MSc SBDE Title Page

UCL Candidate Code: FBXX7

Module Code: BENV0088

Module Title: Integrated Building Systems Simulation

Coursework Title: Performance Analysis of Integrated Building Systems
Module Leader: Ivan Korolija (i.korolija@ucl.ac.uk)

Date: 05.05.2020

Word Count: 2560

By submitting this document, you are agreeing to the Statement of Authorship:

I certify that the attached coursework exercise has been completed by me and that each and every quotation, diagram or other piece of exposition which is copied from or based upon the work of other has its source clearly acknowledged in the text at the place where it appears.

I certify that all field work and/or laboratory work has been carried out by me/us with no more assistance from the members of the department than has been specified.

I certify that all additional assistance which I have received is indicated and referenced in the report.

Please note that penalties will be applied to coursework which is submitted late, or which exceeds the maximum word count. Information about penalties can be found in your Course Handbook which is available on Moodle: ***https://moodle-1819.ucl.ac.uk/course/view.php?id=11655&section=41***

- ***Penalties for late submission***
- ***Penalties for going over the word count***

In the case of coursework that is submitted late and is also over length, then the greater of the two penalties shall apply. This includes research projects, dissertations and final reports.

Contents

Introduction and context

To meet coursework learning outcomes and design objectives it was important to create parametric workflow that is flexible, validated and with easy to develop detailed systems. There are many building energy modelling (BEM) tools which typically are disconnected or centralized. "Instead of a single all-powerful tool or a disjointed set of tools to address our contemporary dilemma, a cohesive toolkit that seeks enhanced workflows between software might be far more effective." [1]. Ladybug tools [7] is a cohesive toolkit that through API's connects with multiple tools such as EnergyPlus (E+), OpenStudio (OS) and Radiance. It has all mentioned adjectives and is developed inside visual programming language - Grasshopper inside 3D modelling environment Rhino 6 and additionally it is free, open source with wide community of users.

Although Ladybug tools have many advantages, even detailed HVAC systems have implicit controls. In future OpenStudio will probably be connected to Modelica and functional mock-up interface through "Spawn-of-EnergyPlus" project [6] which will greatly enable integration with building automation systems with explicit controls.

Different construction features, schedules, setpoints and HVAC systems were explored to optimise an office building located in London that needed retrofit to yield good results. It was multi-objective optimisation consisted of carbon footprint and thermal comfort. Additionally, client required this office space to be class A and to include mechanical ventilation and air-conditioning with airflow per person no less than 12.5 l/s when occupants are present.

In this coursework 14 default systems were simulated, followed by sensitivity analysis. Next, detailed HVAC system was created with multiple controls being implemented and optimisation that used evolutionary algorithm.

United Kingdom's plans to decarbonise the grid [2] will probably change CIBSE TM46 coefficients in favour of electricity. Implementing solutions that use electricity give better potential for future as also implementing renewable energy sources on site to lower carbon footprint even more.

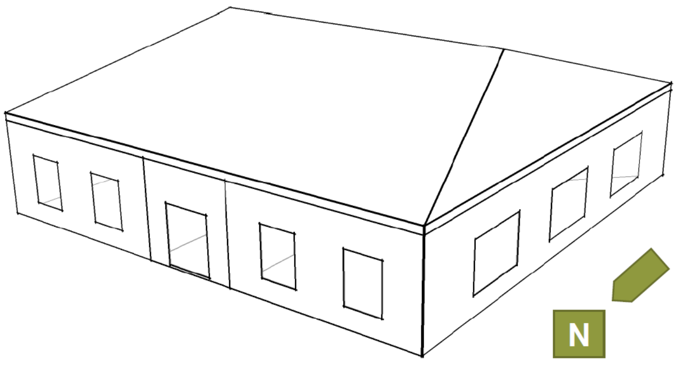


Figure 1 - Existing office building [Source: Coursework brief]

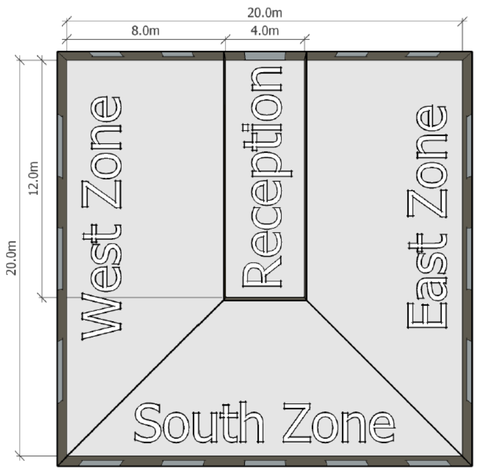


Figure 2 - Floor plan [Source: Coursework brief]

Methodology and assumptions

The methodology consists of four main parts:

1. Create and validate model results - Ladybug with EP-launch
2. Change model from step 1 and create batch of models by running simulations in parallel
3. Create script in python for sensitivity analysis to process data from step 2
4. Run evolutionary algorithm on parametric model and save iterations on website for client brief

To create all above stages, three plugins from Grasshopper Ladybug family were used: Ladybug mainly for weather data connection and visualisation, Honeybee for creating EnergyPlus and OpenStudio models [11], Ironbug for detailed HVAC systems creation [12]. Additionally, python programming language with SALib library for sensitivity analysis [5], Colibri package from TT Toolbox plugin [13] for DesignExplorer website for client briefing [10] and Octopus grasshopper plugin for evolutionary optimisation [14].

1. Create and validate model results - Ladybug with EP-launch

Model's geometry, materials and construction was imported to Ladybug from the ExistingBuilding.idf model. Other assumptions were translated to the Ladybug environment.

Imported idf file had five zones where attic's boundary representation (brep) was opened. Honeybee defines geometries and necessary information for energyplus to create Honeybee zones. It is not possible to run simulation with opened geometry. Attic did not have floors in the idf file, four floor surfaces were added to the original idf file and imported once again.

Most of the parameters were incorporated with the simple nodes, but radiant fraction for people, light and equipment gains cannot be manipulated with the built-in nodes. OpenStudio measure was added with ruby code to include radiant fraction for gains, which unfortunately extended simulation time. Most of the radiant fraction measure code was developed by Chris Mackey [15]

Detailed HVAC system - baseboard units with its parameters were incorporated using Ironbug to create corresponding model with the same detailed HVAC system in the Ladybug toolkit.

