Experiment 1 - Geometric vs constant bounds

Before

Hypothesis

Geometric bounds are better. If the Brumley&Tuveri method is used, even in the presence of noise.

Setup

- The dimensions: D = {50, 52, ..., 140}
- The number of signatures used: N = {500, 600, ..., 7000, 8000, 9000, 10000}
- The constant bounds: L = {2,3,4}
- The data source: data = {real, sim}
- each run 5 times, random sampling $n \in N$ signatures, constructing lattice from $d \in D$ shortest, then either doing constant bounds with $l \in L$ or geometric bounds from 2 bits on. Doing SVP with progressive BKZ.

2 * [(3 * 68 * 45) + (1 * 68 * 45)] = 24 480 tasks

each task does 5 runs of attack: 122 400 runs of attack.

Schedule tasks with (l, n, d), where $l \subseteq L$ or $\emptyset$ for geometric bounds:

```
for data in {real, sim}
    for d in D
        for n in N
            schedule one const task with ({2,3,4}, n, d) and three times the walltime
            schedule one geom task with (0, n, d)
```

That makes 12 240 tasks.

Outputs

Each task outputs {real,sim}_{geom,const2,const3,const4}_{n}_{d}.csv with 5 lines for the 5 runs:

seed, success, duration, last_reduction_step, info, #liars, real_info, bad_info, good_info

Visualizations

For both real and sim data:

- 3D plot, x:N, y:D, z: number of successes; this for geom,const2,const3,const4.
- 3D plot, x:N, y:D, z: last reduction step; this for geom,const2,const3,const4.
- 3D plot, x:N, y:D, z: avg. duration of successful run; this for geom,const2,const3,const4.
- 2D lineplot, x:N, y:sum over d in D (number of successed); all of geom,const2,const3,const4 in one plot.

Why?

- Gives a lot of insight into the parameter space.
- Gives a good baseline for future experiments, can compare in different parts of the parameter space.
- Shows superiority of out parameter choice of geometric distribution.

During

Run 25.09. 21:00

- Some tasks failed to run, due to some CPython error (likely missing Cython?) or some weird architecture, or the fact that fpylll was likely compiled with march=native or mtune=native? TODO: Determine the reason and rerun.
- Some tasks had issues due to the use of fish shell to script the task run, when run in parallel it needs to lock and access some user specific files, which took too much time, sometimes timed out: `<E> fish: Unable to open universal variable file '/storage/brno11-elixir/home/j08ny/.config/fish/fishd.elmo5-15.hw.elixir-czech.cz': Stale file handle <W> fish: Locking the universal var file took too long (25.683 seconds). Could not chdir to home directory <E> fish: Your personal settings will not be saved. <E> fish: Unable to locate config directory derived from $HOME: '/storage/ostrava1/home/j08ny/.config/fish'. <E> fish: The error was 'File exists'. <E> fish: Please set $HOME to a directory where you have write access.`
- Bad(incomplete) tasks will either not have resulting csv files, or those files will not have 5 lines or those lines will contain the regex "0,.,LLL", collect those and rerun.

