Integrating a secondary constraint into SAIL to generate wheelcases that allow for more freedom

Jan Kruska

12.02.2020

Problem

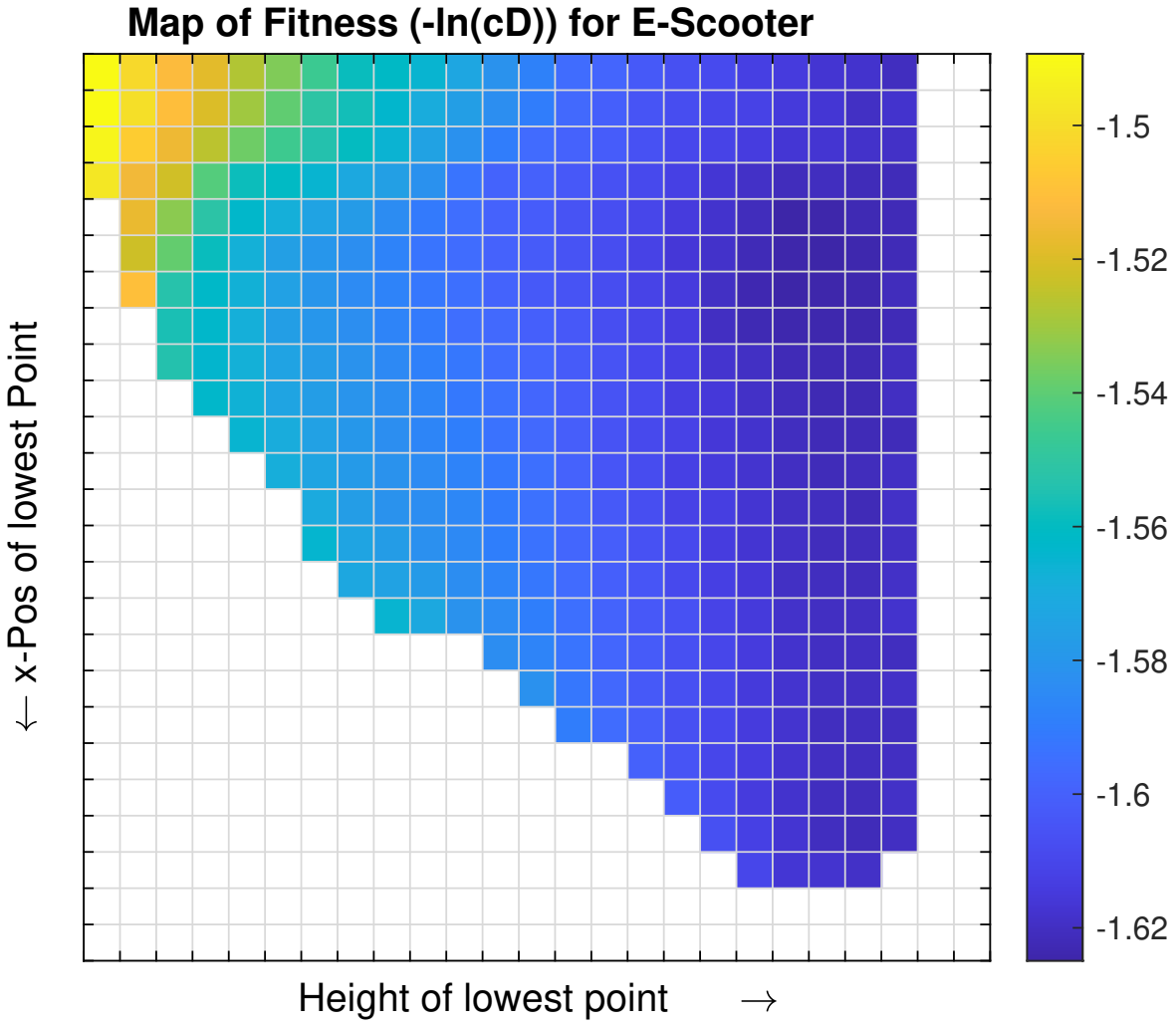
Method

Results

Problem

The existing wheelcase inhibits the maximum steering angle of the wheels

- Generate wheelcases that do not inhibit wheel movement while still maintaining some aerodynamic optimality