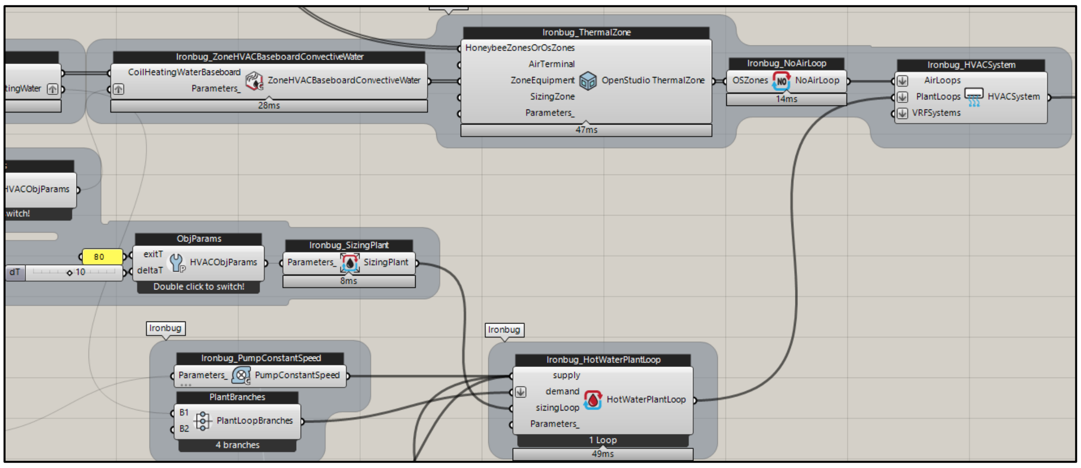
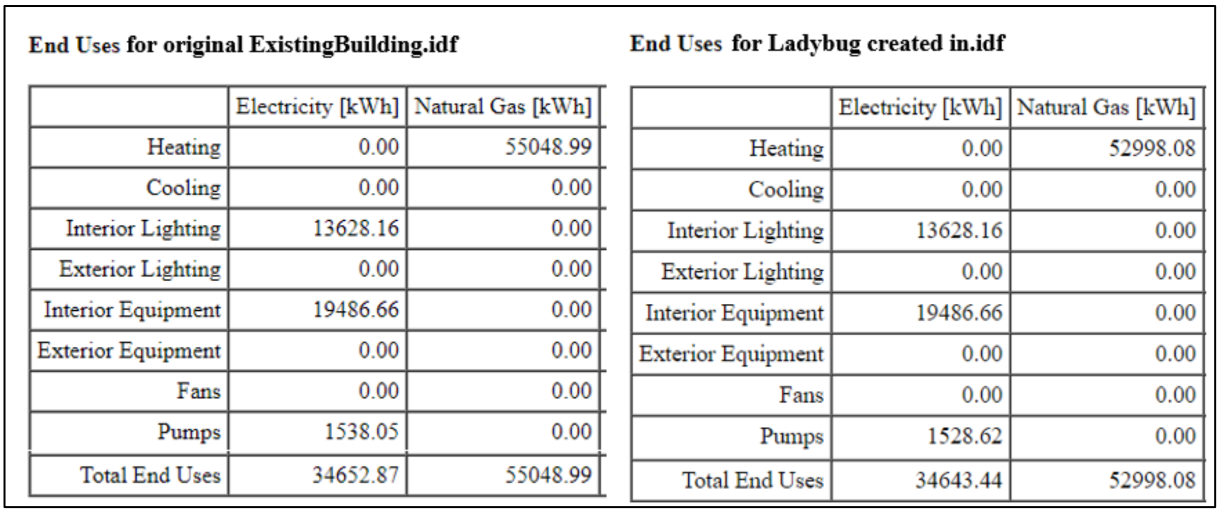


Figure 1 - Fragment of detailed system developed in Grasshoper with plugin Ironbug

As both models use the same engine EnergyPlus 8.9 they should yield the same results. Matching parameters was left at point when difference was below 5%, as it was important to progress with the coursework and results were already similar to a certain extent.



End Uses for original ExistingBuilding.idf

	Electricity [kWh]	Natural Gas [kWh]
Heating	0.00	55048.99
Cooling	0.00	0.00
Interior Lighting	13628.16	0.00
Exterior Lighting	0.00	0.00
Interior Equipment	19486.66	0.00
Exterior Equipment	0.00	0.00
Fans	0.00	0.00
Pumps	1538.05	0.00
Total End Uses	34652.87	55048.99

End Uses for Ladybug created in.idf

	Electricity [kWh]	Natural Gas [kWh]
Heating	0.00	52998.08
Cooling	0.00	0.00
Interior Lighting	13628.16	0.00
Exterior Lighting	0.00	0.00
Interior Equipment	19486.66	0.00
Exterior Equipment	0.00	0.00
Fans	0.00	0.00
Pumps	1528.62	0.00
Total End Uses	34643.44	52998.08

Figure 2 - End use tables for both models

Objectives were calculated with regard to current location and building type. Carbon footprint was calculated with national standard CIBSE TM46 coefficients. [16]

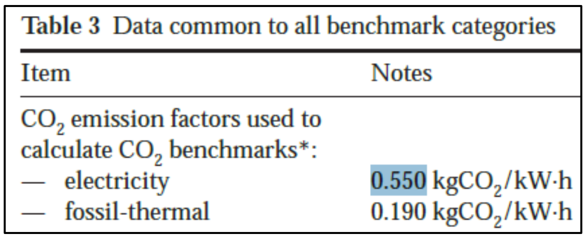


Table 3 Data common to all benchmark categories

Item	Notes
CO_2 emission factors used to calculate CO_2 benchmarks*:	
— electricity	0.550 $kgCO_2/kW{\cdot}h$
— fossil-thermal	0.190 $kgCO_2/kW{\cdot}h$

Figure 3 - CO_2 emmission factors from CIBSE TM46 [16]

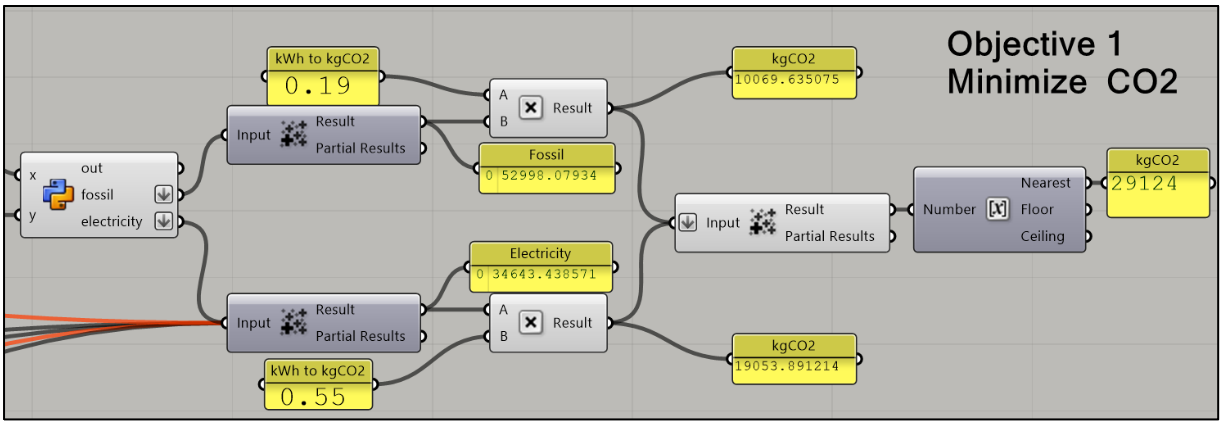


Figure 4 - Carbon footprint objective calculations

Thermal comfort objective was calculated with Fanger PMV model and ASHRAE standard 55 [4] as a function of the lowest percentage of comfortable time during occupancy from all zones, where comfortable time is when a predicted percentage of people dissatisfied (PPD) was below 10%. Input parameters for PMV model such as: clothing, wind speed, metabolic rate and occupancy hours were taken from original idf file and weren't changed. Seperate PMV model from Ladybug was used rather then unmet hours from EnergyPlus for better control and results analysis.