Few reruns

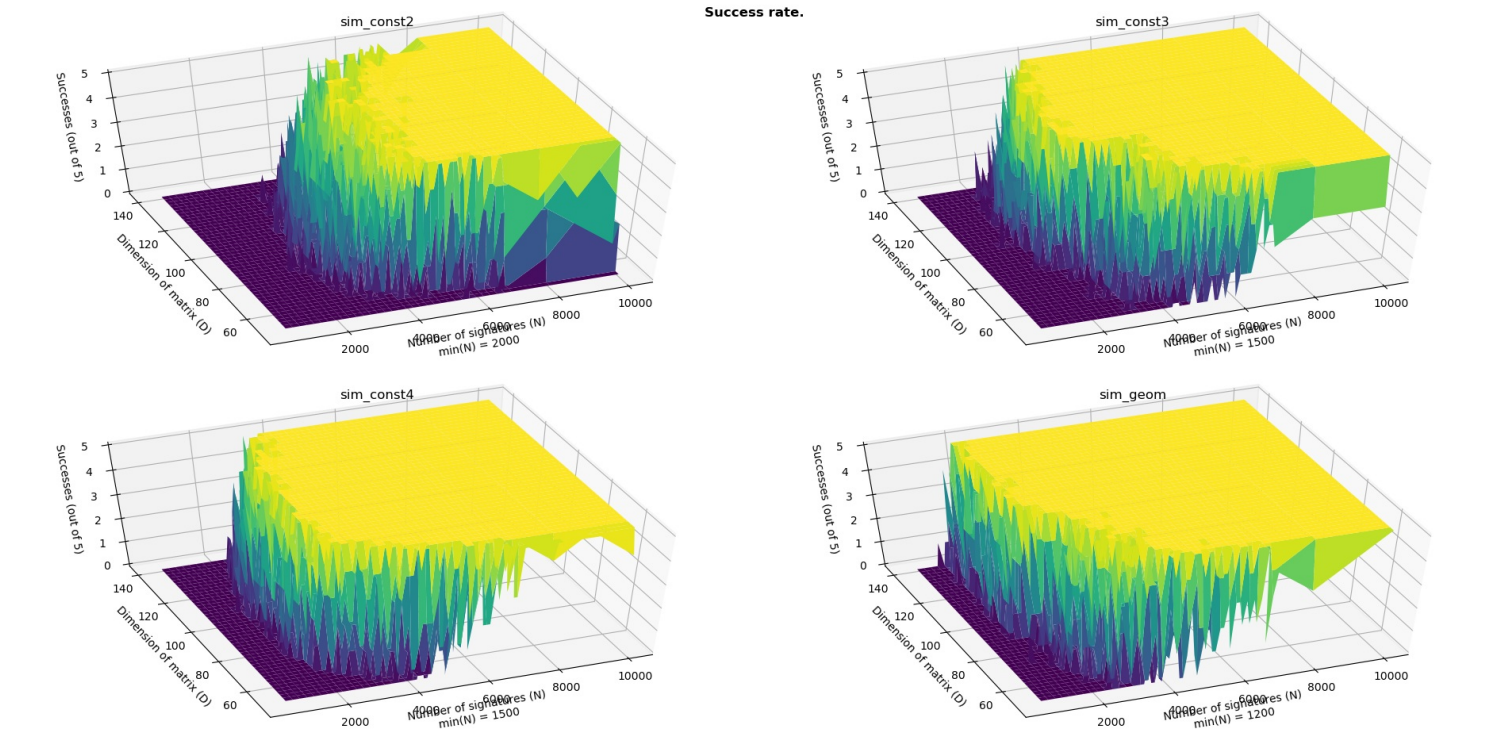
- First did a rerun of those tasks with "0,.,LLL" which failed due to a C runtime error in G6K native module. Then I actually fixed (recompiled G6K without -march=native) and rerun again.
- Then also did a rerun for those tasks that were missing the results files, likely due to the above error or some other (ran out of time).

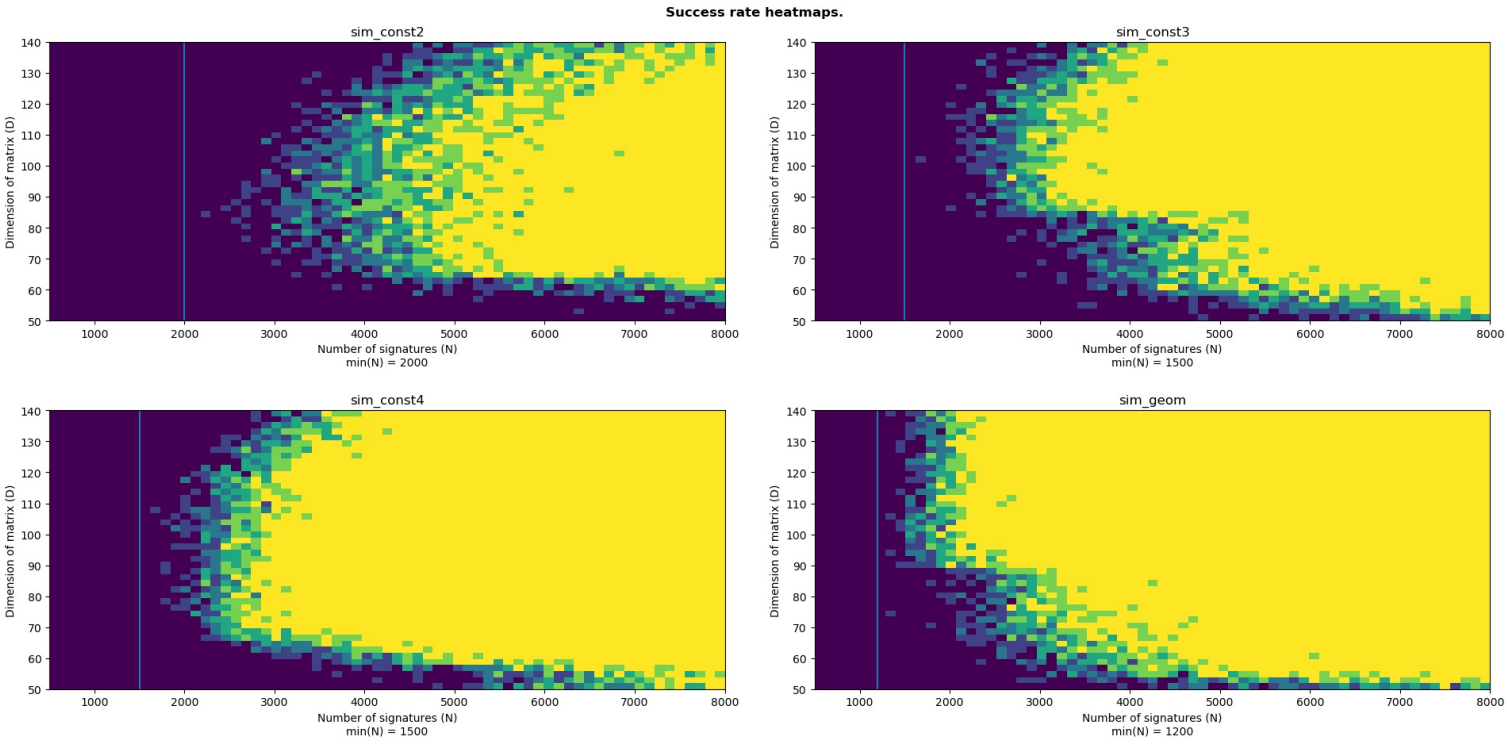
TODO: Real data!!!!!