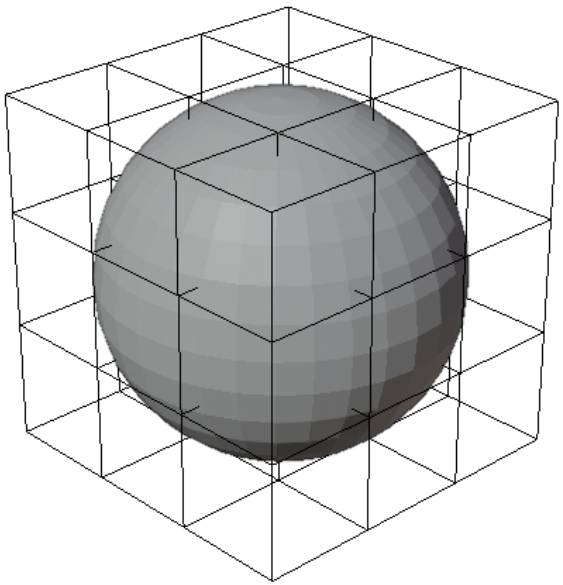

Map of Fitness (-ln(cD)) for E-Scooter
← x-Pos of lowest Point
Height of lowest point →
-1.5
-1.52
-1.54
-1.56
-1.58
-1.6
-1.62

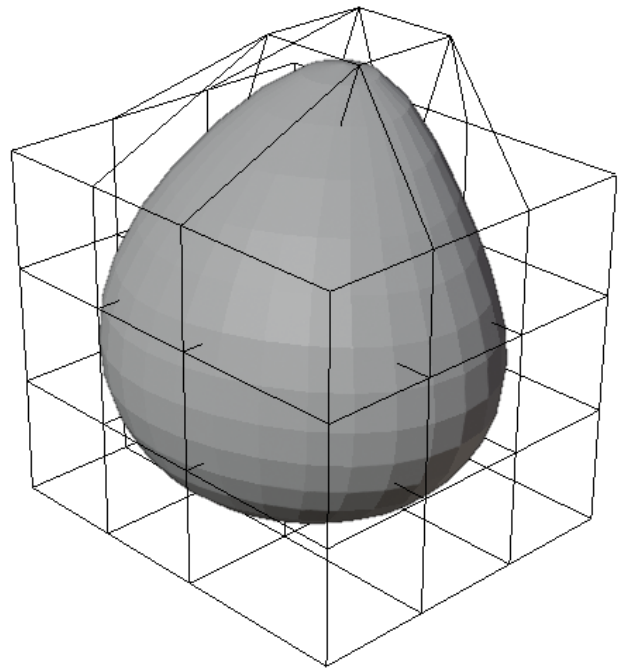
Categorization

- MAP-Elites needs Categories to assign individuals their respective cells
- Currently the chosen Categories are:
  1. Width of the velomobil
  2. x-Position of the widest Point[1]

[1] The velomobil is directed towards -X, so bigger values correspond to the widest point being further back

Free Form Deformation



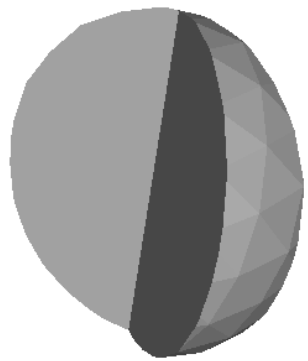


Soft Constraint vs. Hard Constraint

- Two different solutions, which both violate the constraint, can still differ in terms of the severity of the violation
- The current wheelcase does violate the constraint, so at the beginning of the process no non-trivial solution, that fulfills the constraint, is known
- Because of those two considerations a soft constraint, seemed more useful for the problem at hand

