accim: a Python library for adaptive setpoint temperatures in building performance simulations

Daniel Sánchez-García[1*], David Bienvenido-Huertas[2] and William O'Brien[3]

[1]Grupo Termotecnia, Higher School of Engineering, University of Cádiz, Cádiz, Spain

[2]Department of Building Construction, University of Granada, Granada, Spain

[3]Department of Civil and Environmental Engineering, Carleton University, Ottawa, Ontario, Canada

*****Author to whom correspondence should be addressed;

Abstract

Building performance simulations (BPS) can be used to estimate the energy required to deliver indoor environmental conditions acceptable for the occupants. Although the adaptive approach has been historically addressed only to naturally-ventilated spaces, recent research has found it could also be applied to air-conditioning spaces. Thus, it is possible to use setpoint temperatures based on adaptive comfort models as energy saving measures. This study presents a seamless methodology based on the use of accim, an open-source software tool to automate the use of adaptive setpoint temperatures in building performance simulations. accim allows to use script-based workflows to perform all actions within the development of a simulation-based thermal comfort study. A case study is used to demonstrate the capabilities of accim. The results show accim provides a wide range of new possibilities for developing studies related to the energy implications of adaptive thermal comfort.

Keywords: adaptive thermal comfort; building performance simulation; adaptive setpoint temperatures; computational approach; Python

1. Introduction

1.1. Research context

Building performance simulation (BPS) is being used more and more often throughout the life of a building for energy saving measures and building energy consumption analysis, among others. It has become more important in the formulation of regulations that support the goal of lowering energy usage and emissions in the construction industry, as well as in the design and operation of low energy, high performance buildings (Hong, Langevin, and Sun 2018). Building energy modelling software programmes may generally be divided into: (i) software programmes with built-in simulation engines, such as EnergyPlus (Crawley et al. 2001), TRNSYS (Beckman et al. 1994) or ESP-r (University of Strathclyde 2002); (ii) Graphical User Interfaces for some specific simulation engine, such as eQUEST (Hirsch), Designbuilder (DesignBuilder Software Ltd 2021), OpenStudio (Guglielmetti, Macumber, and Long 2011) or Modelkit (Big Ladder Software Ltd) among others; (iii) parametric and optimisation tools, such as jEplus (Yi) and GenOpt (Wetter 2001); and (iv) plugins that provide specific performance analysis for other applications, such as the Python package eppy (Santosh et al. 2004), MLE+ (Bernal et al. 2012), EpXL (Schild) or Ladybug Tools (Roudsari) among others.

Computational repeatability, reproducibility and replicability has grown more crucial to academics, designers, and practitioners as BPS gets more integrated into various elements of the design of architecture and the procedures involved in making decisions. The lack of a computational environment containing crucial software and applications to conduct it, as well as a workflow in which BPS and data analysis are integrated, are the major causes of issues in simulation reproducibility. There are currently very few instances in the literature that address how to do reproducible research in the BPS area.

Also, it is important to consider not just the energy type but also the way buildings are used (Allouhi et al. 2015). To address this need, adjustments in setpoint temperatures have been considered as a widespread energy saving measure (Monge Palma et al. 2023; Hoyt, Arens, and Zhang 2015). As a means of promoting natural gas independence, several governments, notably the Greek and Spanish ones, restricted the heating setpoint temperatures to 19 and the cooling setpoint temperatures to 27°C in August 2022. This may be the most recent example of setpoint temperature adjustments made to conserve energy. Also, the Danish government adopted heating-setpoint energy-saving policies for public buildings, that fixed it at 19°C. Within this research field, adaptive comfort models have recently been introduced as a strategy for incorporating buildings' resilience into decreased energy usage, especially when considering climate change scenarios. According to the requirements of ASHRAE 55 (ANSI/ASHRAE 2020) and EN16798-1 (European committee for standardization 2019) and based on 1998 de Dear's and Brager's first regression model (de Dear and Brager 1998), adaptive comfort models are appropriate exclusively for spaces with natural ventilation, given these must also be non-cooled/non-heated spaces. However, certain considerations must be given. Since there were so few mixed-mode (MM) building information in the initial RP-884 database, the 1998 research lacked enough information on them to draw any firm conclusions. Some studies previously shed light on the reconciliation of adaptive comfort and air-conditioning (Yun, Lee, and Steemers

2016; Mui and Chan 2003). Also, after the development of the ASHRAE Global Thermal Comfort Database II (Földváry Ličina et al. 2018), Parkinson et al. (Parkinson, de Dear, and Brager 2020) re-examined the original ASHRAE adaptive model in 2020 considering this larger database. All three types of buildings, that is air-conditioned (AC), naturally-ventilated (NV), and mixed-mode (MM), showed outstanding concordance with the adaptive model when the independent variable was inside temperature instead of outside temperature. This led to an evaluation of the constraints related to the adaptive comfort models in MM and AC constructions to be reevaluated. The results from 1998 could only agree with the 2020 re-evaluation if the significant association between the internal and exterior temperatures in NV spaces was acknowledged. Therefore, what the 1998 research understood to be an adaptation to the exterior environment was an adaptation to the indoor environment, which in turn, was closely related to the outdoor environment. As such, the use of adaptive setpoint temperatures (Sánchez-García, Rubio-Bellido, et al. 2019) may help to achieve thermal comfort.

Adaptive setpoint temperatures can be explained as setpoint temperatures that take the values of the adaptive comfort limits, therefore making sure the temperature falls within adaptive comfort zone all the time the HVAC system is active, but with an energy consumption lower than the resulting from PMV-based ones. These have been applied using:

(i) very basic methods, such as the manually separate simulation of each month and the later merge of results to obtain the simulation throughout the year (Sánchez-Guevara Sánchez, Mavrogianni, and Neila González 2017), in which ASHRAE's simplified monthly mean prevailing mean outdoor temperature method for the calculation of the adaptive comfort limits was used instead of the daily weighted-mean method;

(ii) intermediate methods, which could use the daily weighted-mean prevailing outdoor temperature, but a time-consuming, difficult, and error-susceptible manual process that involved the following steps needed to be carried out (Sánchez-García, Bienvenido-Huertas, et al. 2019): setpoint temperatures were first calculated using an Excel spreadsheet; an Schedule:Compact object had to be created, so that setpoint temperatures could be pasted in it; the EPW file for the location in question was chosen; and finally, the adaptive setpoint simulation for each pair of setpoints and climate zone was run. The handling of other files was also necessary. Besides, each time an adaptive setpoint temperature (AST) and EnergyPlus Weather (EPW) file were combined, which might be hundreds or thousands of times, this procedure had to be repeated.

There are also some studies in which adaptive setpoint temperatures were applied, however the method was not described (Wang, Pattawi, and Lee 2020; Kramer et al. 2015; Dhaka, Mathur, and Garg 2012). Then, methods progressed using the Energy Management System (EMS) module of EnergyPlus, that could automate some parts of that process, which led to the development of the Adaptive-Comfort-Control-Implementation Script (ACCIS) (Sánchez-García, Martínez-Crespo, et al. 2023). This EMS script allows the calculation of the adaptive comfort limit values and implementation on the adaptive setpoint temperatures as simulations are run. Also, the user can customise the adaptive setpoint temperature using different arguments, which allow the user to select the comfort model, acceptability levels as well as certain parameters related to mixed-mode, such as minimum outdoor temperature or maximum wind speed (Sánchez-García, Martínez-Crespo, et al. 2023). Nonetheless, there were still some tasks that remained manual, mainly related to existing elements in the model, which could not be automated with EMS. These tasks were automated with a Python parser for EnergyPlus building energy models, named eppy (Santosh et al. 2004), which allows to translate objects into lists. As a result, the Python package named accim (Sánchez-García, Bienvenido-Huertas, and Rubio-Bellido 2021) was developed, which allowed to fully automate the implementation of adaptive setpoint temperatures.

The main use of accim is the implementation of adaptive setpoint temperatures as energy-saving measures. These are especially relevant to two topics: (i) energy poverty, since vulnerable households (those whose energy consumption is below half of the median) usually minimise the use of air-conditioning units and use natural ventilation instead, and therefore are more suitable for adaptive comfort conditions (Bienvenido-Huertas, Sánchez-García, Rubio-Bellido, and Pulido-Arcas 2021; Bienvenido-Huertas, Sánchez-García, Rubio-Bellido, and Marín-García 2021; Bienvenido-Huertas, Sánchez-García, and Rubio-Bellido 2021); and (ii) climate change, given the need to reduce the building energy consumption to help its mitigation, and the adaptation of human being to warmer environments. An example of the former, is the application of adaptive setpoint temperatures based on the EN16798-1 considering all occupant expectations, which led to energy poverty reductions ranging between 43% for Category I and 98.5% for Category III in Seville (Bienvenido-Huertas, Sánchez-García, and Rubio-Bellido 2021). An example of the latter is the use of accim to analyse energy savings based on the use of an adaptive local comfort model for Japan compared to the international ASHRAE 55 adaptive model in present and future scenarios. accim simplified the creation of tables and figures to present these results. The findings revealed that total energy demand decreased by 18% to 91% in cold climate zones but increased by 17% to 51% in warm climate zones, depending on the specific Representative Concentration Pathway scenario (Sánchez-García, Bienvenido-Huertas, et al. 2023).

1.2. Research gap

The fields of adaptive thermal comfort and building energy efficiency have evolved largely independently, with extensive research dedicated to each. Adaptive thermal comfort studies focus on optimizing occupant satisfaction, while energy efficiency research emphasizes reducing energy consumption and greenhouse gas emissions in buildings. However, there is a notable lack of studies that integrate these two domains to explore their interdependencies systematically.

The use of adaptive setpoint temperatures offers a potential solution for reconciling thermal comfort and energy efficiency objectives. By adapting setpoint temperatures to align with comfort models, this approach aims to balance occupant satisfaction with energy savings. Despite its potential, the practical application of adaptive setpoint temperatures has been hindered by fragmented methodologies. Tools like ACCIS have automated portions of the process, but the overall workflow—spanning data preparation, simulation, and analysis—remained largely manual and inconsistent.

This fragmentation highlights a critical gap in the literature: the lack of a unified methodology to seamlessly integrate adaptive thermal comfort and energy efficiency studies. Addressing this gap is essential to advance both fields and to enable more efficient and reproducible research at the intersection of comfort and energy performance.

1.3. Research aim

Thus, there is a necessity of procedures to ease the development of studies considering adaptive thermal comfort and energy demand. This Python library aims to fill this gap, not only by providing the tools, but also a seamless methodology to efficiently automate the entire process. This library has been partially presented in two previous studies: in the first one, the module used to apply the adaptive setpoint temperatures in the building energy models was explained (Sánchez-García, Bienvenido-Huertas, and Rubio-Bellido 2021), which only represents a step of the entire process; then, in the second one, the EMS code was explained in a deeper level (Sánchez-García, Martínez-Crespo, et al. 2023), in which the interactions among the objects were described.

However, the present paper expands the scope to capture the entire simulation process, and bases its originality statement in the following points: (i) it presents an overview of accim, considering the remaining modules not previously presented, (ii) provides a seamless Python-based methodology to ease the development of energy and thermal comfort studies, and (iii) demonstrates its capabilities providing an example of adaptive comfort and energy demand study from preparation of data to analysis and visualization of results.

As a result, this study introduces a Python-based approach that fully automates the adaptive setpoint simulation process, integrating key steps such as weather data preprocessing, adaptive setpoint application, simulation execution, and result analysis. Section 2 firstly presents an overview of the modules for data analysis, simulations and implementation of adaptive setpoint temperatures, and secondly describes the case study used in this work. Section 3 firstly describes the application of the methodology; secondly, analyses the results of the case study and lastly compares accim to the main alternative method. Section 4 discusses the results and Section 5 describes the limitations.

2. Methodology

2.1. Modules for data analysis, simulations and implementation of adaptive setpoint temperatures

The Python package accim is mainly composed of three modules: `accim.data`, `accim.sim` and `accim.run`. The structure and use of the main user-callable classes, methods and functions of the project are:

- `data`
  - `data_preprocessing`
    - `rename_epw_files`: prior to simulation, the EPW files need to be formatted to follow a certain file name pattern. This is necessary for the later analysis of the CSV files resulting from simulation, considering variables such as city and climate change future scenario.
  - `data_postprocessing`
    - `Table`: allows to read the CSV files and generate a pandas DataFrame instance.
      - `format_table`: allows to filter the columns of the DataFrame instance.
      - `scatter_plot`: generates a PNG file containing a scatter plot based on multiple arguments.
      - `scatter_plot_with_baseline`: similar to method `scatter_plot`, but allows to compare some specific variant with the others.
      - `time_plot`: generates a PNG file containing a line plot with time on the x-axis.
      - `wrangled_table`: allows to perform multiple reshaping methods to obtain summary tables.
- `sim`
  - `accis`

        - `addAccis`: used to apply adaptive setpoint temperatures based on multiple input arguments.
- `run`
    - `run`
        - `runEp`: used to automatically run simulations and generate results in suitable format.

Although the classes, methods and functions have been briefly explained above, a more detailed description can be found in Appendix A. When all of these are used, a seamless and script-based methodology can be carried out to automatically perform all actions required to develop a study based on adaptive setpoint temperatures.