After

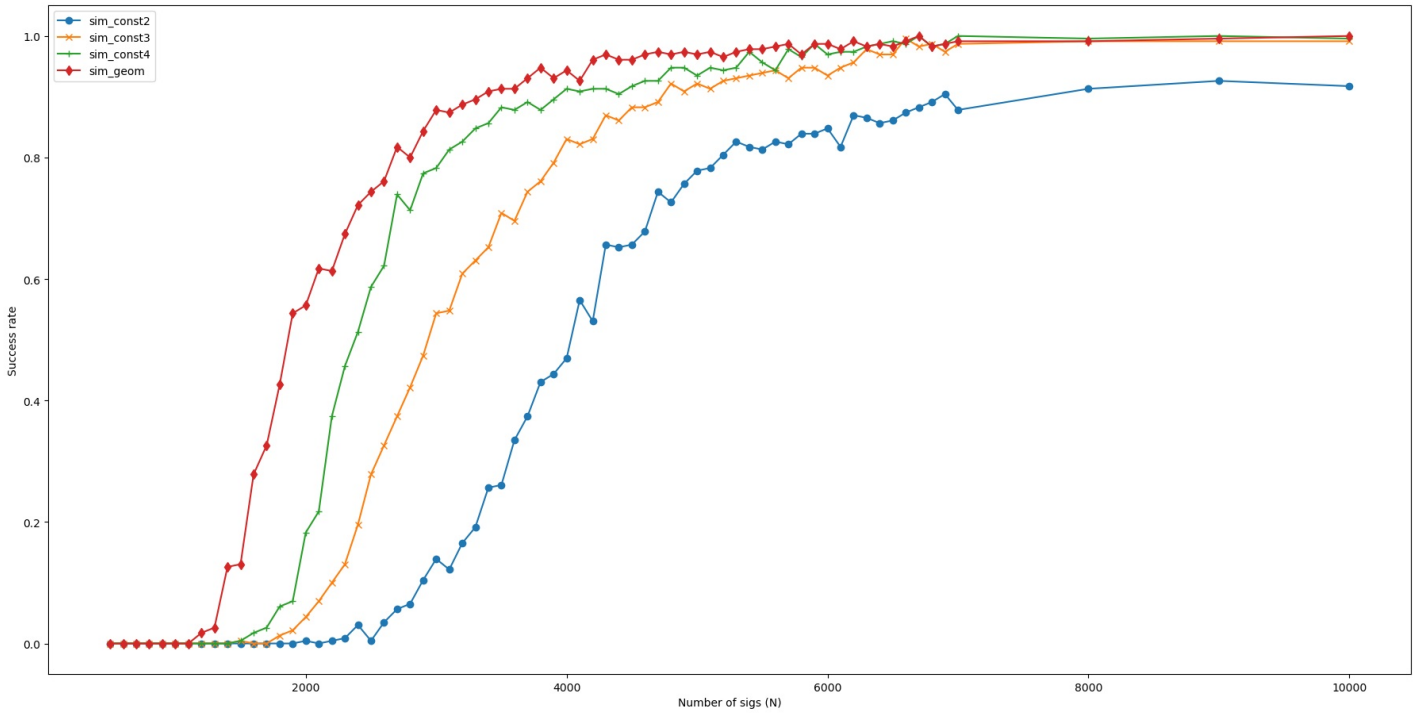
Figures

(only simulated data)



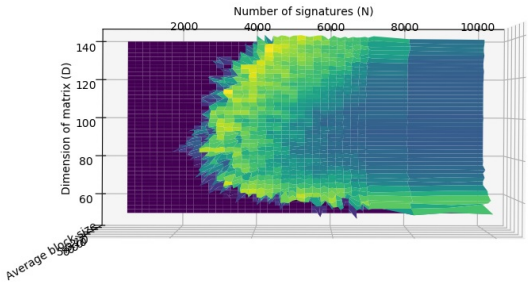


Success rate (over all dimensions).

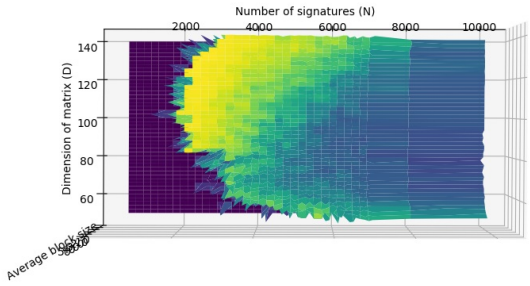


Average block size in successful runs.