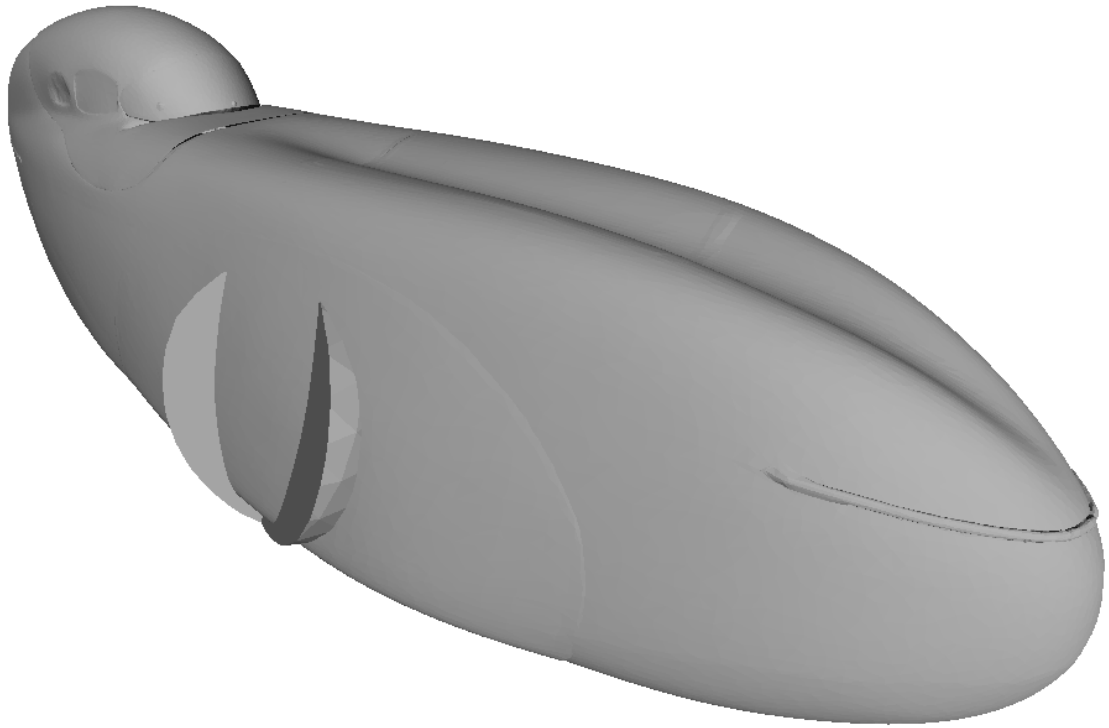
Steering Volume

The space needed by the wheel for all possible steering angles can be modeled as a volume:



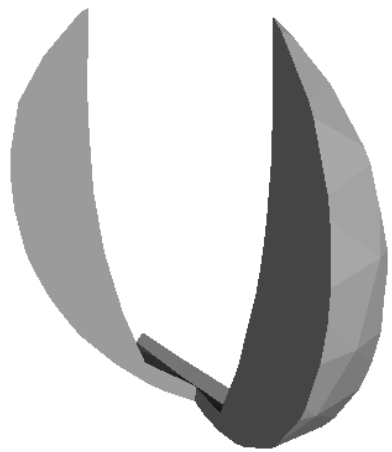
Constraint + Velomobil

This volume intersect with the current wheelcase



Constraint Volume

The Constraint can be calculated as the volume outside of the wheelcase, using mesh-difference operation

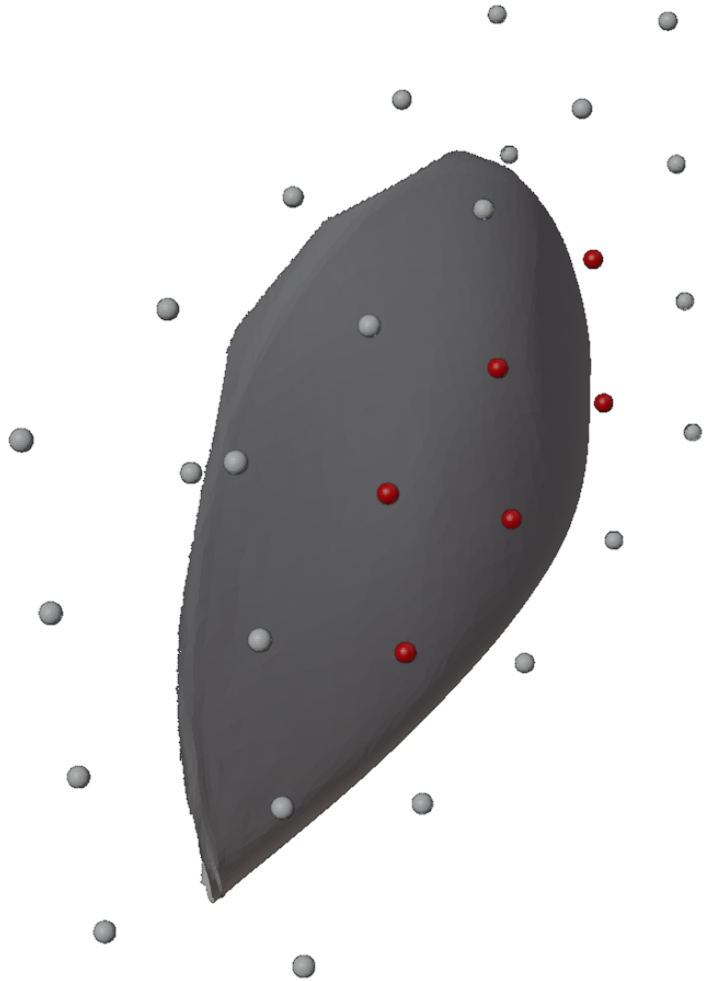


This does not perfectly map to the real constraint, but it is stipulated, that the volume outside of the wheelcase is negatively correlated with the largest possible steering angle.

