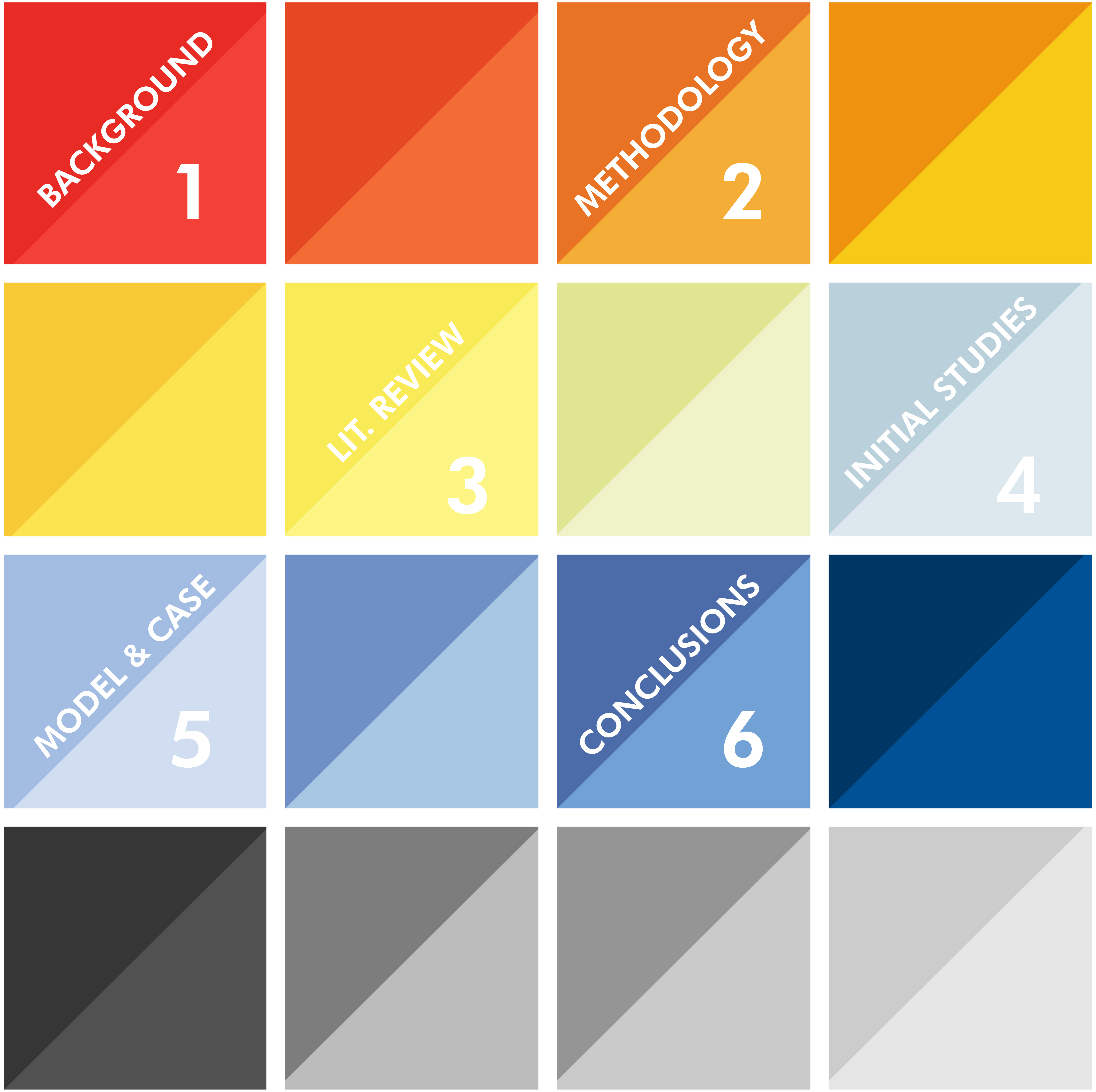
COMPUTATIONAL METHOD FOR EARLY-STAGE DESIGN OPTIMIZATION OF NATURALLY VENTILATED TERMINALS


BACKGROUND
1
METHODOLOGY
2
LIT. REVIEW
3
INITIAL STUDIES
4
MODEL & CASE
5
CONCLUSIONS
6

BACKGROUND

36%
of all CO_2 emitted is from the built environment

40%
of all energy consumed is by the built environment

12%
of all CO_2 emitted is from manufacturing and construction

2050
by 2050 all buildings in Europe need to be carbon neutral

20-66%
of all energy consumed in buildings is from HVAC

8%
of company OPEX are related to the building

12%
of embodied CO_2 in buidlings is from HVAC

35%
of initial building costs is from building services

BACKGROUND

20-86%

of all energy consumed at airports is from HVAC

15%

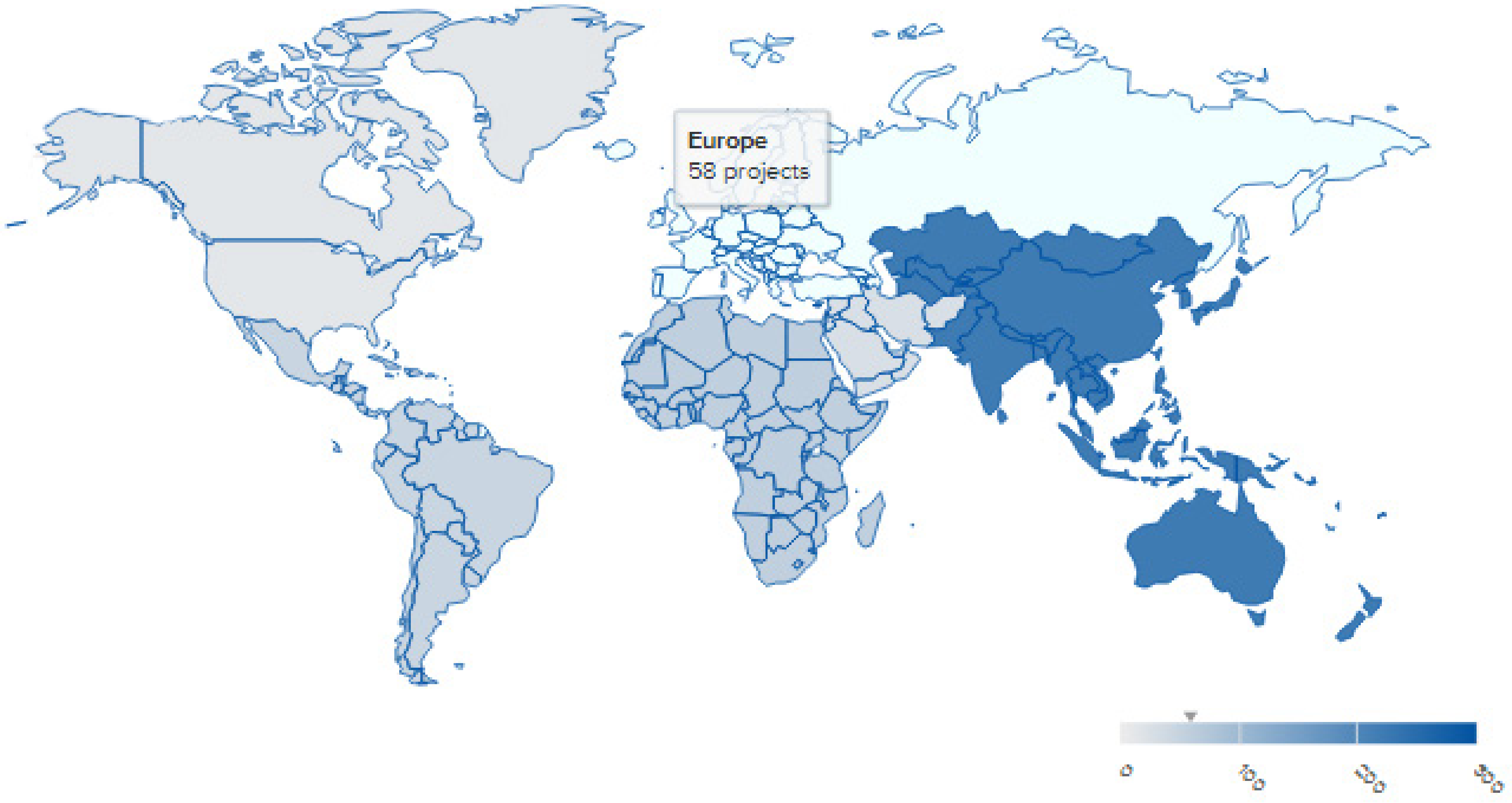
of airport OPEX from energy consumption

2X

passenger traffic to double by 2040

$1tn

to be invested into airport construction



BACKGROUND

Airport	Climate	Terminal Concept	LCT	Passengers '09 [mln]	Total Energy [GWh]	Terminal Area [m2]	Total CO2 [ton]	kWh/pax	kWh/m2	kgCO2/pax	kgCO2/m2
London Heathrow	Cfb		No	66.0	976.0	597,000.0	332,000.0	14.8	1,634.8	5.0	556.1
London Gatwick	Cfb		Yes	32.4	213.8	258,000.0	99,700.0	6.6	828.7	3.1	386.4
London Stansted	Cfb		Yes	20.0	106.1	91,000.0	52,474.0	5.3	1,165.9	2.6	576.6
Paris Charles de Gaulle	Cfb		No	57.9	1,015.0	?	?	17.5	?	?	?
Frankfurt Airport	Cfb		No	50.9	766.1	?	226,100.0	15.0	?	4.4	?
Amsterdam Schiphol	Cfb		No	43.6	331.8	650,000.0	132,000.0	7.6	510.5	3.0	203.1
Eindhoven Airport	Cfb		Yes	1.7	5.8	14,800.0	?	3.4	391.9	?	?
Istanbul Atatürk Airport	Csa		No	29.8	238.1	377,000.0	107,150.0	8.0	631.6	3.6	284.2
Izmir Adnan Menderes	Csa		No	6.2	18.9	295,000.0	6,830.0	3.0	64.1	1.1	23.2
Ankara Esenboga	Csa		No	6.1	95.7	182,000.0	15,460.0	15.7	525.8	2.5	84.9
Zurich Airport	Cfb		No	21.9	305.5	370,000.0	34,326.0	14.0	825.7	1.6	92.8
Hong Kong Airport	Cwa		No	69.7	?	746,000.0	?	4.0	374.0	?	?
Stavanger Airport	Cfb		No	4.5	?	46,000.0	?	3.6	349.0	?	?
Bergen Airport T3	Cfb		No	7.0	?	52,500.0	?	1.1	144.0	?	?
Galapagos Eco Airport	BSh		No	0.5	?	6,027.0	?	1.0	47.0	?	?

BACKGROUND

"…the air-conditioning industry whose simple aim, not unreasonably, is to capture market share and generate profit, or a gestural architecture with no environmental presence or meaning…"

"…the marked reluctance historically in the design community to acquire such expertise for fear of destroying free artistic expression" is "anti-scientific" and "…may be the principle barrier to a sustainable future for the built world."

C.A. Short (2018)

So, can we design buildings without BS?*
A low-tech approach for a high-tech sector?

**BS: Building Services*

BACKGROUND

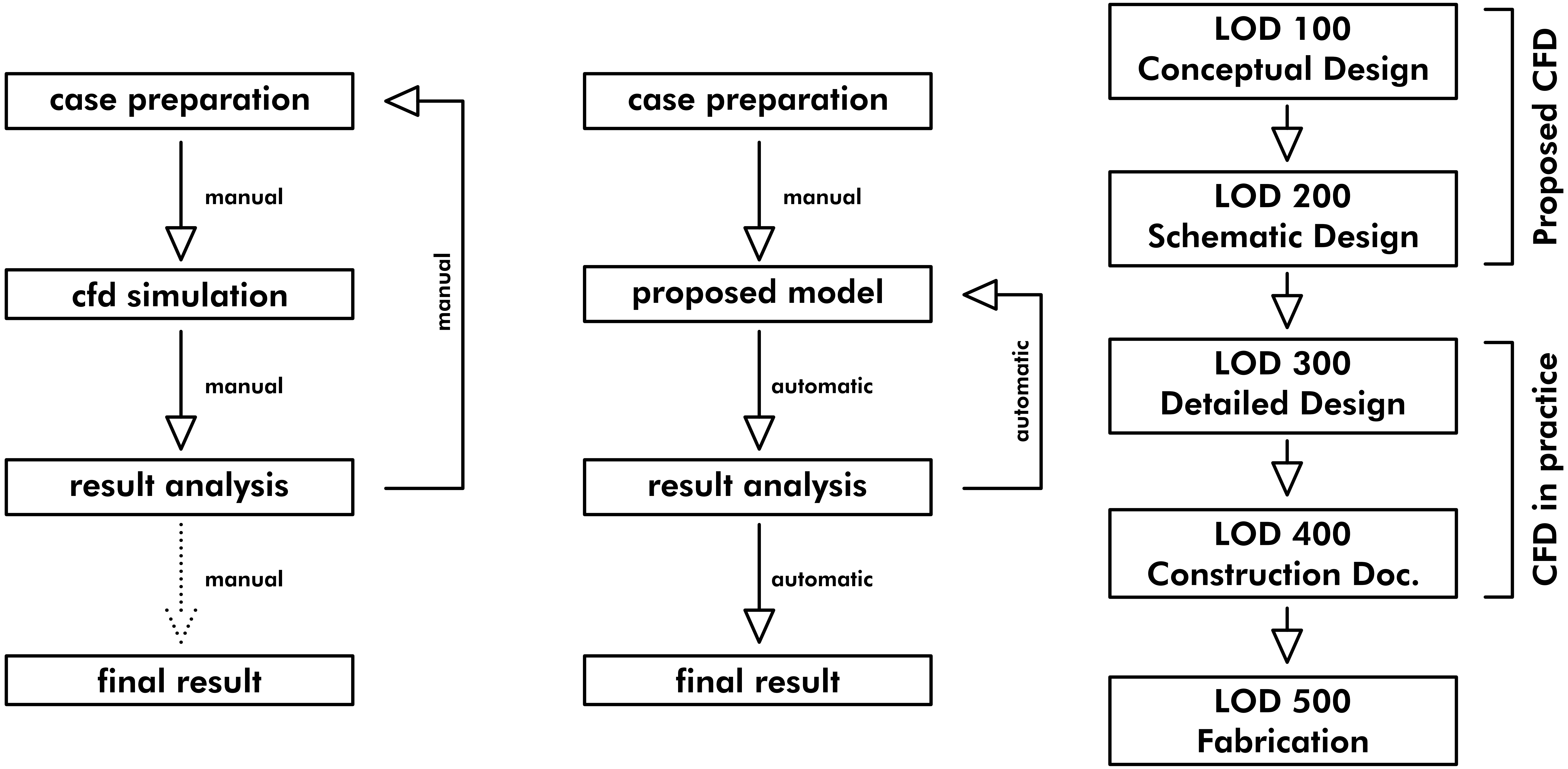
Mechanical ***ventilation is easy to design: it is*** ***predictable*** ***it is*** ***generic*** ***with little change from project to project.***

Natural ***ventilation is difficult to design: it is*** ***'unpredictable'*** ***and*** ***specific*** ***to each project.***

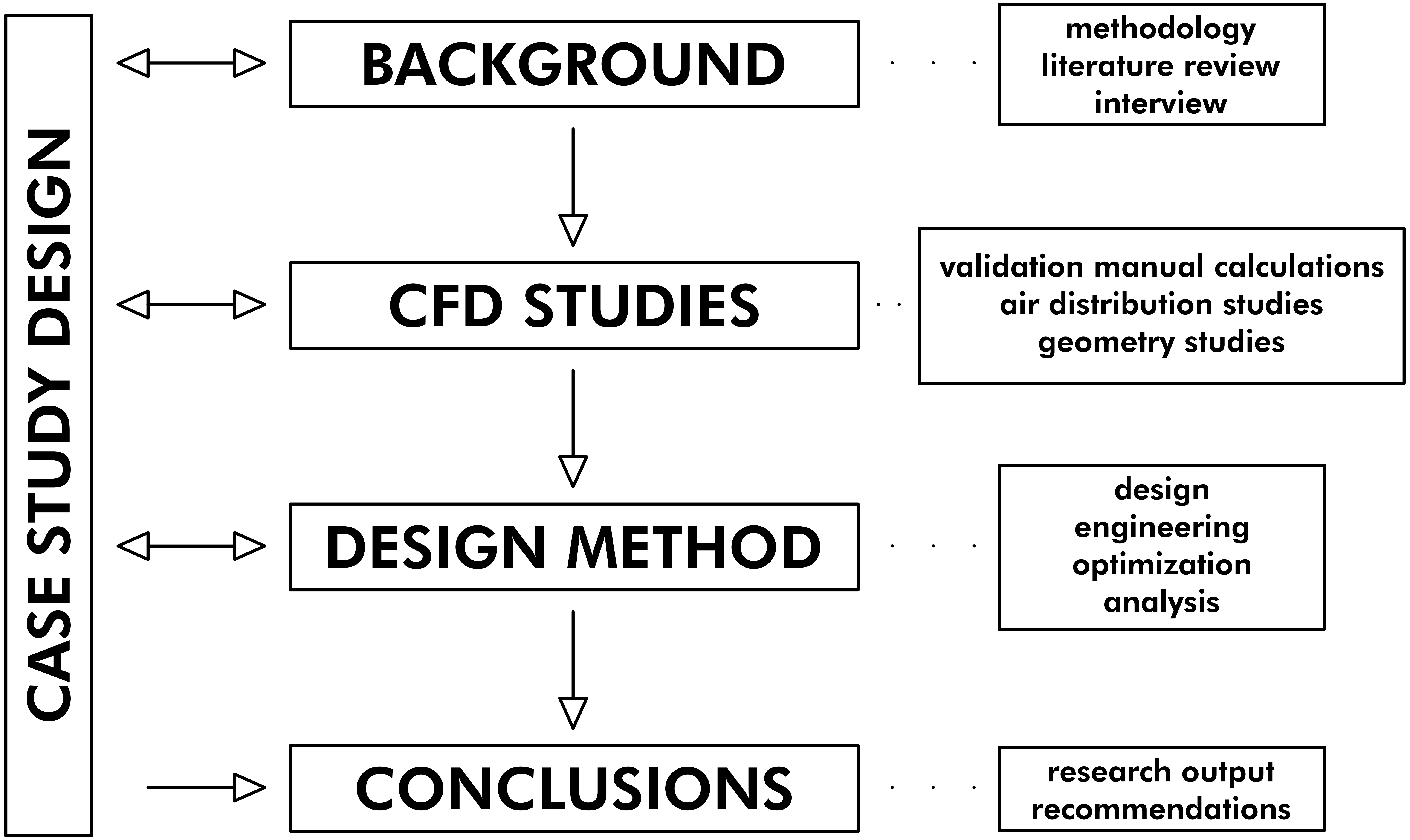
Natural ventilation needs to be ***integrated*** ***into the design from the start.***

METHODOLOGY

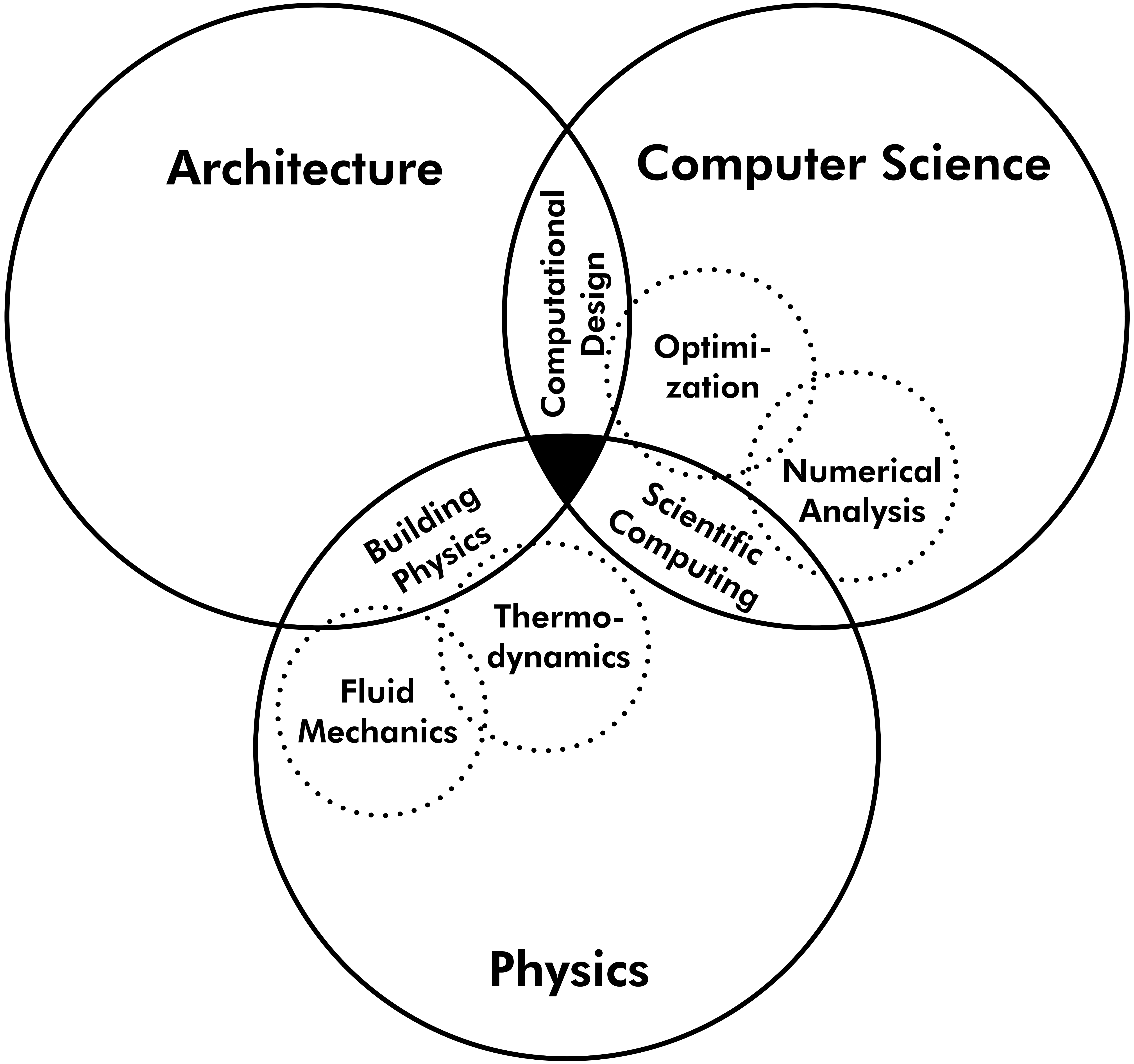
Research objective: development of a computational method for early-stage design optimization of naturally ventilated terminals.