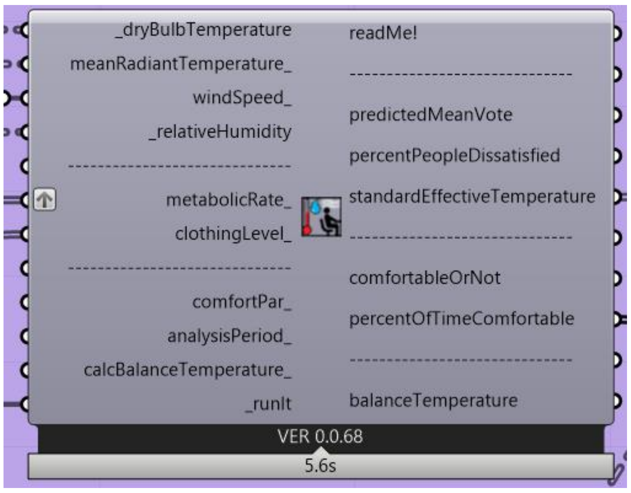


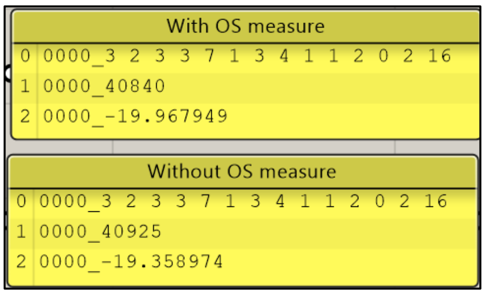
Figure 5 - Ladybug object for running PMV model

2. Change model from step 1 and create batch of models by running simulations in parallel

To prepare data for sensitivity analysis it is important what kind of data do we need and how much. Morris method was chosen as the one already known from the previous coursework. In the beginning I wanted to perform sensitivity analysis for each system to see which parameter influence which system more and how do they corelate with one another. As one system needs 140 model inputs and outputs in total, I needed 1960 models. With Morris method it is good practice to have at least 10*(k+1) model inputs and outputs, where number of parameters k = 13 (7 construction features and 6 schedules) and number of different HVAC systems was n=14 we have 1960 models in total. For detailed information about each parameter and corresponding bounds look appendix 2.

$$n * 10 * (k + 1) = 14 * 10 * (13 + 1) = 14 * 140 = 1960$$

OpenStudio measure with radiant fraction was not included in sensitivity analysis to lower time for generating each idf file. Model inputs were generated with random sample to create batch of idf files to be later simulated in parallel using multiple threads. Each system had the same set of inputs to lower the possibility of favouring randomly one system over another. Minimum fresh air requirement was added to this analysis - 12.5 l/s of fresh air per person when occupied.



```
With OS measure
0 0000_3 2 3 3 7 1 3 4 1 1 2 0 2 16
1 0000_40840
2 0000_-19.967949

Without OS measure
0 0000_3 2 3 3 7 1 3 4 1 1 2 0 2 16
1 0000_40925
2 0000_-19.358974
```

Figure 6 - Single model with and without OS measure.

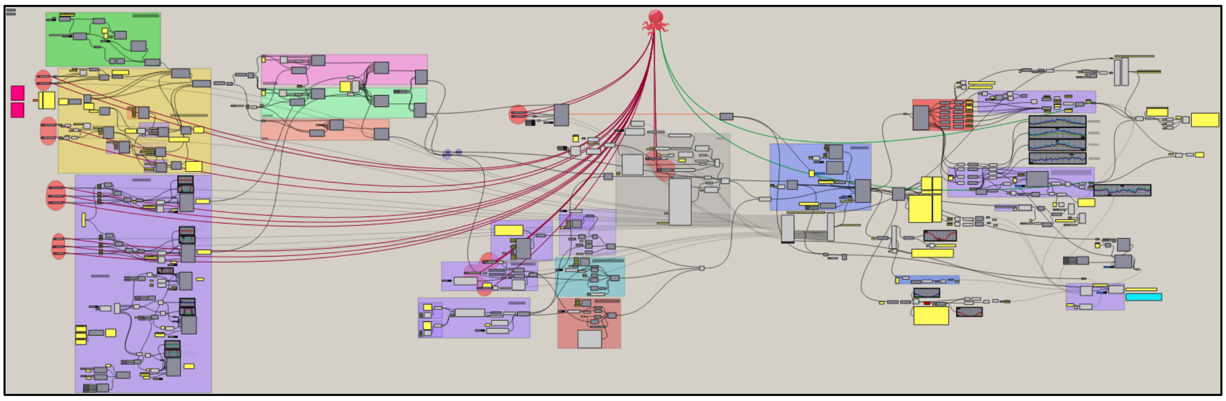
Each iteration had set of inputs and outputs which were saved to panel and later exported to text file. Due to issues probably related to ram capacity generated list of inputs did not match list of outputs. The whole batch could not be used for sensitivity analysis as lists were created separately and could not be matched. For next result generation, each input and output received ID number, which corelated inputs to the outputs (zero – input parameters, first - carbon footprint [kgCO2] and second - thermal comfort, percentage of comfortable time [%]).ID number was added before the value, in this case 0000.

3. Create script in python for sensitivity analysis to process data from step 2

Data generated in Ladybug and Honeybee was saved to text files to be later analysed with python scripts. The basis for the scripts were taken from GitHub repository [5] and customized to serve the purpose of this analysis. Two python scripts were created, first for each system with 140 models and second script for all systems with 1960 models. Python scripts are included in the appendix 3 and 4. In total for each objective 15 SA graphs were created.

4. Run evolutionary algorithm on parametric model and save iterations on website for client brief

Single system was chosen after comparing results from step 2 and 3. Optimisation concerned only radiant system with DOAS. Additionally to previously explored 14 parameters from step 2 multiple additional improvements and parameters for the system were incorporated such as heat source replacement from gas boiler to heat pump with vertical ground source loop, DOAS heating gas coil changed to electrical, DOAS dual-setpoints for supply air, changing supply water setpoint both for heating and cooling water loops, changing loops maximum length, pressure head on fans, radiant control throttling range. Each parameters input was controlled with number slider and outputs were converted to numbers. Model created this way was ready for optimisation with Octopus plugin. Octopus uses evolutionary algorithm to find best solutions. [6]



Above picture represents 3rd and last grasshopper script where Octopus connects to each number slider with **dark pink** wires and objectives are connected with **green** wires. Grasshopper scripts and high-resolution screen captures are provided with other supplementary files.