2.2. Case study

In this work, a case study is presented to demonstrate the capabilities of accim. The full process, from data pre-processing to data analysis is carried out. In this case study, the aim is the analysis of energy demand resulting from the use of adaptive setpoint temperatures based on a local adaptive comfort model for office buildings in India, named IMAC-C (Manu et al. 2016), and the comparison with ASHRAE 55 adaptive model and static setpoints from the Indian Building Code (Bureau of Indian Standards 2016), in different operation modes. In this study, 2 different locations, a present and a future scenario, are considered, resulting in 4 different EPW files (Table 1).

The building energy model used is the 2018 IECC DOE Commercial Reference Building Prototype Small Office (U.S. Department of Energy 2023) (Table 1). It has 6 thermal zones: a thermal zone in the centre of the building (CORE_ZN), surrounded by 4 thermal zones (PERIMETER_ZN_1 to 4), all of them air-conditioned and located on the ground floor, and a non-air-conditioned attic (Figure 1). Since this is a well-known model, no further details are provided, however these are available online in (U.S. Department of Energy 2023).

Some of the objectives of this case study include the simulation considering MM, something that accim performs with Airflow Network objects, so that natural ventilation is calculated considering wind pressure coefficients among other factors instead of the assumption of a natural ventilation flow rate based on a schedule. However, the original building energy model from DOE do not consider such objects, and therefore have been imported to DesignBuilder and then exported to account for them. The MM used in accim is change-over, and therefore do not allow air-conditioning and natural ventilation to be used at the same time. Specifically, windows are opened when all the following conditions are met at each simulation timestep:

- If no heating and no cooling are needed
- if outdoor temperature < min outdoor temperature; min outdoor temperature is calculated using a delta value from the heating setpoint temperature, and it is used to prevent from introducing air excessively cold. So, for instance, if that delta value was set to 5 and heating setpoint was 18°C, when outdoor temperature fell below 13°C windows would be closed.
- if outdoor temperature < operative temperature
- if wind speed < max wind speed; max wind speed is set as a value, which when exceeded, windows are closed to prevent from introducing high-speed airflows
- if operative temperature < cooling setpoint temperature
- if operative temperature > ventilation setpoint temperature (VST, usually equal to the neutral or comfort temperature)

Table 1: Input files.

Extension	Filename
EPW	Current_Ahmedabad-hour.epw
	Current_Shimla-hour.epw
	RCP852100_Ahmedabad-hour.epw
	RCP852100_Shimla-hour.epw
IDF	SmallOffice_NewDelhi.idf

EPW: EnergyPlus Weather file
IDF: Input Data File

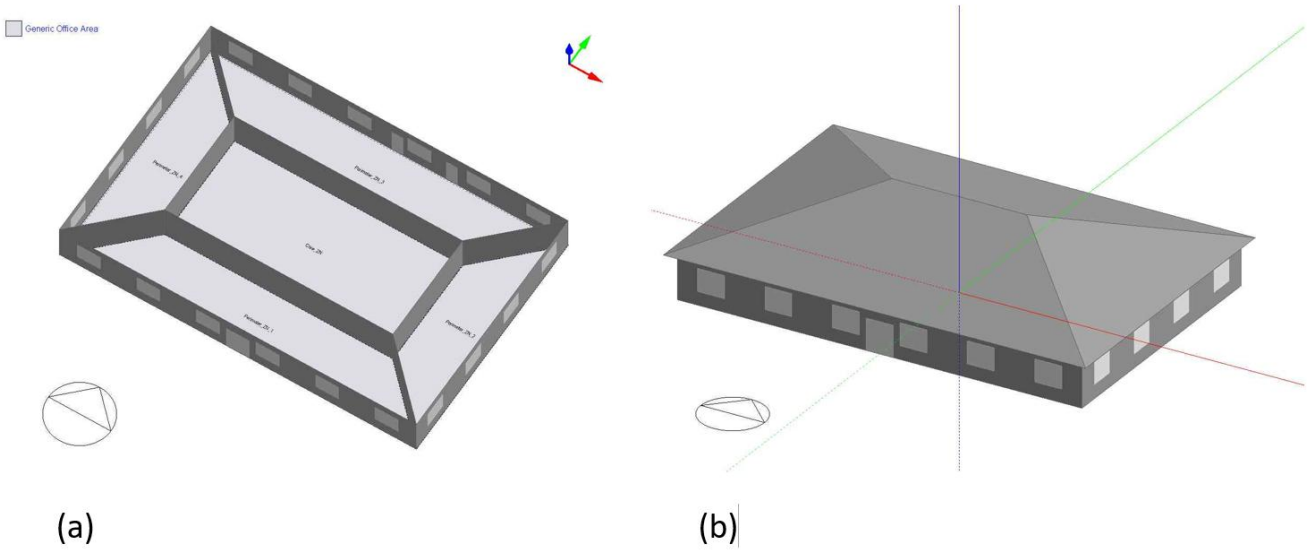

Figure 1. 2018 IECC DOE Commercial Reference Building Prototype Small Office. (a) View of thermal zones. (b) Overall view.

3. Use of the methodology

In this section, the case study is carried out using the proposed methodology based on the use of accim. Firstly, the process is briefly explained, focusing on input and outputs. Secondly, the results of the case study are analysed. Lastly, the proposed methodology is compared with other alternative methods.

3.1. Applying the methodology

The process for this study comprises three main phases: data pre-processing, running simulations, and data analysis, each carefully designed to ensure a systematic and accurate investigation of building performance under varying conditions (Figure 2). A detailed version of this section is available in Appendix B.

The first phase, data pre-processing, involves organizing and preparing the required input files. EPW files are renamed using a structured format that reflects their geographic location, representative climate scenario, and time frame (Table 2). This ensures that the files are not only easy to identify but also compatible with subsequent simulation tools. User interaction plays a crucial role during this step, as the system proposes new names based on extracted metadata, and the user must validate or amend these suggestions. In parallel, adaptive setpoint temperatures are incorporated into the building model files. This adjustment generates multiple versions of building Input Data Files (IDFs), representing different adaptive scenarios (Table 3). The output IDFs are named based on the settings input by the user and described in Table 4.

Table 2: Input and output EPW files.

Extension	Input	Output
EPW	Current_Ahmedabad-hour.epw	India_Ahmedabad_Present.epw
	Current_Shimla-hour.epw	India_Shimla_Present.epw
	RCP852100_Ahmedabad-hour.epw	India_Ahmedabad_RCP85-2100.epw
	RCP852100_Shimla-hour.epw	India_Shimla_RCP85-2100.epw

Table 3: Input and output IDF files.

Extension	Input	Output
IDF	SmallOffice_NewDelhi	SmallOffice_NewDelhi[CS_IND IMAC C NV[CA_80[CM_3[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1[NS_X.idf
		SmallOffice_NewDelhi[CS_IND IMAC C NV[CA_80[CM_3[HM_1[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1[NS_X.idf
		SmallOffice_NewDelhi[CS_IND IMAC C NV[CA_80[CM_3[HM_0[VC_X[VO_X[MT_X[MW_X[AT_0.1[NS_X.idf
		SmallOffice_NewDelhi[CS_IND IMAC C NV[CA_80[CM_0[HM_0[VC_X[VO_X[MT_X[MW_X[AT_0.1[NS_X.idf
		SmallOffice_NewDelhi[CS_INT ASHRAE55[CA_80[CM_3[HM_0[VC_X[VO_X[MT_X[MW_X[AT_0.1[NS_X.idf

Table 4: Description of output IDF files.

ComfStand	ComfMod	HVACmode	Definition	Acronym
CS_IND IMAC C NV	CM_3	HM_2	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in MM operation	IND_Adap_MM
		HM_1	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in NV operation	Ind_Adap_NV
		HM_0	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in AC operation	IND_Adap_AC
	CM_0	HM_0	PMV-based or nearly static setpoint temperatures based on Indian Building Code, in AC operation	IND_Stat_AC
CS_INT ASHRAE55	CM_3	HM_0	adaptive setpoints based on ASHRAE 55 horizontally extended beyond applicability limits, in AC operation	ASH_Adap_AC

In the second phase, running simulations, the prepared input files are used to execute building performance simulations. This step involves systematically combining the modified building models with the formatted climate data to simulate a variety of scenarios. Each simulation run generates a unique output file that reflects the interplay of climatic conditions and adaptive building strategies. In this case study, the process resulted in twenty distinct simulations, derived from combinations of five building models and four climate scenarios. The simulations were executed automatically, leveraging predefined parameters to ensure consistency and minimize manual intervention. The output files generated during this phase form the foundation for subsequent analysis, encapsulating detailed information about energy demand and indoor thermal conditions for each scenario. The simulation outputs are stored in a structured format to facilitate streamlined analysis in the next phase.

The final phase, data analysis, involves interpreting the simulation results through both quantitative and visual methods. The energy demand data is first organized into comprehensive tables, highlighting the performance differences among various adaptive strategies. These tables not only summarize energy consumption but also calculate energy savings achieved through adaptive measures, offering valuable insights into their efficiency under different conditions. Beyond tabular data, visualizations play a pivotal role in conveying the findings. Scatter plots and time-series graphs are used to illustrate the relationships between adaptive strategies, energy demand, and indoor comfort levels. These visual tools help uncover patterns, such as the effectiveness of adaptive cooling strategies in reducing energy use while maintaining comfort.

In conclusion, based on input files (i.e. EPW and IDF files) and using the script-based methodology, the tool provides insights into the comparison of different adaptive approaches, revealing these influence energy performance and occupant well-being across diverse climates and scenarios. These insights are presented in the next section.

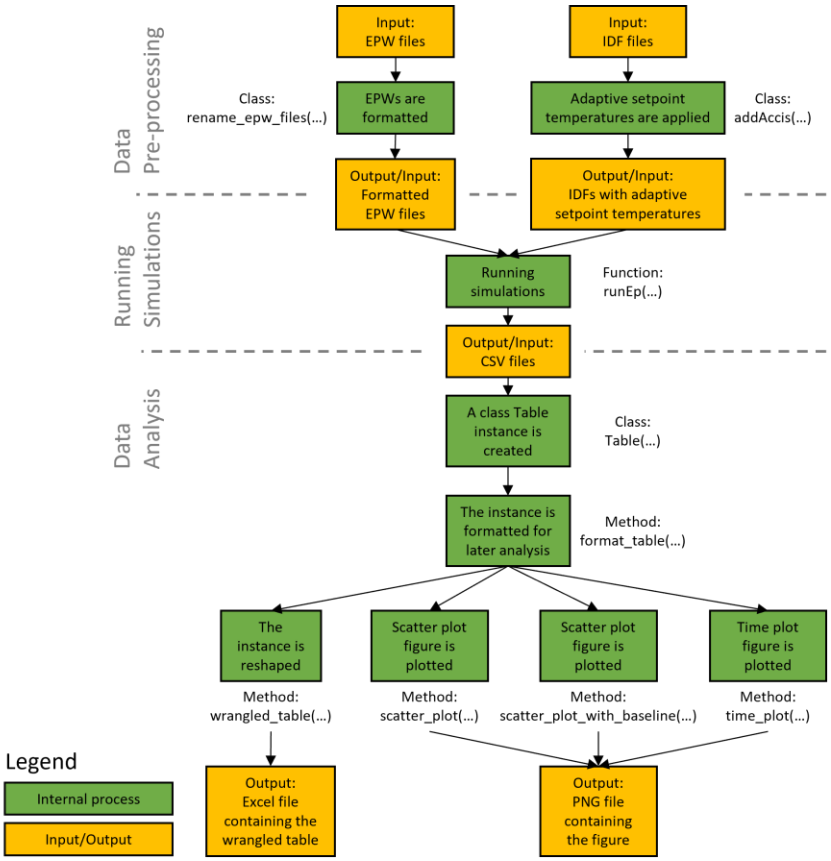

Figure 2. Overall process for the development of the case study

3.2. Results of the case study

The results of the study, if arranged in table format, contains data on energy demand for each simulation, as well as the comparison of these in terms of energy saving. In this case, for clarity purposes, that table has been divided into Table 5 and Table 6.

Table 5 presents absolute energy demand values, revealing significant differences across adaptive and static strategies under varying climatic conditions. In Ahmedabad, the static setpoint configuration (IND_Stat_AC) exhibits the highest cooling energy demand, with 283.83 kWh/m² in the present scenario. In contrast, IND_Adap_MM reduces this demand to 122.32 kWh/m², reflecting its superior performance. Under future climatic conditions (RCP85-2100), IND_Adap_MM continues to outperform, reducing energy demand to 190.9 kWh/m², compared to 340.09 kWh/m² for IND_Stat_AC. Similarly, in Shimla, IND_Adap_MM demonstrates its efficiency by achieving cooling energy demands as low as 48.78 kWh/m² (present scenario) and 75.88 kWh/m² (RCP85-2100). These results emphasize the versatility of IND_Adap_MM in optimizing energy consumption across diverse climatic contexts.

Table 6 offers a comparative analysis of energy savings achieved by mixed-mode adaptive settings (IND_Adap_MM) relative to other configurations, including both static (IND_Stat_AC) and air-conditioned adaptive (IND_Adap_AC and ASH_Adap_AC) settings. Savings are presented as percentages, with positive values indicating reductions and negative values reflecting increases in energy demand. For clarity purposes, the calculation of the percentages is shown for each setting in cooling, heating and total energy demand (i.e. 1-(IND_Adap_MM/IND_Stat_AC)). For cooling demand in Ahmedabad, savings with IND_Adap_MM range from 44% to 57% in the present scenario, depending on the baseline configuration, and from 24% to 44% under the RCP85-2100 scenario. Similarly, in Shimla, IND_Adap_MM achieves cooling energy savings ranging from 68% to 71% in the present scenario and from 60% to 62% under RCP85-2100. These values emphasize the adaptability of IND_Adap_MM, consistently delivering significant reductions in cooling demand across diverse climates and future projections.