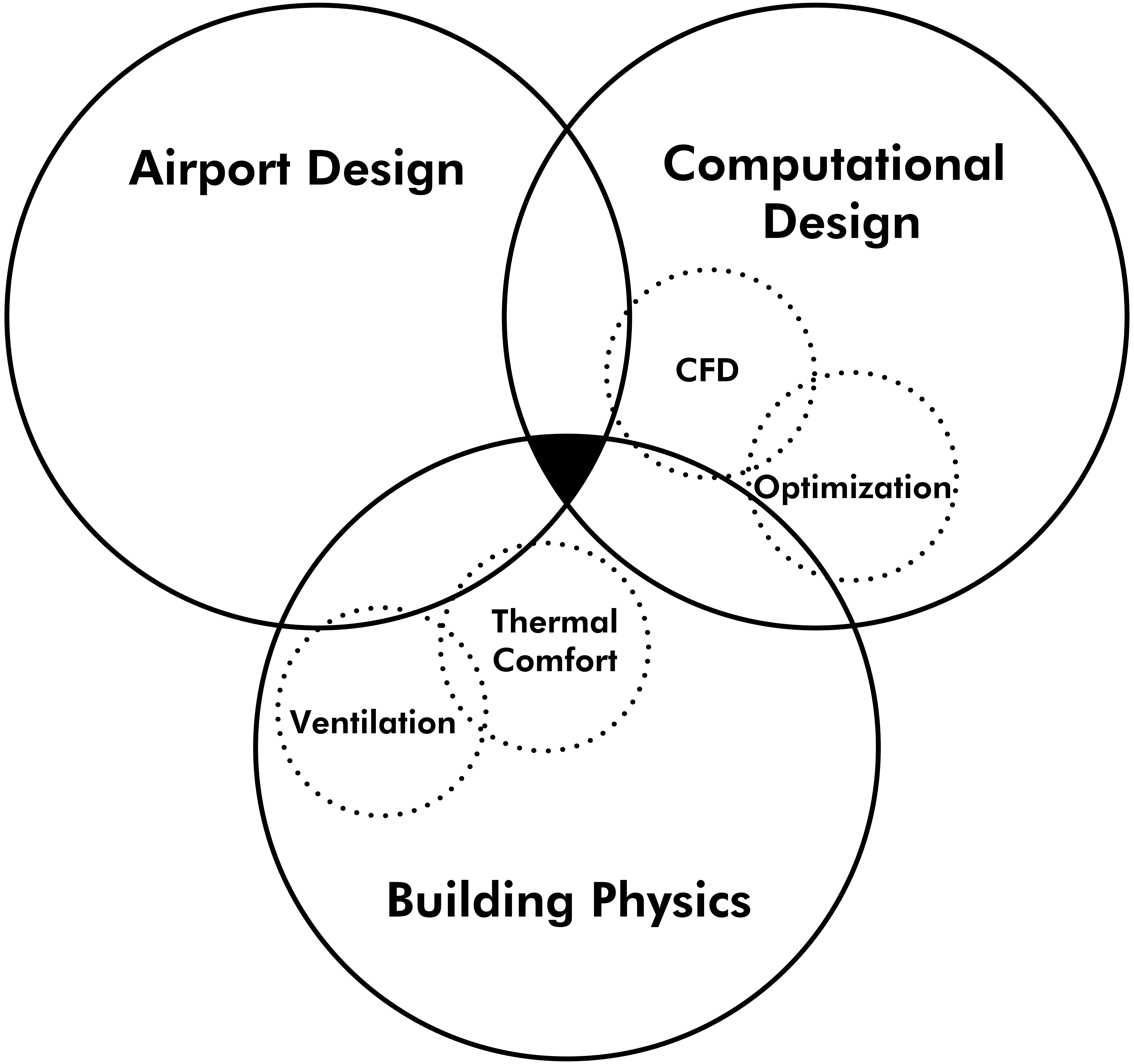
METHODOLOGY



LITERATURE REVIEW



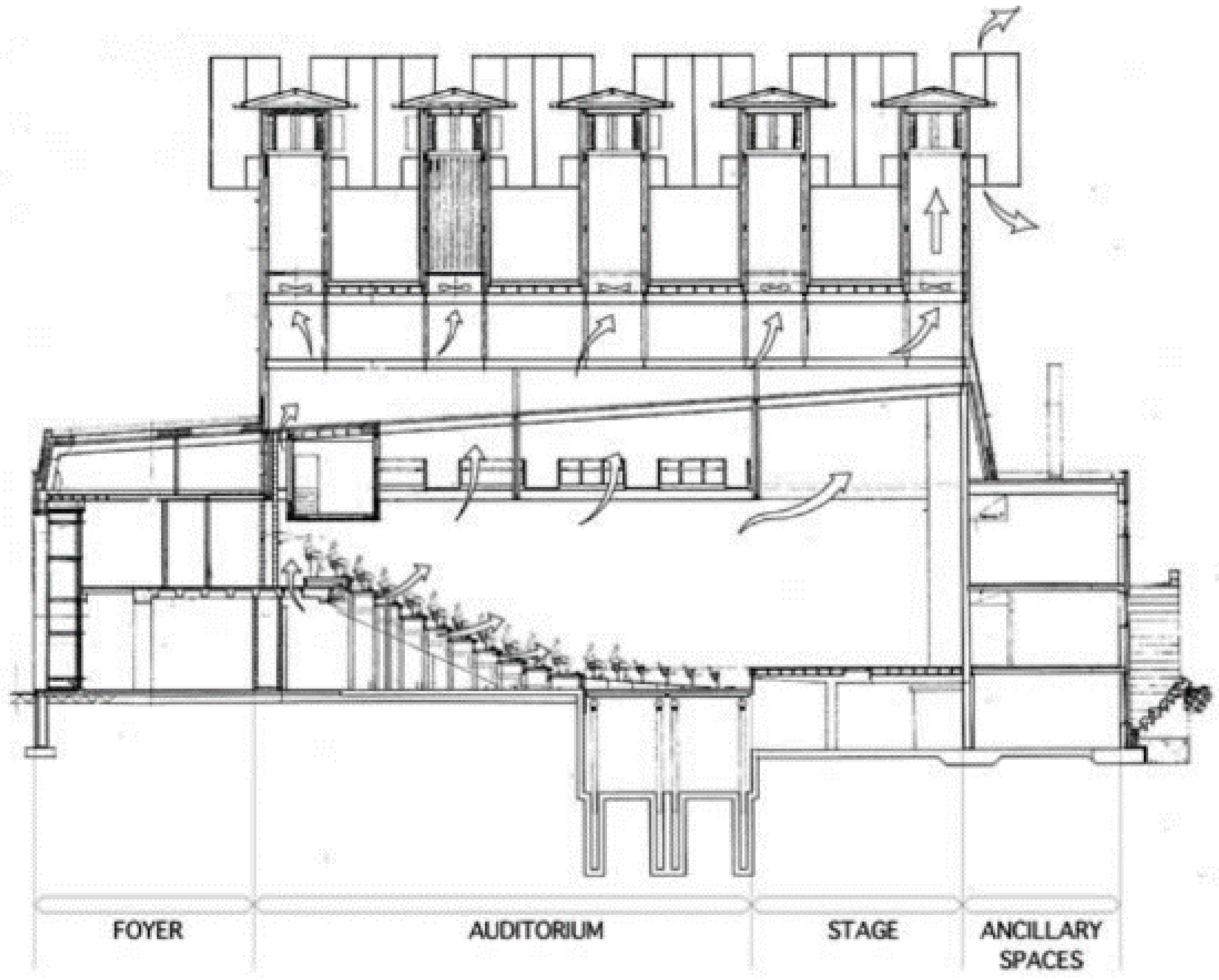
LITERATURE REVIEW



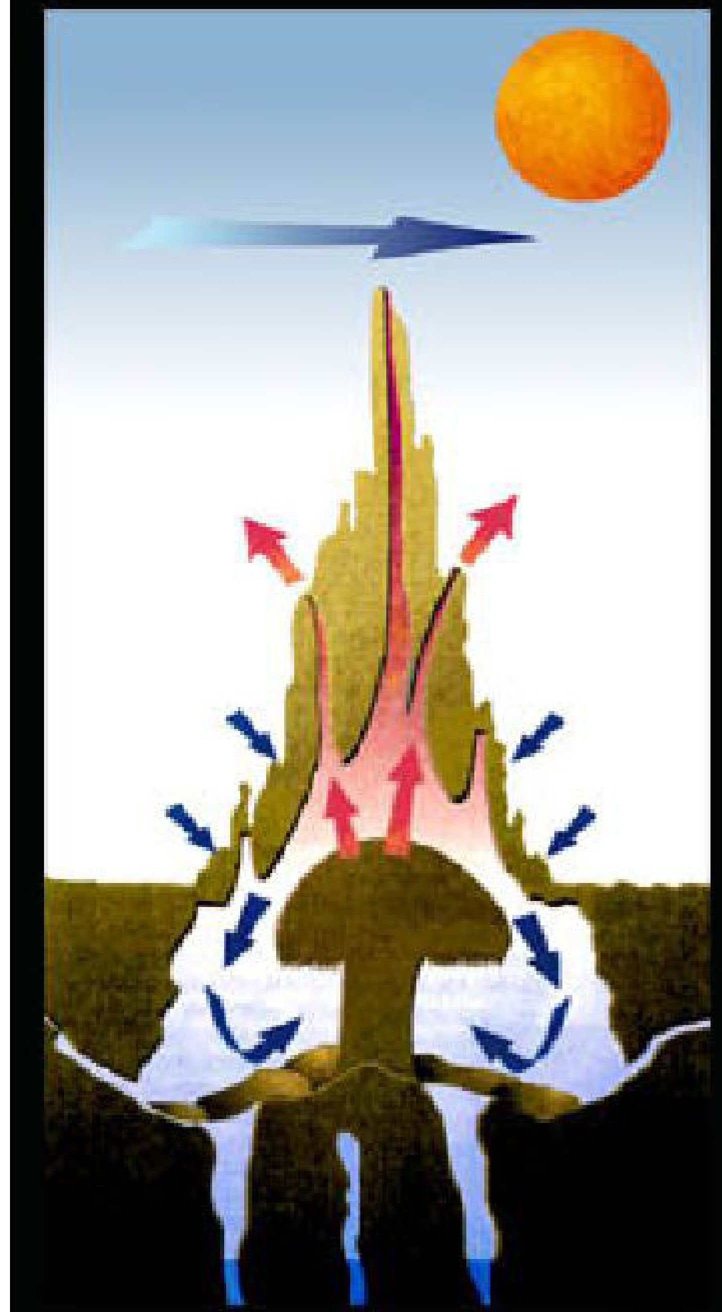
LITERATURE REVIEW

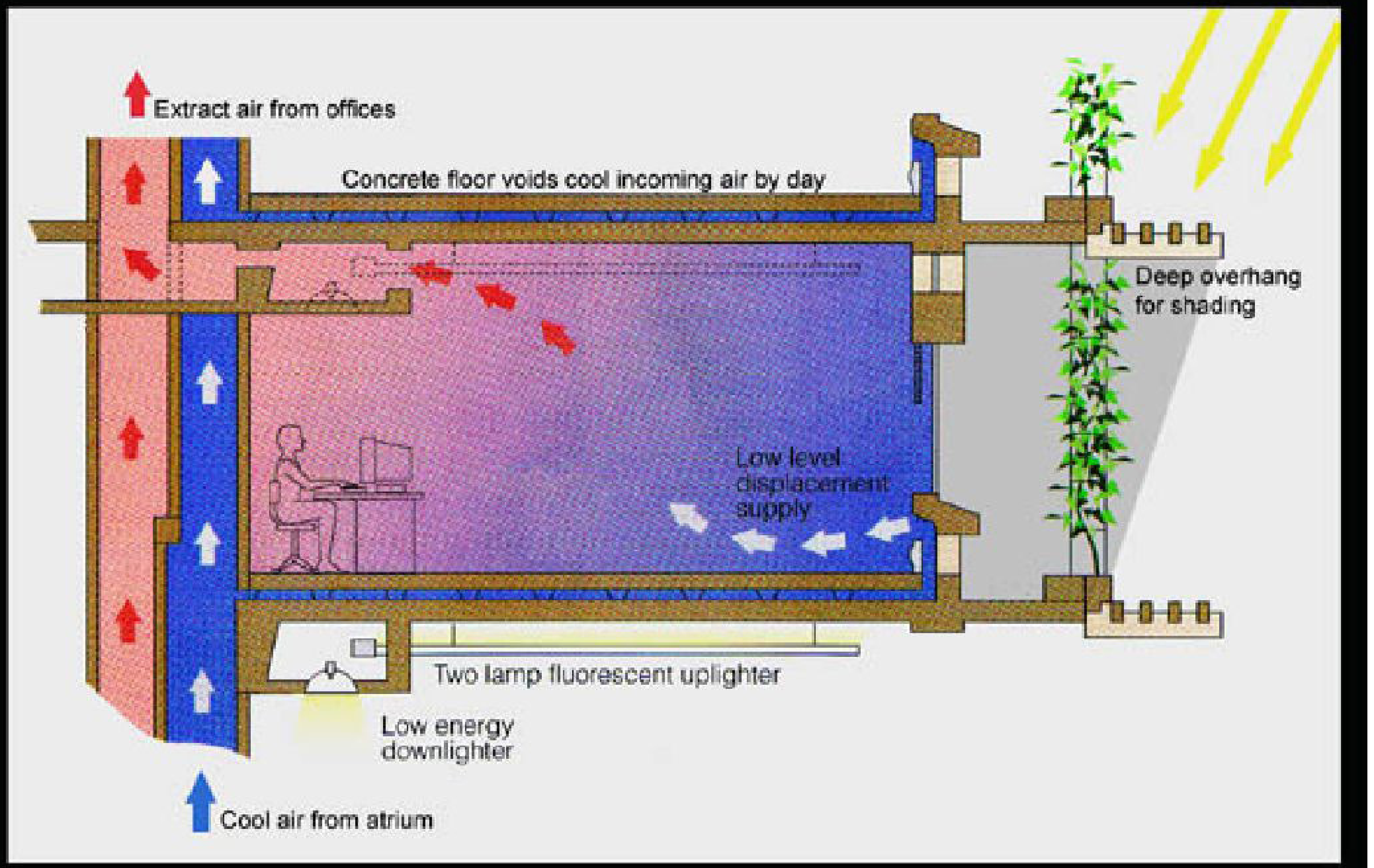


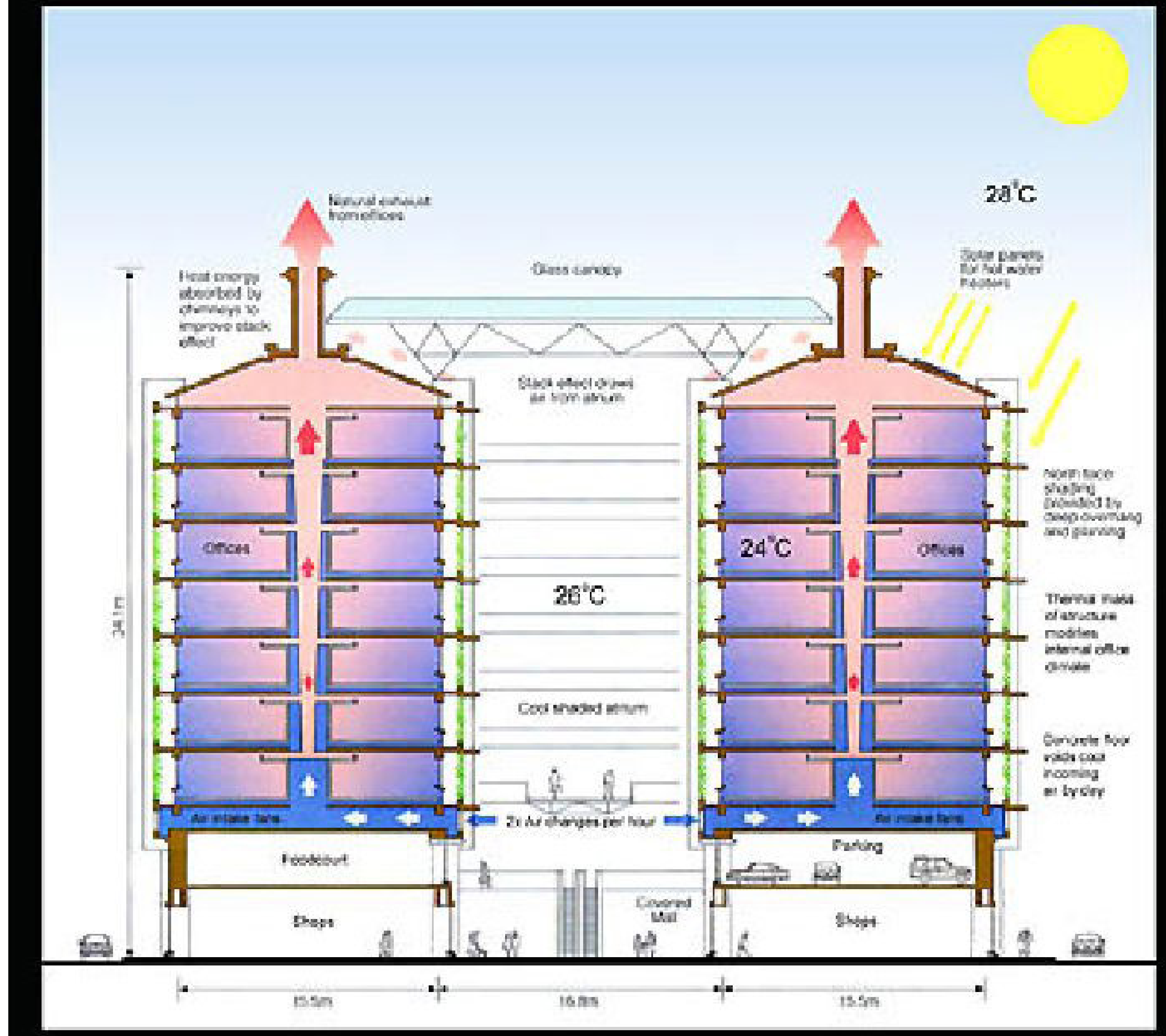

LITERATURE REVIEW

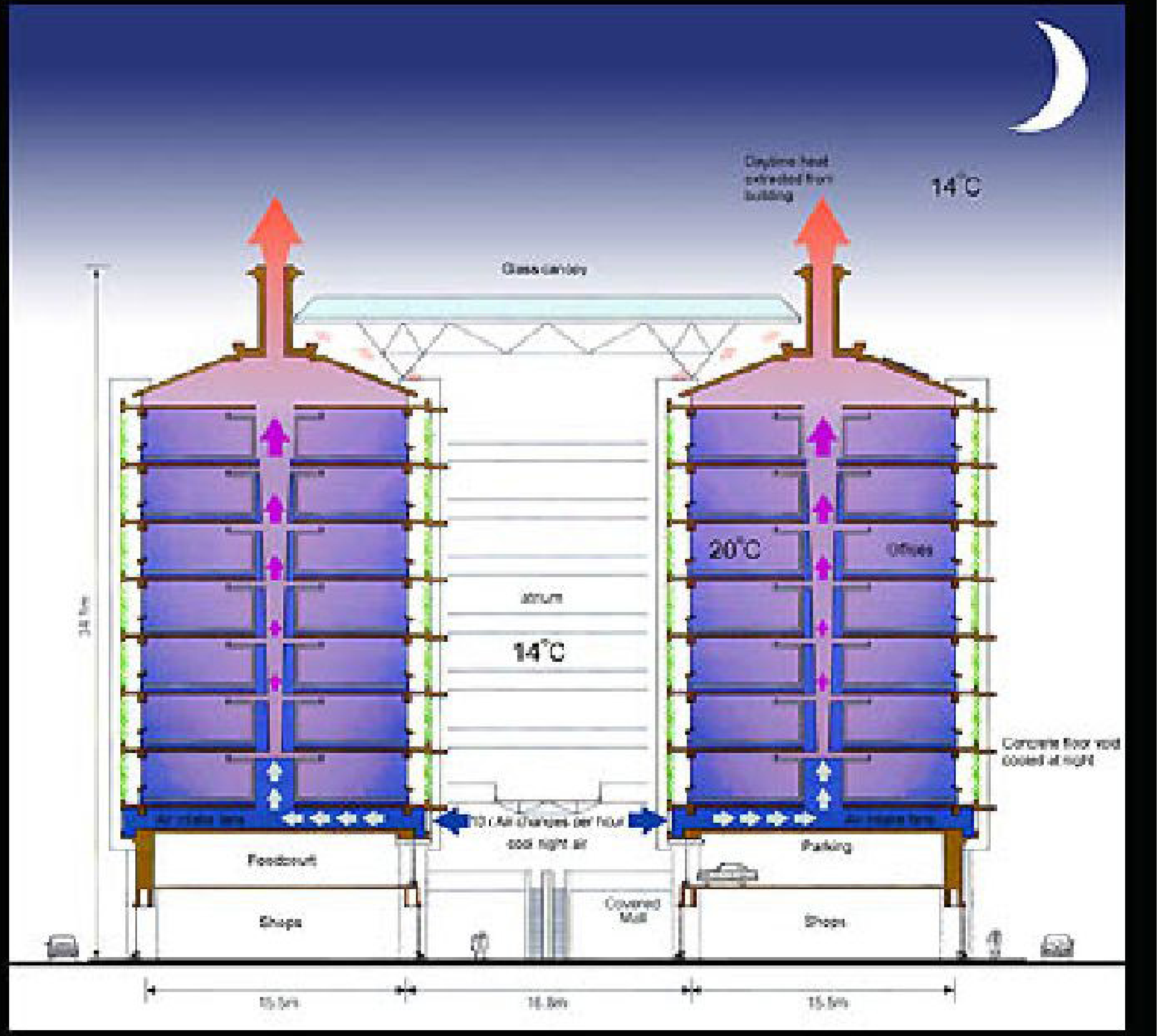


LITERATURE REVIEW









INITIAL STUDIES

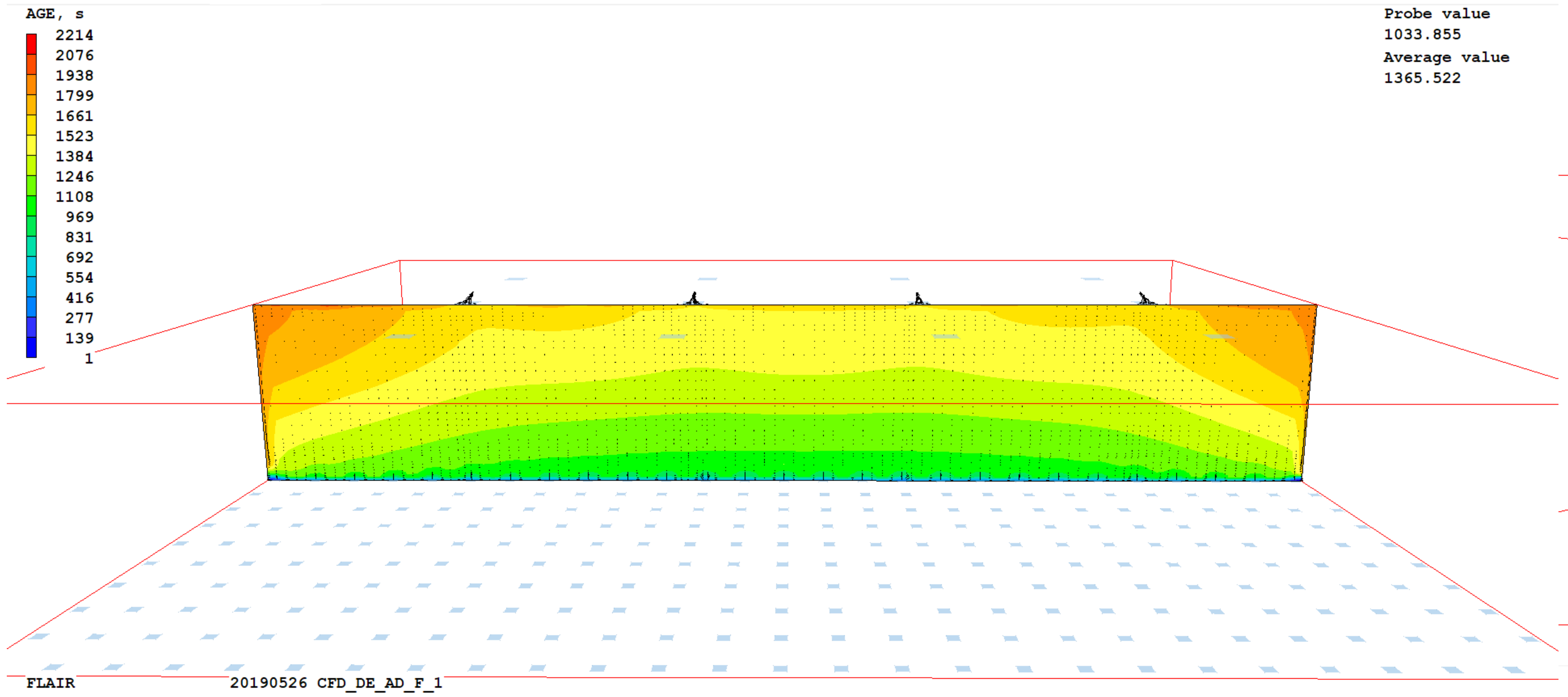
1. Compare manual calculations to CFD
2. Explore various CFD settings
3. Explore design alternatives

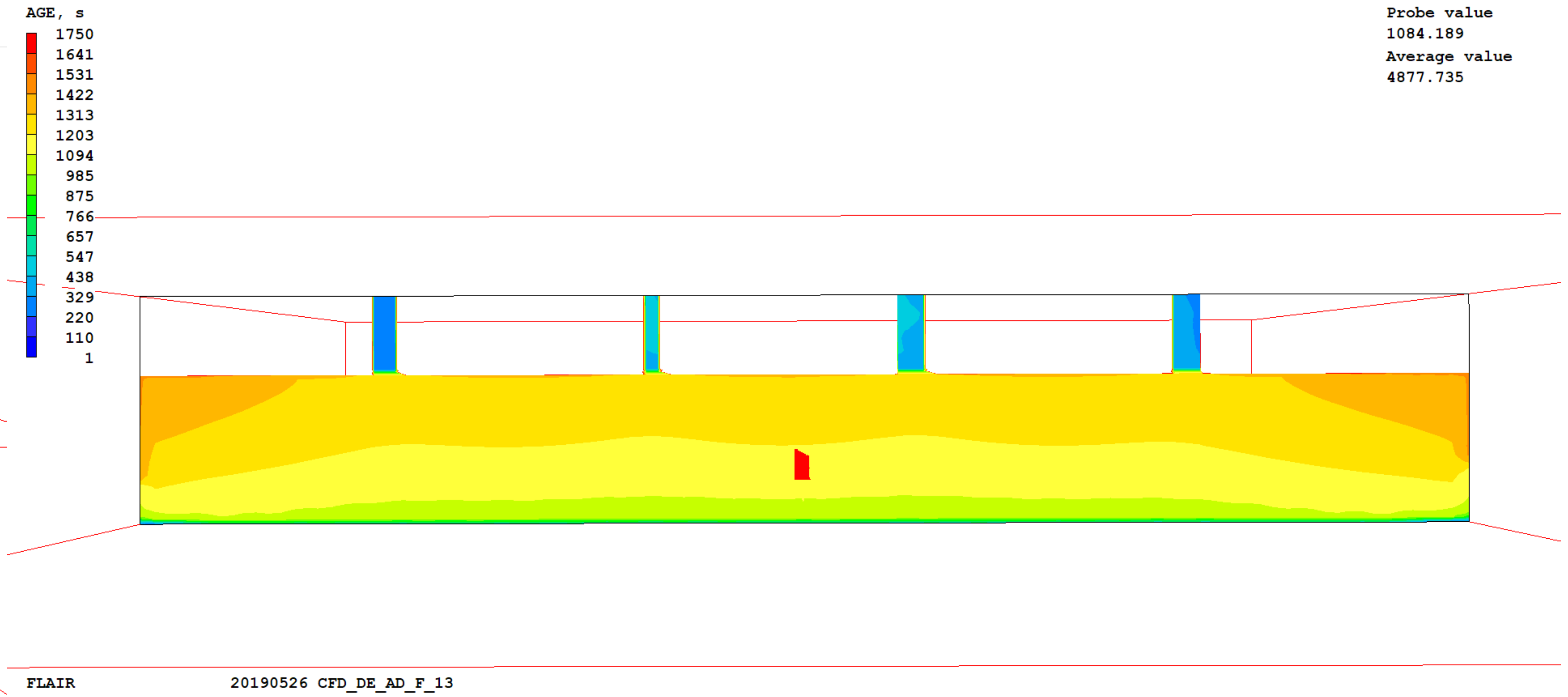
	Air distribution - Floor 1	Air distribution - Floor 2	Air distribution - Floor 2B	Air distribution - Floor 3	Air distribution - Floor 4	Air distribution - Floor 5	Air distribution - Façade 1	Air distribution - Façade 2
Model Used	AD_F_1	AD_F_2	AD_F_2B	AD_F_3	AD_F_4	AD_F_5	AD_F_6	AD_F_7
Description								
Sweeps	140	140	140	140	140	140	140	140
Run time [s]	659	362	373	181	179	180		
Residence time [s]	1587	1644	1750	1616	1703	1673		
ACH [1/h]	2.27	2.19	2.06	2.23	2.11	2.15		
Volume flow rate [m3/s]	31.7	30.6	28.8	31.2	29.6	30.1		
# of inlets	512	36	18	6	4	3	4	4
A inlets [m2]	184	432	216	420	440	210	440	210
Inlet geometry	0.6x0.6m square	2.16x2.16m square	2.16x2.16m square	1.0x70.0m gutter	1.7m wide gutter along perimeter	1m wide gutter	1.7m high opening along perimeter (2.1m high)	0.85m high opening along perimeter (2.1m high)
# of outlets	12	12	12	12	12	12	12	12
A outlets [m2]	22.2	56	56	56	56	56	56	56
Outlet geometry	1.36x1.36m square	2.16x2.16m square	2.16x2.16m square	2.16x2.16m square	2.16x2.16m square	2.16x2.16m square	2.16x2.16m square	2.16x2.16m square
v inlet, max [m/s]	0.2	0.2	0.4	0.2	0.2	0.4	0.2	0.4
dP [Pa]	3.26	4.16	4.12	3.4	4.16	3.6		
Average PMV at 1.5m height	0.278	0.291	0.3	0.291	0.109	0.228		
Average PMV at 1.1m height	0.271	0.271	0.267	0.274	0.064	0.175		
Average PMV at 0.1m height	-0.032	0.003	-0.031	0.033	-0.137	-0.14		