Analysis and discussion

Looking just at the predicted percent of discomfortable hours during occupied hours for the existingbuilding model, we can see that in the middle (which represents summer) certainly it is not comfortable.

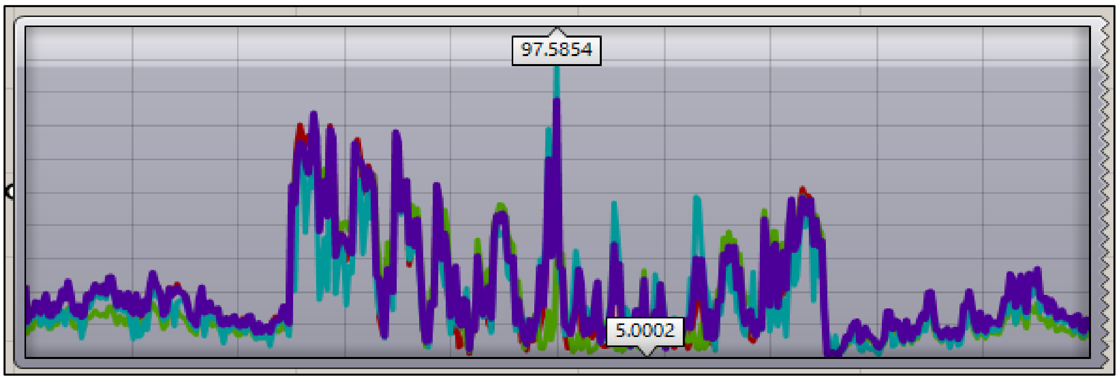


Figure 7 - Predicted percentage of dissatisfied people for baseline model

Calculated thermal comfort objective was **18,6 %** as it calculates percentage of comfortable time during occupancy and takes smallest number from all zones, where:

- West office zone **18.6 %**
- East office zone 19.4 %
- Reception 21.3%
- South office zone 26.1%

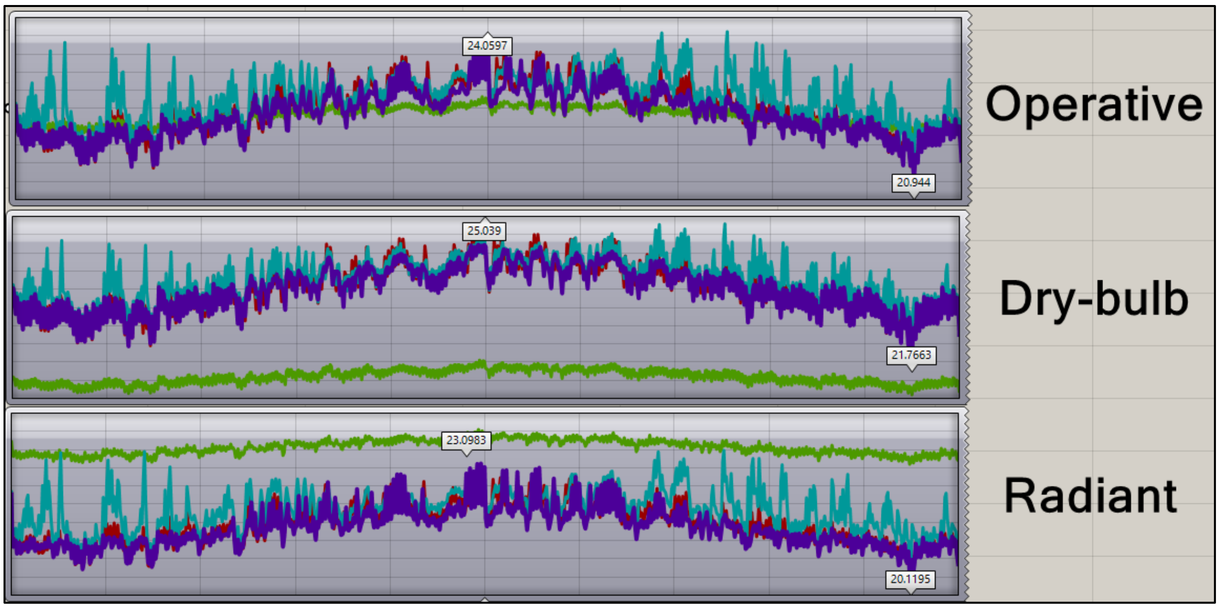


Figure 8 - Operative, Dry-bulb and radint temperatures for each zone for the occupied period (3320 hours). Green: Reception, turquoise: south office zone, red: west office zone, violet: east office zone.

In certain days and weeks there is high variation of radiant temperature in other spaces than reception as solar gains fall on offices surface area. That could be flattened by adding thermal mass, lowering delta temperature between heating and cooling setpoints, lowering hysteresis for radiant system control throttling etc.

Energy balance graph shows what type of loads we have. High infiltration load during the whole year – infiltration rate was set to 0,7 [ach] which confirms that this is leaky building. In summer contributors from biggest to smallest are solar gains, interior equipment, lighting and people. Note: Natural ventilation was not modelled in original file.