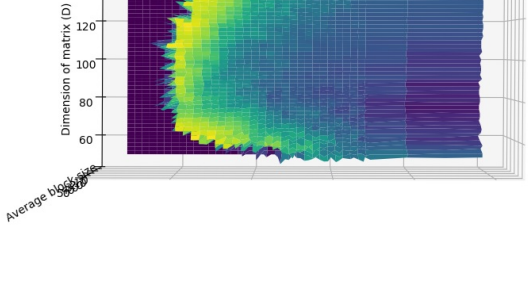
sim_const2



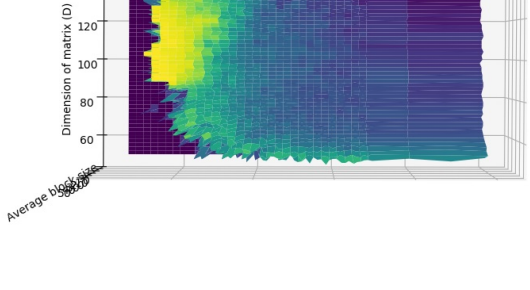
sim_const3



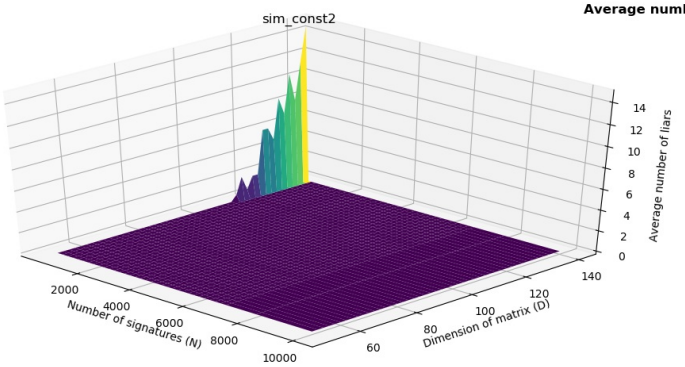
sim_const4

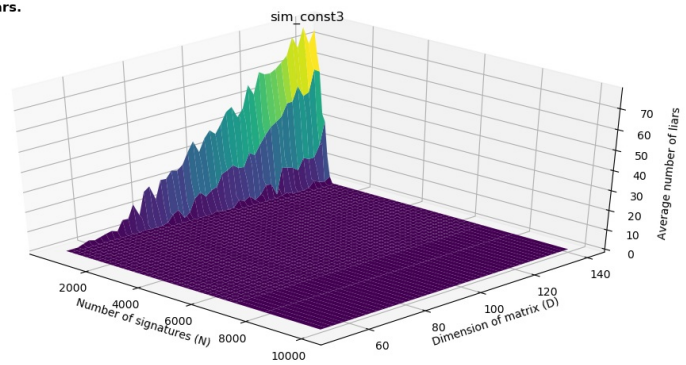


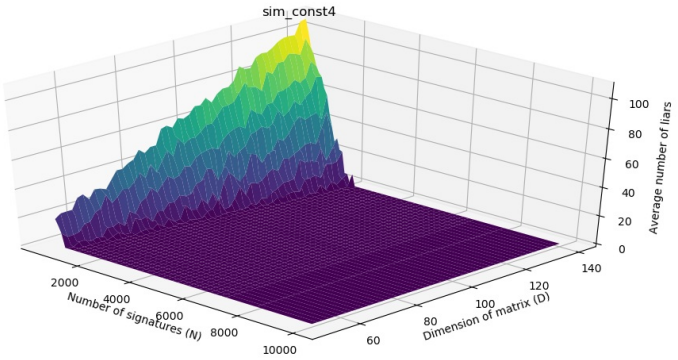
sim_geom

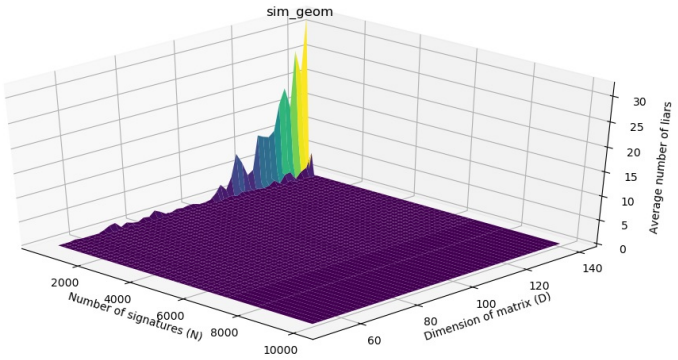


Average number of liars.