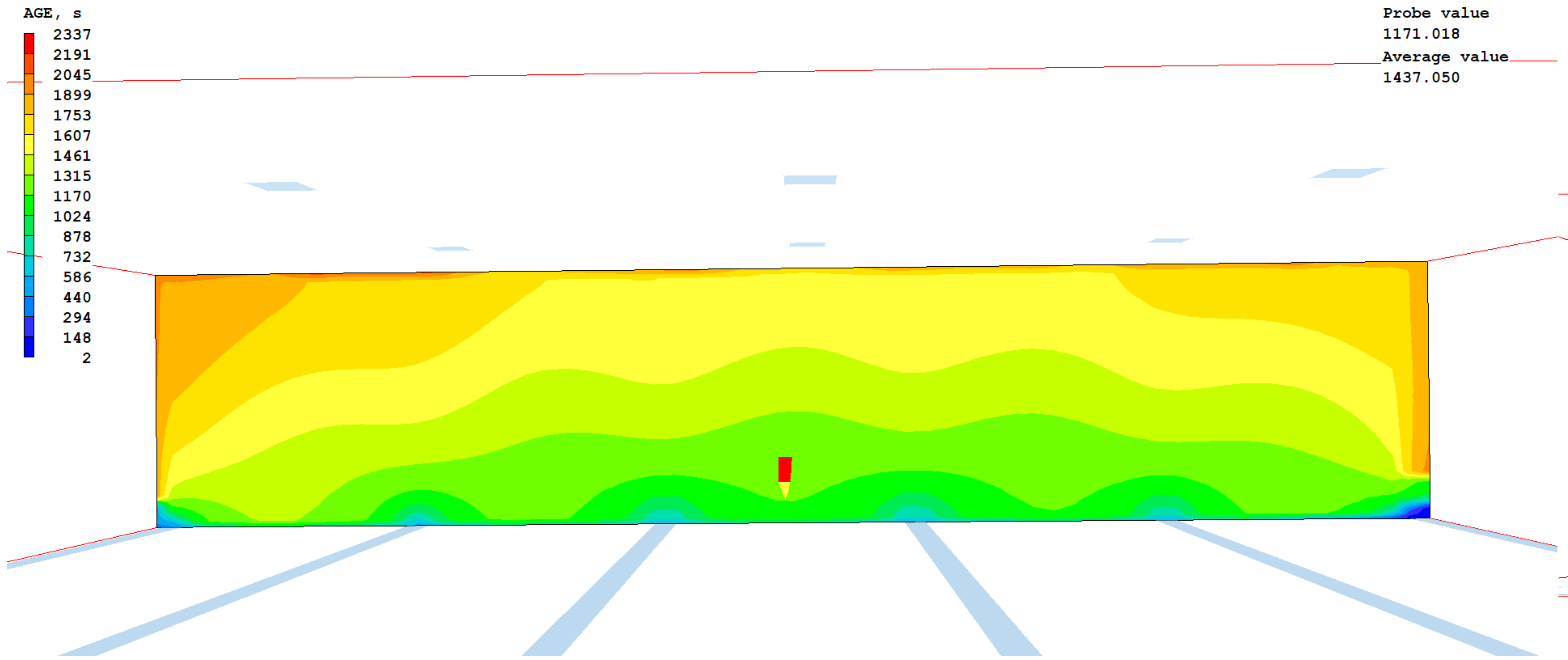
	Hall Geometry 1	Hall Geometry 2	Supply Plenum 1	Supply Plenum 2	Supply Plenum 3	Solar Chimney 1	Wind load	Season Testing - Winter	Season Testing - Summer
Model Used	AD_F_8	AD_F_9	AD_F_10	AD_F_11	AD_F_12	AD_F_13	AD_F_14	AD_F_15	AD_F_16
Description	Roof tapers towards openings at shallow angles	Trumpet shaped roof towards openings	1m deep supply plenum under floor, open all sides	1m deep supply plenum under floor, open all corners	1m deep supply plenum under floor, open one side	Solar chimneys of 4m high with 300W/m2 radiation	1m deep supply plenum under floor, open all corners	Inlet temp - 10C	Inlet temp 27C
Sweeps	140	140	140	140	140	140	140	140	140
Run time [s]	876	937	294	295	224	847		354	343
Residence time [s]	812	862	1090	1245	2085	1988		1599	1669
ACH [1/h]	4.43	4.18	3.3	2.89	1.73	1.81		2.25	2.16
Volume flow rate [m3/s]	55.9	55.5	5.77	5.05	3.02	25.3		31.5	30.2
# of inlets	512	512	1	1	1	512		36	36
A inlets [m2]	184	184	20	20	20	184		432	432
Inlet geometry	0.6x0.6m square	0.6x0.6m square	1.0x20m gutter in middle	1.0x20m gutter in middle	1.0x20m gutter in middle	0.6x0.6m square		2.16x2.16m square	2.16x2.16m square
# of outlets	12	12	4	4	4	12		12	12
A outlets [m2]	56	56	5.4	5.4	5.4	56		56	56
Outlet geometry	2.16x2.16m square	2.16x2.16m square	1.16x1.16m square	1.16x1.16m square	1.16x1.16m square	2.16x2.16m square		2.16x2.16m square	2.16x2.16m square
v inlet, max [m/s]	0.2	0.2	0.2	0.2	0.2	0.2		0.2	0.2
dP [Pa]	2.47	2.98	2.24					4.6	3.95
Average PMV at 1.5m height	-0.105	-0.098	-0.284	-0.18	0.374	1.07		-3.58	1.58
Average PMV at 1.1m height	-0.12	-0.116	-0.298	-0.197	0.352	1.06		-3.69	1.57
Average PMV at 0.1m height	-0.66	-0.68	-0.646	-0.603	-0.128	0.624		-4.8	1.44

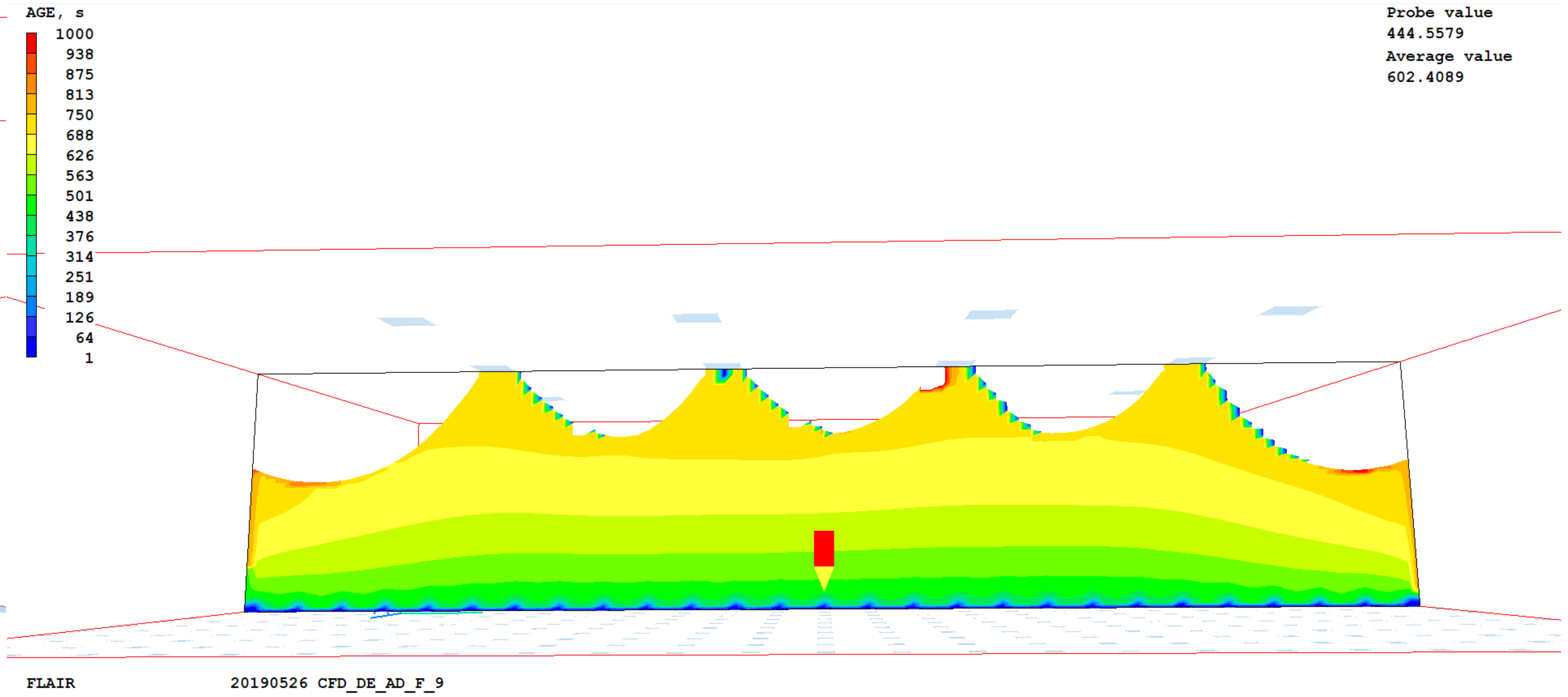
	Boussinesq	Constant Rho	Laminar	Radiation OFF	Regular
Sweeps	860	860	-	860	860
Run time [s]	2134	1605	-	1679	1851
Residence time [s]	882	61	-	851	854
Difference vs 1500 sweeps	3.28%	-92.86%	-	-0.35%	n/a
ACH [1/h]	4.08	59	-	4.23	4.21
Difference vs 1500 sweeps	-3.09%	1301.43%	-	0.48%	n/a
Volume flow rate [m3/h]	5.44	78.64	-	5.64	5.62
Difference vs 1500 sweeps	-3.20%	1299.29%	-	0.36%	n/a
% error - P1	1.07E-01	1.32E-02	-	1.48E-02	3.64E-02
% error - U1	2.15E-01	7.52E-02	-	5.21E-02	1.19E-01
% error - V1	1.77E-01	8.07E-02	-	5.37E-02	1.18E-01
% error - W1	3.54E-01	1.46E-01	-	9.68E-02	2.07E-01
% error - KE	4.93E+01	9.11E-01	-	1.80E+01	8.95E+00
% error - EP	1.63E+00	2.20E-01	-	7.03E-01	3.95E-01
% error - T3	5.05E-01	1.96E-03	-		6.28E-03
% error - AGE	1.18E-02	2.04E-01	-	3.65E-01	6.06E-01
% error - TEM1	3.65E-02	9.09E-04	-	1.27E-03	4.44E-03
Average PMV at 1.5m height	-0.93	-1.66	-	-0.23	-0.212
Difference vs 1500 sweeps	338.68%	683.02%	-	8.49%	n/a
dP	1.84	193	-	1.96	1.95

INITIAL STUDIES

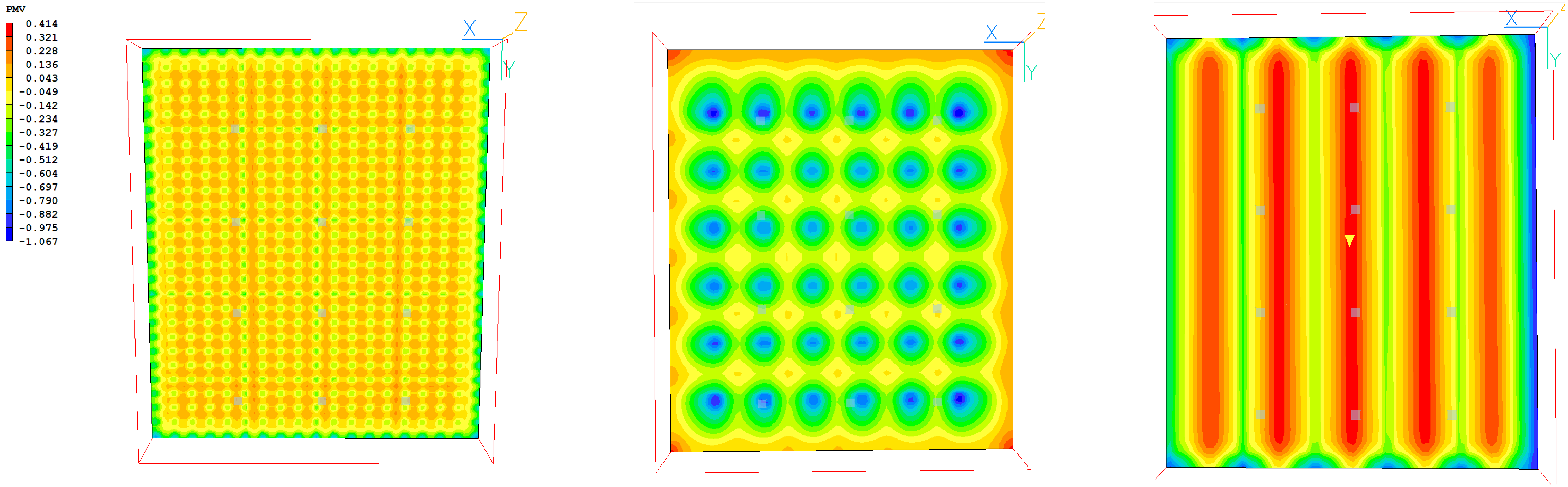




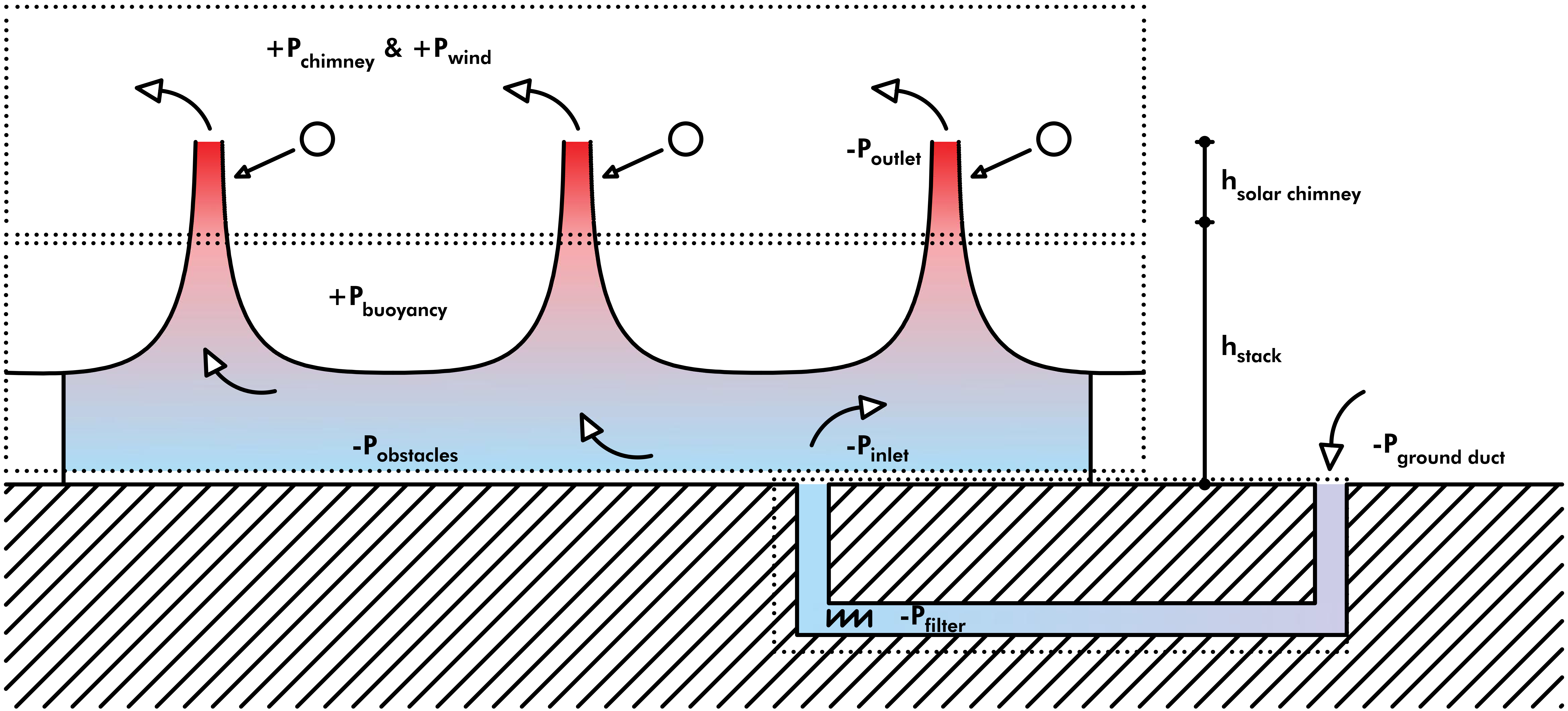




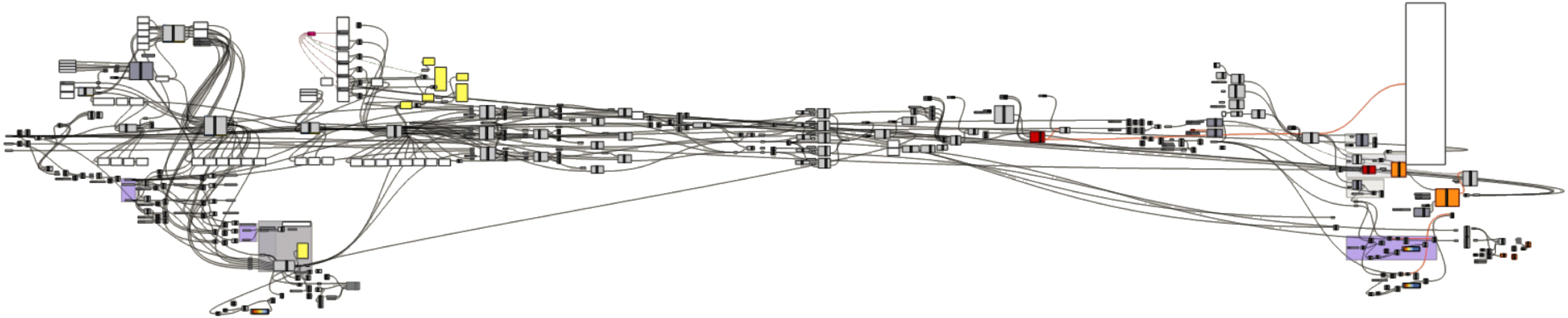
INITIAL STUDIES



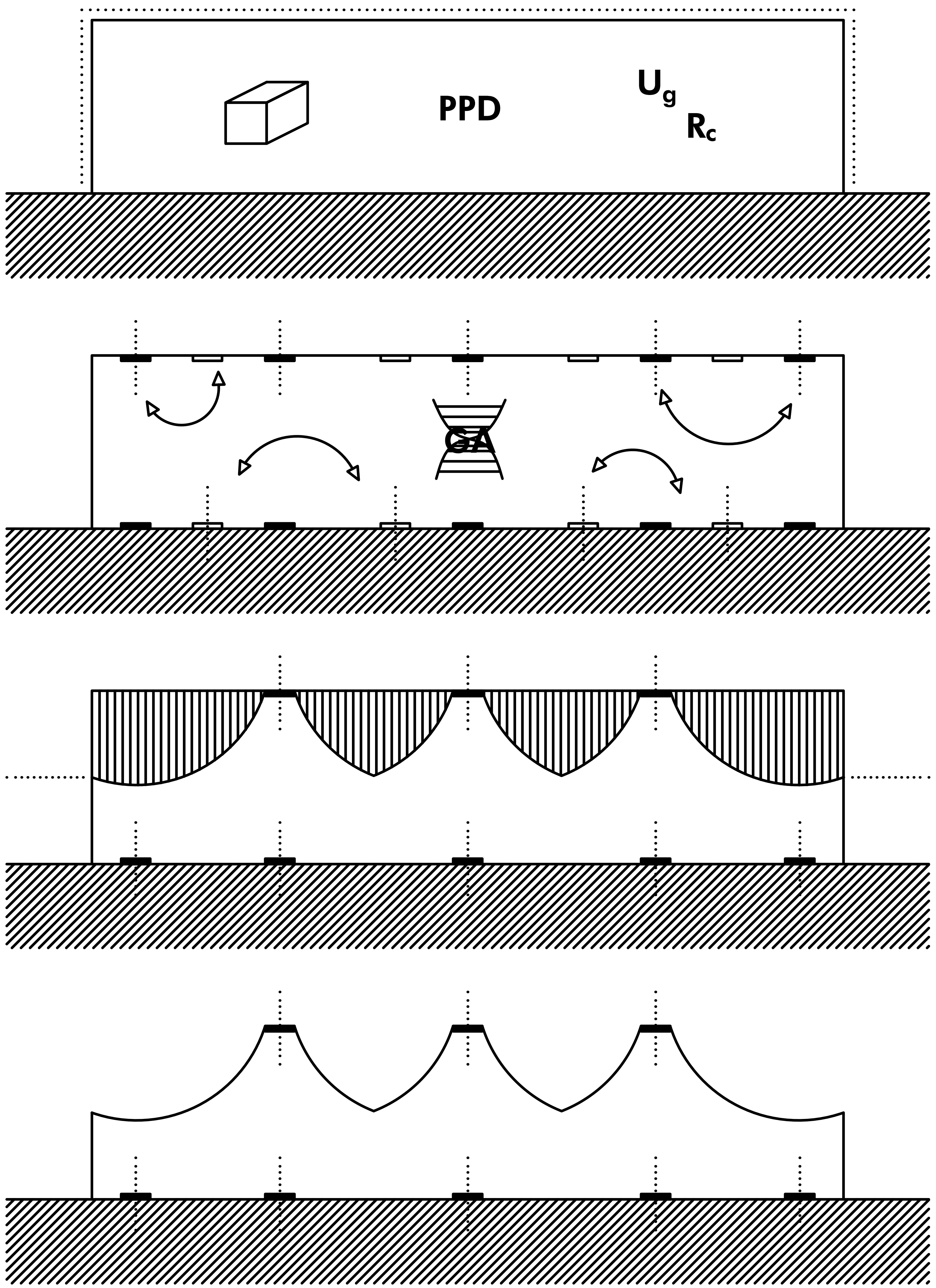
MODEL & CASE

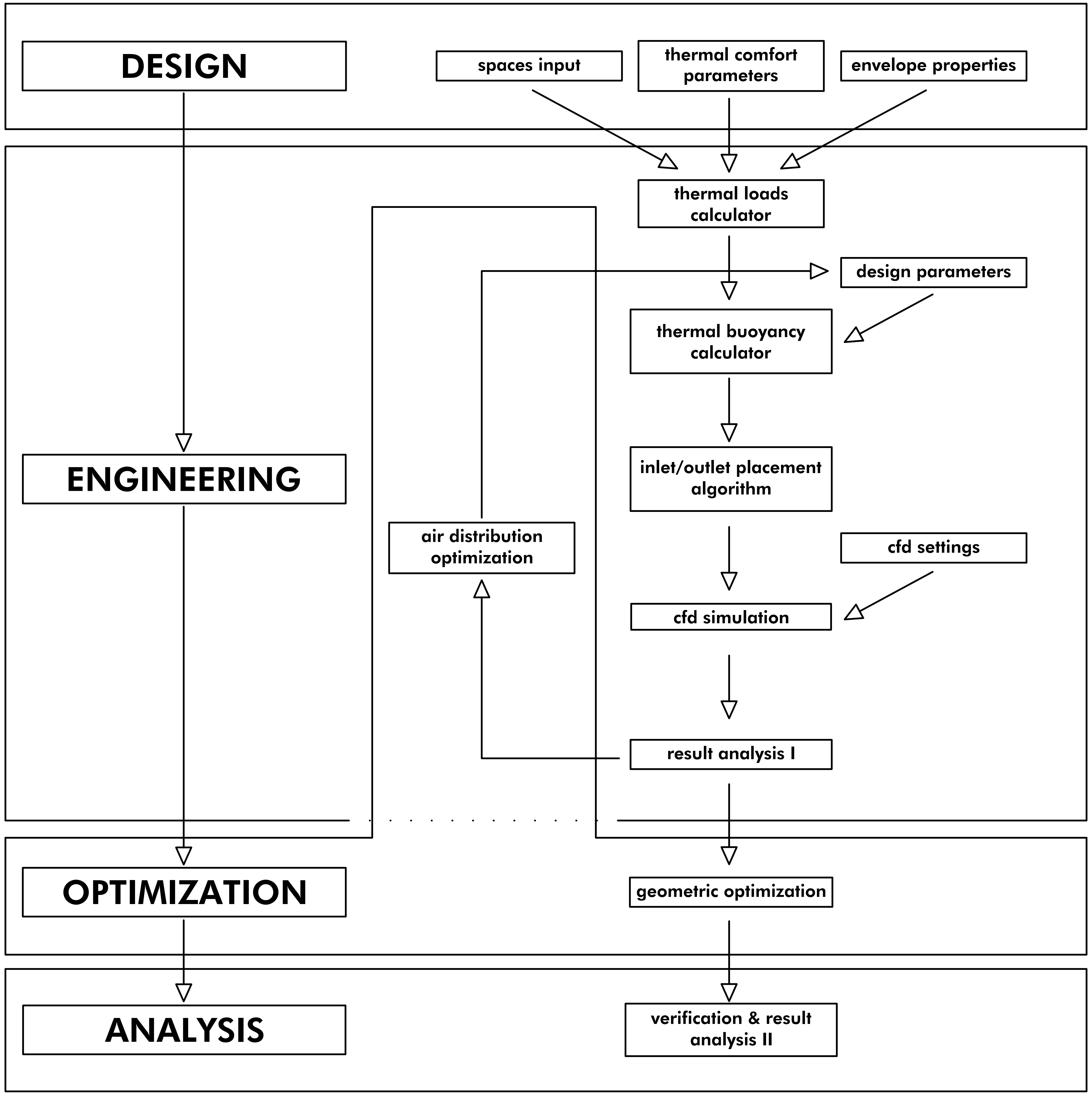

+P chimney & +P wind
-P outlet
h solar chimney
+P buoyancy
h stack
-P obstacles
-P inlet
-P ground duct
-P filter