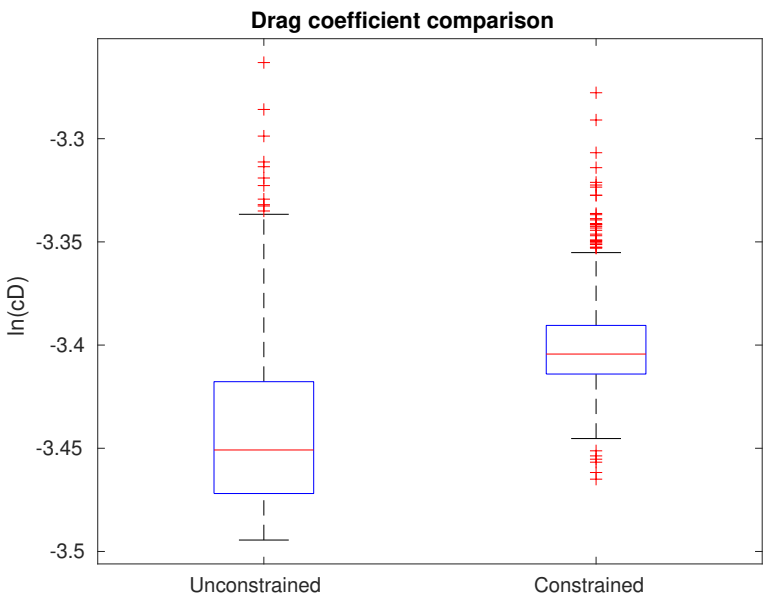
SAIL-Parameters

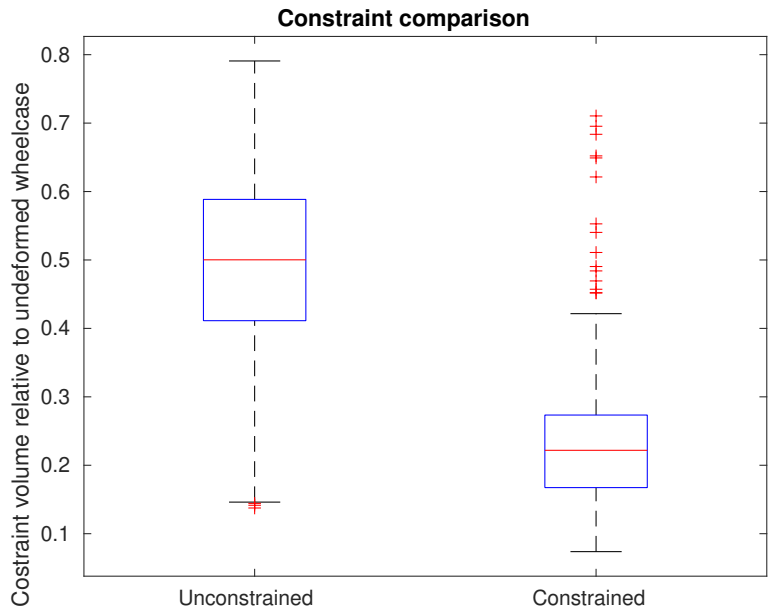
1. Real Evaluations: 1000
2. Generations Acquisition: 2048
3. Generations result: 8192
4. Children per Gen : 32
5. Map-resolution: 25x25