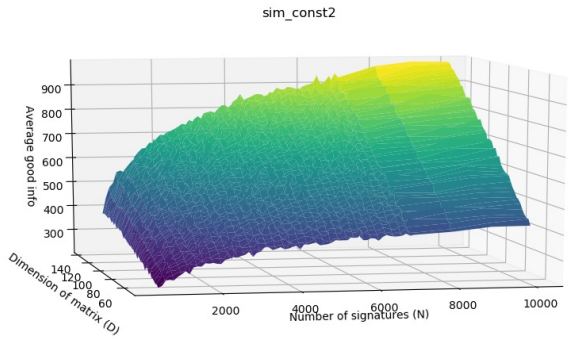


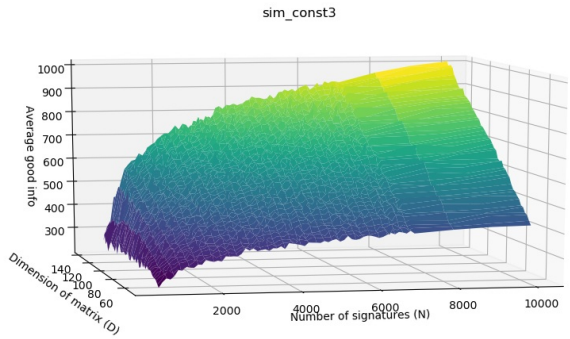


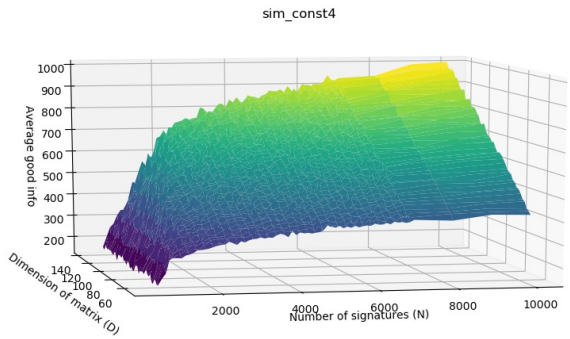


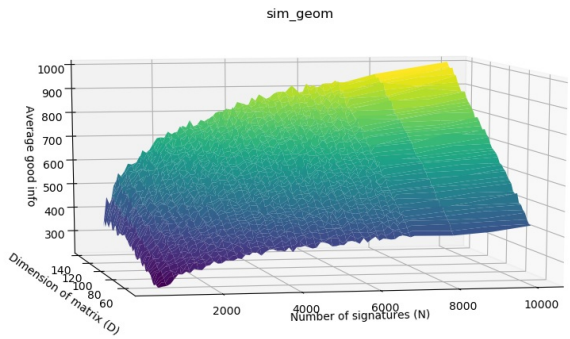


Average good info.



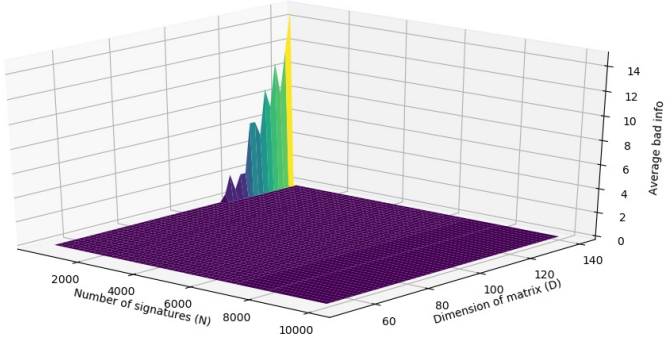




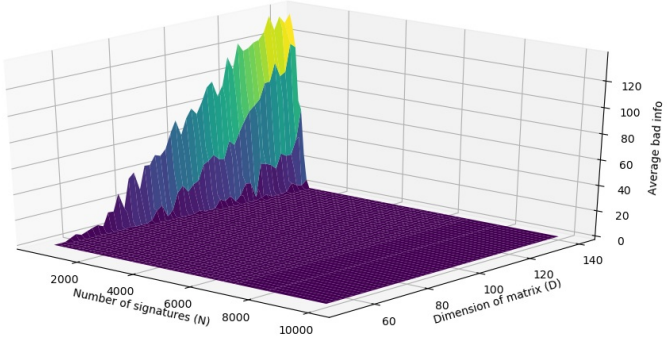


Average bad info.