The negative values observed in Table 6 correspond to increases in energy demand, primarily in heating scenarios. For instance, IND_Adap_MM shows a slight increase in heating demand compared to IND_Stat_AC in Shimla (10.42 kWh/m² vs. 6.62 kWh/m²). This increase results in a percentage change of -57%, which appears significant but is inconsequential due to the very small absolute values involved. Such variations, while mathematically notable, have negligible practical implications given the low heating energy demands in these scenarios.

Overall, the results in Table 5 and Table 6 confirm that IND_Adap_MM offers substantial energy savings across different settings, particularly in cooling demand, which is a critical factor in hot climates. The strategy demonstrates remarkable adaptability, ensuring reduced energy consumption and enhanced indoor comfort even under extreme climate projections. These findings reinforce the importance of adopting IND_Adap_MM as a cornerstone of sustainable building practices, particularly in the face of evolving climate challenges.

Table 5. Simulation results (part I): energy demand values.

EPW_City_or_subcountry		Ahmedabad		Shimla	
EPW_Scenario		Present	RCP85	Present	RCP85
EPW_Year		Present	2100	Present	2100
Building_Total_Cooling Energy Demand (kWh/m2) (summed)	IND_Stat_AC	283.83	340.09	151.41	200.27
	IND_Adap_AC	217	250.84	167.47	191.53
	IND_Adap_MM	122.32	190.9	48.78	75.88
	ASH_Adap_AC	250.7	291.64	163.01	199.53
Building_Total_Heating Energy Demand (kWh/m2) (summed)	IND_Stat_AC	0	0	6.62	1.08
	IND_Adap_AC	0	0	0.04	0
	IND_Adap_MM	1.08	0.09	10.42	2.44
	ASH_Adap_AC	0	0	0.61	0
Building_Total_Total Energy Demand (kWh/m2) (summed)	IND_Stat_AC	283.83	340.09	158.03	201.35
	IND_Adap_AC	217	250.84	167.52	191.53
	IND_Adap_MM	123.39	191	59.2	78.33
	ASH_Adap_AC	250.7	291.64	163.62	199.53

Table 6. Simulation results (part II): energy saving.

EPW_City_or_subcountry		Ahmedabad		Shimla	
EPW_Scenario		Present	RCP85	Present	RCP85
EPW_Year		Present	2100	Present	2100
Building_Total_Cooling Energy Demand (%) (summed)	1-(IND_Adap_MM/IND_Stat_AC)	0.57	0.44	0.68	0.62
	1-(IND_Adap_MM/IND_Adap_AC)	0.44	0.24	0.71	0.6
	1-(IND_Adap_MM/ASH_Adap_AC)	0.51	0.35	0.7	0.62
Building_Total_Heating Energy Demand (%) (summed)	1-(IND_Adap_MM/IND_Stat_AC)	-inf	-inf	-0.57	-1.26
	1-(IND_Adap_MM/IND_Adap_AC)	-inf	-inf	-233.02	-inf
	1-(IND_Adap_MM/ASH_Adap_AC)	-inf	-inf	-16.09	-inf
Building_Total_Total Energy Demand (%) (summed)	1-(IND_Adap_MM/IND_Stat_AC)	0.57	0.44	0.63	0.61
	1-(IND_Adap_MM/IND_Adap_AC)	0.43	0.24	0.65	0.59
	1-(IND_Adap_MM/ASH_Adap_AC)	0.51	0.35	0.64	0.61

-inf: infinite, resulting from division by zero

The performance outcomes of adaptive setpoint strategies in building energy simulations are also depicted in Figure 3, Figure 4 and Figure 5, offering valuable insights into energy demand and indoor comfort dynamics across various scenarios. These figures highlight the comparative advantages of IND_Adap_MM against other configurations while illustrating patterns and relationships critical for understanding energy-saving potential and thermal performance.

These trends are shown in Figure 3, which organizes data by setpoint settings and climatic scenarios. The figure highlights the ability of adaptive strategies like IND_Adap_MM to reduce energy use while maintaining indoor comfort, allowing temperatures to fluctuate within acceptable ranges. For instance, in Ahmedabad under RCP85-2100, IND_Adap_MM achieves significantly lower cooling energy demands compared to static setpoints while maintaining similar comfort levels. This figure includes a simulation of free-running conditions in the middle row, providing insights into the operative temperatures that would occur without an HVAC system. It also illustrates the extent to which adaptive comfort limits would be exceeded. Similar data is captured in time-series plots in Figure 4, offering insights into the temporal distribution of heating and cooling loads. This figure also provides insights into the responsive nature of the adaptive setpoints, changing daily based on the prevailing mean outdoor temperature variations.

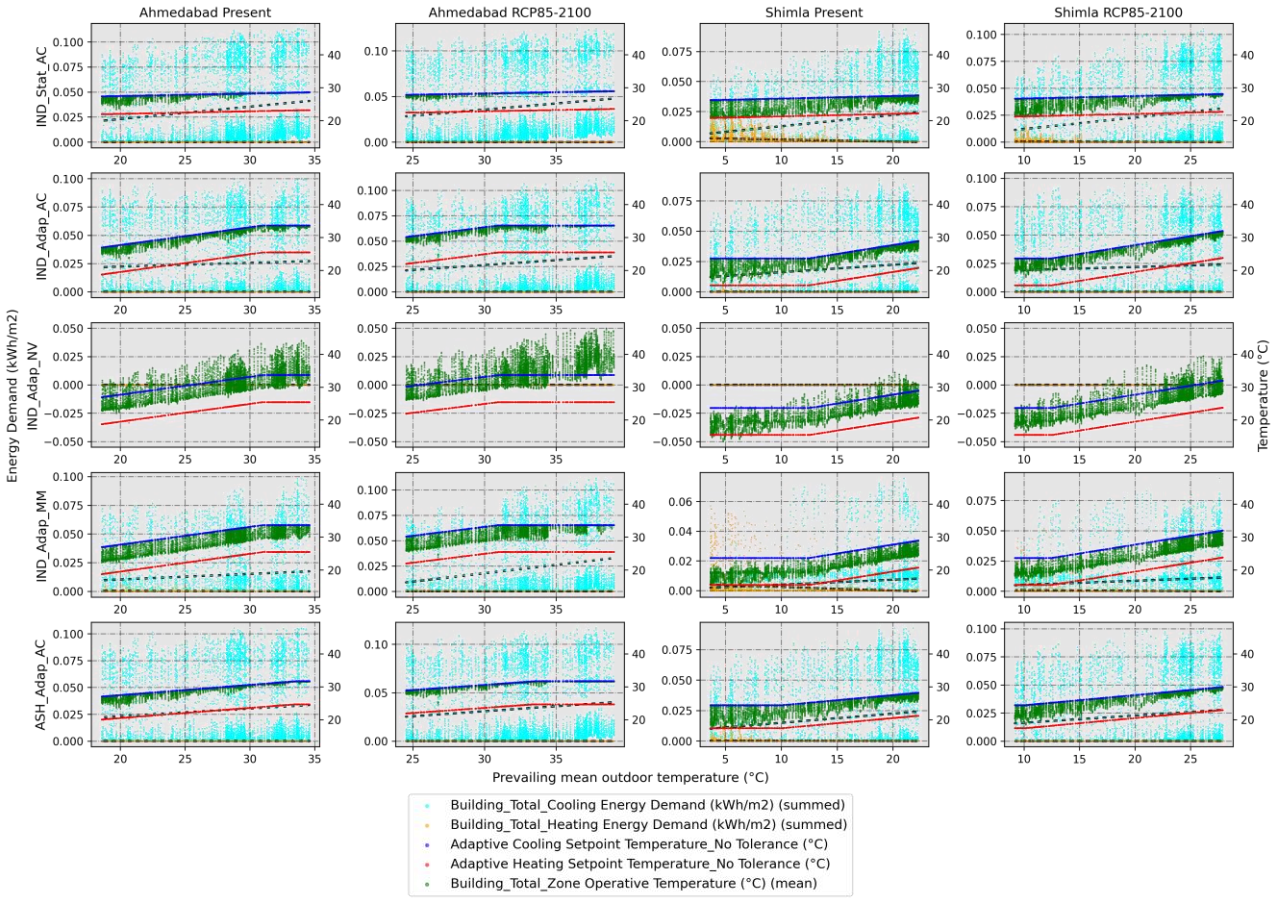

Figure 3. Adaptive comfort scatterplots

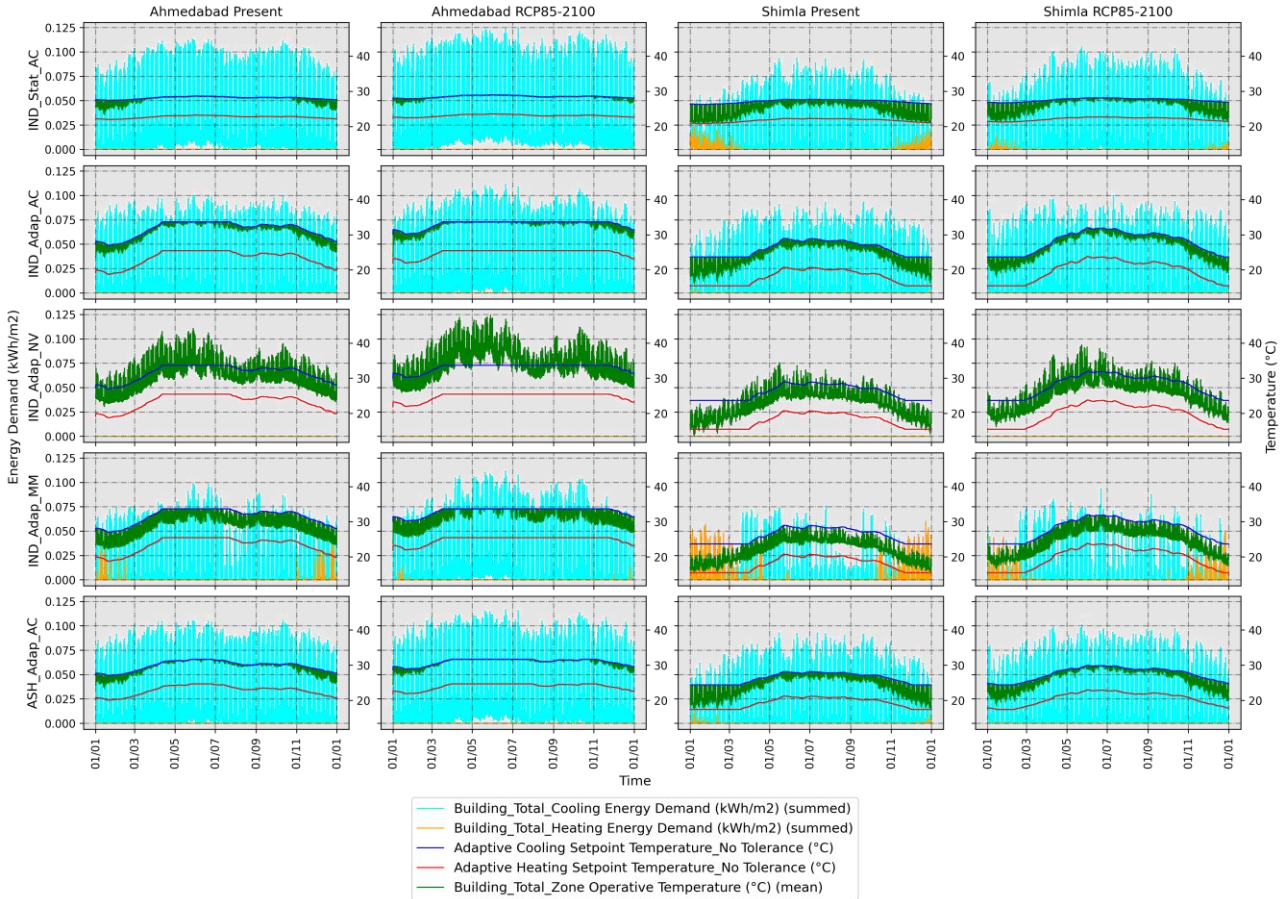

Figure 4. Adaptive comfort time-series plots

The comparative performance of IND_Adap_MM against other configurations is depicted in scatter plots that compare hourly energy demands, with IND_Adap_MM serving as the baseline. This comparison is detailed in Figure 5, where the x-axis represents IND_Adap_MM values and the y-axis corresponds to the energy demand of other configurations. Points that align along the diagonal represent similar energy usage, while deviations indicate differences. In Ahmedabad, many points cluster near the 25% line, demonstrating that IND_Adap_MM requires just a quarter of the energy used by static and other adaptive configurations for cooling in numerous cases. A similar trend is observed in Shimla, where IND_Adap_MM consistently shows substantial savings in cooling energy demand, reinforcing its position as the most energy-efficient configuration under various scenarios.