MODEL & CASE

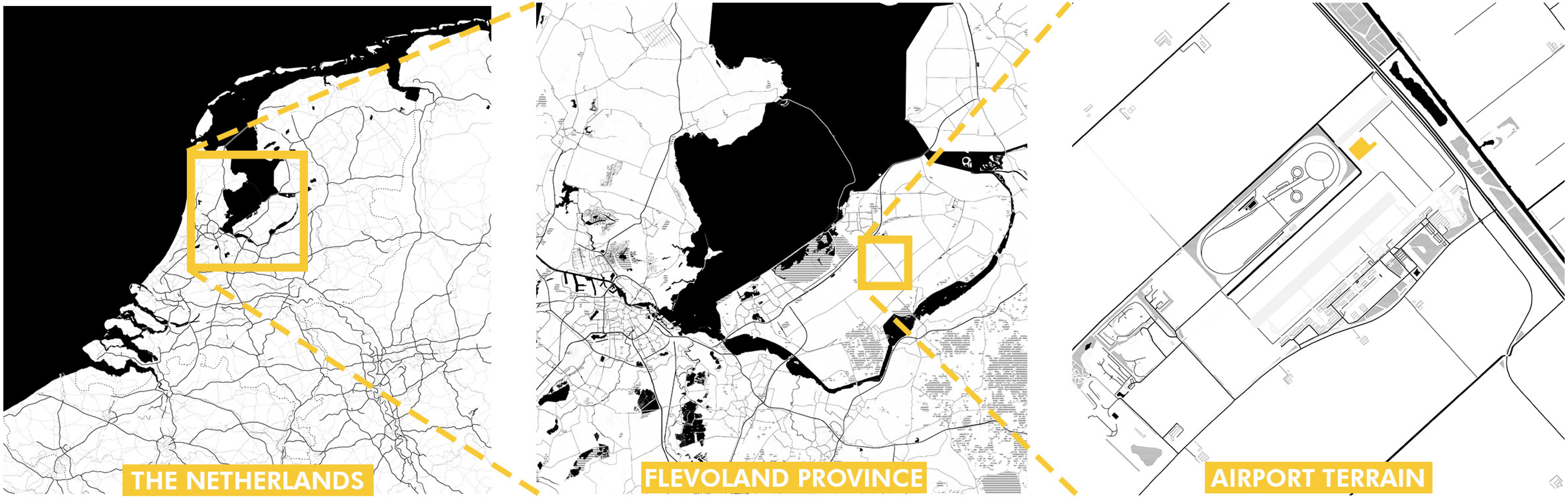


MODEL & CASE


PPD
U_g
R_c
GA

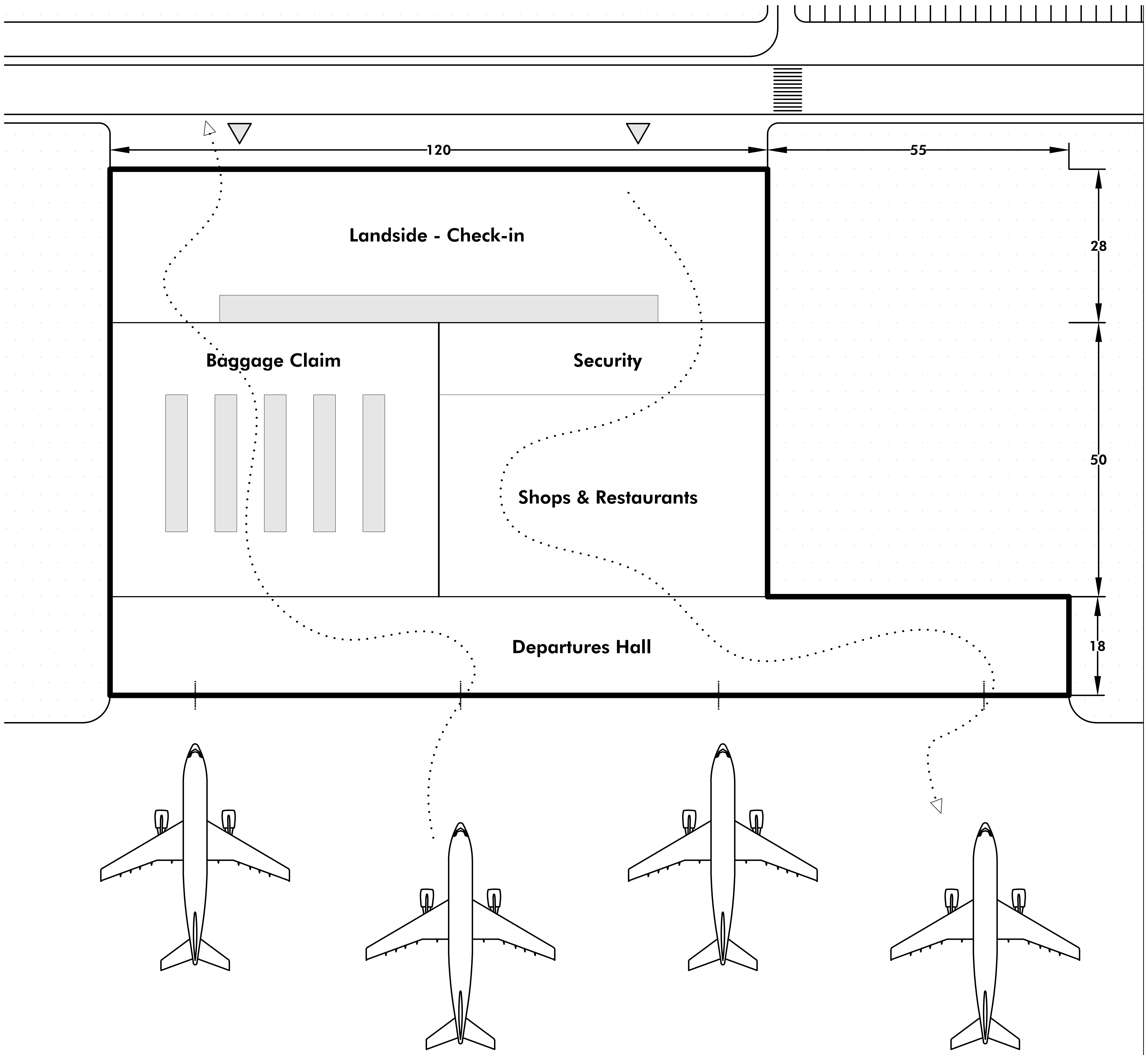

DESIGN
spaces input
thermal comfort parameters
envelope properties
thermal loads calculator
design parameters
thermal buoyancy calculator
ENGINEERING
inlet/outlet placement algorithm
air distribution optimization
cfd settings
cfd simulation
result analysis I
OPTIMIZATION
geometric optimization
ANALYSIS
verification & result analysis II

MODEL & CASE

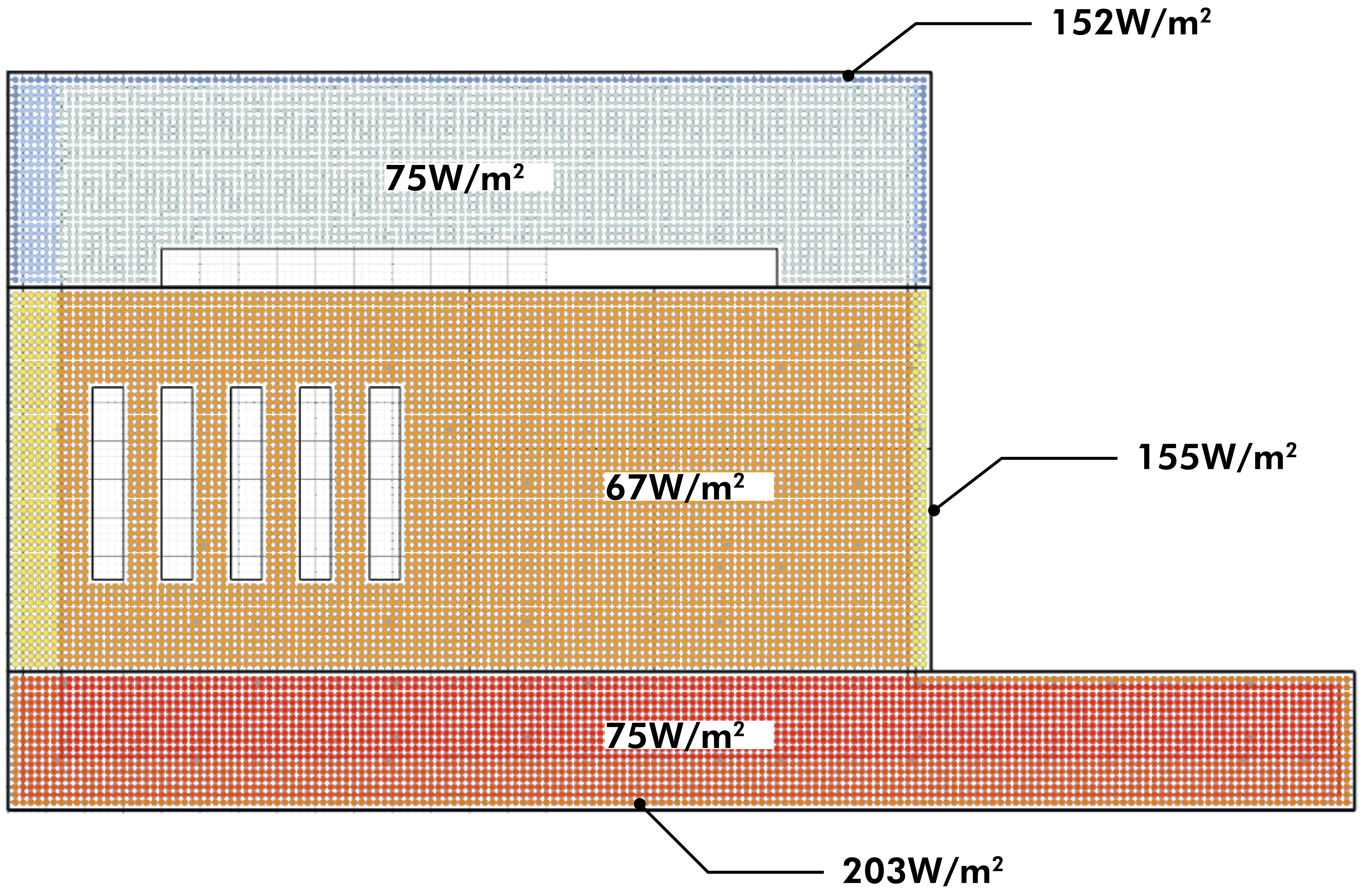


MODEL & CASE

Space Name	Landside Check-in	Main Hall	Airside Departures
Level of Service	1.8	1.4	1.8
PPD	15	5	15
Comfort Class	III	I	III
Design Season	Summer		
Outside Temperature	27		
Air Supply Temperature	18		
Clothing Level	0.5		
Metabolic Rate	1.2		
Façade Area [m2]	1760	3480	3192
Façade WWR [-]	70	40	70
Façade Ug [W/m2K]	1	1	1
Façade Rc [m2K/W]	6.5	6.5	6.5
Façade gcombined [-]	0.1		
Equipment Loads [W/m2]	7	7	7



MODEL & CASE



Solar Loads (avg) [W/m2]	20.8	12.7	35.5
Transmission Loads (avg) [W/m2]	0.96	0.58	1.6
Total Heat Load [W/m2]	96.5	88	111.9
Total Heat Load [kW]	285.7	484.1	352.4
dT stratification	1		
Ttarget	27.2	24.7	27.2
qv,min [m3/s]	25.7	59.6	31.7

MODEL & CASE

```
Occupancy Calculator
import rhinoscriptsyntax as rs
import math

for i in range(len(spaces)):
    centroids= rs.SurfaceAreaCentroid(spaces)
space_centerpoints=centroids[::2]

area_per_space_list = rs.SurfaceArea(spaces)
area_per_space=area_per_space_list[::2]
area_per_space_list= []
LOS_per_space_list=list(LOS_per_space)
occupancy_per_space= []

for i in range(len(spaces)):
    a=area_per_space[i]
    b=LOS_per_space[i]
    c=a/b
    occupancy_per_space.append(round(c))

Thermal Comfort Parameters I
dT_balance_list = []
design_condition_list = []
category_output_list = []

for i in range(len(_PPD)):
    if _season == 1:
        season_output = "Winter"
        clothing_level = _clothing_level[0]
        T_outside_ = -10
        q_sol_ = 50
        if _PPD[i] <= 5:
            dT_balance = 0
        elif 6 <= _PPD[i] <= 10:
            dT_balance = -0.5
        elif 11 <= _PPD[i] <= 15:
            dT_balance = -2.5
        else:
            dT_balance = 0
        dT_balance_list.append(dT_balance)
    elif _season == 2:
        season_output = "Intermediate"
        clothing_level = _clothing_level[1]
        T_outside_ = 10
        q_sol_ = 500
        if _PPD[i] <= 5:
            dT_balance = 0
        elif 6 <= _PPD[i] <= 10:
            dT_balance = -1.5
        elif 11 <= _PPD[i] <= 15:
            dT_balance = -2.5
        else:
            dT_balance = 0
        dT_balance_list.append(dT_balance)
    elif _season == 3:
        season_output = "Summer"
        clothing_level = _clothing_level[2]
        T_outside_ = 27
        q_sol_ = 500
        if _PPD[i] <= 5:
            dT_balance = 0
        elif 6 <= _PPD[i] <= 10:
            dT_balance = 1.5
        elif 11 <= _PPD[i] <= 15:
            dT_balance = 2.5
        else:
            dT_balance = 0
        dT_balance_list.append(dT_balance)
    else:
        season_output = "ERROR - WRONG SEASON"

    category_note = "EN 15251 - Category "
    if _PPD[i] <= 5:
        category_output = category_note + "I"
    elif 6 <= _PPD[i] <= 10:
        category_output = category_note + "II"
    elif 11 <= _PPD[i] <= 15:
        category_output = category_note + "III"
    else:
        category_output = "ERROR - TOO HIGH PPD"
    category_output_list.append(category_output)

    design_condition = 'Season: ' + str(season_out-
put[i]) + """
    """ + 'Comfort Category: ' + str(category_out-
put[i])
    de-sign_condition_list.append(design_condition)

dT_balance_ = dT_balance_list
design_condition_ = design_condition_list
Occupancy Calculator
import rhinoscriptsyntax as rs
import math

for i in range(len(spaces)):
    centroids= rs.SurfaceAreaCentroid(spaces)
space_centerpoints=centroids[::2]

area_per_space_list = rs.SurfaceArea(spaces)
area_per_space=area_per_space_list[::2]
area_per_space_list= []
LOS_per_space_list=list(LOS_per_space)
occupancy_per_space= []

for i in range(len(spaces)):
    a=area_per_space[i]
    b=LOS_per_space[i]
    c=a/b
    occupancy_per_space.append(round(c))
```

```
Envelope Calculator
import math
import numpy as np

q_solar_list = []
q_solar_floor_field_list = []
q_solar_in_zone_list = []
q_transmission_list = []
q_transmission_floor_field_list = []
q_transmission_in_zone_list = []
q_combined_list = []

for i in range(len(_area_per_space)):
    a = _facade_area[i]
    b = _glazing_percentage[i]
    c = _glazing_g_factor[i]
    d = _q_sol
    e = _area_solar_zone[i]
    f = _area_transmission_zone[i]
    g = _area_per_space[i]
    q_solar = float(a*0.01*b*c*d)
    q_solar_list.append(q_solar)
    q_solar_floor_field = q_solar / g
    q_solar_floor_field_list.append(q_solar_floor_
field)
    q_solar_in_zone = q_solar / e
    q_solar_in_zone_list.append(q_solar_in_zone)
    h = _facade_Ug[i]
    j = _facade_Rc[i]
    k = _T_outside
    l = _T_inside
    q_transmission = a*0.01*b*h*abs(k-l)
    q_transmission_list.append(q_transmission)
    q_transmission_floor_field = q_transmission/g
    q_transmission_floor_field_list.append(q_trans-
mission_floor_field)
    q_transmission_in_zone = q_transmission/e
    q_transmission_in_zone_list.append(q_transmis-
sion_in_zone)
    q_combined = (q_transmission + q_solar)/g
    q_combined_list.append(q_combined)

q_solar_floor_field_ = q_solar_floor_field_list
q_solar_in_zone_ = q_solar_in_zone_list
q_transmission_floor_field_ = q_transmission_floor_
field_list
q_transmission_in_zone_ = q_transmission_in_zone_list
q_sol_and_transmission_ = q_combined_list
Occupancy Calculator
import rhinoscriptsyntax as rs
import math

for i in range(len(spaces)):
    centroids= rs.SurfaceAreaCentroid(spaces)
space_centerpoints=centroids[::2]

area_per_space_list = rs.SurfaceArea(spaces)
area_per_space=area_per_space_list[::2]
area_per_space_list= []
LOS_per_space_list=list(LOS_per_space)
occupancy_per_space= []

for i in range(len(spaces)):
    a=area_per_space[i]
    b=LOS_per_space[i]
    c=a/b
    occupancy_per_space.append(round(c))
Occupancy Calculator
import rhinoscriptsyntax as rs
import math

for i in range(len(spaces)):
    centroids= rs.SurfaceAreaCentroid(spaces)
space_centerpoints=centroids[::2]

area_per_space_list = rs.SurfaceArea(spaces)
area_per_space=area_per_space_list[::2]
area_per_space_list= []
LOS_per_space_list=list(LOS_per_space)
occupancy_per_space= []

for i in range(len(spaces)):
    a=area_per_space[i]
    b=LOS_per_space[i]
    c=a/b
    occupancy_per_space.append(round(c))
```

MODEL & CASE

Ventilation Rate Calculator

```
import numpy as np

Q_per_space_per_m2_ = []
Q_per_space_total_= []
Q_people_eq_per_space_ = []
Qperspaceperm2 = int
Qperspacetotal = int
Q_ventilation_per_space_ = []
heatingperm2 = int(_heating_per_m2)
Qventilationperspace = int

for i in range(len(_area_per_space)) :
    a= _occupancy_per_space[i]
    b= _metabolic_rate
    c= _area_per_space[i]
    d= _Q_internal[i]
    e= _Q_transmission[i]
    f= heatingperm2
    Qperspaceperm2 = (a*b)/c+d+e
    Qperspacetotal = a*b+(d+e)*c
    Q_per_space_per_m2_.append(Qperspaceperm2)
    Q_per_space_total_.append(Qperspacetotal)
    Qventilationperspace = a*b+(d+e)*c-f*c
    Q_ventilation_per_space_.append(Qventilationper-
space)
    Q_people_eq_per_space = ((a*b)/c)+d
    Q_people_eq_per_space_.append(Q_people_eq_per_
space)

qvminclimatization = int

if _season == 1:
    Tsupply = _T_supply_max
    T_supply_ = _T_supply_max
elif _season == 2:
    Tsupply = _T_supply_max
    T_supply_ = _T_supply_max
elif _season == 3:
    Tsupply = _T_supply_min
    T_supply_ = _T_supply_min
print('Supply temperature =' + ' ' + str(Tsupply))

q_v_min_per_space_list = []
q_v_min_climatization_list = []
q_v_min_en15251_list = []
_T_target_list = []

for i in range(len(Q_ventilation_per_space_)):
    a= Q_ventilation_per_space_[i]
    b= 1.20
    c= 1008
    d= _T_balance + _dT_balance[i] #+ _dT_thermal_
stratification
    e= Tsupply
    qvminclimatization= a/(b*c*(d-e))
    q_v_min_climatization_list.append(qvminclimatiza-
tion)
    _T_target_list.append(d)

T_target_ = _T_target_list

qvminen15251 = int
qvperperson = int
qvperarea = int
qvperperson_list = []
qvperarea_list =[]

for i in range(len(_PPD)):
    if _PPD <= 10:
        qvperperson = 0.01
        qvperarea = 0.001
    else:
        qvperperson = 0.007
        qvperarea = 0.0007
    qvperperson_list.append(qvperperson)
    qvperarea_list.append(qvperarea)

for i in range(len(Q_ventilation_per_space_)):
    a= _occupancy_per_space[i]
    b= qvperperson_list[i]
    c= _area_per_space[i]
    d= qvperarea_list[i]
    qvminen15251= a*b + c*d
    q_v_min_en15251_list.append(qvminen15251)

#print('qvminen15251 =' + ' ' + str(qvminen15251) + '
' + 'm3/s')
print(q_v_min_en15251_list)

q_v_min_per_space_list = list(map(max, zip(q_v_min_
en15251_list, q_v_min_climatization_list)))
print('qvminperspace = ' + str(q_v_min_per_space_
list))
q_v_min_per_space_= q_v_min_per_space_list
q_v_min_total_ = sum(q_v_min_per_space_)
```

Heat Load Points Calculator

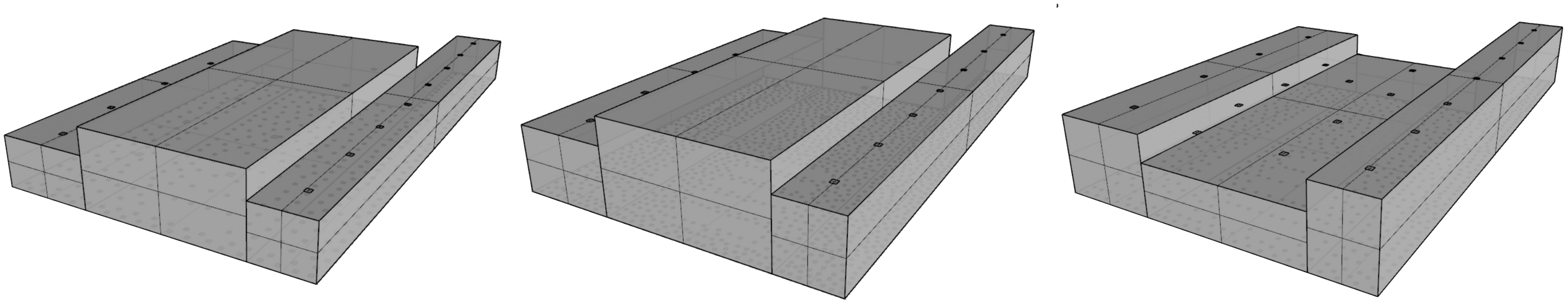
```
import Rhino.Geometry as rg
import Grasshopper as gh
import rhinoscriptsyntax as rs
import clr
clr.AddReference("Grasshopper")
import Grasshopper.Kernel.Data.GH_Path as ghpath
import Grasshopper.DataTree as datatree
from Grasshopper.Kernel.Data import GH_Path
import System
import ghpythonlib.components as ghc

is_inside_list=[]
is_inside_transmission_list=[]
transmission_points_list=[]
transmis-sion_curves=list(transmission_zone_curves)
heat_load_points_list = list(heat_load_points)
datatuples=[]
for i in range(len(heat_load_points)):
    dataTuple=tuple(heat_load_points[i])
    dataTu-ple=(dataTuple[0],dataTuple[1],dataTu-
ple[2])
    datatuples.append(dataTuple)

out_points_list=[]
heat_points_out_list=[]
ylist=[]
zlist=[]
xlist=list(zip(*datatuples))
z_coordinates_list=[]
is_inside_occupancy_list=[]
occupancy_points_list=[]
is_inside_solar_list=[]
solar_points_list=[]

for i in range(len(heat_load_points)):
    z_coordinate=heat_load_points[i][2]
    z_coordinates_list.append(z_coordinate)

for i in range(len(transmission_zone_curves)):
    for j in range(len(heat_load_points)):
        is_inside_occupancy= (spac-es[i].Con-
tains(heat_load_points[j])==rg.PointContainment.In-
side)
        is_inside_occupancy_list.append(is_inside_oc-
cupancy)
        if is_inside_occupancy==True:
            occupan-cy_points_list.append(heat_load_
points)
            z_coordinates_list[j]=z_coordinates_
list[j]+occupancy_equipment_heat_load[i]

        is_inside_transmission= (transmis-si-on_zone_
curves[i].Contains(heat_load_points[j])==rg.PointCon-
tainment.Inside)
        is_inside_transmission_list.append(is_inside_
transmission)
        if is_inside_transmission==True:
            transmis-sion_points_list.append(heat_
load_points)
            z_coordinates_list[j]=z_coordinates_
list[j]+transmission_heat_load[i]

        is_inside_solar= (so-lar_zone_curves[i].Con-
tains(heat_load_points[j])==rg.PointContainment.In-
side)
        is_inside_solar_list.append(is_inside_solar)
        if is_inside_solar==True:
            so-lar_points_list.append(heat_load_
points)
            z_coordinates_list[j]=z_coordinates_
list[j]+solar_heat_load[i]
heat_load_values=z_coordinates_list
```

Buoyancy Calculator

```
import numpy as np
A_effective_per_space_list=[]
dT_per_space_list=[]
h_max_per_space_list=[]

for i in range(len(_q_v_min_per_space)):
    a=float(_g)
    b=float(_c_d)
    c=_q_v_min_per_space[i]
    d=float(_T_supply)
    e=float(_T_target[i])
    f=_h_stack[i]
    g=abs(e-d)/(e+273)
    h=np.sqrt(2*a*f*g)
    Aeffperspace=float
    Aeffperspace= c/(b*h)
    A_effective_per_space_list.append(Aeffperspace)
    A_effective_per_space_=A_effective_per_space_list
    j=abs(e-d)
    dT_per_space_list.append(j)
    dT_per_space_=dT_per_space_list

dP_initial_list = []
h_stack_new_list = []

for i in range(len(_h_stack)):
    d=float(_T_supply)
    e=float(_T_target[i])
    f=float(_dP_extra[i])
    dh = f/(1.21*9.81*(abs(e-d)/(e+273)))
    h_stack_new = round(_h_stack[i] + dh)
    h_stack_new_list.append(h_stack_new)
    dP_initial = 1.21*9.81*_h_stack[i]*(abs(e-d)/
(e+273))
    dP_initial_list.append(dP_initial)

dP_initial_ = dP_initial_list
h_stack_ = h_stack_new_list
```

MODEL & CASE

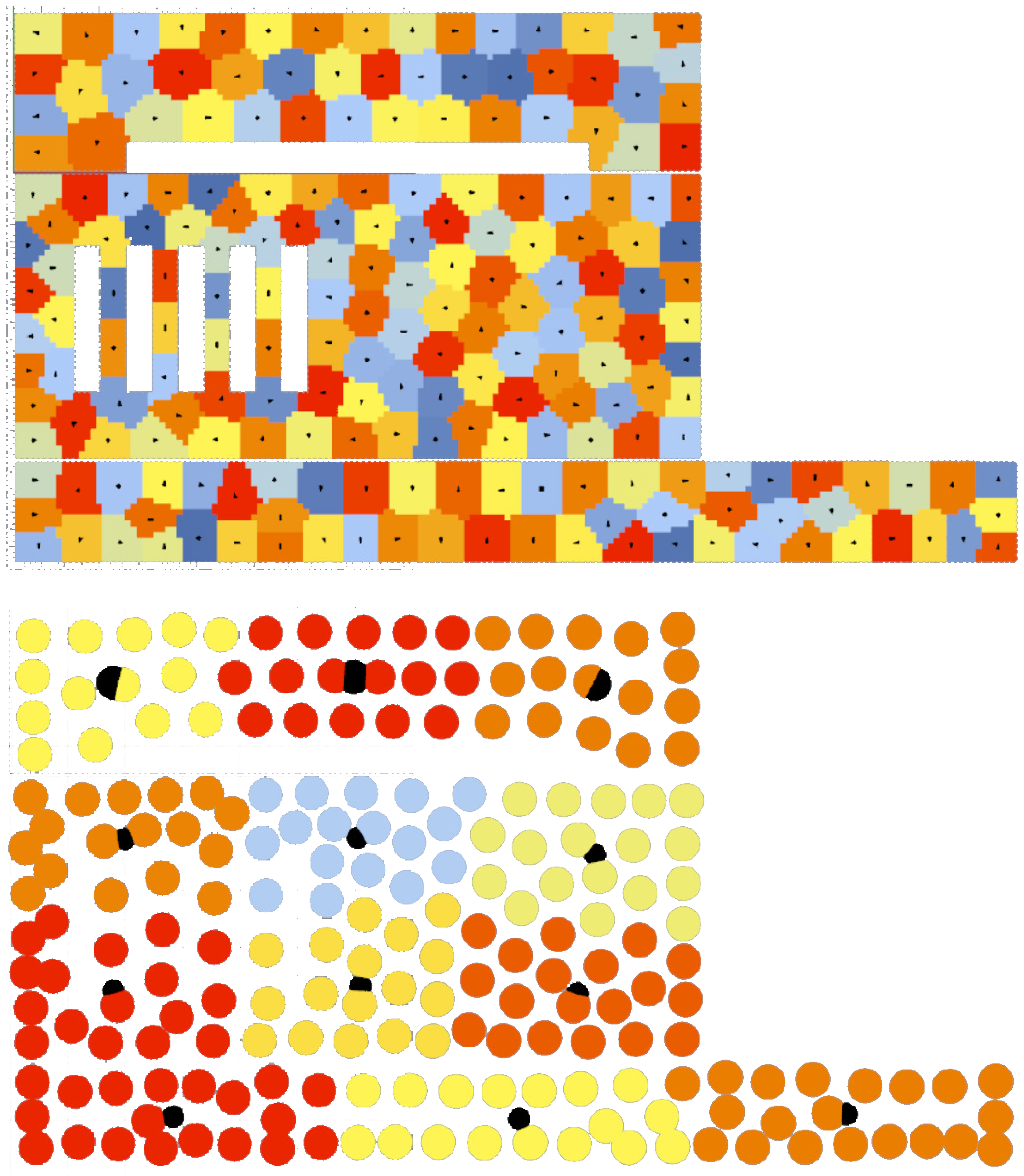
5 design variables in the model
automatic geometry generation