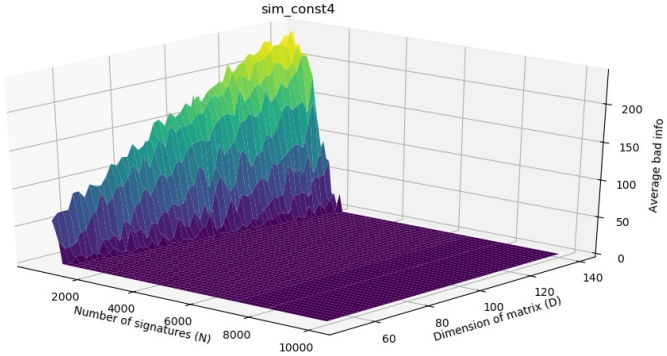
sim_const2



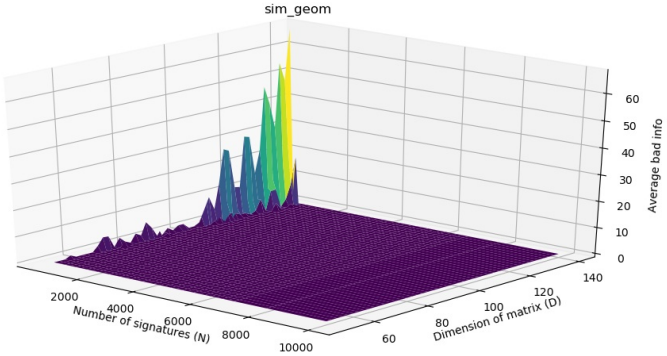
sim_const3



sim_const4

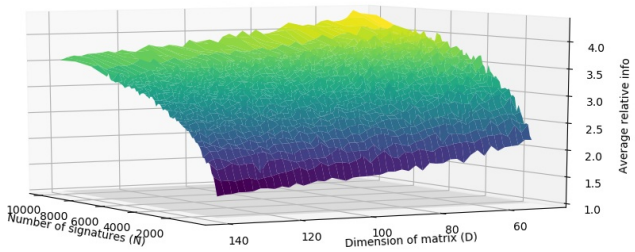


sim_geom

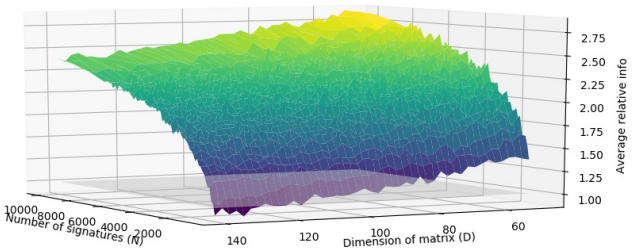


Average relative info.