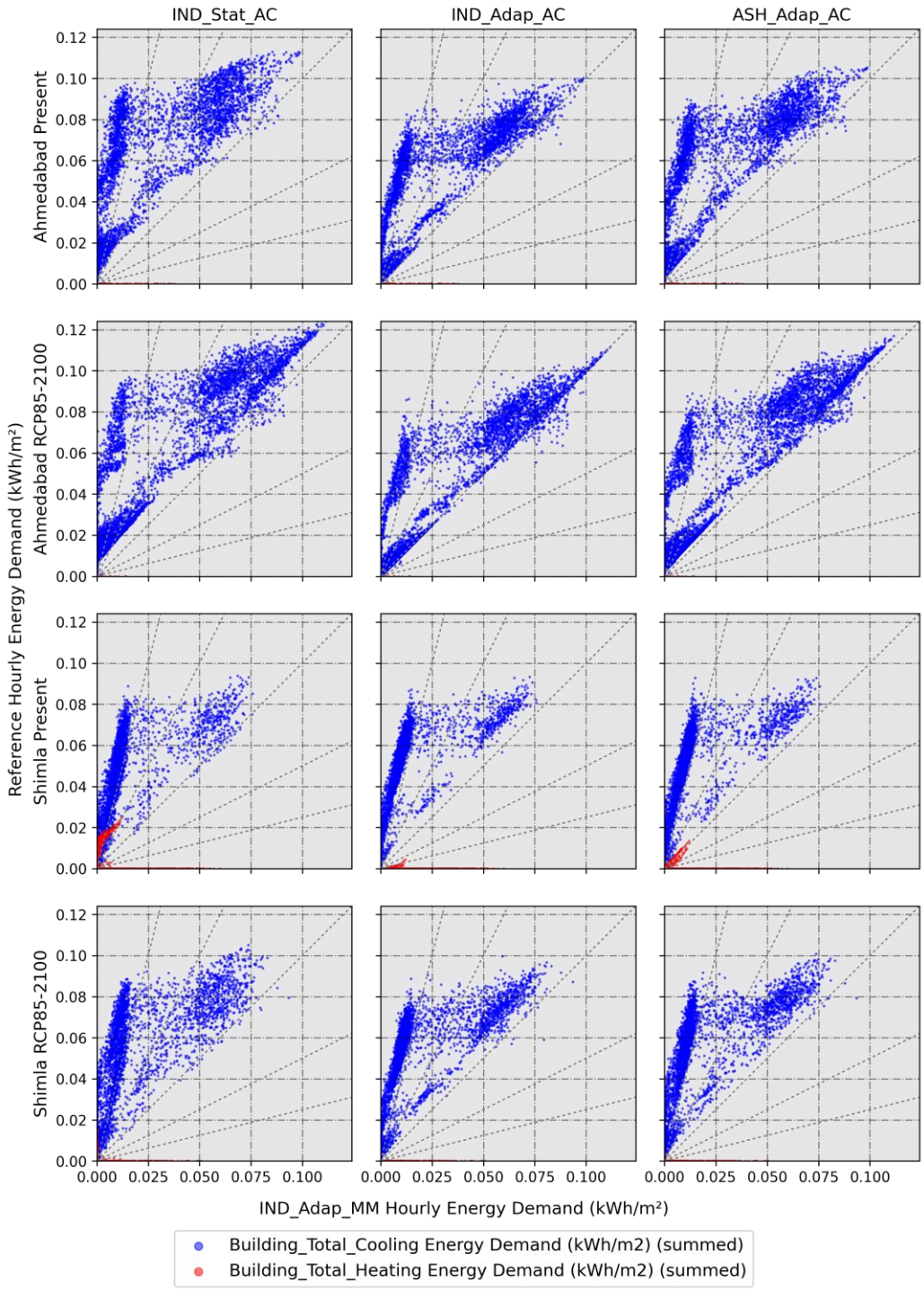

Figure 5. Comparison of hourly energy demands

3.3. Comparison with alternative methods

The main non-automated approach involves defining setpoints using a Schedule:Compact object, where adaptive setpoints are pre-calculated externally and stored in a CSV file for import into EnergyPlus. This method allows for more flexibility compared to manual monthly simulations, as users can input time-varying schedules that better reflect local climate conditions. However, the workflow is still error-prone and time-consuming, as it involves calculating the setpoints, formatting the CSV file, and ensuring compatibility with the EnergyPlus model. While this method simplifies some aspects of adaptive comfort implementation, it remains dependent on manual intervention and does not fully automate the process, making it challenging for large-scale studies or simulations with diverse scenarios.

Compared to the previous method, the accim Python library represents a step forward by fully automating the implementation of adaptive setpoint temperatures in EnergyPlus. It integrates every aspect of the workflow, from weather file preprocessing and adaptive setpoint implementation to the automated simulation and result visualization. By streamlining these processes, accim eliminates the manual steps required in other methods and significantly reduces the risk of errors. It also supports advanced customization, allowing users to define parameters such as comfort models, acceptability thresholds, and mixed-mode operation rules directly within the script. While accim requires some familiarity with Python programming, its comprehensive approach and scalability make it an efficient and accessible solution for balancing thermal comfort and energy efficiency across multiple locations and scenarios.

4. Discussion

The highly customizable input arguments that accim can admit coupled with the capability to compute the adaptive setpoint temperature values “on the go” provides a wide range of new opportunities to investigate the energy implications of adaptive thermal comfort in AC or MM buildings. In particular, new research possibilities are opened, considering the use of adaptive thermal comfort to alleviate energy poverty and mitigate the climate change. Since accim allows handling a large number of simulations using adaptive setpoint temperatures, it allows to perform studies with numerous locations, therefore re-scaling the possibilities from local to national, continental, or even global scope.

Classical investigations are based on the search for a case study, the development of a simulation model, its validation and obtaining results through limited approaches in operational patterns. With the newly developed methodology, the analysis approach can be virtually unlimited for operational patterns. The great versatility of the methodology allows it to be adapted to any operational pattern and adaptive thermal comfort model developed in future years. This allows future regional models to be developed to be included in energy analysis studies and not have a single focus on thermal comfort, as up to now. Through the developed methodology, the duality of energy analysis and thermal comfort can be implemented more efficiently in future research. In addition, the methodology seeks independence from commercial software and optimizes the use of the free EnergyPlus tool. This favours the use of the methodology, by having an open-source methodology accessible to everyone.

5. Limitations

Limitations are firstly related to the building energy model itself. There are two ways to use accim: using a model with no HVAC system at all, in which accim will add an autosizable Variable Refrigerant Flow (VRF) system for each occupied zone, or using a building energy model with a fully modelled HVAC system. In the latter, if the HVAC system is not properly set and sized, the number of hours exceeding heating and/or cooling setpoints (i.e. unmet hours) might be unacceptable.

Unrelated to the technical aspects of setpoint control there are a series of practical questions concerning the implementation of an adaptive setpoint temperature that remain unanswered. First, the adaptive comfort model was derived on field measurements in naturally ventilated buildings. While the adaptive comfort principles are assumed to apply to any occupant, the expectations of an occupant in an air-conditioned building may be more stringent compared to someone in a naturally ventilated building. The ±3.5K range embedded in the ASHRAE 55 adaptive model would likely be narrower in contexts with fewer adaptive opportunities. Second, recent attempts to understand long-term thermal comfort in air-conditioned buildings (Li et al. 2020) suggest that large variations in the indoor daily temperature range may lead to greater dissatisfaction for building occupants. While the magnitude of satisfactory temperature ranges is unclear, it’s unlikely to be the ±3.5K range permissible in the current adaptive model. Similarly, an acceptable rate of change in indoor temperatures across multiple days is not known but is assumed to be more conservative than that derived by the adaptive comfort model. Third, the limits of applicability in ASHRAE Standard 55 preclude the use of the adaptive comfort model in air-conditioned buildings. As a result, few engineers would consider implementing an experimental setpoint control without it being endorsed by the leading industry authority. Lastly, adaptive setpoint temperatures are based on the assumption that occupants will adapt to indoor temperature in fully air-conditioned or mixed-mode spaces as if they were on naturally ventilated spaces. Although there are strong evidence (Parkinson, de Dear, and Brager 2020; Sun et al. 2024; Yun, Lee, and Steemers 2016), experimental studies using adaptive comfort limits as setpoint temperatures are still needed to validate this approach.

6. Conclusion

BPS is a technique that is widely used to compare alternative design and retrofit choices and explore how buildings perform in terms of their environmental and energy implications. Additionally, adaptive comfort models have recently been put out as a strategy to reduce energy consumption, particularly when taking into account the usage of adaptive setpoint temperatures. However, using adaptive setpoint temperatures in building simulation requires a high level of technical expertise.

This paper firstly aims to provide an overview of the unique open-source software solution available for automating and facilitating building performance simulations - especially with adaptive setpoint temperatures - and result interpretation. Secondly, it aims to provide a seamless and accessible Python-based methodology for users with basic experience programming, in which code snippets can be easily re-used for different studies.

This research employs a case study to show the tool's capabilities. In this study, the proposed methodology is firstly applied to carry out the case study, which involves data pre-processing, running the simulations and lastly generating figures and tables to analyse the data. Secondly, the data analysis shows the following conclusions:

- In Ahmedabad, using mixed-mode adaptive setpoints (IND_Adap_MM) reduced cooling energy demand by up to 57% under present conditions and 44% in future scenarios (RCP85-2100).
- In Shimla, the same approach achieved cooling energy demand reductions of up to 65% in current climates and 61% in future projections.

Lastly, alternative methods are compared to accim. Unlike manual or partially automated approaches, accim offers a fully automated, error-resistant solution for implementing adaptive setpoint temperatures in building performance simulations. It streamlines the entire process—from data preparation to result analysis—reducing complexity, enhancing reproducibility, and supporting large-scale studies with minimal user intervention.

This tool will be helpful for policymakers, at the integration of homes' energy performance into the low-carbon built environment, in particular when considering future scenarios under the influence of climate change. Future lines of research might include the development of the tool to include new thermal comfort models from around the world, new arguments to define custom adaptive comfort models and a new module for parametric and optimisation studies.

Conflict of interest

No potential competing interest was reported by the authors.

Data availability

The data that support the findings of this study are available in the following website, which contains the EPW and IDF files as well as a Jupyter Notebook containing all code snippets:

https://github.com/dsanchez-garcia/accim/tree/master/accim/sample_files/jupyter_notebooks/research_paper_case_study_v0-7-3

Other data potentially interesting to the reader are:

- Github repository: https://github.com/dsanchez-garcia/accim/tree/master
- PyPI project: https://pypi.org/project/accim/
- Documentation: https://accim.readthedocs.io/en/master/

Acknowledgements

The authors acknowledge the support provided by the Thematic Network 722RT0135 "Red Iberoamericana de Pobreza Energética y Bienestar Ambiental (RIPEBA)" financed by the call for Thematic Networks of the CYTED Program for 2021. Also, the authors would like to acknowledge the Thematic Network 723RT0151 "Red Iberoamericana de Eficiencia y Salubridad en Edificios" (IBERESE) financed by the call for Thematic Networks of the CYTED Program for 2022 for supporting this research. Finally, the authors would like to thank Prof. Dr. Thomas Parkinson, from University of Sydney's IEQ Lab, for his assistance in the explanation of the assumption and limitations of adaptive setpoint temperatures.

Appendix A. Detailed methodology

At this appendix, the operation of all modules, classes, methods and functions will be explained in detail. Figure A1 shows all relevant elements of the project for this study. Since `accim.sim` has already been explained in a previous study, `accim.data` and `accim.run` will be emphasized. All explanations are addressed to accim version 0.7.3, therefore these might be different for previous and newer versions. Similar documentation for the latest release can be found at the website (see Section Data availability). In order to help the reader not to get lost among the modules, classes, methods and functions, all processes described below have been broken down into numbered lists consistently with the workflows (Figure A2). The coding of those numbered lists follows the structure:

A. (for each class, function, etc)

A.a (for every step within the process)

A.a.1 (when step above is divided in multiple branches)

A.a.1.1 (when step above is divided in multiple branches)

A.a.2 (and so on)

The methods, functions and classes have been enumerated based on the process that should be followed for developing a comfort study to be consistent with Section 3; however, the methodology is explained following the project structure in Figure A1: module `accim.sim` contains function B, module `accim.run` contains function C, and `accim.data` contains class A, class D and methods E to G.

```
accim                                          Color legend
+---data                                       Package
|   +---data_postprocessing.py                 Module
|   |   |   Table                              Python file
|   |       |   format_table                   Class
|   |       |   scatter_plot                   Method
|   |       |   scatter_plot_with_baseline     Function
|   |       |   time_plot
|   |       |   wrangled_table
|   +---data_preprocessing.py
|       |   rename_epw_files
+---sim
|   |   accim_Base.py
|   |   accim_Base.py
|   |   accim_Base_EMS.py
|   |   accim_ExistingHVAC.py
|   |   accim_ExistingHVAC_EMS.py
|   |   accim_IDFgeneration.py
|   |   accim_Main.py
|   |   accim_VRFsystem.py
|   |   accim_VRFsystem_EMS.py
|   |   accis.py
|       |   addAccis
+---run
    |   run.py
        |   runEp
```

Figure A1. Project structure.