FFD-Configuration



Comparison non-constrained vs. constrained





Drag-maps

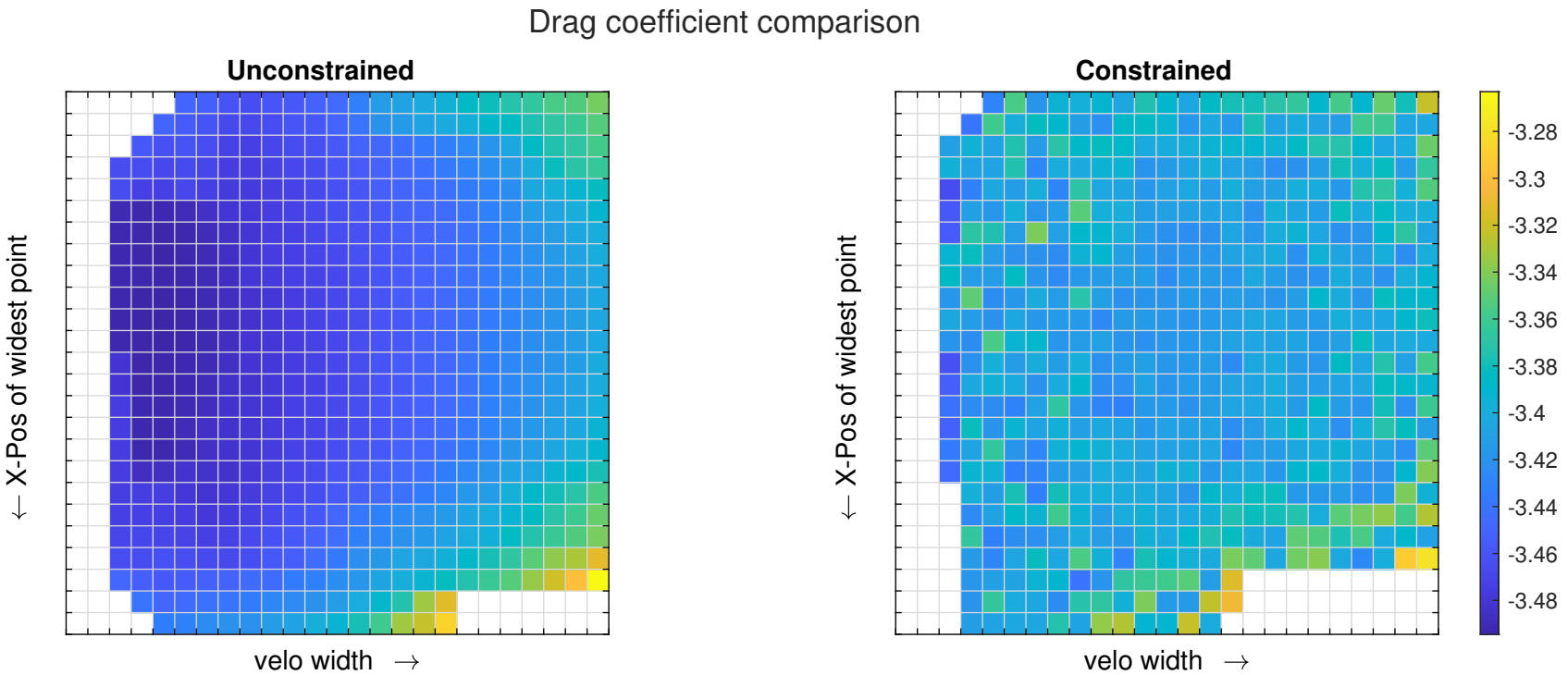


Figure: Maps of final drag values

Constraint-maps

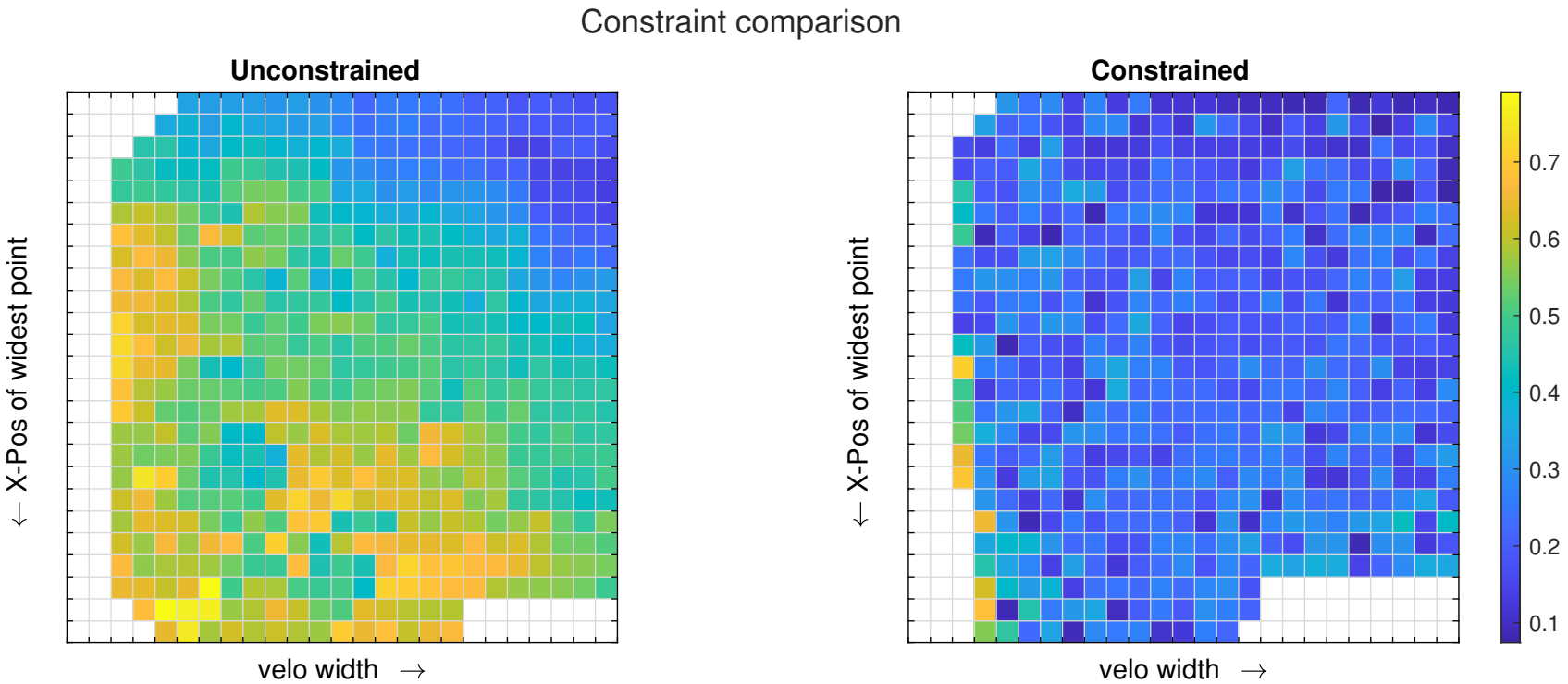
