set of new basic factors. However, using LFopt, there are only $\binom{n-1}{r-1}$.

The following graphs shows us, for each possible number of factors, the average percentage of new sets of basic factors which:

- Are discarded because they are linearly dependent
- Are discarded because they do not contain the last factor
- Are kept for further analysis



We see that the biggest difference that the LFopt makes, is when there is a large number of factor. This is convenient since the number of sets to test is proportional to n. This proves that the LFopt will yield large time gains for designs with a lot of factors.

Number of iterations

We've seen that the LFopt decreases the number of new set of designs to test. We will now check if, among those remaining sets, the number of iterations needed to obtain an rL-smaller design (if such design exists), is lower using the LFopt algorithm.

To check, we computed the maximal number of iteration needed to reach the rL-smaller design as $\binom{n}{r} \times r!$ when using the normal algorithm and as $\binom{n-1}{r-1} \times r!$ when using the LFopt algorithm. Then we simply computed the percentage of that maximum, that was actually needed to reach the rL-smaller design.

```
T = 10×3 table
```

	n	Average number of iterations needed	Max number of iterations possible
1	5	18	96
2	6	44	240
3	7	113	480
4	8	154	840
5	9	218	1344
6	10	365	2016
7	11	816	2880
8	12	187	3960
9	13	2125	5280
10	14	5376	6864

It is clear, from the table, that even though the theoretical number of iterations needed to reach the rL-smaller design is small, the actual average number of iterations is lower.