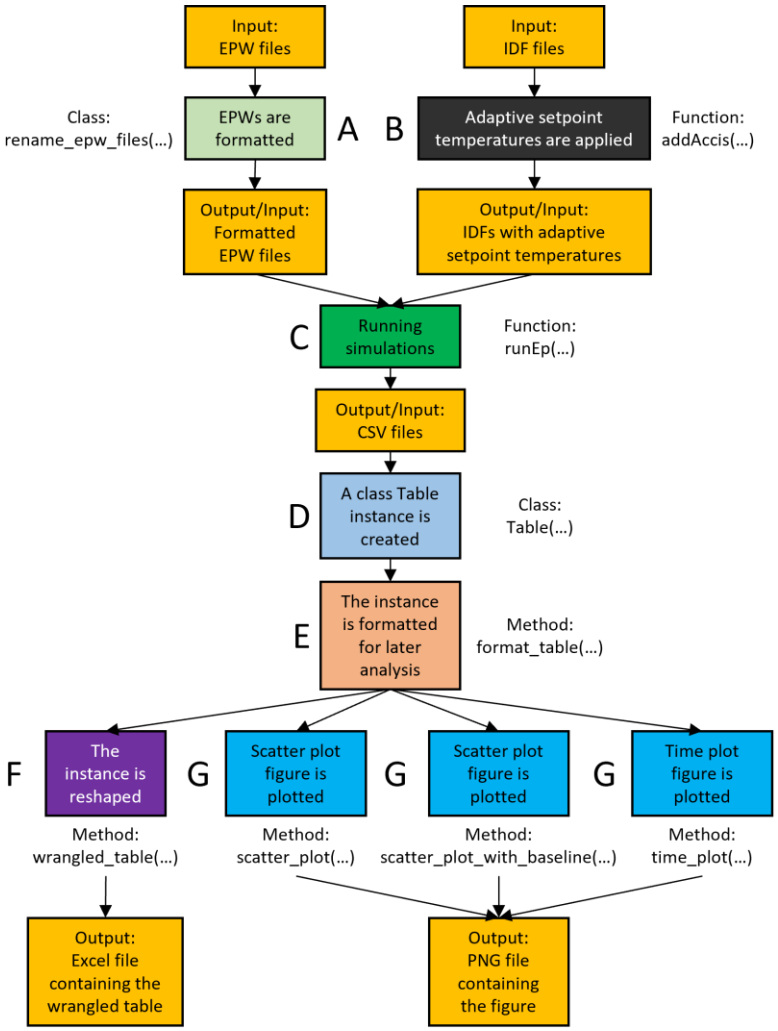

Figure A2. Flowchart of the entire process.

1. accim.data

This module contains 2 Python files: `data_preprocessing` and `data_postprocessing`. The first one is used to prepare the EPW files prior to running the simulations, while the second one is used to analyse the data resulting from the simulations. Both are explained in the following sections.

1.1. data_preprocessing

At the final stage, information related to the EPW files, such as location, RCP scenario and year (if climate change is considered) can be analysed. The data analytics module takes csv files with information delimited by a separator in its name, but to do so, the EPW file names need to be formatted prior to simulation following a certain pattern: Country_City_RCPscenario-Year.

To do so, the class `rename_epw_files` can be used. Once the class has been instantiated, following process can be divided in 2 stages: steps A.a. to A.f, in which new names for the EPW files are proposed (Figure A3); and steps A.g. to A.j., in which the user needs to review the proposed names and make any corrections if necessary, and confirm actions to be performed (Figure A4). EPW names can be very different mainly depending on the source. In this case, EPWs have been downloaded EnergyPlus and OneBuilding websites, as well as Meteonorm software. These different sources have been considered to minimise the number of unexpected issues that may arise from the different patterns they follow.

A.a. Existing EPW files in the folder are scanned, and a Pandas DataFrame instance (`epw_df`) is created to map the current EPW names.

A.b. If climate change is considered in the EPW, then the current file name should contain some data regarding the RCP scenario and the year. Therefore, the current file name is scanned to search for any matches between the scenarios and year. Only RCP 2.6, 4.5, 6.0 and 8.5, and years from 2000 to 2100 in 10-years intervals are looked for. This tool does not consider the previous scenarios published in the Special Report on Emissions Scenarios (SRES), since these are already superseded and should not be used. If no match is found, then accim assumes the EPW is for present scenario and informs the user. Then, accim updates the Pandas

DataFrame instance with this information, specifically the columns `EPW_scenario`, `EPW_year` and `EPW_scenario_year`. The data stored in `EPW_scenario_year` will be later used in the new EPW name, namely in the placeholder RCPscenario-Year.

A.c. Next step is defining information for the placeholders Country and City. To do so, accim opens every EPW file and extracts and stores the latitude and longitude.

A.d. Then accim obtains the address for these coordinates from OpenStreetMap. From that geographic data, accim extracts the country code. This is passed through `pycountry.countries.get()` to return the name of the country in English language, and be stored in column `EPW_country`, which will replace the placeholder Country. At this point, the address for those coordinates is also stored for later use.

A.e. For the city information, accim searches for any matched between substrings within the current name and geographical information. If no match is found, then the string "UNKNOWN" is stored in the column `EPW_city_or_subcountry`, which will similarly replace the placeholder City.

A.f. At this point, all required information to compose the new EPW names has been gathered (although there might be some "UNKNOWN" strings), and new names for the EPW files can be proposed.

A.g. Since the new EPW names have been already proposed, accim needs to check if there are duplicated EPW new names and if there is missing information (previous "UNKNOWN" strings). If so, these will need to be corrected by the user in the next step.

A.h. Then, the user is informed of the old and new EPW names. After reviewing them, the user is asked to enter the IDs of the EPWs which have not been properly renamed.

A.i. Then, for each ID the user has previously entered, accim provides the address previously stored in Stage 1. This information as well as the old EPW name, which should contain information of the city, is helpful for the user, which is asked to enter the correct city name for that EPW file. At this point, the user also is asked to enter the correct city names for duplicate EPWs or those with missing information.

A.j. Finally, accim informs the user of the full final list of EPW names. If some of these are not properly renamed yet, these can be excluded. The user is asked to enter the IDs if necessary and, at last, to confirm the copy and rename of the EPW files and deletion of the old ones.

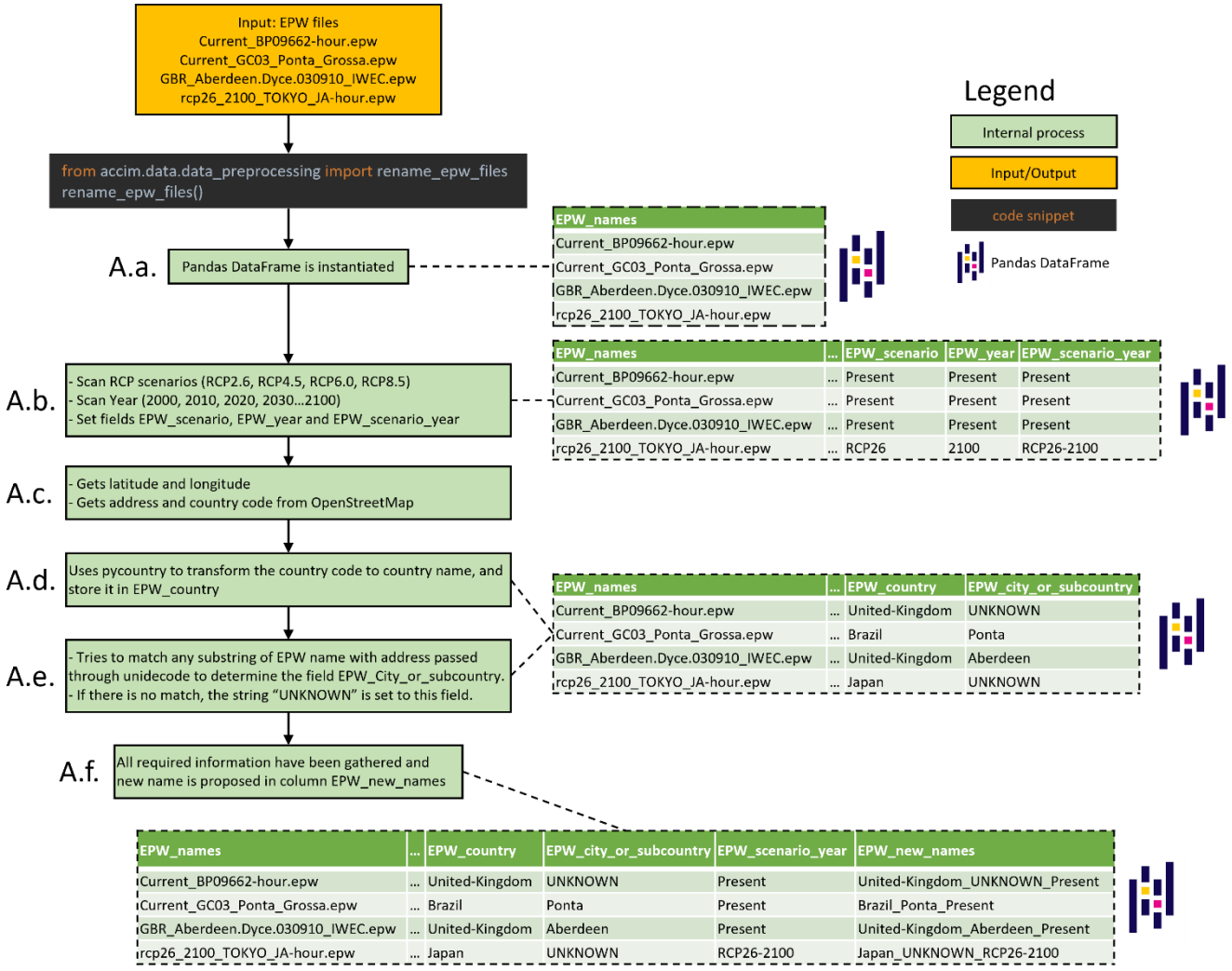

Figure A3. Flowchart for the steps of `rename_epw_files` (A), part 1.

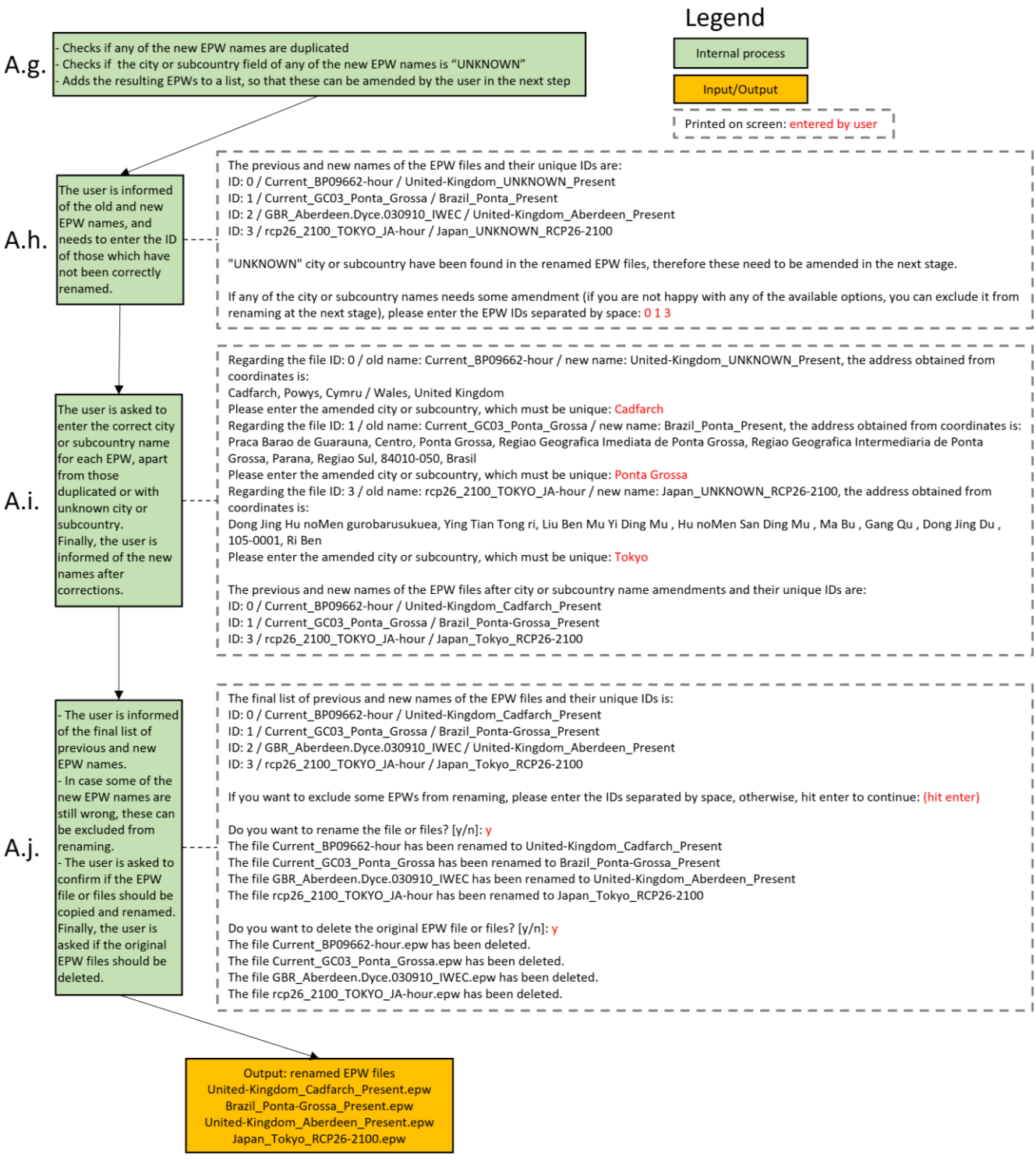

Figure A4. Flowchart for the steps of `rename_epw_files` (A), part 2.