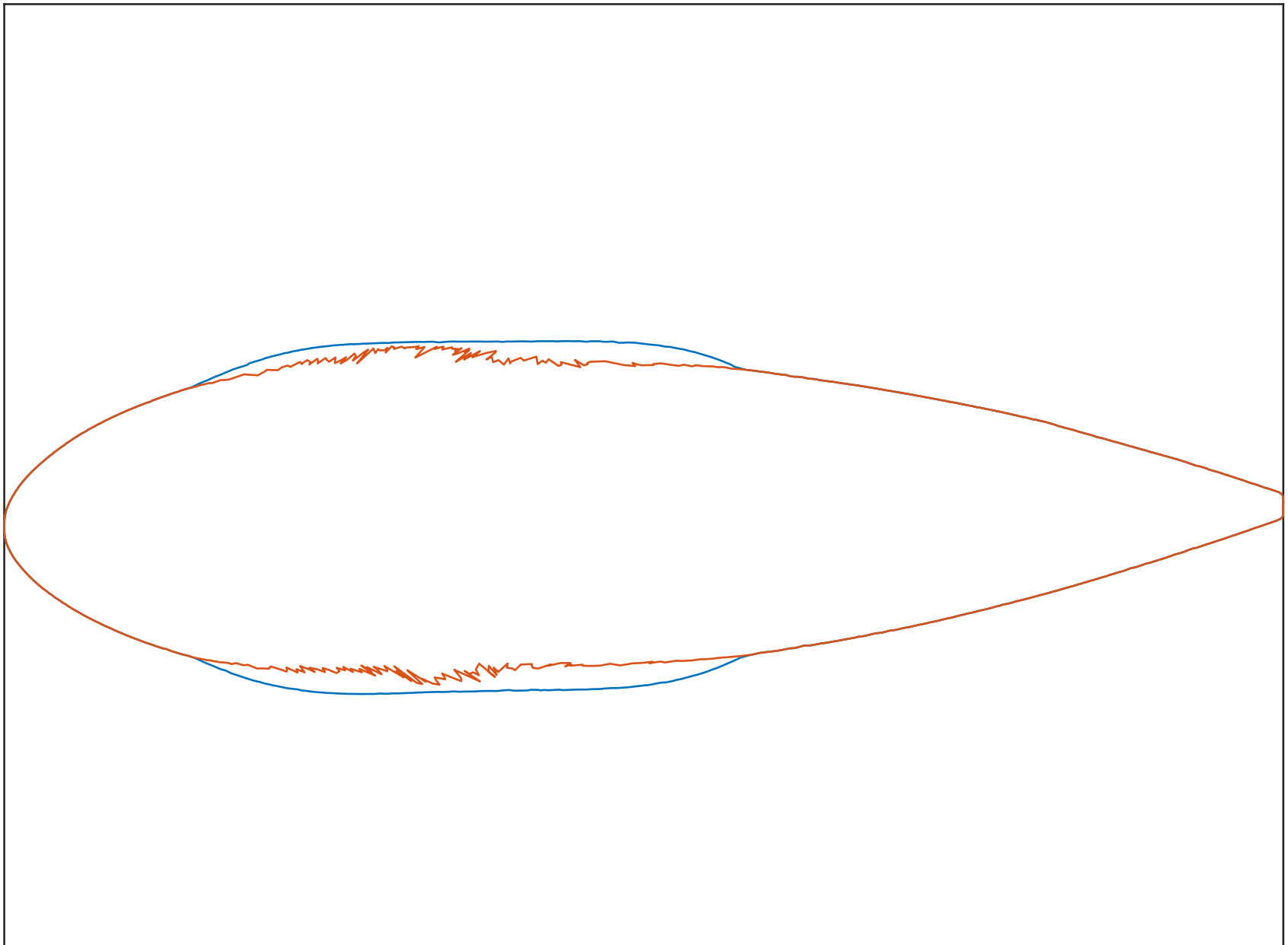
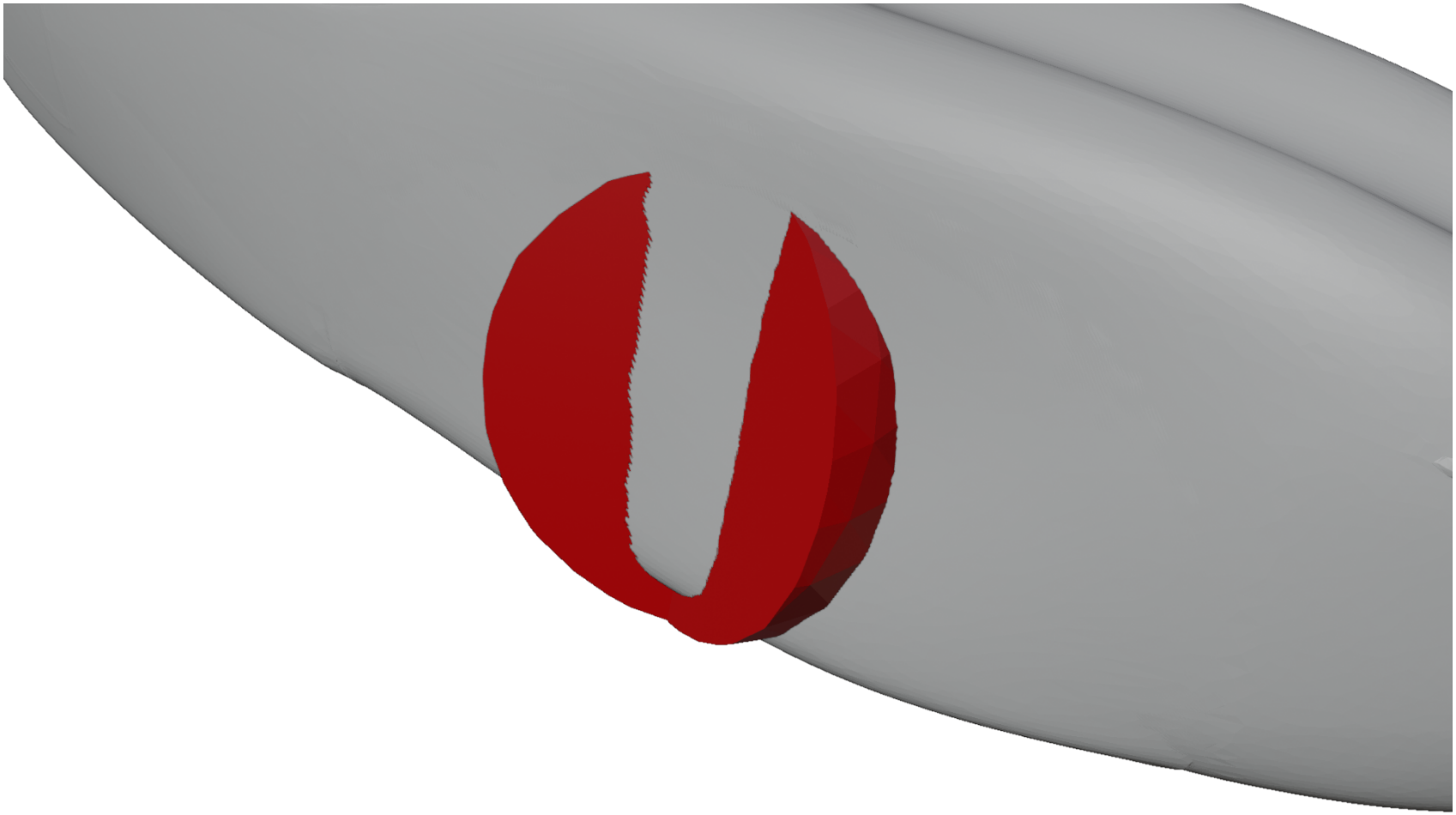