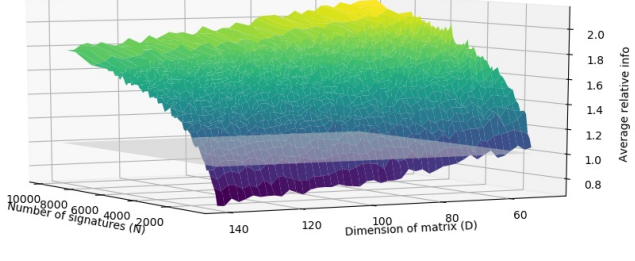
sim_const2



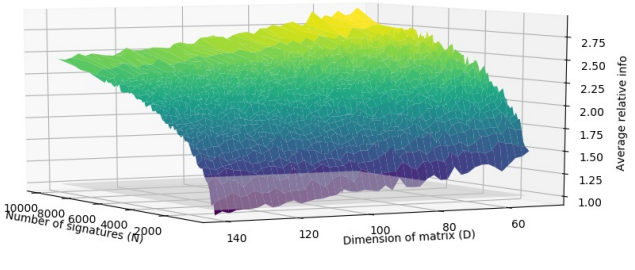
sim_const3

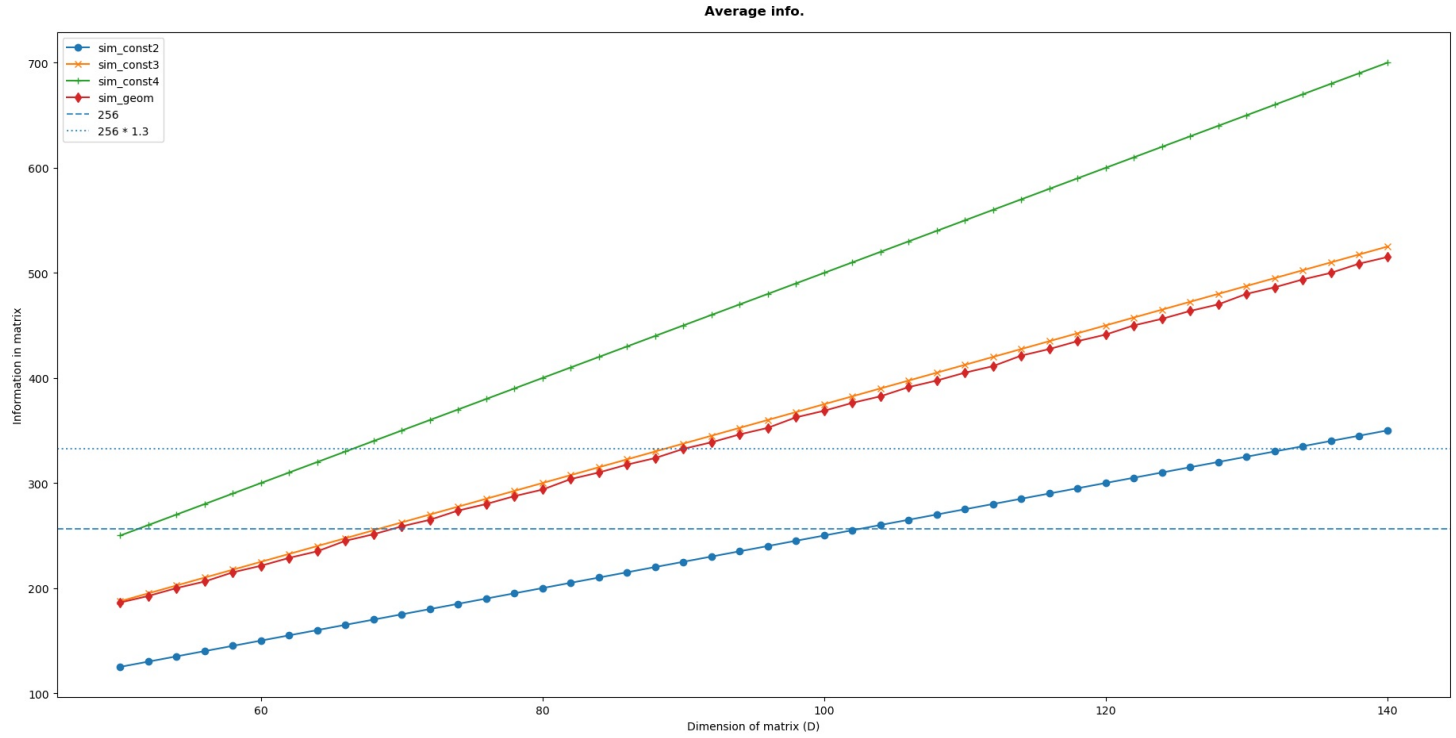


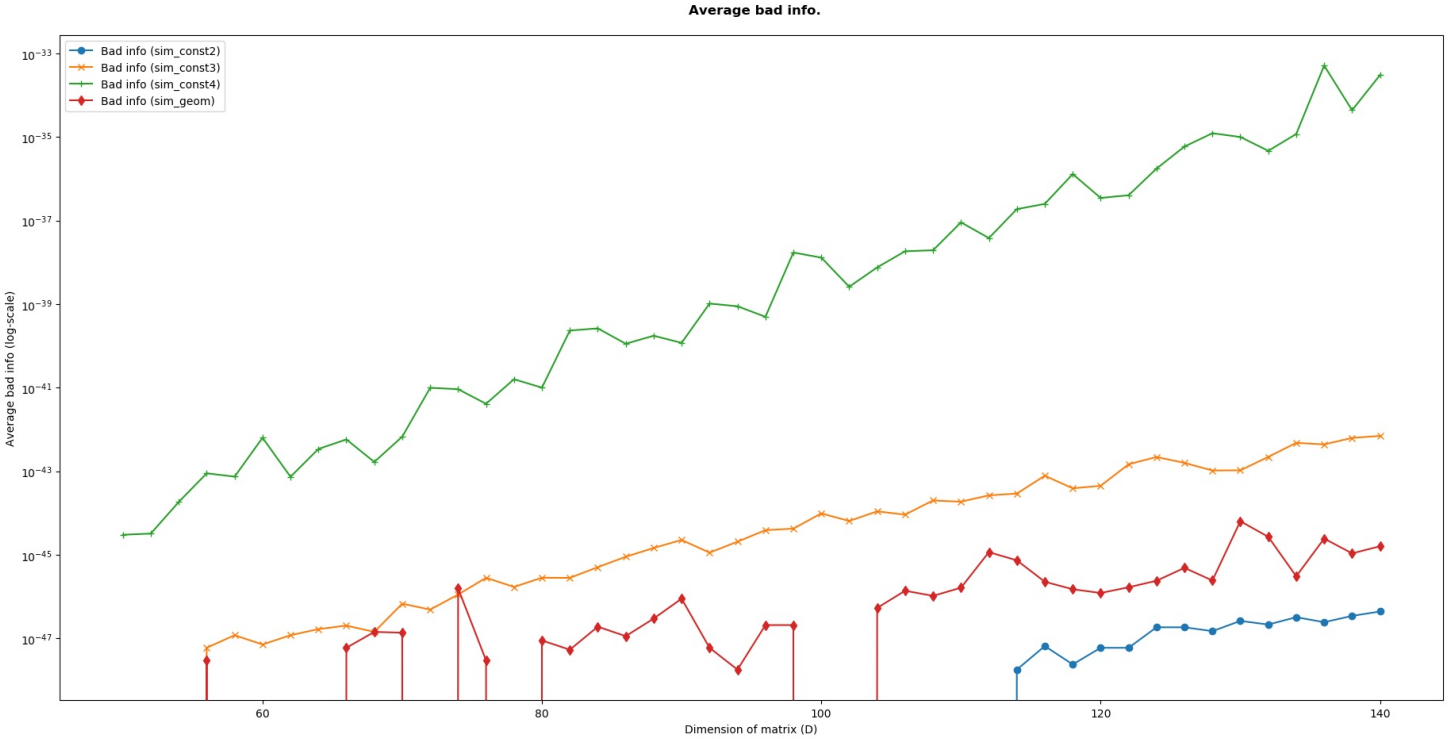
sim_const4

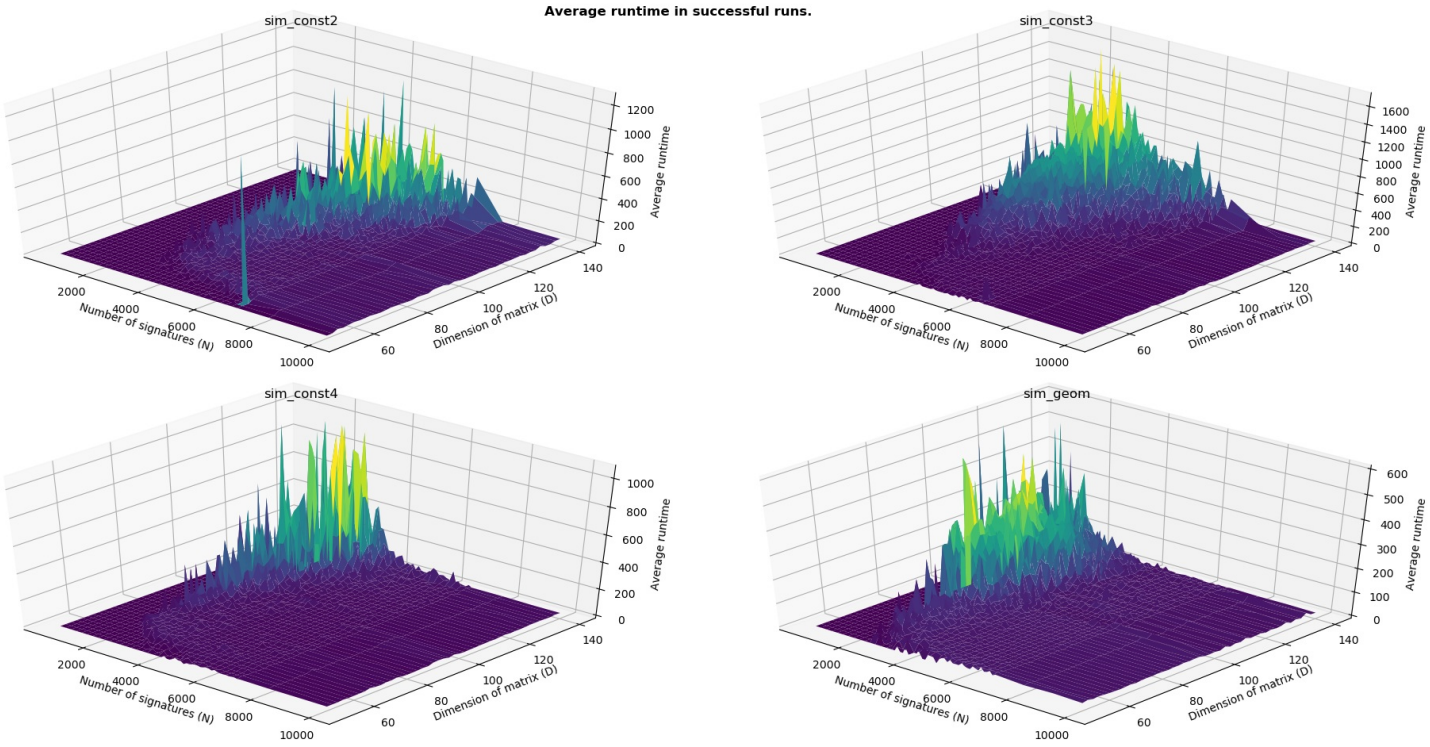


sim_geom




Average info.
sim_const2
sim_const3
sim_const4
sim_geom
256
256 * 1.3
Information in matrix
Dimension of matrix (D)


Average bad info.
Bad info (sim_const2)
Bad info (sim_const3)
Bad info (sim_const4)
Bad info (sim_geom)
Average bad info (log-scale)
Dimension of matrix (D)



Conclusions

- Geometric bounds are better than any tested constant bounds, at least on simulated data. This is because they are closer to the real distribution of the bit-length data in D-shortest signatures. They are more aggressive than constant-2 bounds, and so will have more errors, but utilize the information way better. In simulated data, the errors with geom-bounds only arrise due to the random nature of the nonce generation process (e.g. after 4n signatures **around** n signatures will have at least 2 zero msb, this is only true asymptotically).
- Runtime of the attack is very short, so more exploration into larger lattices or more reduction (higher block sizes) can be performed.
- There is some interesting behavior happening at dimension 90 with the geom bounds and at dimension around 80 with the const3 bounds, where the success rate suddenly jumps left as well as average block size increases. This might be due to some implementation details of fplll and its BKZ implementation though.