1.2. data_postprocessing

The purpose of this module is to analyse and visualise the simulation results. Following sections provide a deeper insight of the processes at each step. This module is the only one that requires a basic understanding of Python syntax. The recommended procedure involves the use of an Integrated Development Environment (IDE).

1.2.1.Table

The class `Table` is used to generate a pandas DataFrame instance from the simulation results. The available arguments for class `Table` are shown in Table A1:

Table A1. Arguments for class `Table`.

Argument	type	Description	Admissible values
`datasets`	list	The list of csv files that need to be analysed. If omitted, accim will take into account all available csv files in the path.	Any list
`source_concatenated_csv_filepath`	string	The path of the csv already concatenated in case it has been generated	Any string
`source_frequency`	string	The frequency of the simulation results	"hourly", "daily", "monthly" or "runperiod"
`frequency`	string	The frequency of the analysed data	
`frequency_agg_func`	string	The function it performs when aggregating the csv rows	"sum" or "mean"
`standard_outputs`	bool	Used to filter only standard outputs for accim	True or False
`concatenated_csv_name`	string	accim concatenate the CSVs and export it with the entered name; if omitted, the csv is not exported	Any string
`level`	list	accim aggregates the zone results for block or building	"block" and/or "building"
`level_agg_func`	list	The function to be performed for level argument	"sum" and/or "mean"
`level_excluded_zones`	list	The list of zones that need to be excluded from level computations.	Any list
`block_zone_hierarchy`	dict	A dictionary to map the block/zone structure.	Any dict
`split_epw_names`	bool	Used to split the EPW name in the pattern Country_City_RCPscenario-Year	True or False
`normalised_energy_units`	bool	Used to show energy units per m2	True or False
`rename_cols`	bool	Used to rename the columns to a more friendly format	True or False
`energy_units_in_kwh`	bool	Used to show energy units in kWh	True or False

Figure A5 shows the process that takes place when a `Table` instance is created. Given there are some CSV files at the path where the `Table` instance is created, which are going to be taken as input files, the process comprises the following steps, named consistently with Figure A5:

D.a. Once the `Table` class has been instantiated (and stored in variable `dataset_runperiod` in this example), after some actions to avoid later errors, accim needs to know what the source of the data is.

D.a.1. If the argument `source_concatenated_csv_filepath` has been entered when the class was instantiated, then accim will set some variables (`source_frequency`, `frequency`, `frequency_agg_func` and `standard_outputs`) based on the concatenated csv file name and create a DataFrame instance.

D.a.2. Otherwise, accim checks if the datasets argument has been entered when the class was instantiated.

D.a.2.1. If so, accim stores the csv files which have been specified in the list of the argument datasets.

D.a.2.2. Otherwise, accim gets all available csv files in the folder which meets certain requirements in its name, in order to avoid other csv files resulting from EnergyPlus simulations, such as those related to zone sizing (ending with Zsz.csv) or tables (ending with Table.csv).

D.a.2.3. Then, accim starts to iterate through every csv file from previous step. The loop includes, among other actions, to create a DataFrame instance, to filter the columns, to define which type of aggregation will be performed for each column, and finally, to perform the aggregations of rows. An example of this would be, defining that operative temperature should be averaged, and then average the rows to transform, for instance, hourly information into daily information. At last, concatenates all instances into a single DataFrame.

D.b. Once the DataFrame has been instantiated, named `df`, accim performs a number of operations focused to detect if there have been errors when aggregating the rows.

D.c. Then, accim needs to know if the argument `concatenated_csv_name` was entered when the class was instantiated.

D.c.1. If so, accim exports the DataFrame instance into a csv file. This file is aimed to be read at step A.1, and therefore, allows to avoid unnecessary computational effort by skipping the A.2 branch if `Table` needs to be instantiated more than once. Afterwards, accim continues to next step.

D.d. In this step, accim tries to detect the hierarchical pattern of blocks and zones (i.e. which zones belong to each block), which is going to be needed to compute totals for block and/or building levels. In case of IDFs generated with Designbuilder, the zones are generally named following the pattern “BlockX:ZoneY”, therefore probably only in this case the hierarchy would be detected. If it is not detected, the user is asked to enter all blocks names, and then, for each block, all thermal zones. If the user previously knows the IDF has been generated with other GUI, such as OpenStudio, and therefore the hierarchy is not going to be detected, it is possible to skip the information request by specifying the hierarchy as dictionary object in argument `block_zone_hierarchy`.

D.e. Next, the user can exclude zones from level computations. This is useful, for instance, to exclude a naturally-ventilated space from the average of operative temperature in air-conditioned spaces. To do so, the user can use the argument `level_excluded_zones`, which takes a list of strings. If this argument is omitted, then the user will be asked to enter the zones to be excluded if necessary. Finally, accim will check the zones entered by the user actually exists in the data.

D.f. At this point, the block/zone hierarchy and the excluded zones are known, and therefore, the level computations can be performed. The tool will detect if block totals have been requested in argument `level`, and if so, the function to be performed in argument `level_agg_func` (sum and/or mean). For instance, accim can sum all zones belonging to a certain block to provide the block total for a certain output. A similar approach is taken also for building level.

D.g. Then, energy units are updated based on the user requests at arguments `normalised_energy_units` and `energy_units_in_kwh`.

D.h. If the user requested normalising the energy units in argument `normalised_energy_units`, then accim will divide the energy outputs by floor area at zone, block and building levels.

D.i. If requested at argument `split_epw_names`, accim will split the column EPW to make the columns for country, city and RCP scenario and year. To work properly, the EPW file name should be in the CSV file name, following the pattern Country_City_RCPscenario-Year, as previously shown in function `rename_epw_files`.

D.j. Finally, if requested at argument `rename_cols`, all columns are renamed, so that the output is more user-friendly and easier to read and understand.

Therefore, the output of this process is a pandas DataFrame instance which can be accessed by the user at the variable `df`. In the example shown in Figure A5, the input csv files number is 4. Therefore, since “runperiod” have been specified at the argument frequency, rows have been aggregated grouped by the run period, resulting in 4 rows at the variable `df`. For instance, if “monthly” were specified, the number of rows of `df` would be 4*12=48, i.e. one row per month. The number of columns in `df` is 99, which varies depending on multiple factors, such as the number of zones in the building, the building and/or block level aggregations and the functions requested, among others.

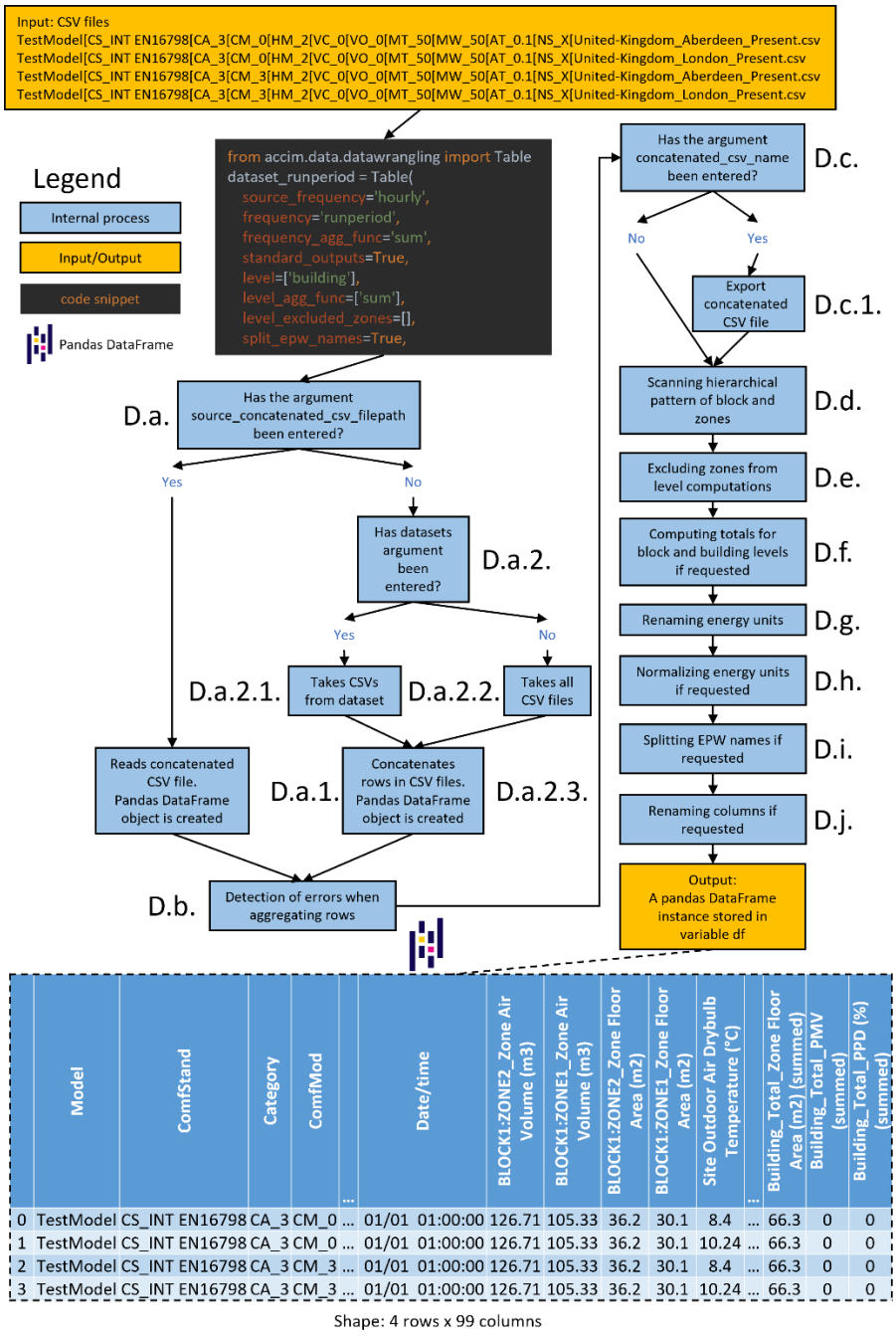

Figure A5. Flowchart for class `Table`.

1.2.2.format_table

`format_table` is a method of class `Table`. It is used to prepare the DataFrame instance `df` for later analysis. Usually, `df` has an important number of columns which usually make it difficult to be handled. Therefore, this method filters the columns to keep those specified by the user. Arguments of `format_table` are shown in Table A2.

Table A2. Arguments of `format_table`.

Argument	Type	Description	Admissible values
`type_of_table`	string	Used to get already defined tables	“all”, “energy demand”, “comfort hours”, “temperature” or “custom”
`custom_cols`	list	Used if custom is used in `type_of_table`. The list of columns to keep.	Any list

Figure A6 shows the process that takes place then the `format_table` method is called:

E.a.First, accim sets the index columns. These columns will not be deleted under any circumstances.

E.b. Then, depending on the value entered at argument `type_of_table`, different columns are filtered. Entering certain arguments returns a predefined range of columns, which allows for an easier use of the method.

E.b.1. If it is “all”, accim will return all columns.

E.b.2. If it is “temperature”, accim will filter all columns in `df` that contains certain temperature terms.

E.b.3. If it is “comfort hours”, accim will filter all columns in `df` that contains certain comfort terms.

E.b.4. If it is “energy demand”, accim will filter all columns in `df` that contains certain energy demand terms.

E.b.5. If it is “custom”, then accim will search for the columns entered in the argument `custom_cols` in all columns of the DataFrame instance `df`, and it will filter them.

Finally, the DataFrame instance `df` is returned with filtered columns, which are those extracted from csv file names (treated as categorical values) and the ones requested when calling the method (treated as numerical values). In this example, the latter are:

- “Building_Total_Cooling Energy Demand (kWh/m2) (summed)”.
- “Building_Total_Heating Energy Demand (kWh/m2) (summed)”.

At this point, the variable `df` is ready to be wrangled with the method `wrangled_table` or otherwise, it is ready to be the source of data to be visualised with methods `scatter_plot`, `scatter_plot_with_baseline` or `time_plot`.