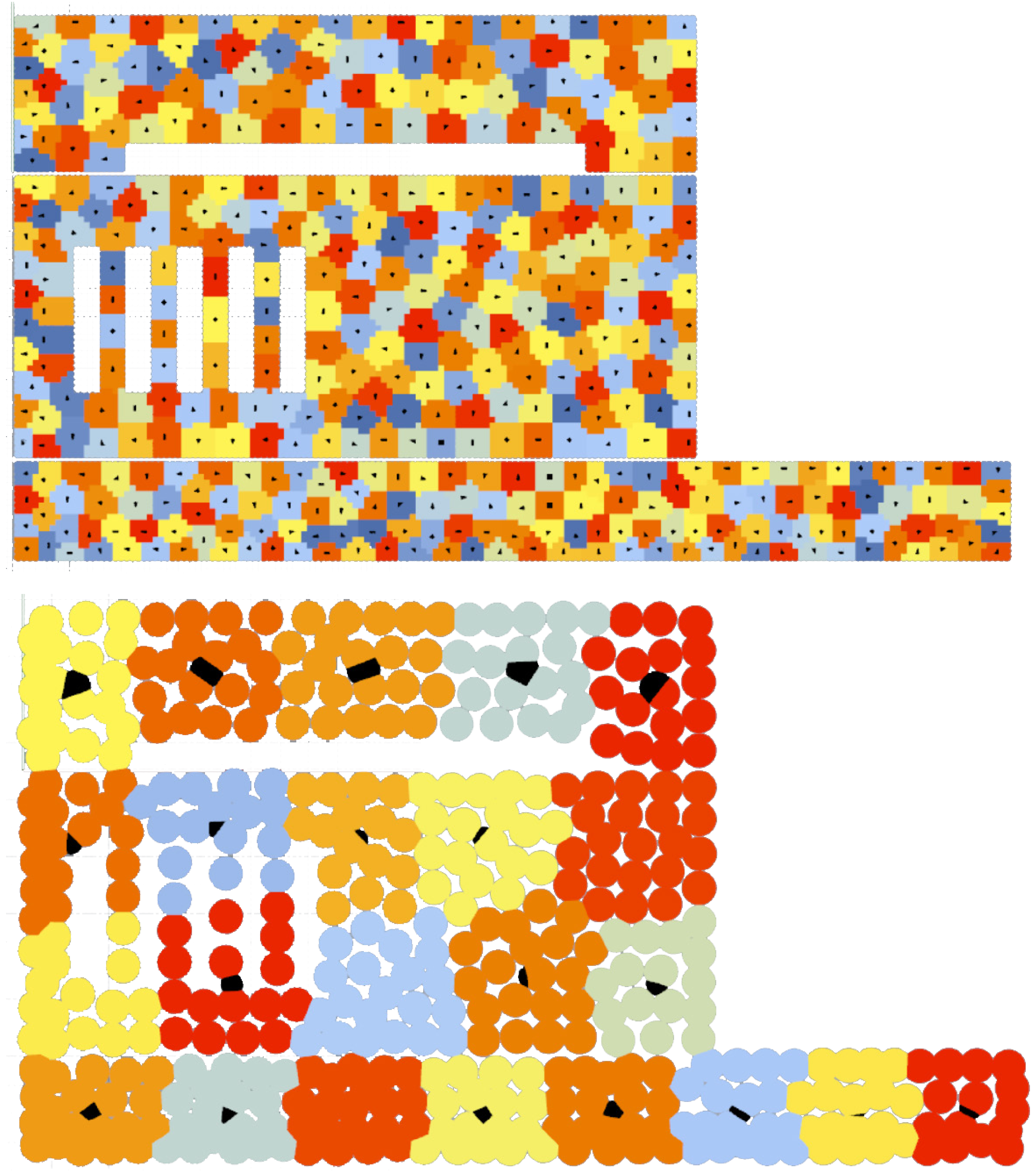
	v_inlet	A_inlet	A_outlet	h_stack_1	h_stack_2
	{0}	{0}	{0}	{0}	{0}
0	0.2	0.36	3.24	12	12
1	0.4	0.81	5.76	16	16
2	0.6	1.44	12.96	20	20
3	0.8	2.25	23.04	24	24
4	1.0	3.24	51.84	28	28

MODEL & CASE

```
K-means Preparation
import Rhino.Geometry as rg
import Rhino
import rhinoscriptsyntax as rs

is_inside_list=[]
heat_points_out_list=[]
z_coordinates_list=[]

for point in heat_load_points:
    is_inside=(spaces.Contains(point,rg.Plane.
WorldXY,0.01)==rg.PointContainment.Inside)
    is_inside_list.append(is_inside)
    if is_inside==True:
        heat_points_out_list.append(point)

for i in range(len(heat_load_points)):
    is_inside= (spac-es.Contains(heat_load_
points[i])==rg.PointContainment.Inside)
    is_inside_list.append(is_inside)
    if is_inside==True:
        z_coordinates_list.append(heat_load_val-
ues[i])

heat_points_out=heat_points_out_list
z_coordinates=z_coordinates_list
import sklearn
import sklearn.cluster as sk
from sklearn.cluster import KMeans
import numpy as np
import math as math

K-means Inlets (Weighted)
numbers=np.array(_z_coordinates)

highest_value=max(_z_coordinates)
print(highest_value)
weights=np.divide(numbers, highest_value)

a=list(weights)
weights=np.array(weights)
print(weights)
datapoints=np.array(_points)
kmeans=KMeans(n_clusters=int(_n_in-
lets),init="k-means++",random_state=0,n_jobs=-1).
fit(datapoints, y=None, sam-ple_weight=weights)
labels= list(kmeans.labels_)
centres=kmeans.cluster_centers_
x_centres=list(centres[:,0])
y_centres=list(centres[:,1])
z_centres=list(centres[:,2])
```



Weighted K-means distribution for inlets (top)
K-means distribution for outlets (bottom)



Same model with increased facade transmission

MODEL & CASE

CFD settings preparation
- input of custom OpenFOAM dictionaries
- speed and comparative performance

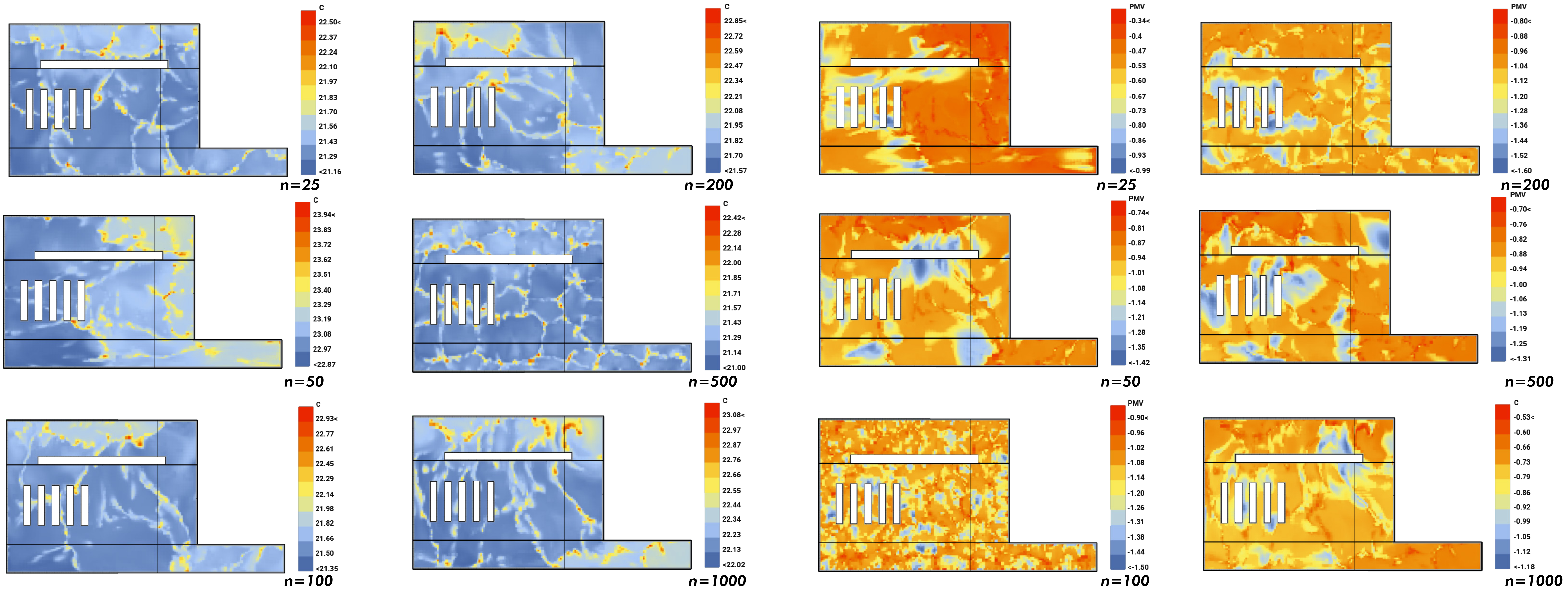
```
topoSetDict and fvOptions
case_folder_windows = 'C:\\Users\\okan-\\butterfly'

topoSetDictScript_A = """
/*--------------------------------*- C++ -*----------------------------------*\

| ========= | |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1706+                                |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
     version     4.0;
     format      ascii;
     class       dictionary;
     location    "system";
     object      topoSetDict;
}
actions
(
"""

fvOptionsScript_A = """
/*--------------------------------*- C++ -*----------------------------------*\

| ========= | |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v1706+                                |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
     version     4.0;
     format      ascii;
     class       dictionary;
     location    "system";
     object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

"""

toposetfile = case_folder_windows + '\\' + case_name + '\\' + 'system\\' + 'topoSetDict'
topoFile=open(toposetfile, "w")
topoFile.write(topoSetDictScript_A)
topoFile.close()

fvoptionsfile = case_folder_windows + '\\' + case_name + '\\' + 'system\\' + 'fvOptions'
fvFile=open(fvoptionsfile, "w")
```

	LS S 12	LS S 12	LS S 12	LS S 12	LS S 12	LS S 12	LS S 12
Model Used	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1
Description	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes	Simplified model of Lelystad with 3 boxes
# of cpu's	12	12	12	12	12	12	12
# of sweeps	25	50	100	200	300	500	1000
PPDavg	32.5	31.9	28.9	26.7	23.9	18.3	11.3
dT(min, max)	1.03	1.42	1.35	1.6	1.3	1	1.1

	LS S 1	LS S 2 P5_test_12	LS S 3	LS S 4	LS S 5	LS S 6	LS S 7	LS S 8	LS S 9	LS S 10	LS S 11	LS S 12
Model Used	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1	Lelystad Simplified 1
Description	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx	Simplified model of Lelystad with 3 boxes of xxx
BlockMesh Factor	1	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.5	0.5	0.5	0.5
# of cells (BlockMesh)	2.5e^6	38000	38000	38000	38000	38000	38000	127000	38000	38000	38000	38000
Grid refinement (direction) (% length) (% cells)	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 30-70 55-45	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	Z-grading 20-80 50-50	no grading
Refinement level inlets & outlets	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)	(2,2)	(4,4)	(3,3)	(3,3)
ExpandMesh	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON
SnappyHexMesh Cells between levels	2	2	4	8	2	2	2	2	2	2	2	1
SnappyHexMesh Extra Layers	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	ON	ON	ON	ON
SnappyHexMesh Surface feature level	2	2	2	2	4	2	2	2	2	2	2	2
time (BlockMesh) [s]	5.1	1.9	1.3	1.9	2	2.2	1.9	4.3	1.9	1.9	1.9	2
time (BlockMesh loading) [s]	72	6.5	4.3	6.5	6.6	6.6	6.5	21.3	6.5	6.5	6.5	6.6
time (SnappyHexMesh) [s]	318	84	96	138	156	60	66	144	90	318	168	24
time (SnappyHexMesh loading) [s]	324	28	30	33	90	31	28	66	28	288	90	7.3
PPDavg	-	-	-	-	-	-	-	-	-	-	9.69	8.29
Tavg	-	-	-	-	-	-	-	-	-	-	25.94	25.52

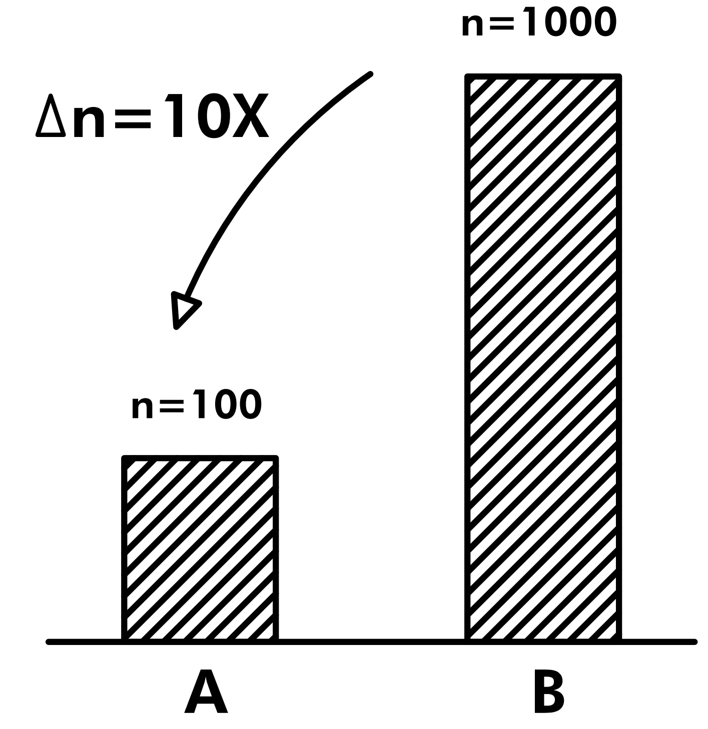
MODEL & CASE

Effects of number of runs on results

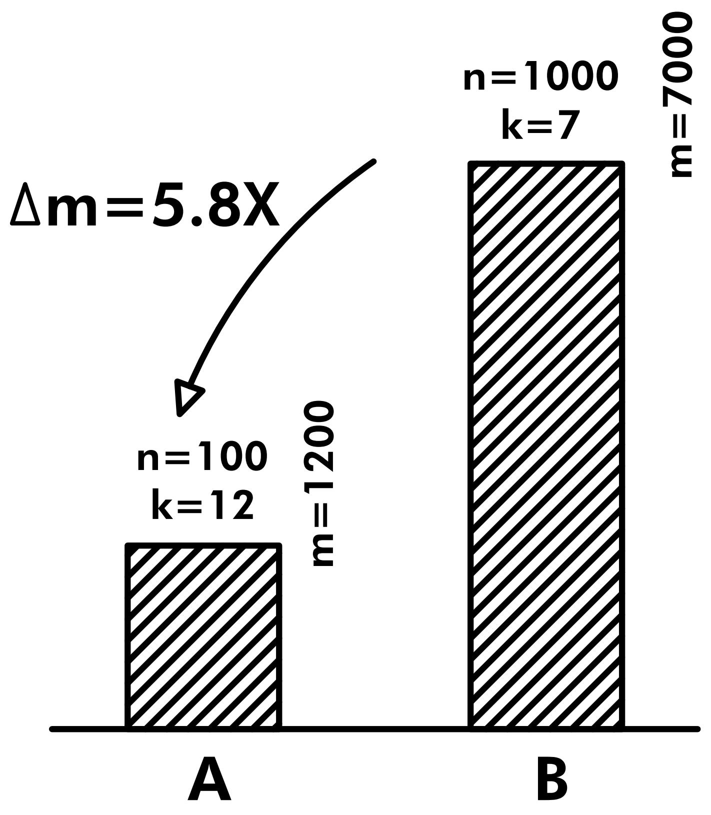


MODEL & CASE

Genetic Algorithm to search for 'best' options



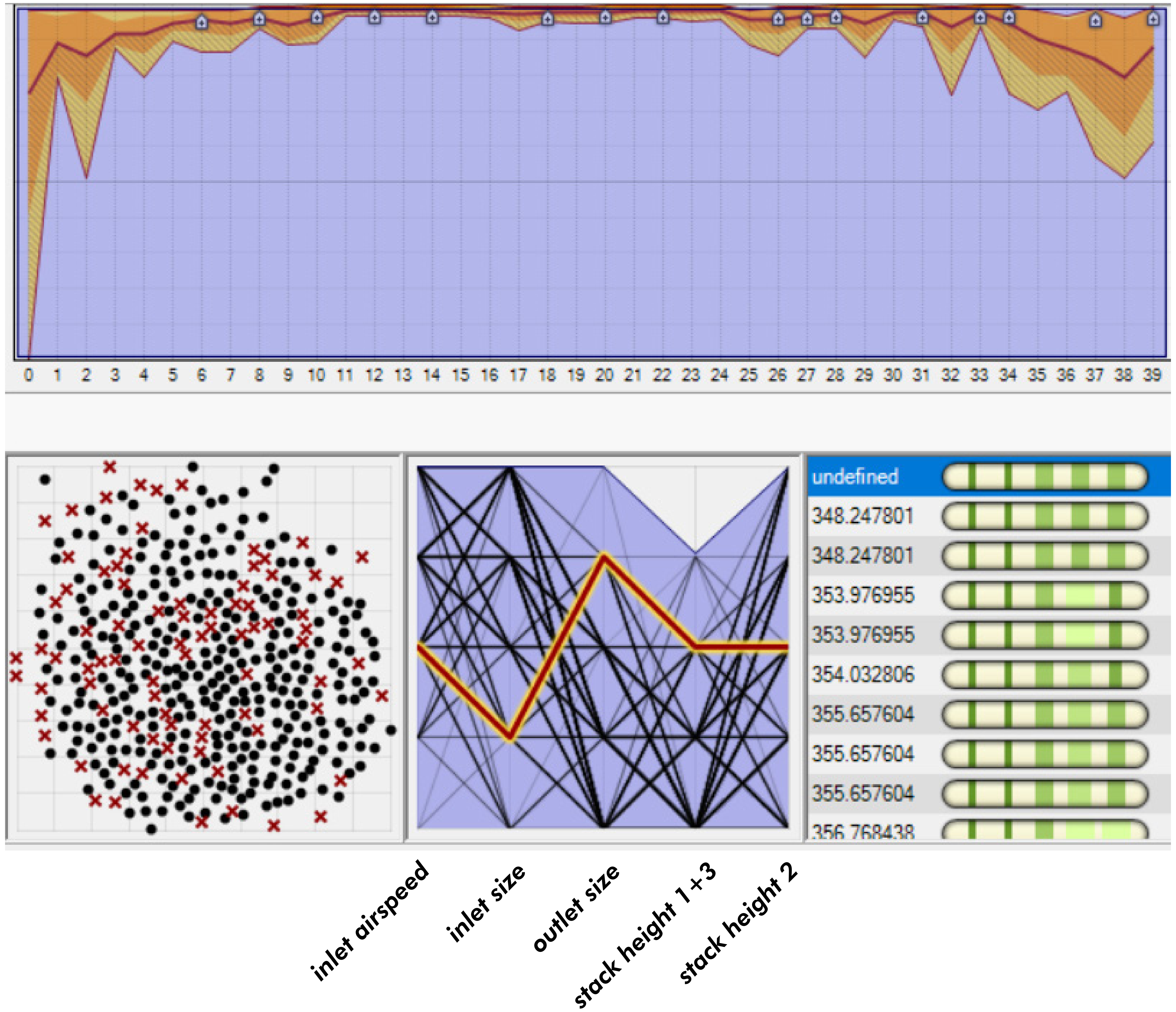
Lower = better
n= inlets + outlets



Lower = better
n= inlets + outlets
k= PPD*100
m=n*k

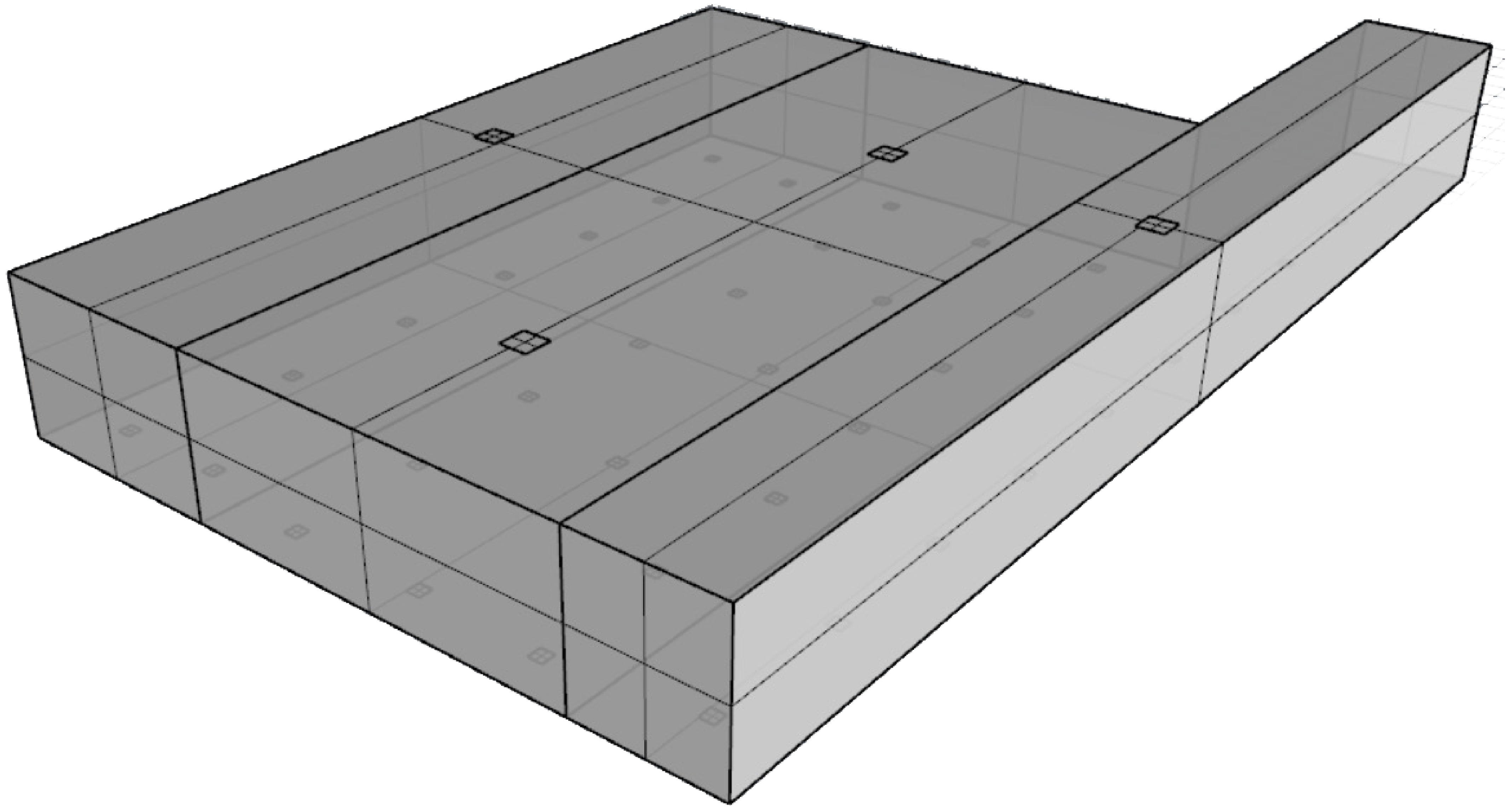
$$\min_{a,b,c,d} (|inlets| + |outlets|)\, PPDavg$$

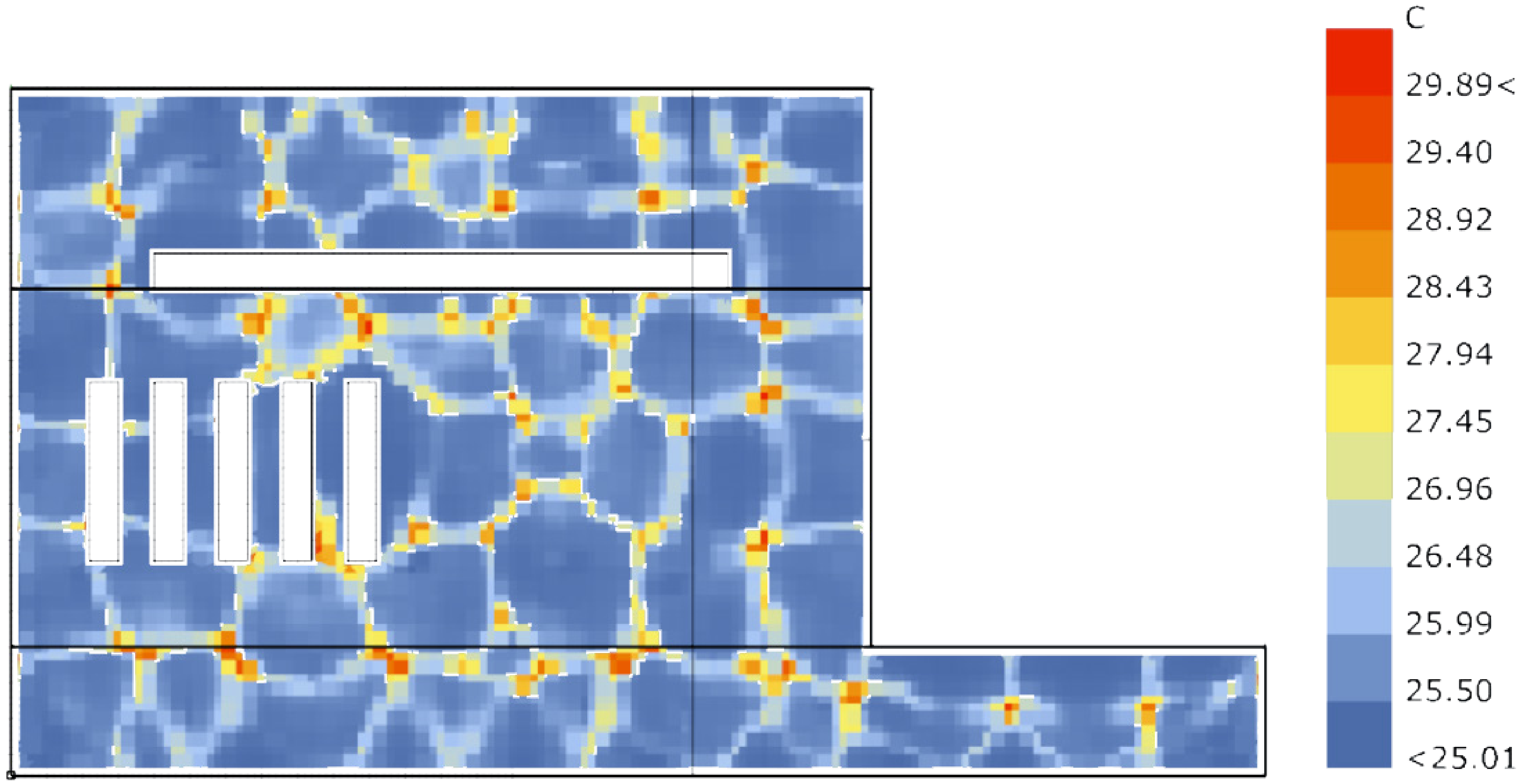
$$subject\ to: = \begin{cases} a \in [a_{min}, a_{max}] \\ b \in [b_{min}, b_{max}] \\ c \in [c_{min}, c_{max}] \\ T_{inlet} = d \\ q_{v,min} = e \end{cases}$$