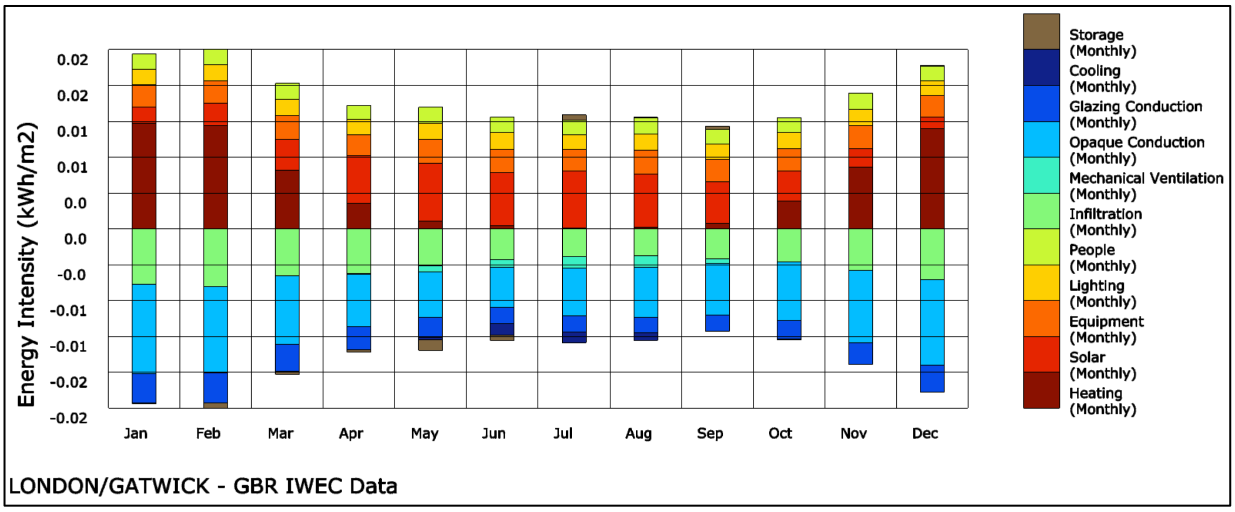
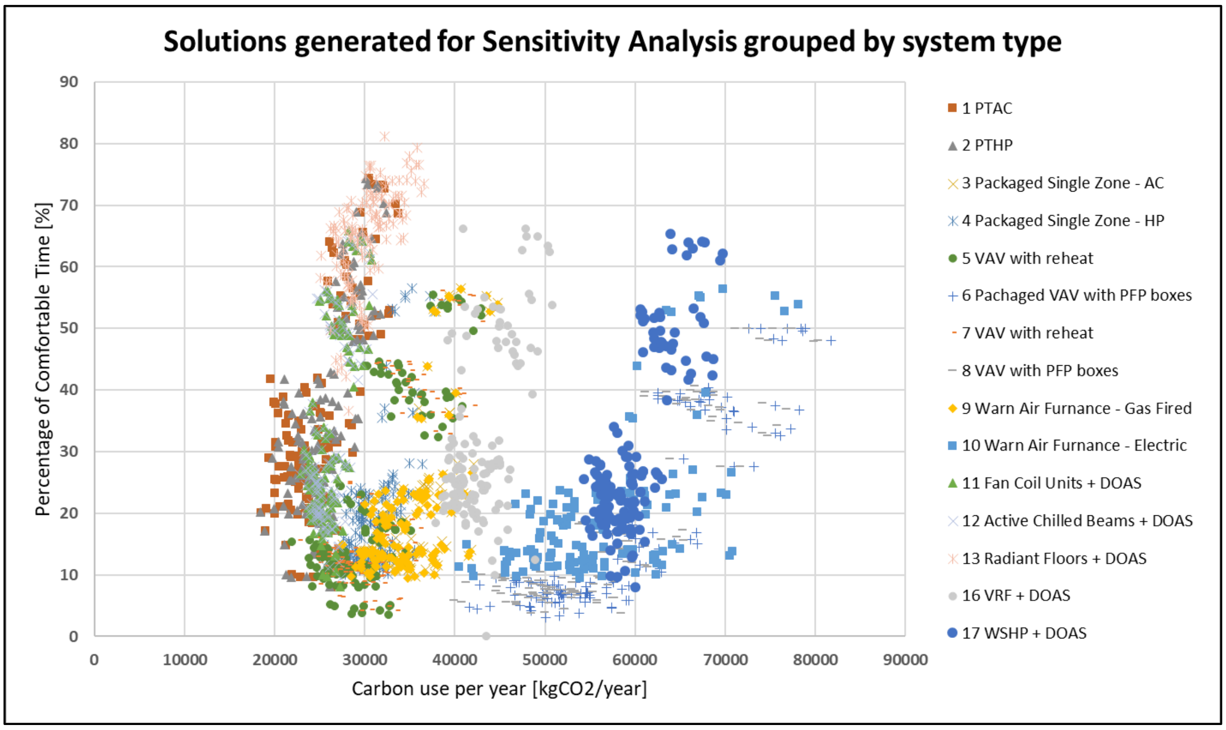


Figure 9 - Energy balance in ideal air loads system for the whole building

Making building more air-tight will lower heat loss in winter but also lower cooling potential of infiltration in summer, nevertheless it is important to seal the building. Solar gains could be lowered by replacing single pane windows with better SGHC glazing or external shading, building has lightweight construction, almost zero insulation etc. There are many possible options to make this building more energy efficient and confortable, however the main aspect of this coursework is to focus on HVAC systems. That being said couple measures were implemented to explore range of values for deeper analysis with goal to understand better HVAC systems rather than making this building passive design.

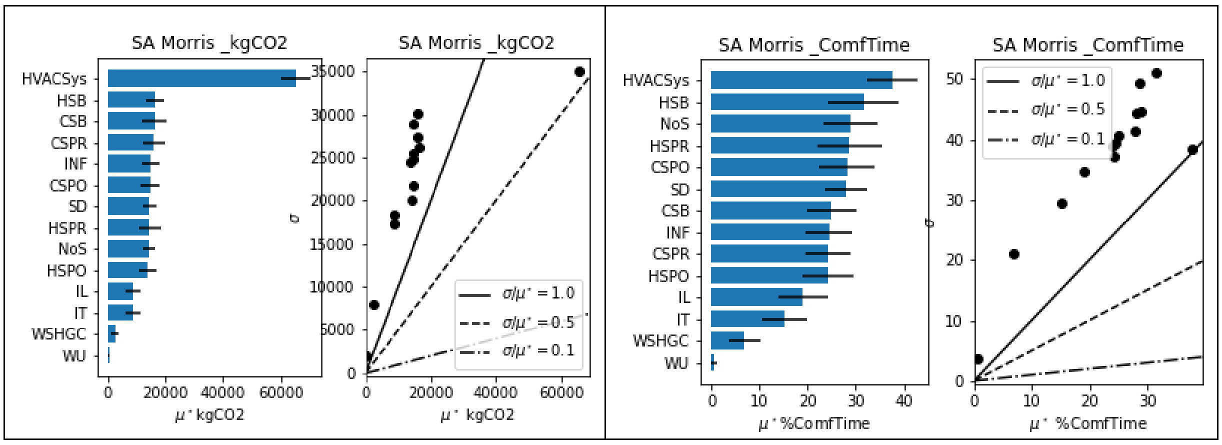
Some systems will perform better than others and it does not solely depend on system having more efficient boiler or pump, but rather on one system not having cooling at all, one bringing heat or coolth only through air and another changing radiant temperature of surfaces.

First simulation batch was created to remove systems that do not cope with chosen objectives and pick one system that performs better than the others. The purpose of that was to narrow field of view to focus only on one system. This was quick and easy way of choosing single system. Unfortunately, this approach is flawed and should not be used without proper sizing and setting parameters correctly as default systems have default values which are very often far from being realistic. Even system number 13 on which analysis was focused on had 80°C supply temperature as default on radiant system without mixer, that was later changed to 40°C. Vertical ground heat exchanger and heat pump were oversized which certainly influenced SCOP calculation and they do not represent real case scenario.