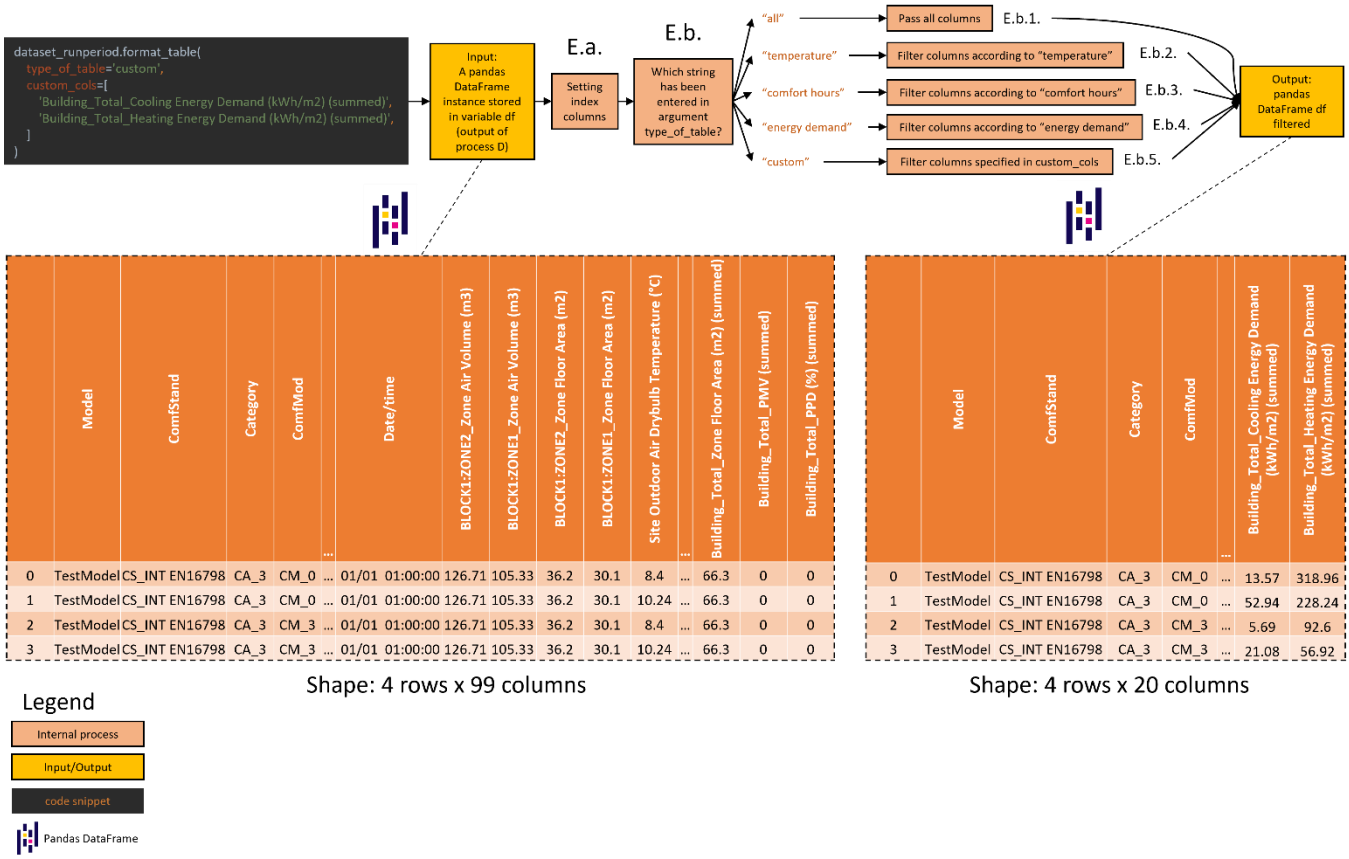

Figure A6. Flowchart for method `format_table`.

1.2.3. wrangled_table

Once the DataFrame `df` has been formatted with `format_table`, it can be reshaped or pivoted with `wrangled_table`, to finally create an Excel file containing the data. Arguments of `wrangled_table` are shown in Table A3.

Table A3. Arguments of `wrangled_table`.

Argument	Type	Description	Admissible values
`reshaping`	string	Used to specify the reshaping method. Operation similar to pandas.	"pivot", "stack", "unstack" or “multiindex”
`vars_to_gather`	list	The list of variables to be analysed. It will combine the values of the variables entered into a single column.	Any list

baseline	string	The instance of all different combinations of variables that will be used as the baseline for comparison	Any string
comparison_cols	list	To calculate absolute or relative (%) differences.	"absolute" and/or "relative"
comparison_mode	list	The point of view to compare the data. Absolute and/or relative differences can be calculated based on the following logic: - for "others compared to baseline" o if "relative", 1-(variant/baseline) o if "absolute", baseline-variant - for "baseline compared to others", o if "relative", 1-(baseline/variant) o if "absolute", variant-baseline	"others compared to baseline" and/or "baseline compared to others"
check_index_and_cols	bool	Used if the user is not sure about the variables to be entered in vars_to_gather	True or False
vars_to_keep	list	The variable or variables to be shown in multiindex	Any list
rename_dict	dictionary	Used to replace strings in the entire DataFrame (values, index or rows, and columns). The dictionary should be in the format {"old string": "new string"}	Any dict
excel_filename	string	The name of the exported excel file	Any string

Figure A7 shows the process that takes place when the method wrangled_table is called. Before calling the method, the user should be aware of the possibilities for analysing the data. For instance, following the example of the 4 CSV files, only 2 of the variables separated by delimiter "[" change:

- ComfMod, where possibilities are "CM_0" and "CM_3" (refers respectively to static and adaptive setpoint temperatures).
- EPW, specifically the field EPW_city_or_subcountry, where the possibilities are "Aberdeen" and "London".

Therefore, all other variables do not seem interesting to be analysed. After calling the method, the following process starts:

F.a. The tool needs to know what reshaping method should use. At this point, index has already been set based on the variables to be analysed. Therefore, if "multiindex" is used, accim does not perform any reshaping at all. It only provides a cleaned view of the DataFrame.

F.a.1. If "pivot", accim will call the DataFrame method pivot_table. At this point, the user must consider that not specifying some variable in argument vars_to_gather will result in the sum of the rows with the same value for that variable. For instance, in Figure A7, EPW_city_or_subcountry has not been specified in vars_to_gather, and therefore, for each possibility of ComfMod, the values of "Aberdeen" and "London" will be summed.

F.a.2. If "unstack", accim will call the DataFrame method unstack. Therefore, accim will keep the variables at vars_to_gather in the columns, and move the variables at vars_to_keep to the rows (or index).

F.a.3. If "stack", accim will call the DataFrame method stack. Therefore, accim will move all the variables in the columns to the rows, and keep a single column for all numerical values. This is useful for plotting data with libraries such as seaborn, since stacked data is the most suitable format.

F.b. The next step consists of the computation of comparison columns based on the data entered in arguments comparison_mode, comparison_cols and baseline, and only takes place if reshaping method is "pivot" or "unstack". In this example, all comparison columns were requested for all comparison modes. For instance, considering the mode "others compared to baseline", the comparison columns relative and absolute allows to know the difference between the baseline ("CM_3") and all other possibilities (in this case only "CM_0") in terms of percentage (1-(CM_0/CM_3), in %) and the meter of the variable (in this case, CM_3 – CM_0, in kWh/m²·year). In this example, the relative and absolute differences show respectively an increase of 1.48 (i.e. 148%) and 39.74 kWh/m²·year of "CM_0" respect the baseline "CM_3".

F.c. Then, all data, columns and indexes of the DataFrame are renamed following the dictionary containing old and new strings entered in argument rename_dict. This allows to improve the readability of the output file (although the user should know at this point "CM_0" and "CM_3" respectively refer to static and adaptive setpoint temperatures, others might not).

F.d. Finally, accim exports the DataFrame instance to an Excel file named after the argument excel_filename.

In this case, the output of this method is an Excel file named "testing_accim.xlsx", and therefore it means this branch of data analysis ends at this point.

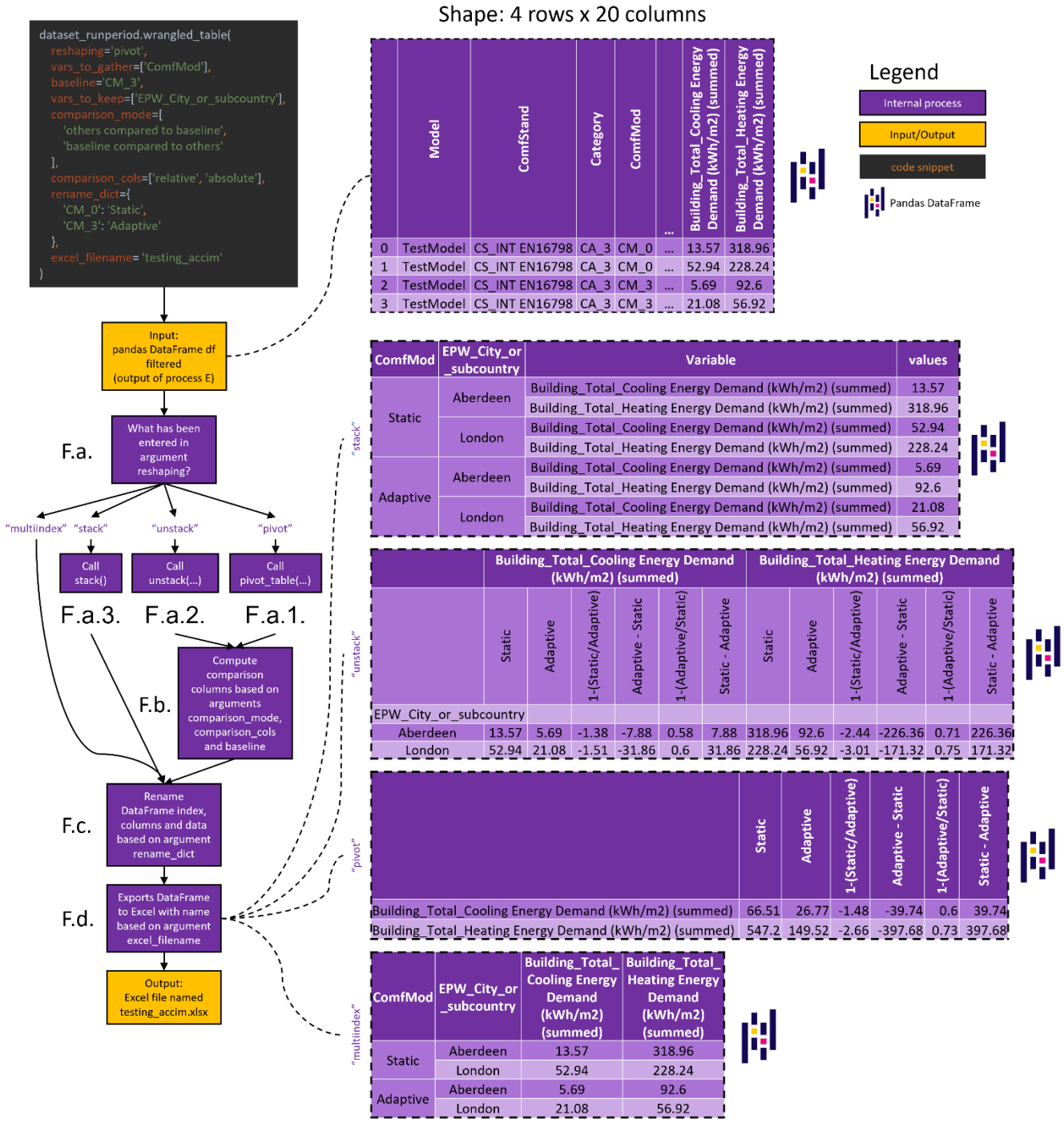

Figure A7. Flowchart for method `wrangled_table`.

1.2.4.scatter_plot, scatter_plot_with_baseline and time_plot

Continuing from the end of `format_table`, data can be visualised. There are currently 3 methods to do so: `scatter_plot`, `scatter_plot_with_baseline` and `time_plot`, whose arguments are shown in Table A4. These figures are computed with matplotlib library, and can be composed of subplots as a function of the variables entered in arguments `vars_to_gather_cols` and `vars_to_gather_rows`. If not all the values resulting from the combination of variables are needed, the user can filter which values should be plotted in columns and rows respectively with the arguments `detailed_cols` and `detailed_rows`, as well as the order of those, with the arguments `custom_cols_order` and `custom_rows_order`. The process is shown in Figure A8, and it is composed of 2 stages: in the first one, (G.a. in flowchart) data to be plotted is gathered and organised in lists, while in the second one (G.b.), subplot axes are created, and data is finally plotted. However, this process is not relevant for the purpose of this research and has no novelty, therefore is not further explained.

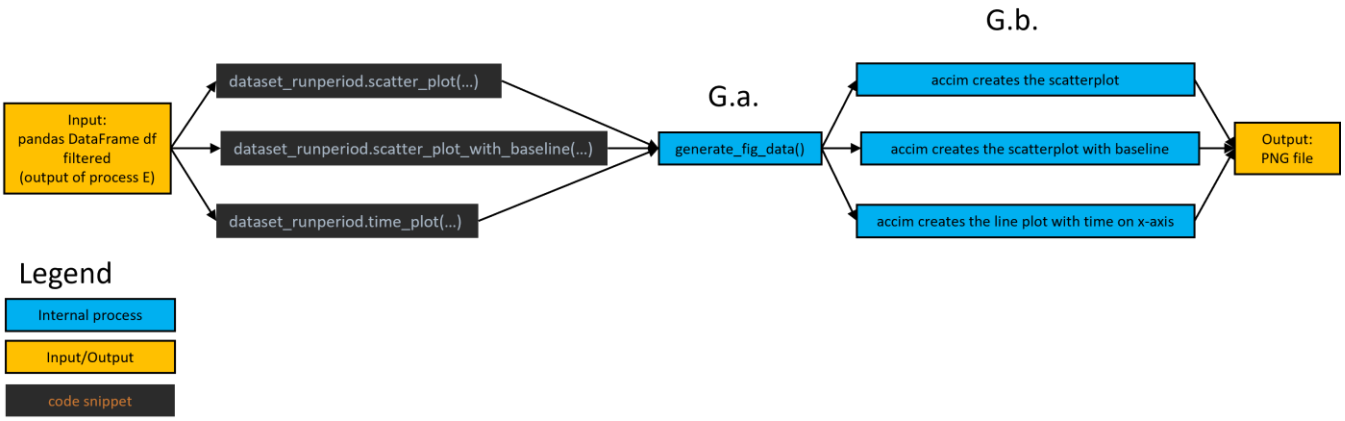

Figure A8. Flowchart of methods `scatter_plot`, `scatter_plot_with_baseline` and `time_plot`.