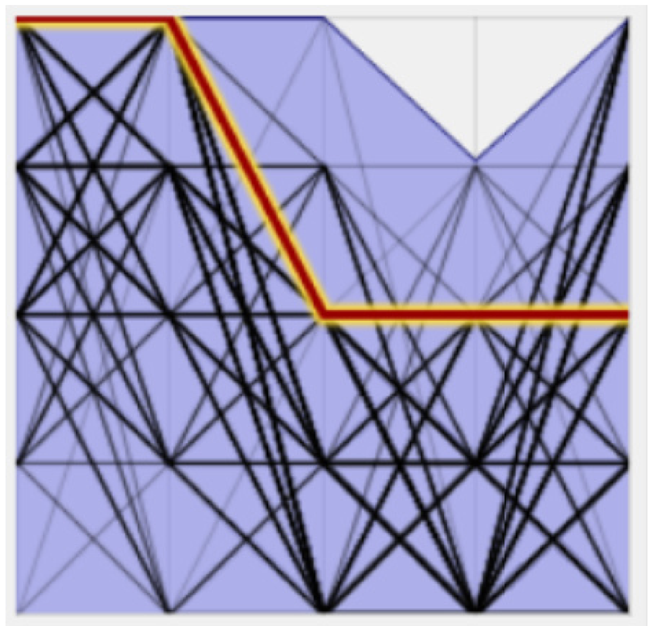


MODEL & CASE

'Best' options: high inlets speeds, few inlets





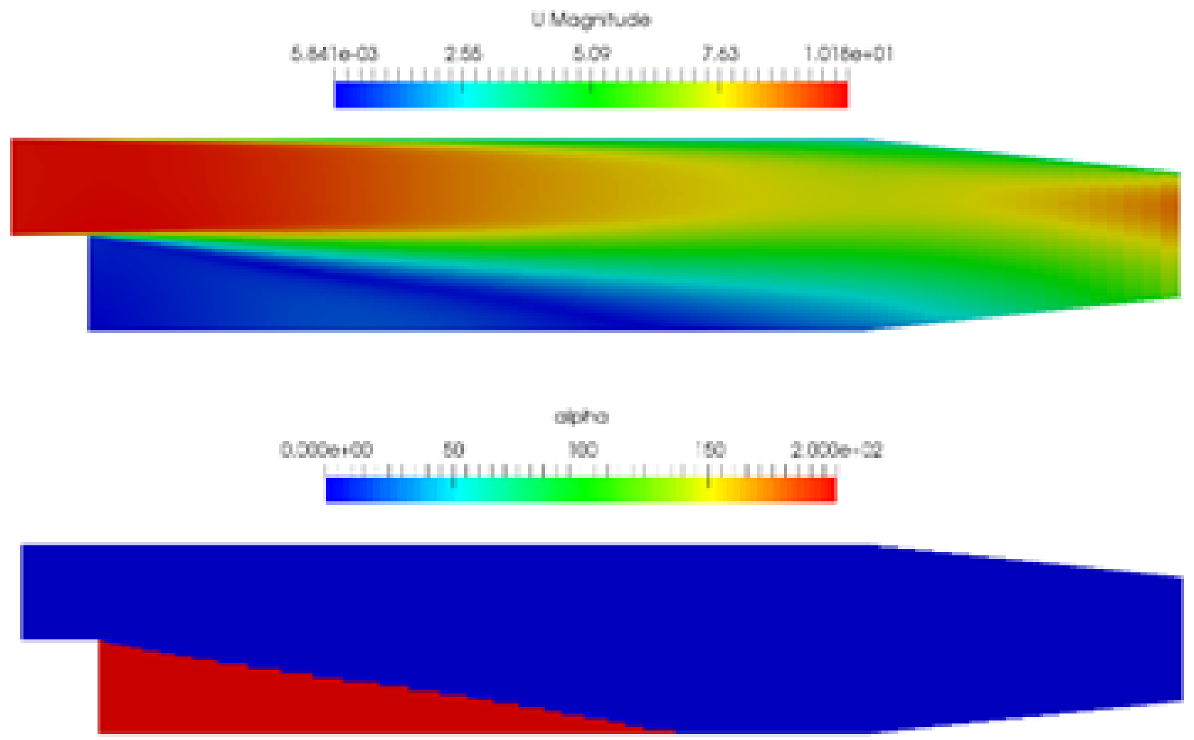


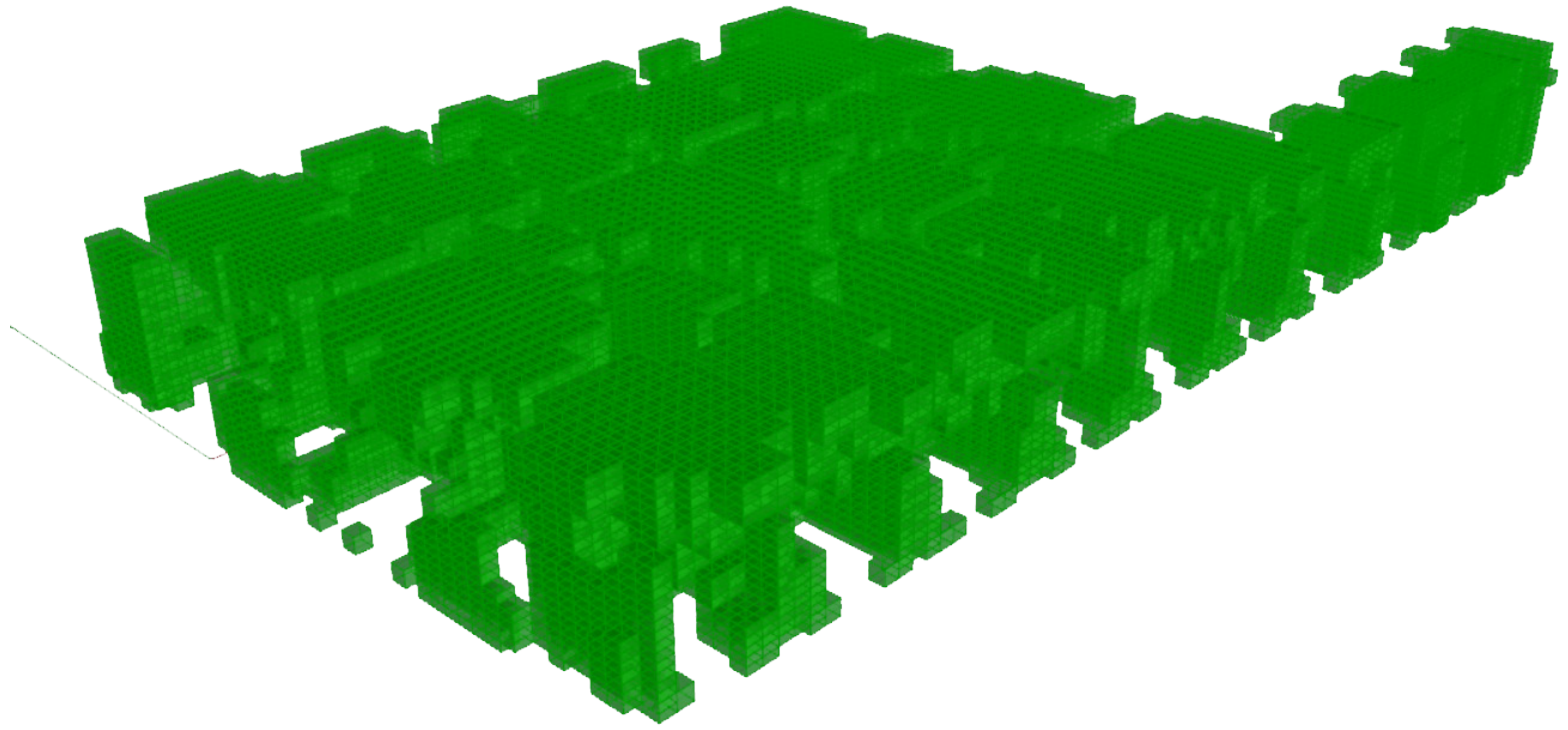
v_{inlet}: 1m/s
a_{inlet}: 3.24m2 (38 inlets)
a_{outlet}: 51.84m2 (4 outlets)
$h_{stack1,3}$: 20m
h_{stack2}: 20m

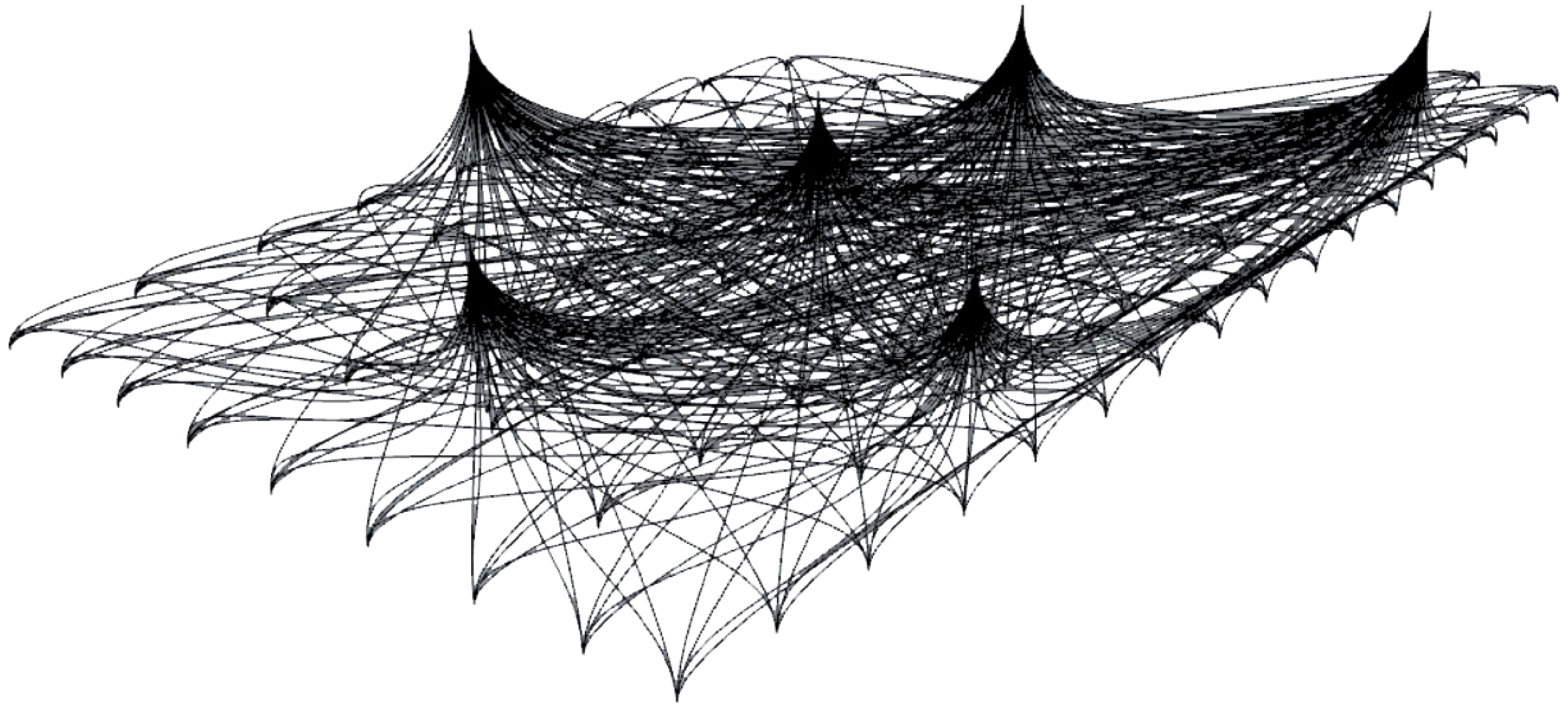
MODEL & CASE

Geometry Optimization

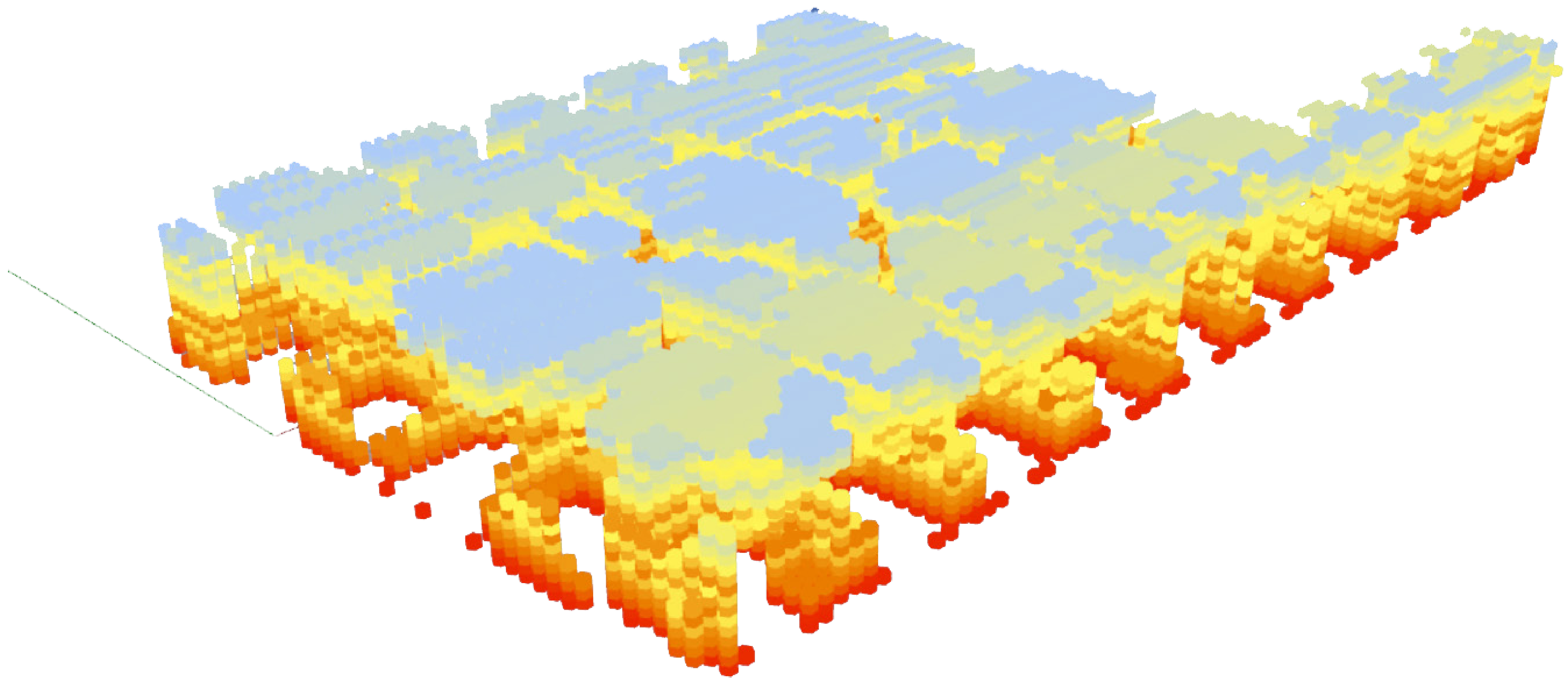
1. ***Adjoint shape optimization***
2. ***Voxel removal***
3. ***Bezier curves***