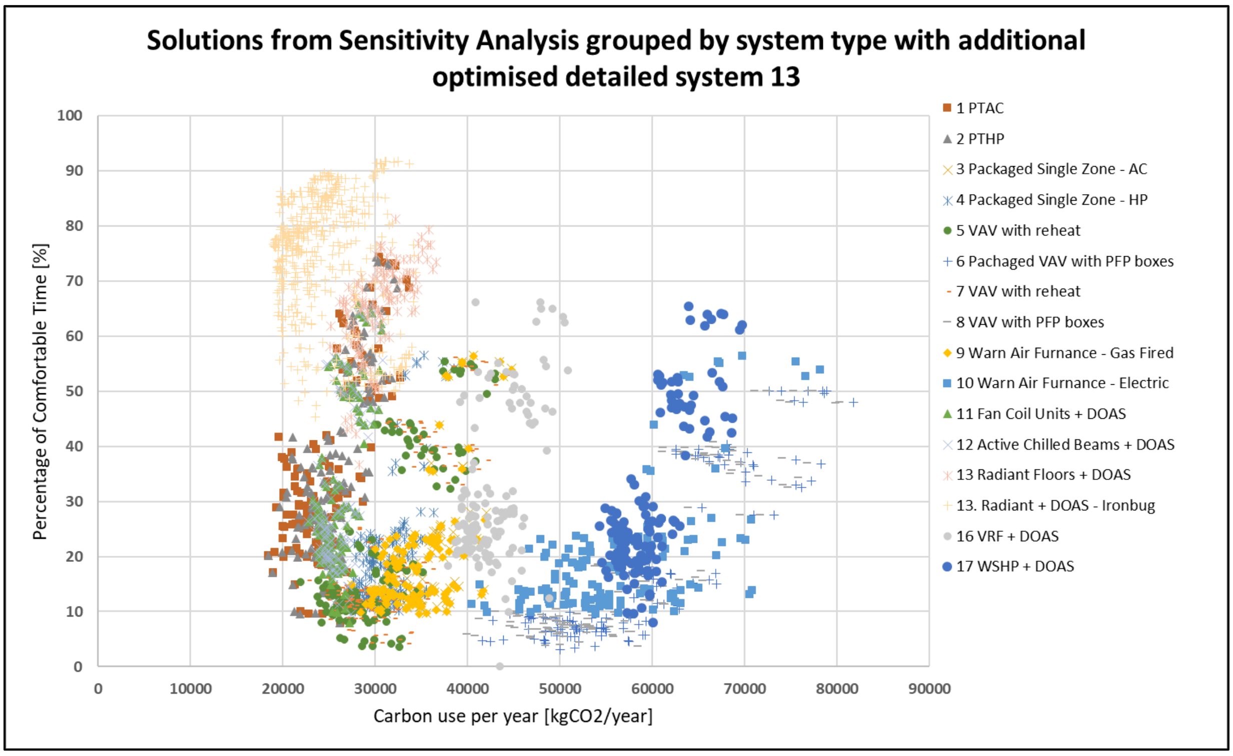
Temperatures for radiant system must be set correctly so that there is no condensation on surfaces. Models were monitored for condensation, information about this is available through DesignExplorer.

Running single simulation took around 60 second but running multiple simulations in a parallel took around 5 seconds on average for a single model. Average time of generating single model for sensitivity analysis without OS measure was 1 second, simulation time 5 seconds and reading results, converting them to our objectives and preparing them for sensitivity analysis script took around 10 seconds. Single batch composed of 2000 models was ready after around 9 hours. As previously mentioned first batch wasn't good for SA, so additional 5,5 hours were needed to re-run generating results. Adding ID for each model took couple of seconds and slowed entire process by 6 milliseconds. Before running first big batch, grasshopper script was tested on smaller 30 model batch, but issue didn't appear during testing phase. OS measure with radiant fraction had small impact on objectives, but extended model generation time by 20 seconds and because of that it was removed from the sensitivity analysis batch run. How big impact radiant fraction has on other systems with other parameters was not checked.



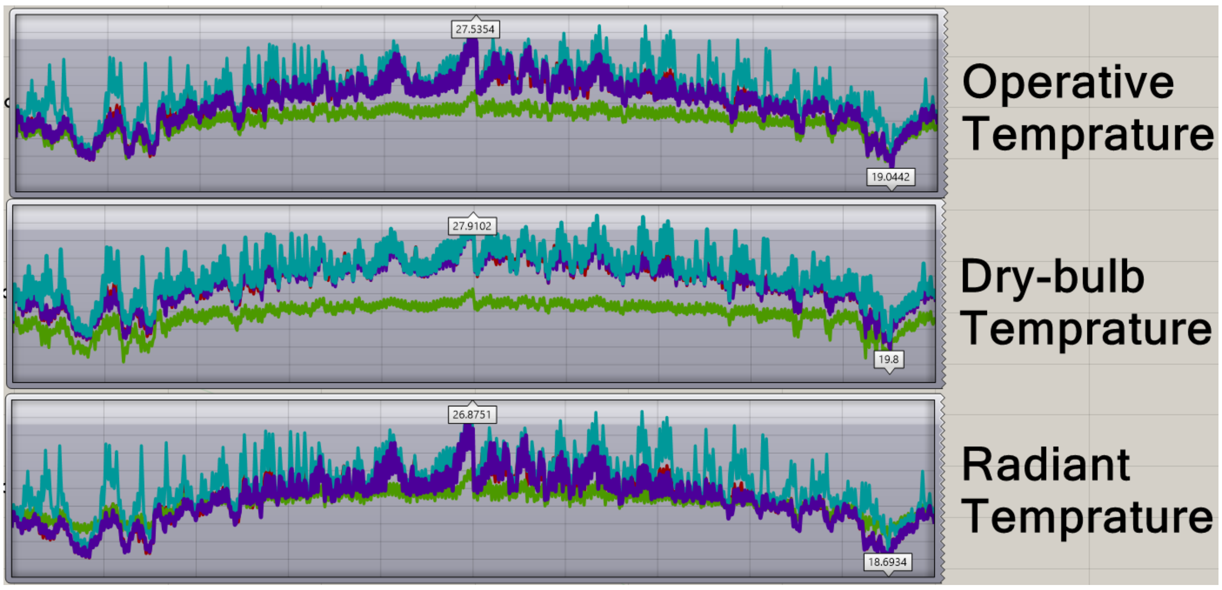
Sensitivity analysis for all systems showed that HVAC system selection is the most sensitive parameter in meeting both objectives. Window U-value and solar heat gain coefficient were at the end of the chart, although bounds were set from 1 to 4 W/m2K which is quite high range. Values are not completely zero and this inaccuracy in the results rather come from small sample size and how the Morris method works rather than these parameters not being sensitive enough. Later it turned out to be an issue in grasshopper, information was not connected correctly, and parameters WU and WSHGC weren't changed in each iteration, just the input name was changed.

Radiant ceiling was on 100% of the ceiling area, typically it would be made of 2-4 m^2 panels and in between there would be air terminals, lighting fixtures, sensors etc which changes how the panels would be sized.



Clearly additional measures with optimisation algorithm resulted in better performing iterations. Algorithm was converging and gave reasonable yield of good solutions, generating more iterations and more generations would give even better solutions. But newer solutions would be better by only small increments. At a distant generation, it would not find better solutions as the good solutions space would deplete. Further measures would have to be added to expand solution space.