Figure: Maps of final constraint values

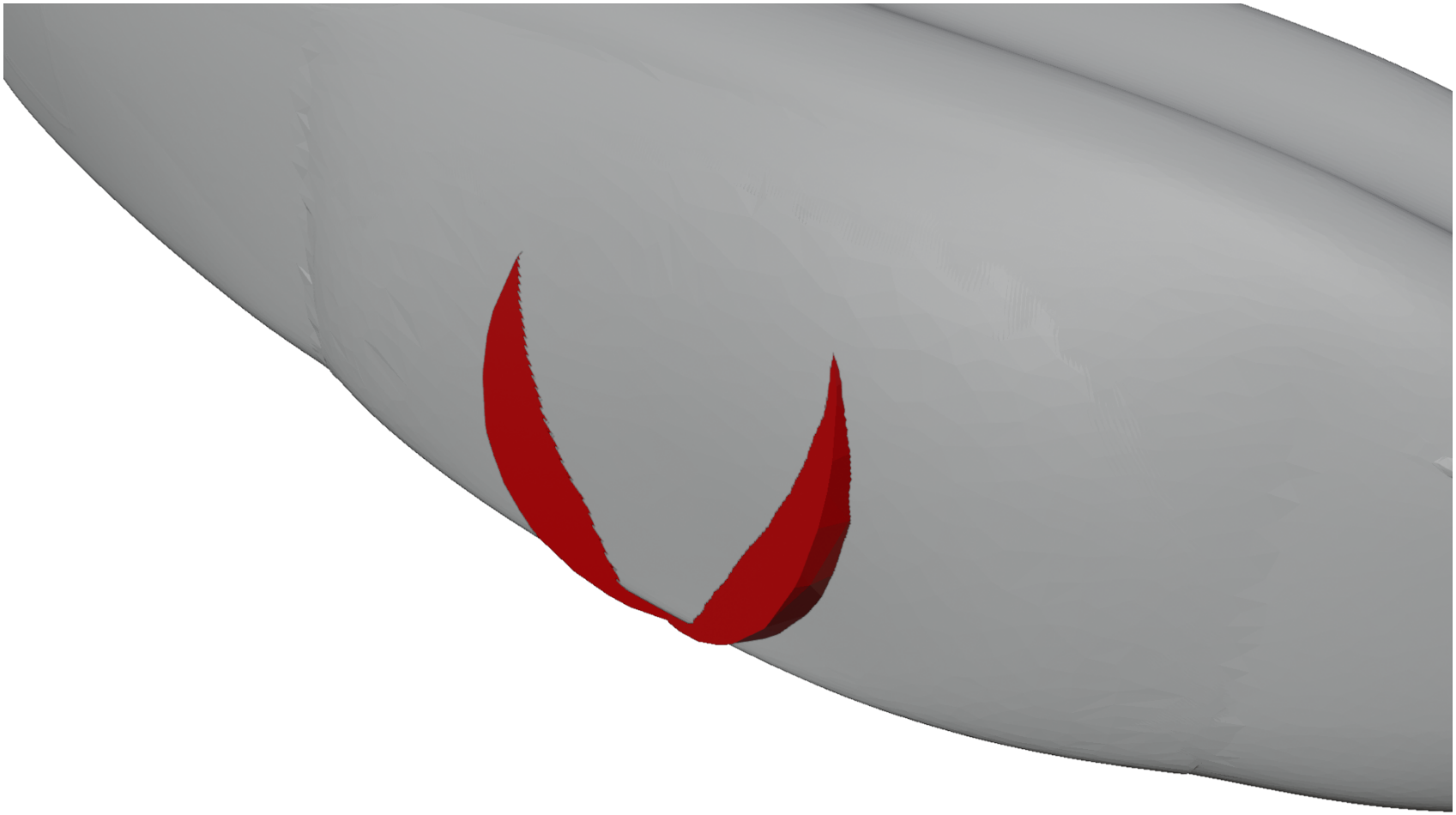
Top down comparison



Unconstrained



Constrained



Discussion

- The resulting cD Values are worse when using a constraint, yet that is to be expected.
- The algorithm works toward fulfilling the constraint, especially in areas in which that is harder
- However solutions that completely fulfill the constraint are still located in aerodynamically bad areas
- Improvements could be made in choosing a different FFD-Configuration, x-Deformations seem largely unimportant whereas y-Deformations of the top row seem very important