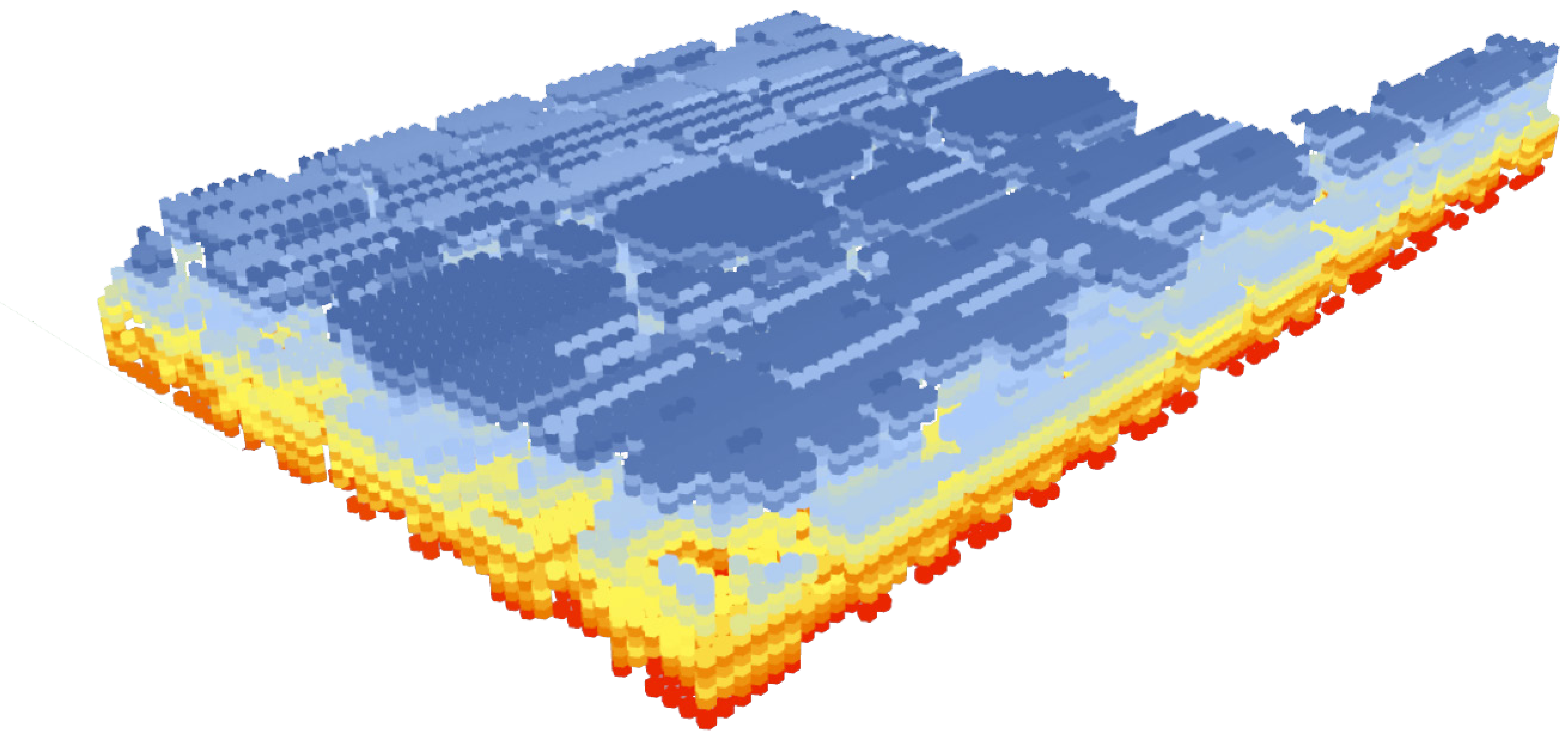




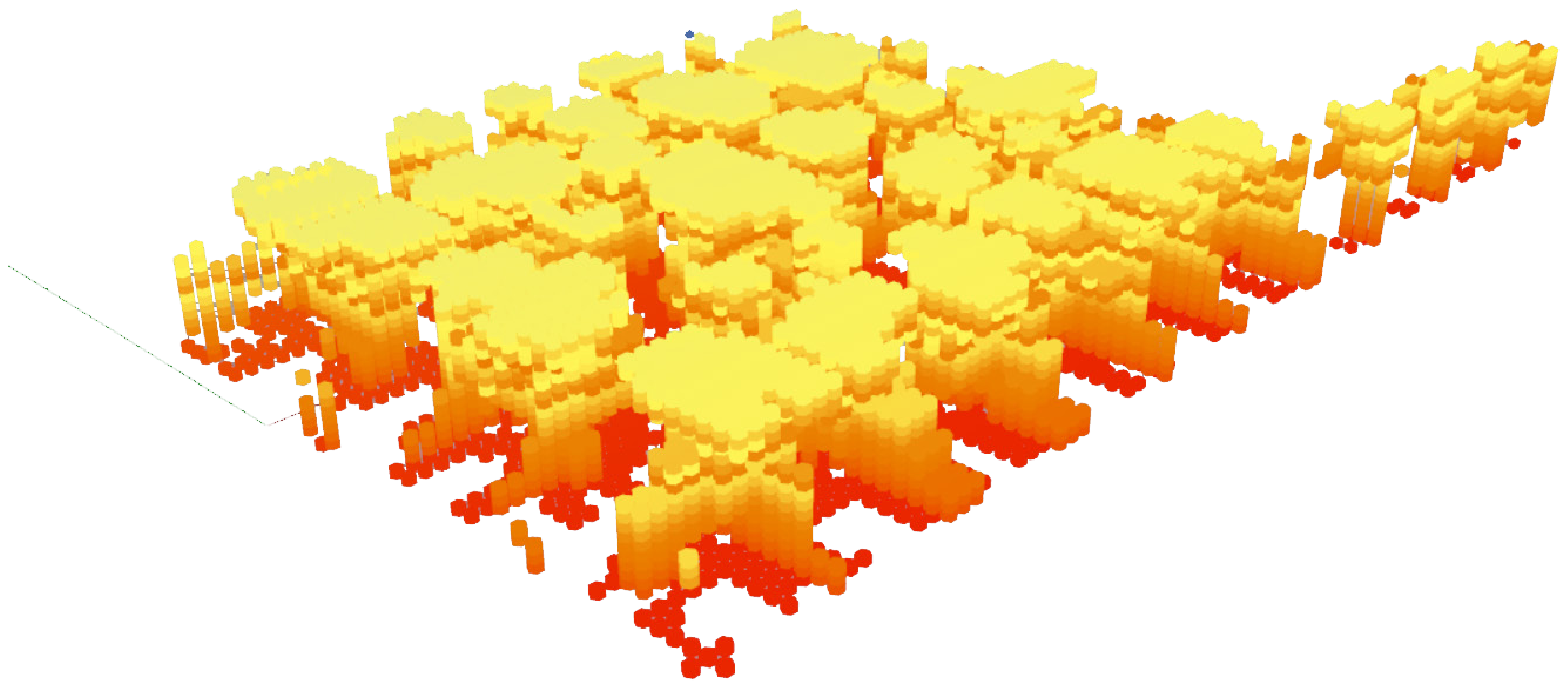
MODEL & CASE



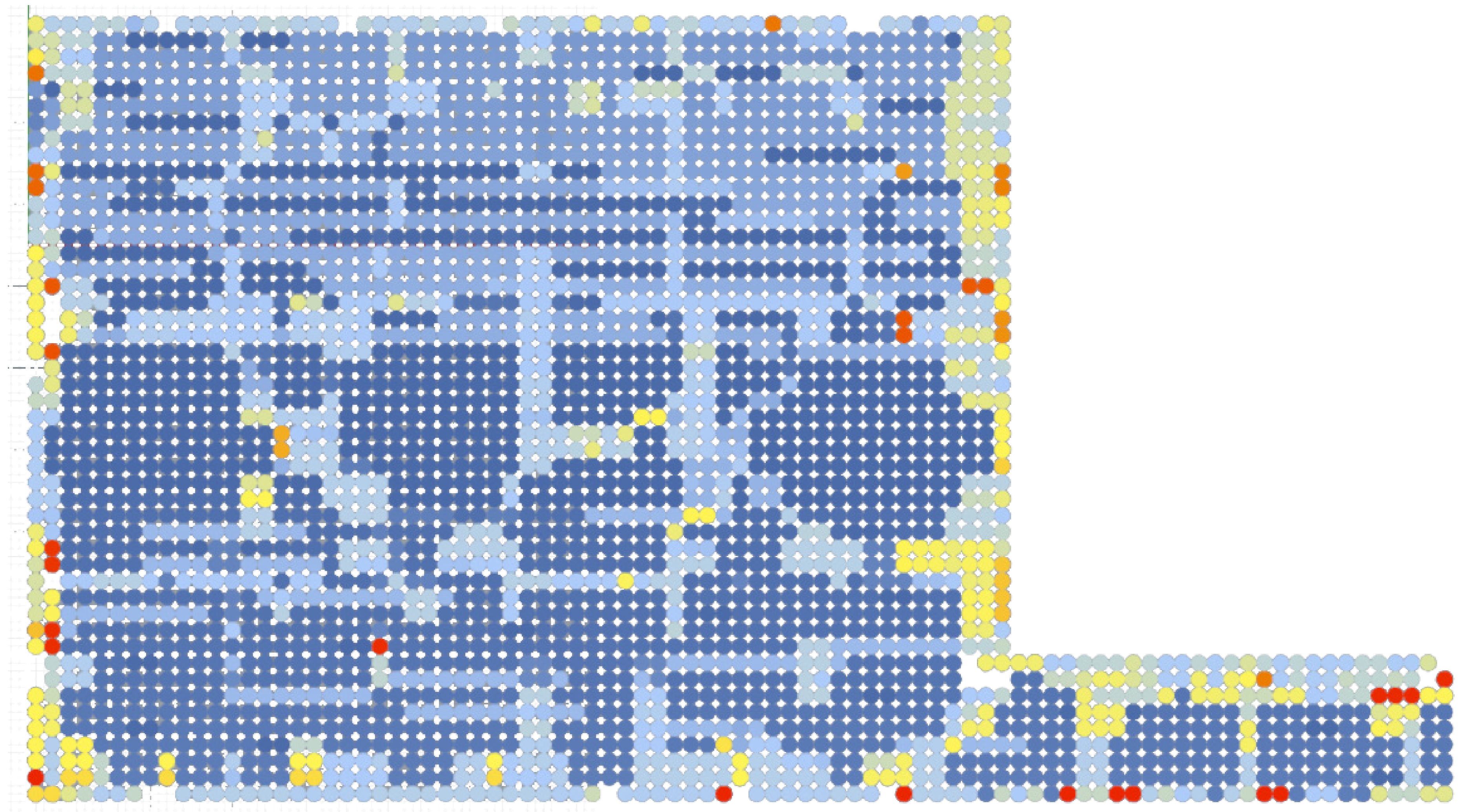
Cull voxels with negative Z component



Cull voxels with velocity < 0.25m/s

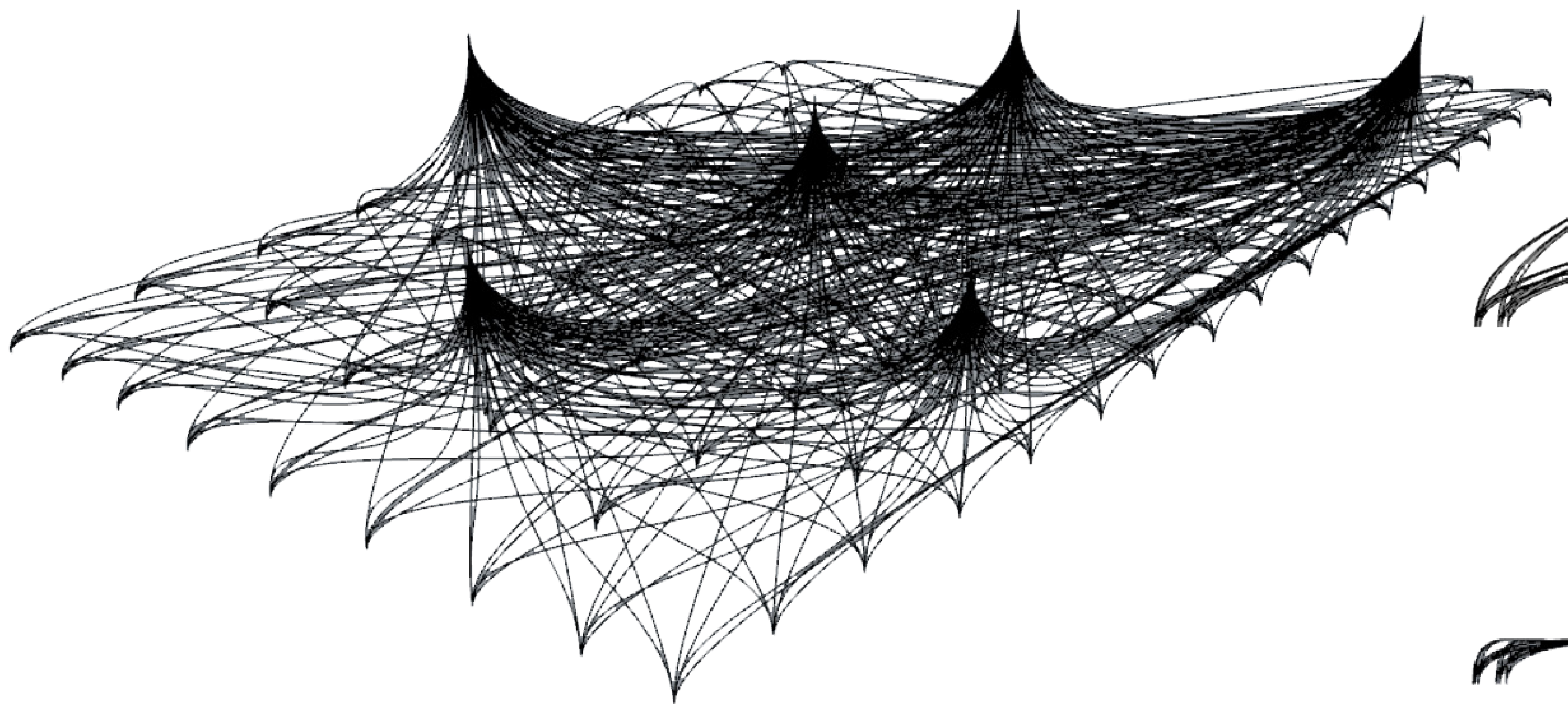


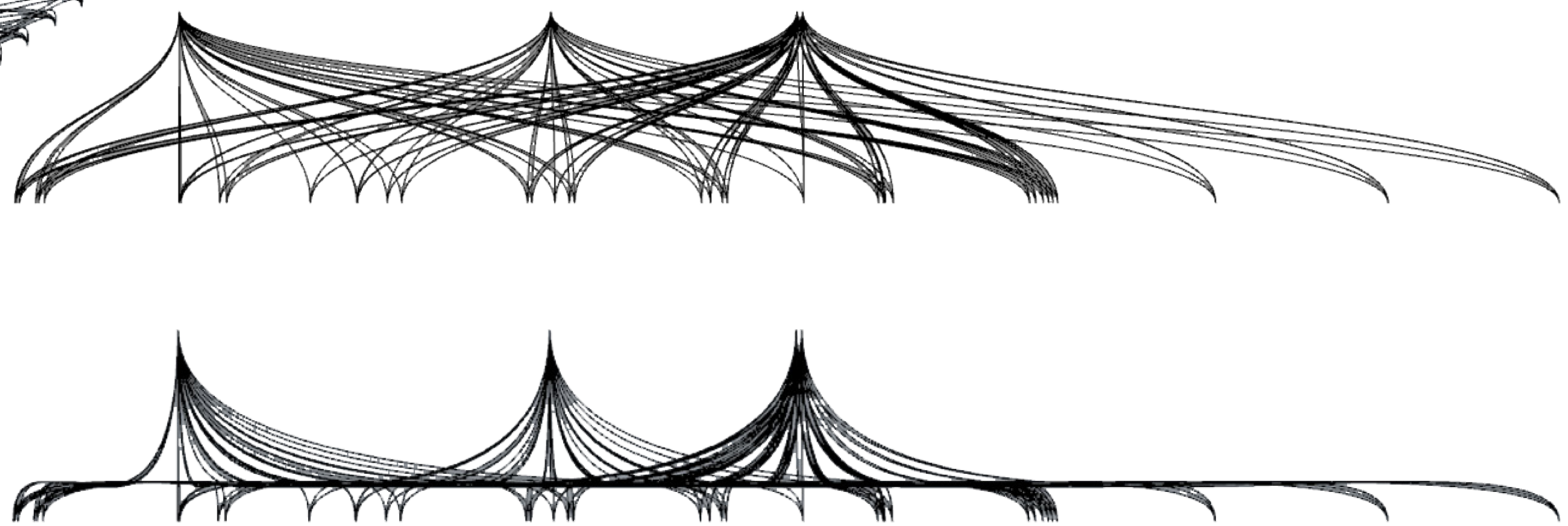
Cull voxels with velocity < 0.35m/s



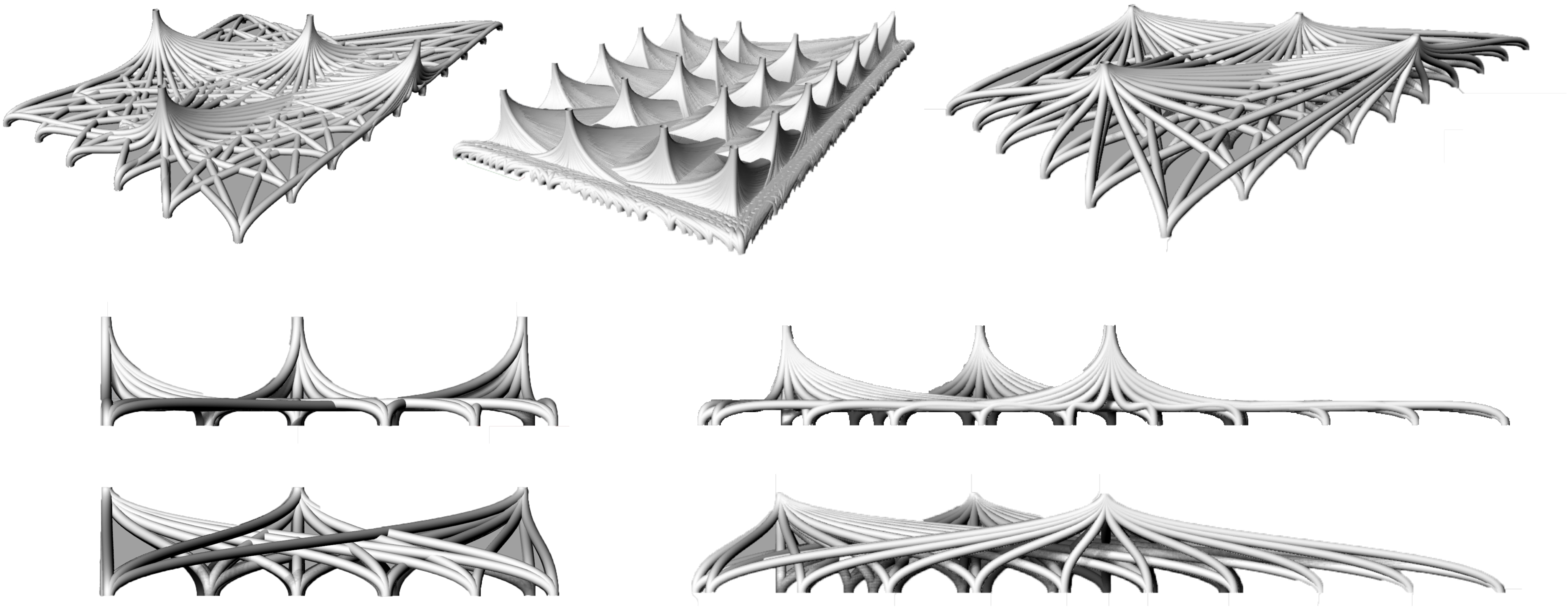
Cull voxels with velocity < 0.25m/s

MODEL & CASE



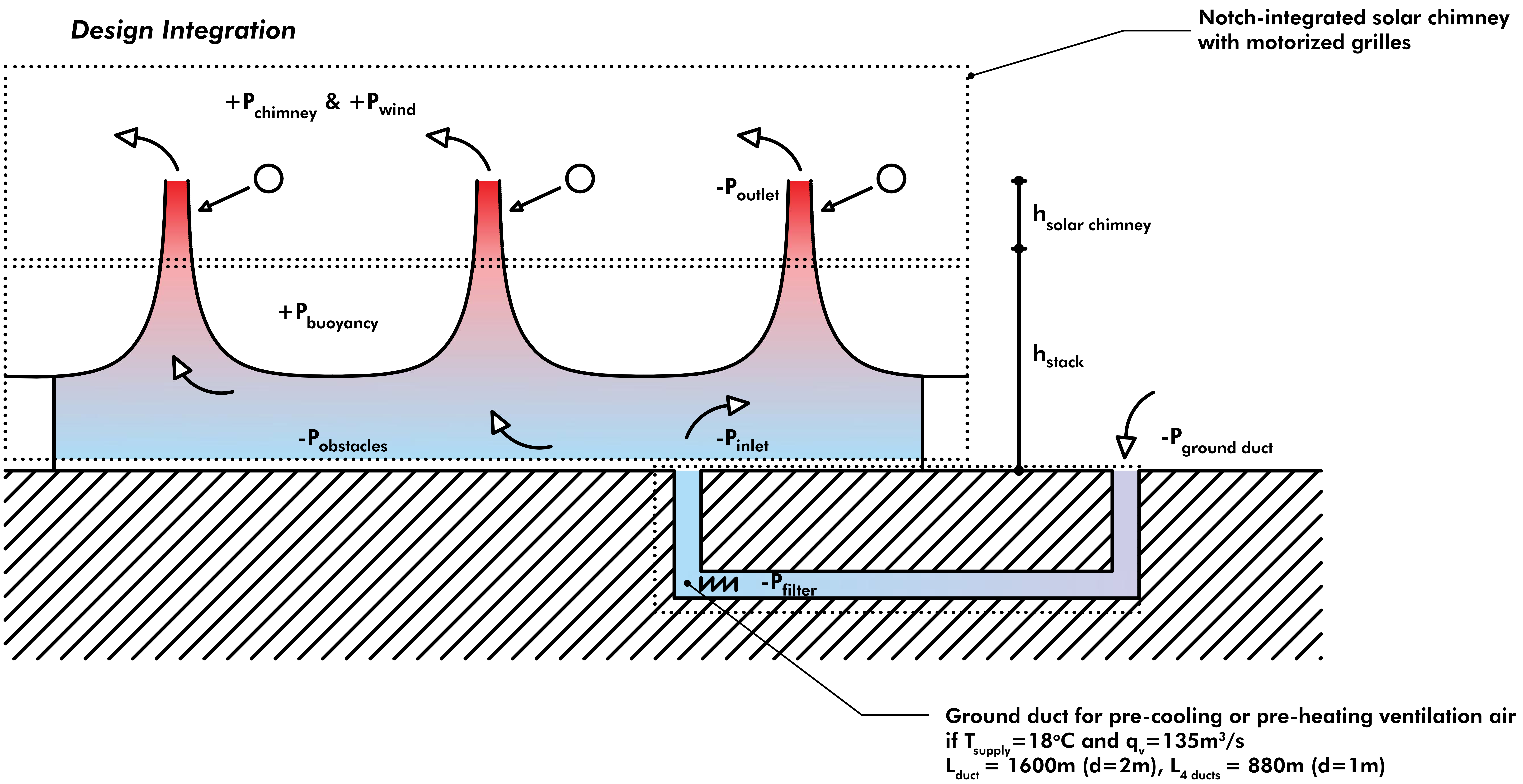


MODEL & CASE



MODEL & CASE

Design Integration



MODEL & CASE

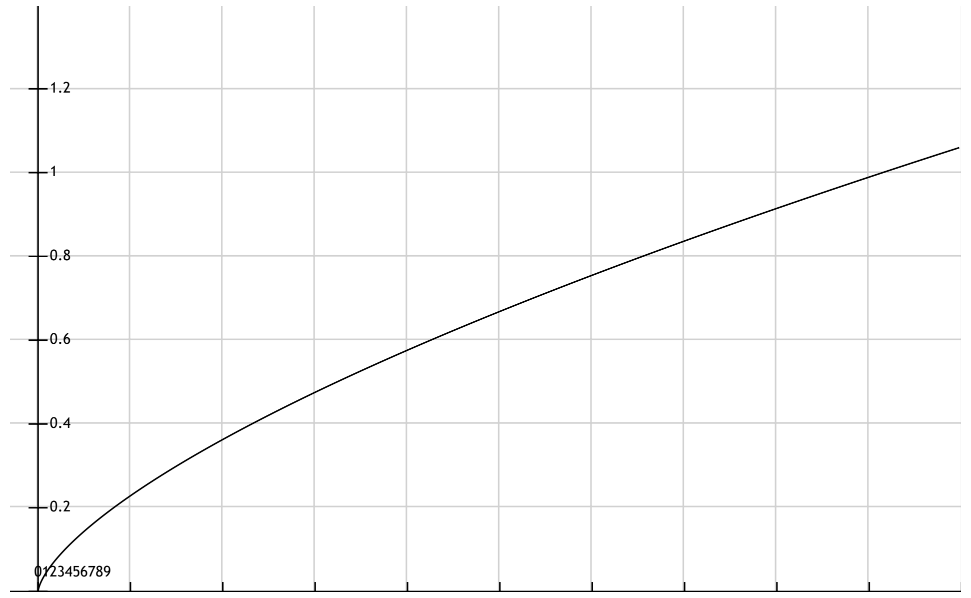
$$dP(h) = 0.22868\ h^{(2/3)}$$

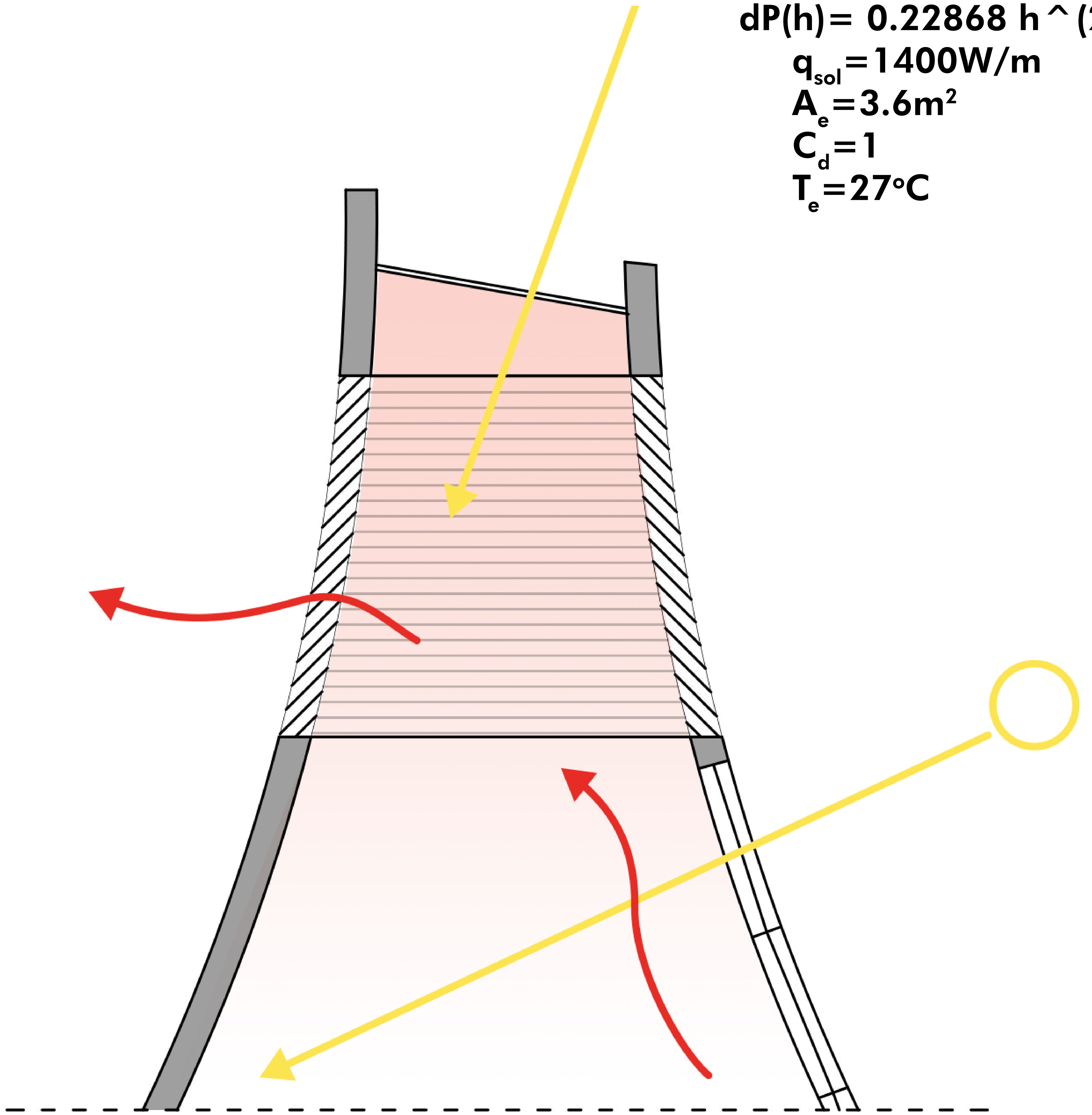
$q_{sol} = 1400 W/m$

$A_e = 3.6 m^2$

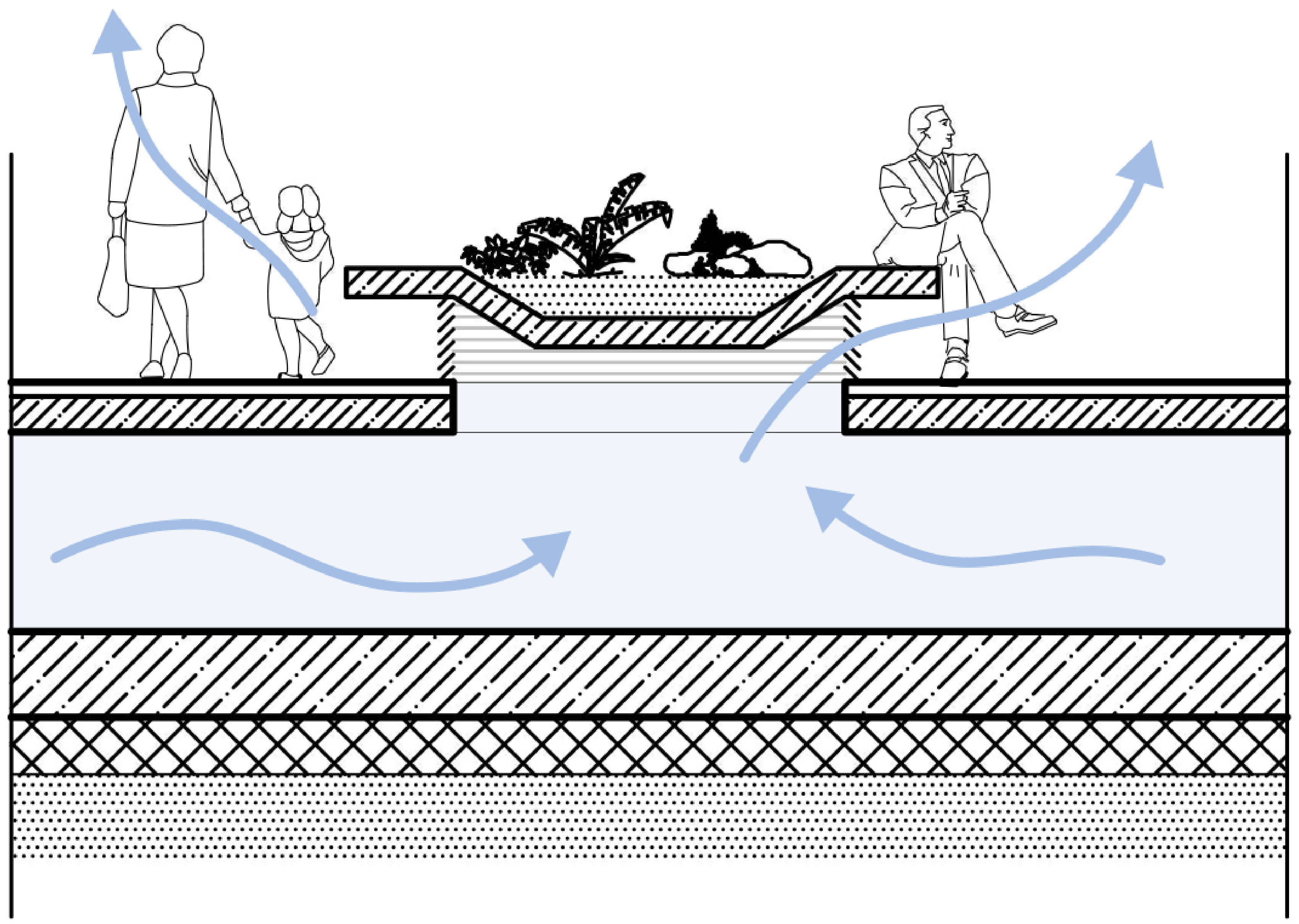
$C_d = 1$

$T_e = 27°C$



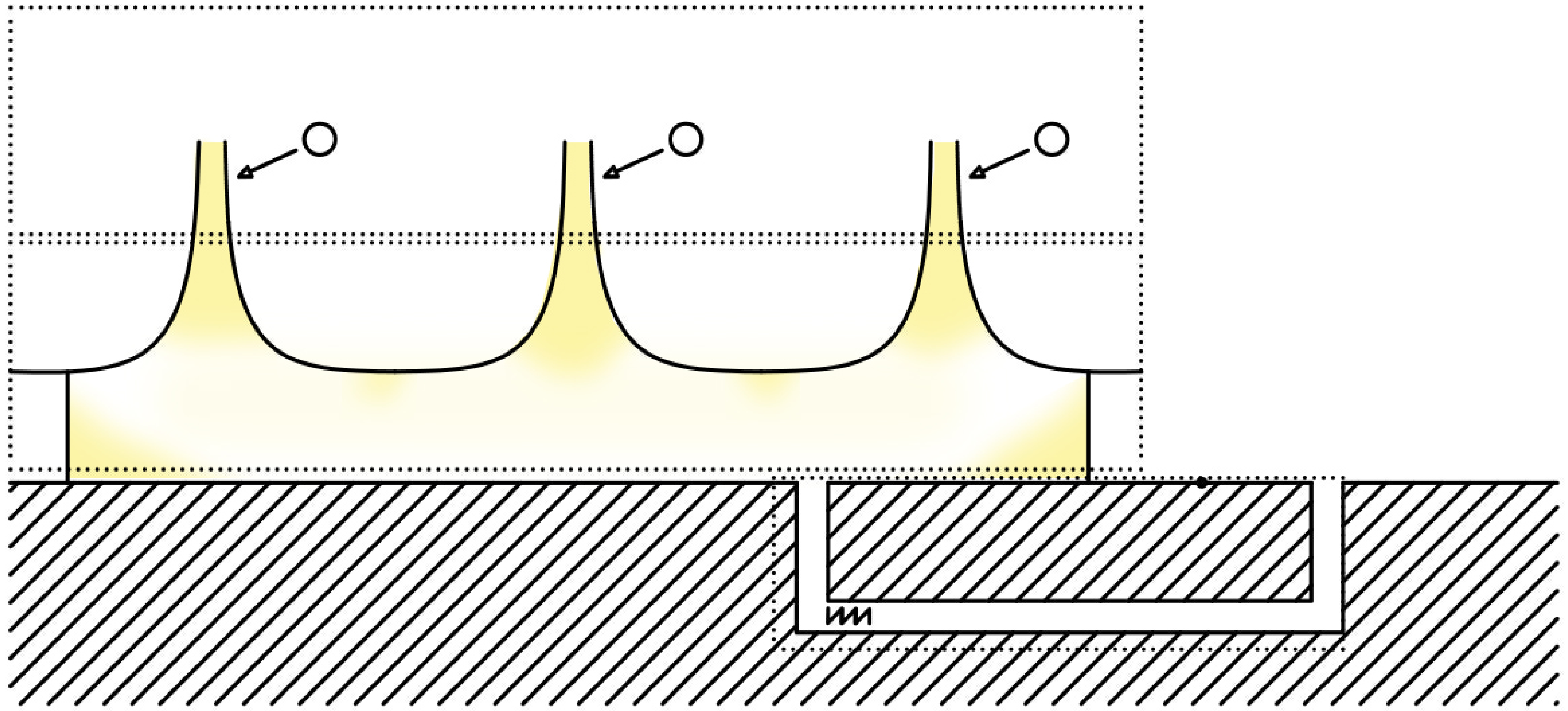


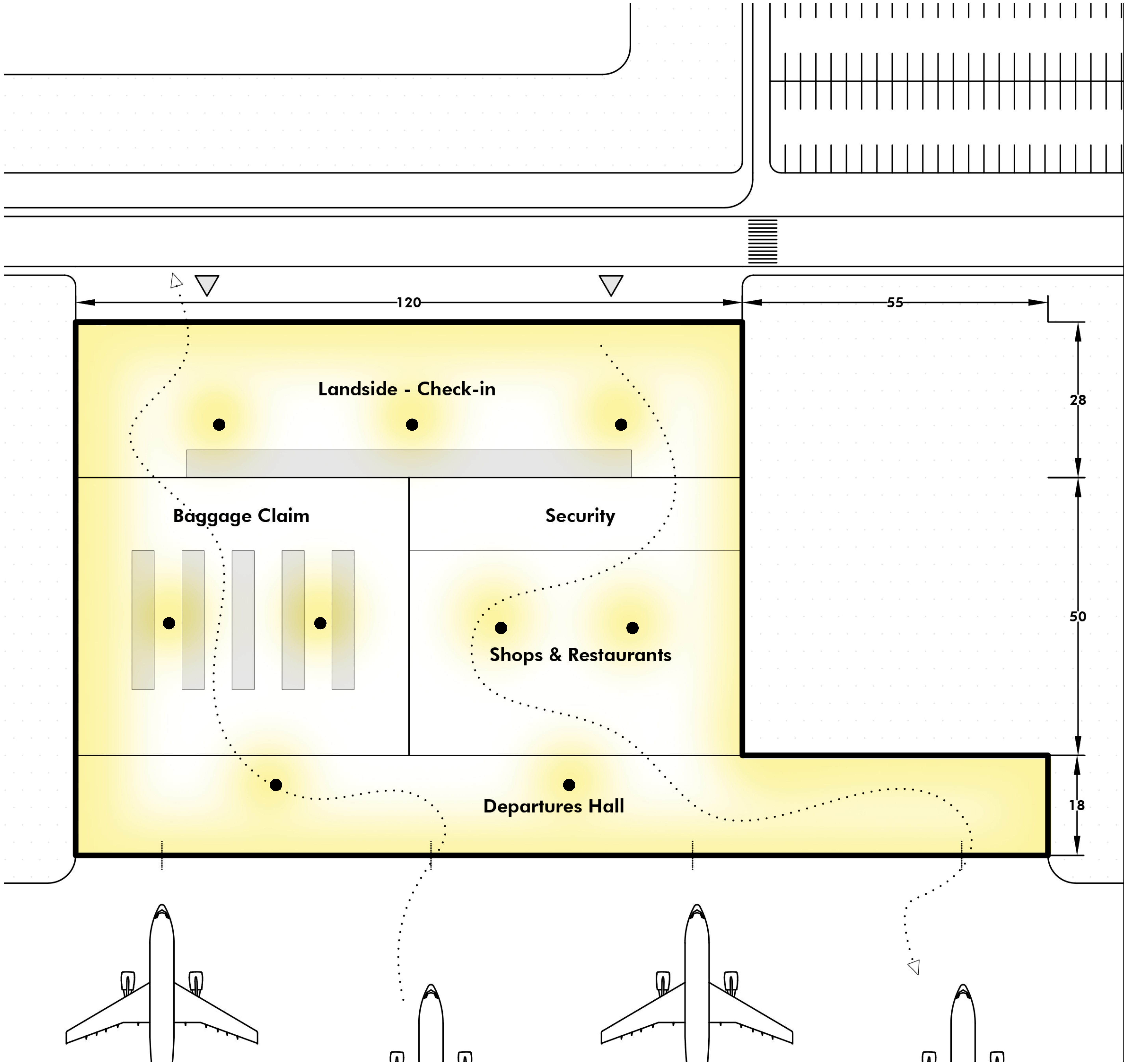
Notch of solar shimney with exhaust



Example of floor inlet with bottom supply plenum

MODEL & CASE




120
55
28
50
18
Landside - Check-in
Baggage Claim
Security
Shops & Restaurants
Departures Hall