Solution from Pareto front with inputs:1_0.6_5_1_32_28_27_16_18_19_0_0_13_0_21_6_1_0_28_18



And corresponding outputs: carbon footprint: 21959 kgCO2, Comfortable time: 86.28 %, Condensation time: 0h, Seasonal coefficient of performance for the heat pump, HP_SCOP: 3.55, Peak cooling load, PeakCL: 7,7 kW and peak heating load, PeakHL: 17,3 kW. The rest of the results are saved on DesignExplorer website: https://bit.ly/2Ylj9SV

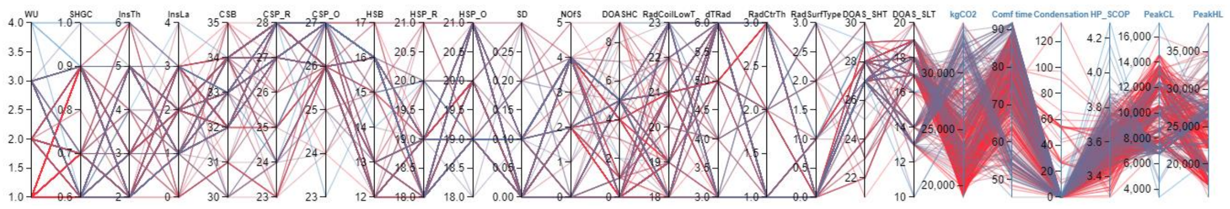


Figure 10 - All results simulated during optimisation visible from DesignExplorer website

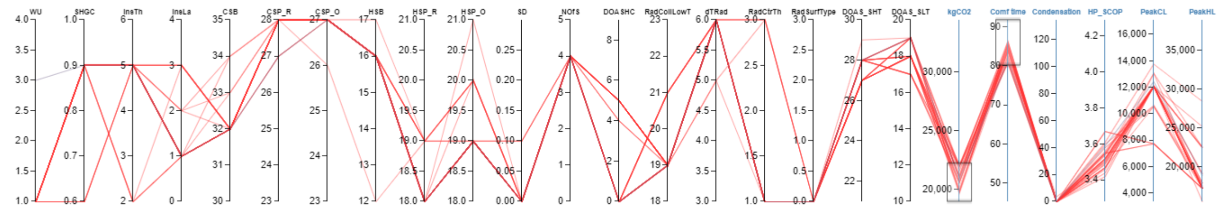


Figure 11 - Few best performing results visible from DesignExplorer website

Algorithm found majority of good solutions to be:

- Active surface on ceilings
- Window U value to be lowest possible
- Small delta temperature between setpoint and setback
- High delta temperature between low and high temperatures which control when heating or cooling is turned on
- Low design capacity for DOAS heating coil

Additional examples of measures (apart from correcting previously mentioned errors) which could be also proposed to the client are generating electricity on-site by implementing photovoltaic array on roof, changing DOAS heating coil source of energy to hot water generated from heat pump, light controls.

Most parts of Grasshopper scripts are very often almost self-explanatory. Tim Peters in The Zen of Python wrote “Explicit is better than implicit”. That rule is a key for creating Grasshopper script or any programming script for the developers and others to understand them easily.

Client brief

To improve current state of the building by lowering carbon footprint and making it more comfortable for the occupants following recommendations are proposed:

1. Invest in more efficient and air-tight windows
2. Add insulation above ceiling
3. Remove baseboard units and implement ceiling mounted radiant panels with heating and cooling
4. Install heat pump with ground heat source
5. Install mechanical ventilation to bring fresh filtered air to the office, with heat exchanger to reduce heat loss and electricity bills.
6. Add controls for temperature, humidity, occupancy and lights.

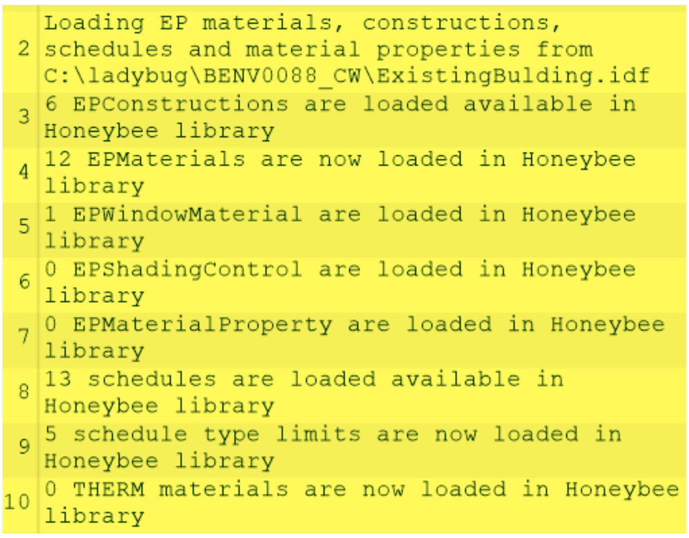
For better results visualisation during the meeting and to answer client questions about not implementing certain solutions and what the difference would be DesignExplorer is provided.

https://bit.ly/2Ylj9SV

References

1. Mackey, C., & Sadeghipour Roudsari, M. (2018). The Tool(s) Versus The Toolkit. In Humanizing Digital Reality. https://doi.org/10.1007/978-981-10-6611-5_9
2. Pye, S. et al. (2015). Pathways to deep decarbonization in the United Kingdom, SDSN - IDDRI.
3. Roudsari, M. S., & Pak, M. (2013). Ladybug: A parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design. Proceedings of BS 2013: 13th Conference of the International Building Performance Simulation Association, 3128–3135.
4. ASHRAE-55. (2017). Thermal environmental conditions for human occupancy. ANSI/ASHRAE Standard - 55.
5. Python library for Sensitivity Analysis SALib. [online]. Available from: https://github.com/SALib/SALib
6. Spawn-of-EnergyPus (Sprawn). [online]. Available from: https://www.energy.gov/eere/buildings/downloads/spawn-energyplus-spawn
7. Ladybug: a parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design. [online]. Available from: http://www.ibpsa.org/proceedings/bs2013/p_2499.pdf
8. National Renewable Energy Laboratory (NREL) github respository. [online]. Available from: https://github.com/NREL/EnergyPlus/tree/bee886ab567181a306a8c2e7c678d6a1063b0bb0/testfiles
9. EnergyPlus version 8.9 documentation, Engineering Reference. [online] Available from: https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v8.9.0/EngineeringReference.pdf
10. Design Explorer tool. [online] Available from: https://tt-acm.github.io/DesignExplorer/
11. Instructions for installing Ladybug + Honeybee. [online]. Available from: https://github.com/ladybug-tools/ladybug-legacy/wiki/Installation-Instructions
12. Ironbug, Grasshopper plugin for detailed HVAC design. [online] Available from: https://github.com/MingboPeng/Ironbug
13. TT Toolbox Grasshopper plugin with colibri 2.0. [online] Available from: https://grasshopperdocs.com/addons/tt-toolbox.html
14. Octopus. Evolutionary algorithm optimisation tool for Grasshopper. [online] Available from: https://www.food4rhino.com/app/octopus
15. Radiant Fraction OpenStudio Measure. [online] Available from: http://hydrashare.github.io/hydra/viewer?owner=chriswmackey&fork=hydra_2&id=Apply_OpenStudio_Measure
16. CIBSE. (2008). CIBSE TM46 Energy benchmarks.