CONCLUSIONS

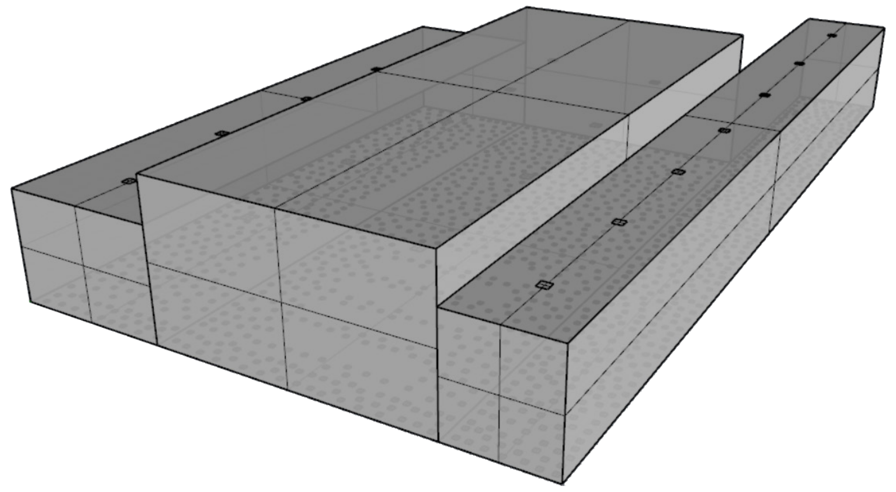
Computational model developed that allows for early-stage design optimization of naturally ventilated terminals

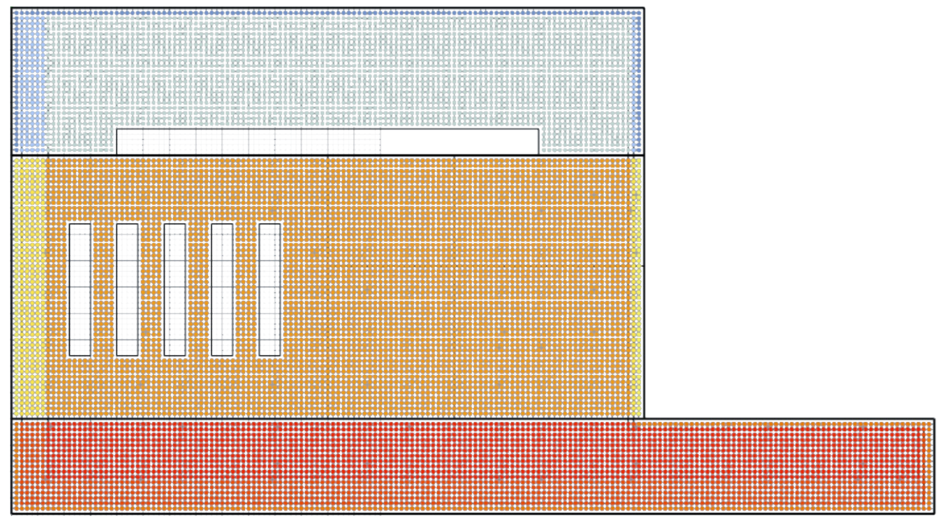
Model runs within a parametric RH/GH environment with Python coding and CFD analyses through OpenFOAM

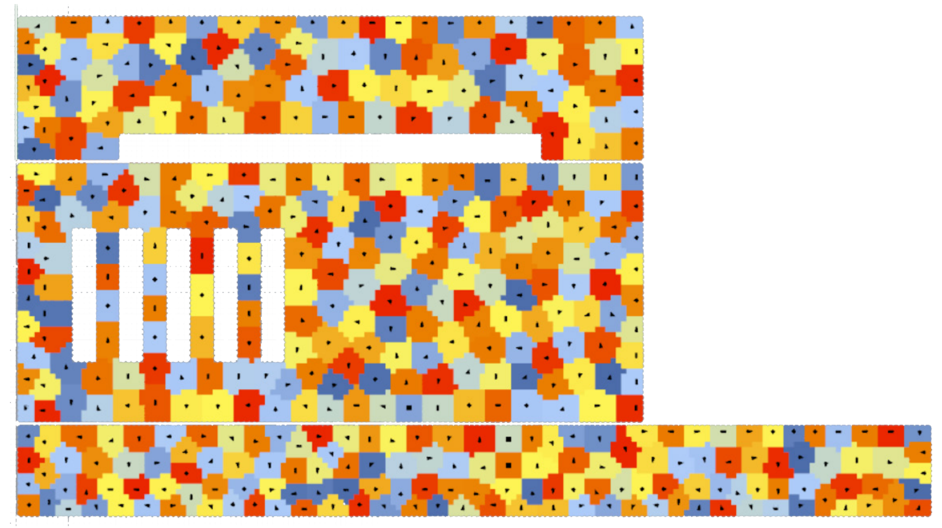
Air distribution parameters are optimized by a genetic algorithm with an aggregated objective

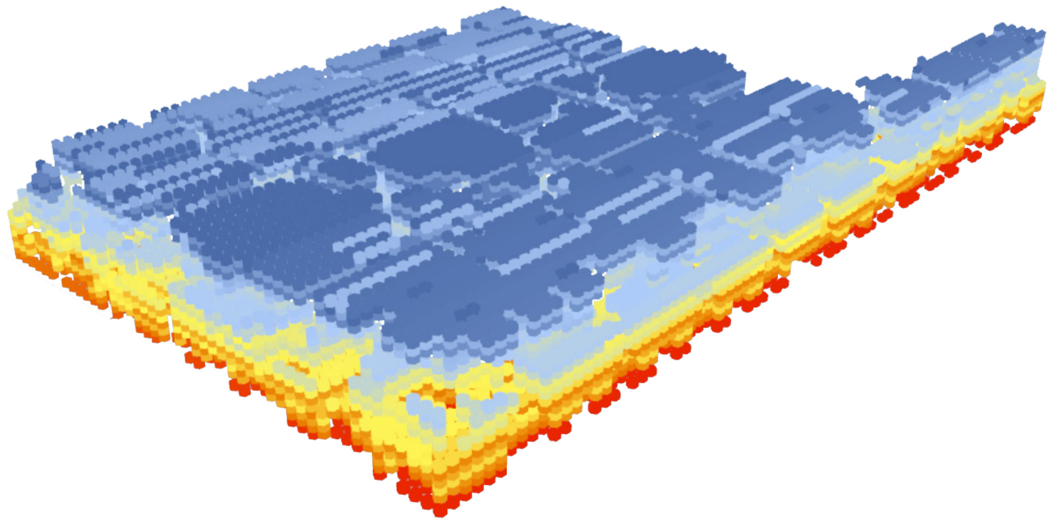
Optimal geometry is indicated by various geometric trials

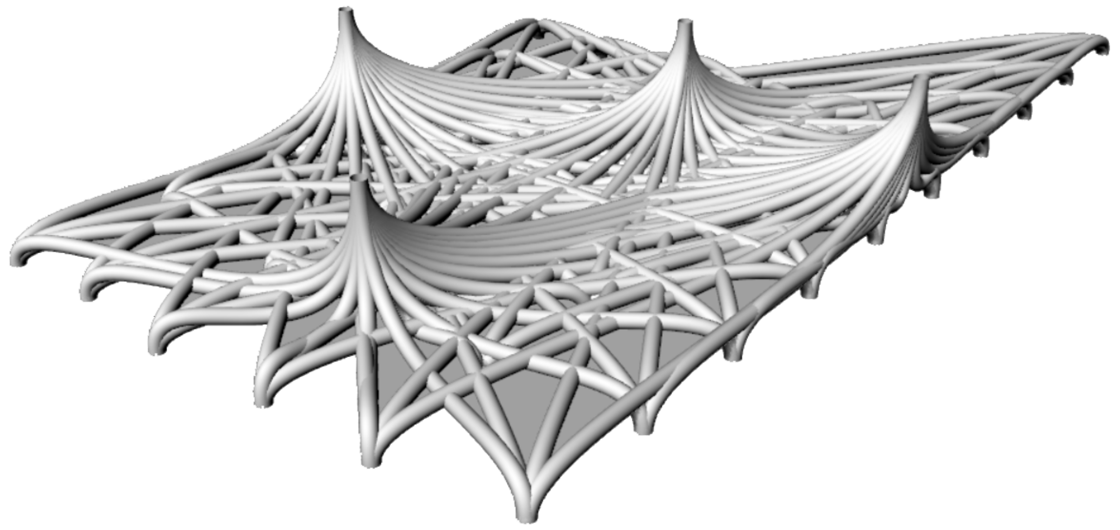
Optimizing geometry can lead up to 50% increased airflow











LIMITATIONS

bUgS!

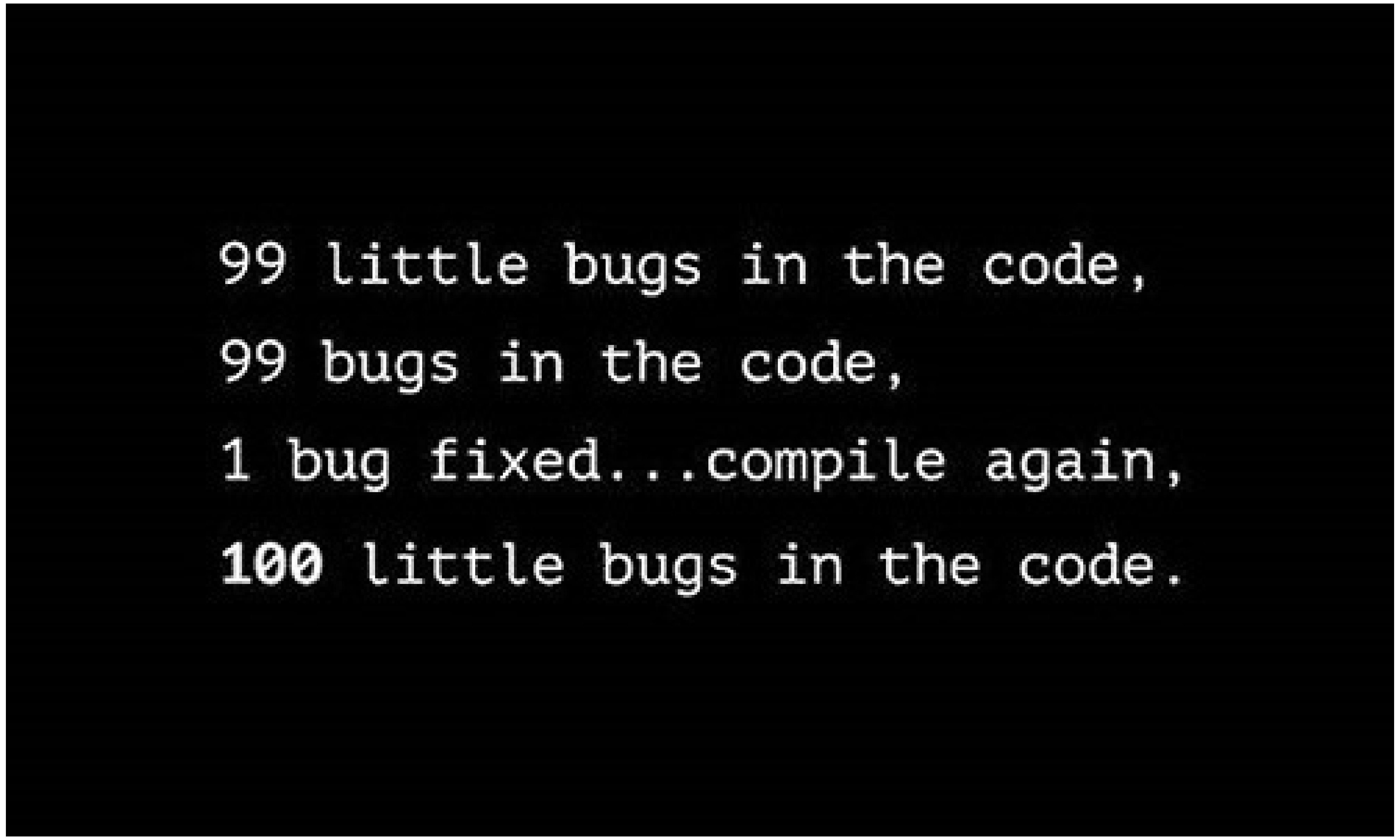
Integration between software suboptimal

Model is limited to lower levels of complexity in geometry

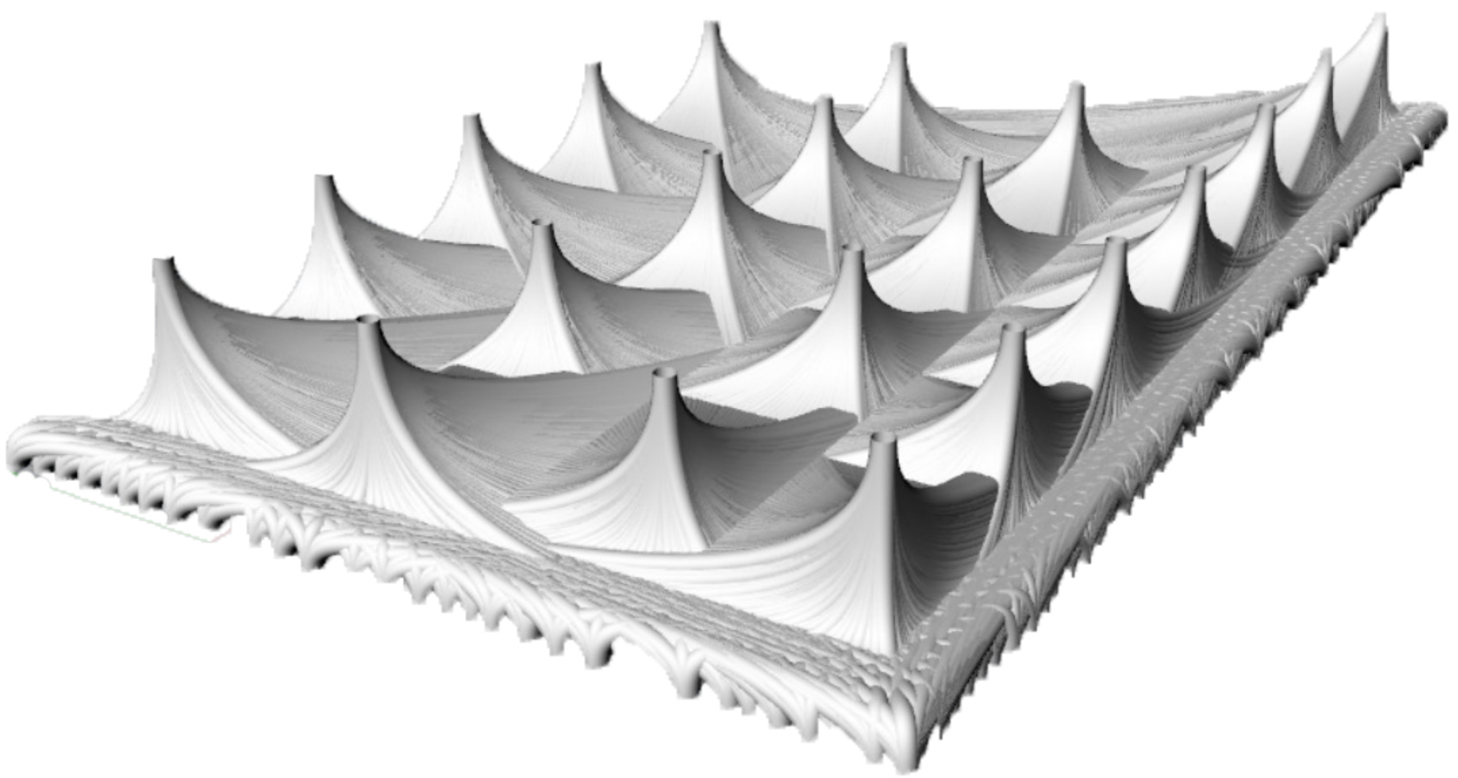
Integration of systems not included (chimneys, ground ducts, etc.)

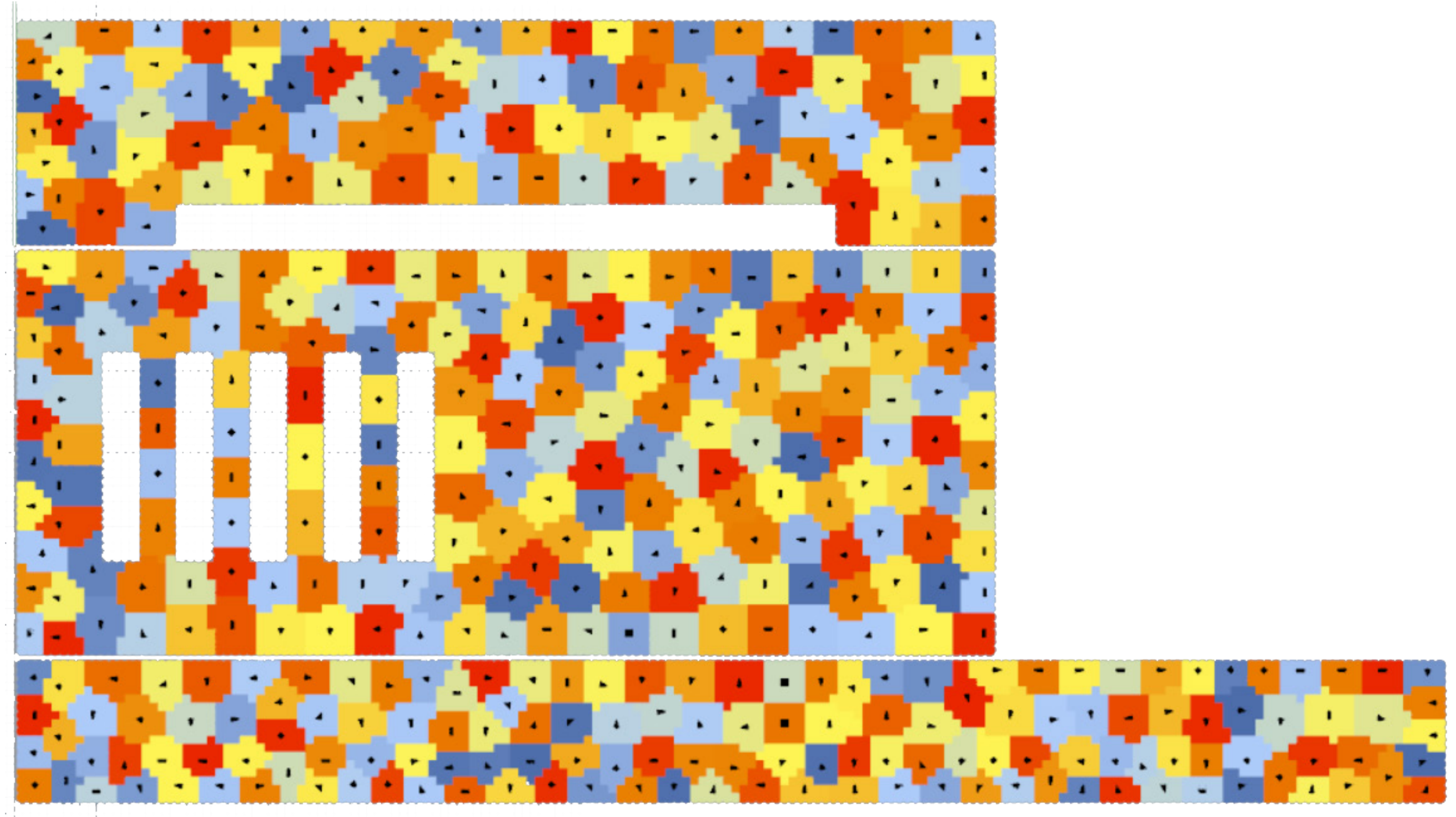
Objective function can be restated to penalize high temperature patches more

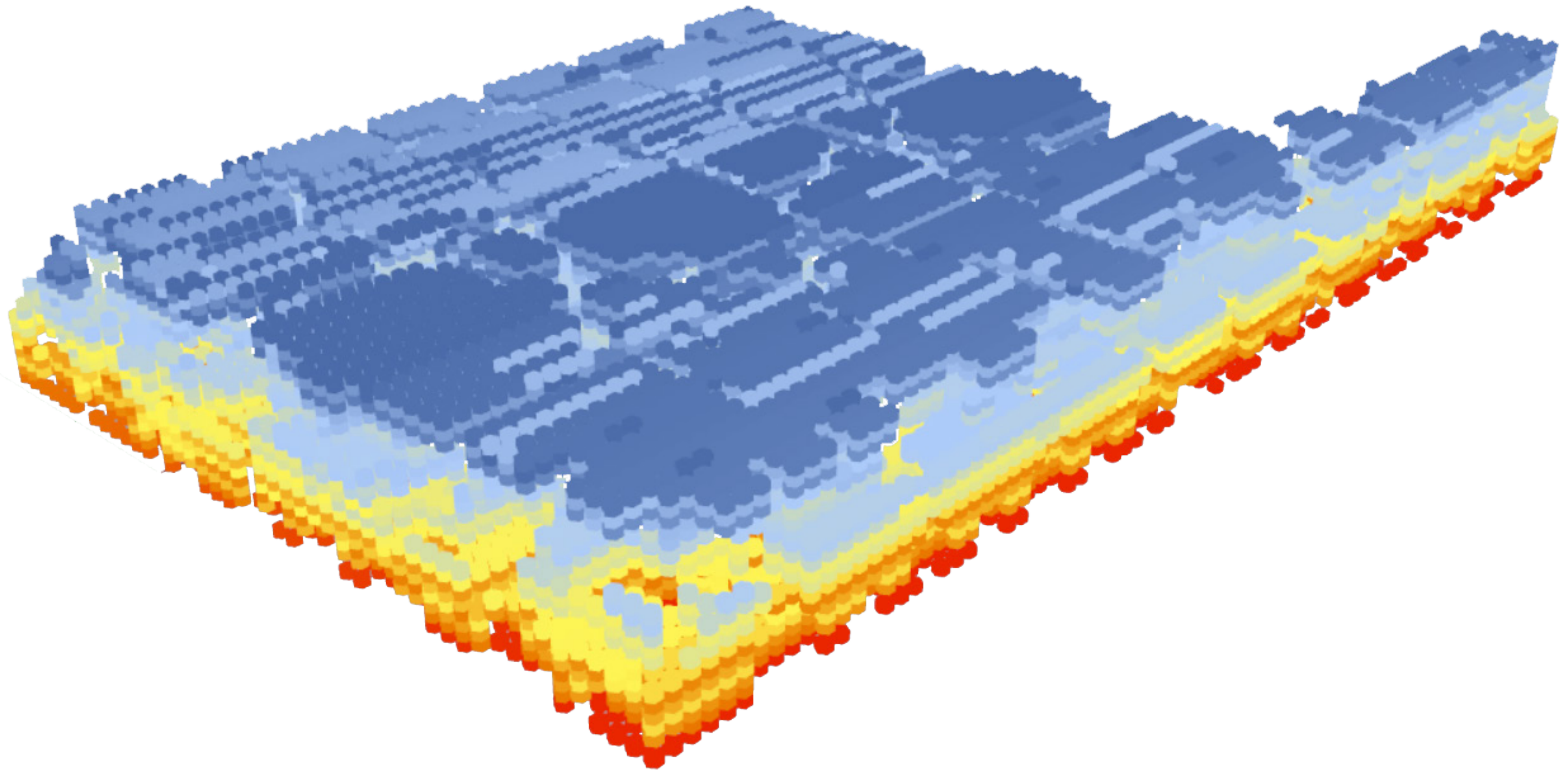
Combination of difficult topic and Solar Decathlon was large challenge

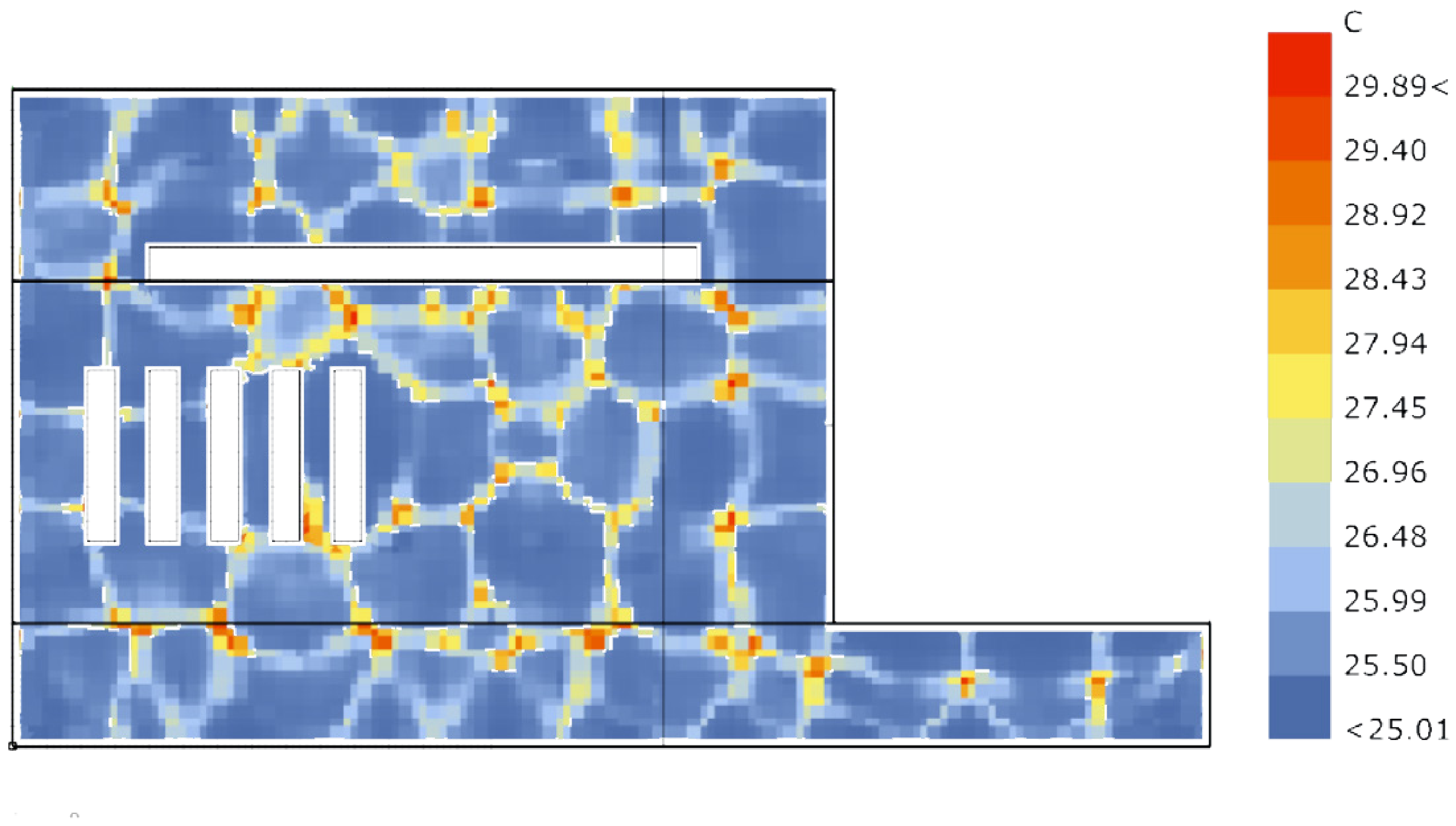


INTERMEZZO










MY KNOWLEDGE
SUFFICES