Appendix 1 – Panel with information what kind of elements were imported to Honeybee library



```
Loading EP materials, constructions,
schedules and material properties from
C:\ladybug\BENV0088_CW\ExistingBulding.idf
6 EPConstructions are loaded available in
Honeybee library
12 EPMaterials are now loaded in Honeybee
library
1 EPWindowMaterial are loaded in Honeybee
library
0 EPShadingControl are loaded in Honeybee
library
0 EPMaterialProperty are loaded in Honeybee
library
13 schedules are loaded available in
Honeybee library
5 schedule type limits are now loaded in
Honeybee library
0 THERM materials are now loaded in Honeybee
library
```

Appendix 2 – Information about each parameter and corresponding bounds for sensitivity analysis

WU – Windows U-value [W/m2K]

WSHGC – Windows solar heat gain coefficient [-]

IT – Insulation thickness [m]

IL – Insulation lamda [W/mK]

INF – infiltration [ach]

CSB – Cooling setback [-]

CSPR – cooling setpoint reception [-]

CSPO – cooling setpoint office [-]

HSB – heating setback [-]

HSPR – heating setpoint reception [-]

HSPO – heating setpoint office [-]

SD – shading depth [m]

NoS – number of horizontal shades [-]

HVACSys – HVAC default system template [-]

```
#name lower_bound upper_bound
WU 1 3
WSHGC 1 6
IT 1 6
IL 0 4
INF 1 7
CSB 0 5
CSPR 0 5
CSPO 0 5
HSB 0 5
HSPR 0 3
HSPO 0 3
SD 0 3
NoS 0 5
HVACSys 1 17
```

Appendix 3 – Python script for sensitivity analysis for whole dataset

```
pip install SALib

from SALib.analyze import morris
from SALib.util import read_param_file
from SALib.plotting.morris import horizontal_bar_plot, covariance_plot, \
   sample_histograms
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files

def fullprint(*args, **kwargs):
  from pprint import pprint
  import numpy
  opt = numpy.get_printoptions()
  numpy.set_printoptions(threshold=numpy.inf)
  pprint(*args, **kwargs)
  numpy.set_printoptions(**opt)

#IMPORT DATA
problem = read_param_file('/content/drive/My Drive/Colab Notebooks/Integrated Design/parameter_file.txt')
#model_input = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/model_input.txt')
#model_output = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/model_output.txt')
data = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/2166.txt')
print("data ndim: ", data.ndim)
print("data shape:", data.shape)
print("data size: ", data.size)

D=problem["num_vars"] #number of variables
print(data[:,0:D].shape)

rows=data.shape[0] #number of inputs/outputs
print("rows: ",rows)
n=divmod(rows,D+1)[0]
print("n: ",n)
model_input=data[:,0:14][0:n*(D+1),:]
model_input.shape

D=problem["num_vars"] #number of variables
for x in [0,1]: #x=1 kgCO2, x=2 %comf_t
  # Prepare data for the analysis for each system
  filename = 'SA Morris '
  rows=data.shape[0] #number of inputs/outputs
  n=divmod(rows,D+1)[0]
  model_input=data[:,0:14][0:n*(D+1),:]
  model_output=data[:,D+x:D+1+x][0:n*(D+1),:]
  print(filename)
  # Perform the sensitivity analysis
  Si = morris.analyze(problem, model_input, model_output, conf_level=0.95,
                    print_to_console=True)
  #create plots
  fig1, (ax1, ax2) = plt.subplots(1, 2)
  #ax1.set_title(filename)
 # ax2.set_title(filename)
  if x == 0:
    horizontal_bar_plot(ax1, Si, {}, sortby='mu_star', unit=r"kgCO2")
    covariance_plot(ax2, Si, {}, unit=r"kgCO2")
    filename_update=filename+"_kgCO2"
  else:
    horizontal_bar_plot(ax1, Si, {}, sortby='mu_star', unit=r"%ComfTime")
    covariance_plot(ax2, Si, {}, unit=r"%ComfTime")
    filename_update=filename+"_ComfTime"
  ax1.set_title(filename_update)
  ax2.set_title(filename_update)
  #save and download plots
  plt.savefig(filename_update)
  files.download(filename_update+".png")
  plt.close()
```

Appendix 4 – Python script for sensitivity analysis for each system type

```
pip install SALib

from SALib.analyze import morris
from SALib.util import read_param_file
from SALib.plotting.morris import horizontal_bar_plot, covariance_plot, \
    sample_histograms
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files

def fullprint(*args, **kwargs):
  from pprint import pprint
  import numpy
  opt = numpy.get_printoptions()
  numpy.set_printoptions(threshold=numpy.inf)
  pprint(*args, **kwargs)
  numpy.set_printoptions(**opt)

#IMPORT DATA
problem = read_param_file('/content/drive/My Drive/Colab Notebooks/Integrated Design/2166/parameter_file.txt')
#model_input = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/model_input.txt')
#model_output = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/model_output.txt')
data = np.loadtxt('/content/drive/My Drive/Colab Notebooks/Integrated Design/2166/2166.txt')
print("data ndim: ", data.ndim)
print("data shape:", data.shape)
print("data size: ", data.size)

for i in [1,2,3,4,5,6,7,8,9,10,12,13,16,17]:
  print(data[data[:,13]==i,0:13].shape)

D=problem["num_vars"] #number of variables
for x in [0,1]: #x=0 kgCO2, x=1 %comf_t
  for HVACsys in [1,2,3,4,5,6,7,8,9,10,12,13,16,17]:

    # Prepare data for the analysis for each system
    filename = 'HVACsys'+str(HVACsys)
    rows=data[data[:,13]==HVACsys,0:13].shape[0] #number of inputs/outputs
    n=divmod(rows,14)[0]
    model_input=data[data[:,13]==HVACsys,0:13][0:n*(D+1),:]
    model_output=data[data[:,13]==HVACsys,14+x:15+x][0:n*(D+1),:]
    print(filename)
    # Perform the sensitivity analysis
    Si = morris.analyze(problem, model_input, model_output, conf_level=0.95,
                    print_to_console=True)
    #create plots
    fig1, (ax1, ax2) = plt.subplots(1, 2)
    ax1.set_title(filename)
    ax2.set_title(filename)
    if x == 0:
      horizontal_bar_plot(ax1, Si, {}, sortby='mu_star', unit=r"kgCO2")
      covariance_plot(ax2, Si, {}, unit=r"kgCO2")
      filename_update=filename+"_kgCO2"
    else:
      horizontal_bar_plot(ax1, Si, {}, sortby='mu_star', unit=r"%ComfTime")
      covariance_plot(ax2, Si, {}, unit=r"%ComfTime")
      filename_update=filename+"_ComfTime"
    #save and download plots
    plt.savefig(filename_update)
    files.download(filename_update+".png")
    plt.close()
```