Table A4. Arguments in common of all figure methods.

Method	Argument	Type	Description
all	`vars_to_gather_cols`	list	The list of variables to be gathered in columns
	`vars_to_gather_rows`	list	The list of variables to be gathered in rows
	`detailed_cols`	list	The list of specific values of `vars_to_gather_cols` to be plotted. Only used if not all values need to be plotted.
	`detailed_rows`	list	The list of specific values of `vars_to_gather_rows` to be plotted. Only used if not all values need to be plotted.
	`custom_cols_order`	list	An ordered list of column names from left to right.
	`custom_rows_order`	list	An ordered list of rows names from top to bottom.
	`cols_renaming_dict`	dictionary	Used to replace strings in the cols. The dictionary should be in the format {"old name": "new name"}
	`rows_renaming_dict`	dictionary	Used to replace strings in the rows. The dictionary should be in the format {"old name": "new name"}
	`figname`	str	Used as the name of the PNG file to be generated
	`figsize`	float	Used as the size of the figure
	`dpi`	int	Used to specify the figure resolution
	`confirm_graph`	bool	Used to skip figure confirmation.
`scatter_plot`	`data_on_x_axis`	string	The column name to be plotted on x-axis
`scatter_plot` and `time_plot`	`data_on_y_main_axis`	list	Multiple spines on the y-axis can be plotted. Therefore, lists with nested lists must be entered following the pattern: [['name_on_1st_y_main_axis', [list of column names you want to plot]], ['name_on_2nd_y_main_axis', [list of column names you want to plot]], etc]
	`data_on_y_sec_axis`	list	Similar to `data_on_y_main_axis`, but in secondary axis
	`colorlist_y_main_axis`	list	A structure similar to `data_on_y_main_axis`, but replacing the column names with the related colour using matplotlib colour notation
	`colorlist_y_sec_axis`	list	A structure similar to `data_on_y_sec_axis`, but replacing the column names with the related colour using matplotlib colour notation
	`ratio_height_to_width`	float	Used to specify the shape of the figure. The height will be multiplied by the number to provide the final height.
	`supxlabel`	str	The label shown on the x-axis
`scatter_plot_with_baseline`	`data_on_y_axis_baseline_plot`	list	The columns to be plotted on y-axis. A list with column names must be entered following the pattern: ['first_column_name', 'second_column_name', etc]
	`colorlist_baseline_plot_data`	list	Similar to `data_on_y_axis_baseline_plot`, but replacing column names with colors using matplotlib color notation
	`baseline`	string	The value resulting from `vars_to_gather_cols` to be compared with all other values

The method `scatter_plot` is used to plot a scatter plot, as the name suggests. It allows to plot data in multiples spines in y-axis. The method `scatter_plot_with_baseline` is also used to plot a scatter plot, but in this case, it is mainly used to compare a value resulting from `vars_to_gather_cols`, used as a baseline, with all other values resulting from that combination of variables gathered in columns. Lastly, `time_plot` is used to make a line plot figure with time on the x-axis. The output of all methods is a PNG file.

2. accim.sim

This module contains several Python files, although only `accis.py` contains a function that can be used: `addAccis`. As the name suggests, this function is used to add an EMS script named Adaptive-Comfort-Control-Implementation Script, which applies adaptive setpoint temperatures based on highly customisable input arguments specified by the user.

The operation of this function is described in Figure A9:

B.a. Given there is a path.
B.b. In which one or multiple IDF files are located.
B.c. The user needs to call that function, either specifying all arguments as shown in the figure, or no argument at all (i.e., `accis.addAccis()`). In the latter, the user will be requested to enter all information guided by clarificatory text in the Python console or CMD terminal.
B.d. Then, accim will generate all output IDFs and will name them based on the arguments entered by the user, to finally save them in the same path where input IDFs were located.

As stated above, this module has already been explained in previous studies (Sánchez-García, Bienvenido-Huertas, and Rubio-Bellido 2021; Sánchez-García, Martínez-Crespo, et al. 2023), therefore no further details are necessary.

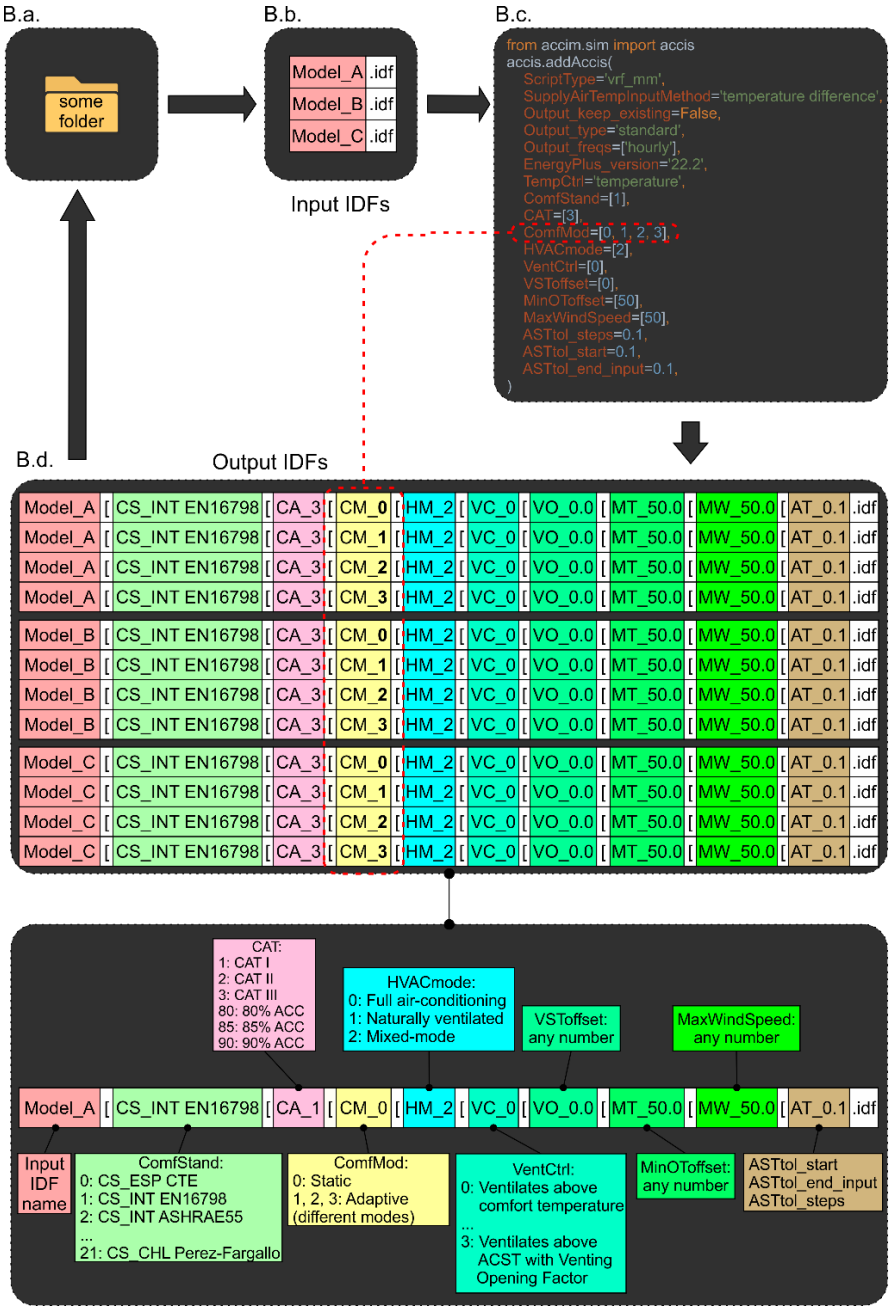

Figure A9. `addAccis` workflow.

3. accim.run

The tool also has a module to perform building energy simulations with EnergyPlus, named `accim.run`. This module contains a Python file, `run.py`, which contains among other functions, a function named `runEp`. This function allows to automatically run all combinations of IDF and EPW files within the path where the function is executed based on arguments (Table A5) specified by user, and names the output files following the pattern "IDF[EPW", therefore using the character "[" as a separator consistently with the whole package. It has been developed using eppy code (Santosh 2023) as a reference, but with modifications to suit the name requirements.

Table A5. Arguments of `runEp`.

Argument	Type	Definition	Admissible values
`runOnlyAccim`	bool	Allows to filter IDFs to only simulate outputs of accim	True or False
`confirmRun`	bool	Allows to skip run confirmation on terminal	True or False
`num_CPUs`	integer	The number of CPUs to be used	Any integer
`EnergyPlus_version`	string	The version of EnergyPlus. It must match the IDF version.	9.1 to 23.1

Taking the renamed EPW files from process A and the IDFs with adaptive setpoint temperatures from process B as input files, in this case the process consists of the following steps, named consistently with Figure A10:

C.a. If argument `EnergyPlus_version` has not been entered, accim asks the user to enter that information on CMD terminal. Then, it searches the IDD file in the default installation path, and stores it in a variable using the eppy method `setiddname`.

C.b. If the user has not confirmed the simulation of only output IDFs from accim with argument `runOnlyAccim`, then accim asks for it on CMD terminal. Then, if requested, accim filters the IDFs.

C.c. Then, accim informs the user of the simulation runs that are going to be performed, which consists of the combination of all renamed EPWs and filtered IDFs within the path where function is being executed. If the user has not confirmed the simulation runs with argument `confirmRun`, then accim asks for it on CMD terminal.

C.d. Finally, if confirmed, accim proceeds with simulation runs, using the number of CPUs specified in argument `num_CPUs`.

The output files are the typical files resulting from EnergyPlus simulation. However, the files that are needed in this procedure are the CSV files, which are ready to be analysed with the data module.

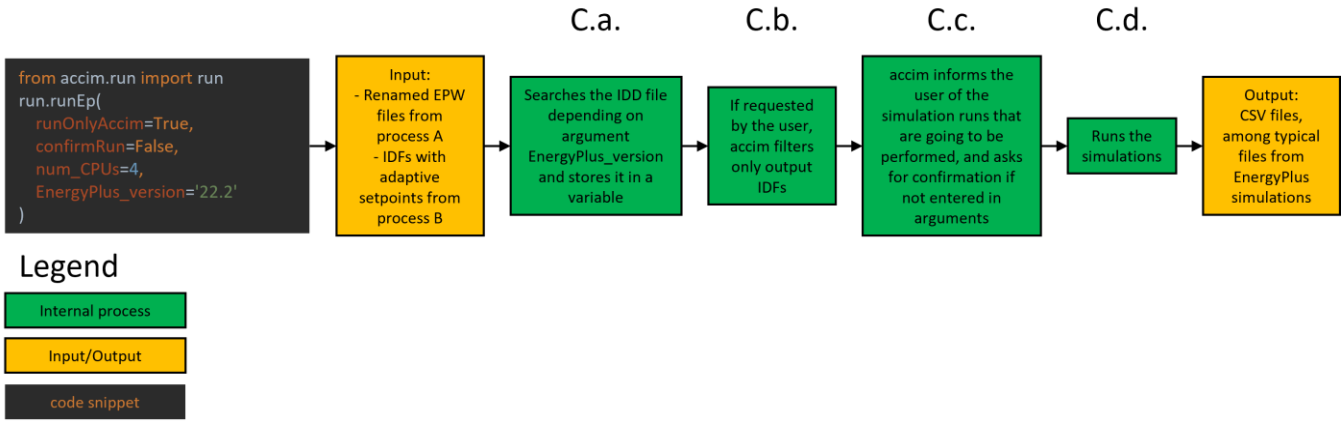

Figure A10. Flowchart of `runEp`.

Appendix B. In-depth details of the methodology based on accim

B.1. Data pre-processing

Prior to simulation, files need to be pre-processed. EPW files need to be renamed following the pattern Country_City_RCPscenario-Year, as previously mentioned. Also, adaptive setpoint temperatures need to be implemented in the IDF files.

B.1.1. EPW files preparation

EPW files can be renamed using the class `rename_epw_files`. The recommended process, suitable for users with no programming background, is:

a. open a CMD dialog pointing at the path where the EPW files to be renamed are located.
b. execute Python by entering “py” or “python”.
c. import the package by entering “`from accim.data.data_preprocessing import rename_epw_files`”.
d. call the function by entering “`rename_epw_files()`”.
e. enter the required information on CMD dialog.

Therefore, after instantiating the class, accim needs to interact with the user throughout the process, shown in Figure B1. The tool will try to identify if the EPW files are for some future RCP scenario, and if no match is found, it considers these for present scenario. Then, based on the addresses obtained from the coordinates, proposes new names for the EPW files, and asks the user if some of these need to be amended, and then, if some of these need to be excluded. In this case, the tool renamed correctly all EPW files at the first instance and no amendments were required. Therefore, the new EPW names are: India_Ahmedabad_Present, India_Shimla_Present, India_Ahmedabad_RCP85-2100 and India_Shimla_RCP85-2100.

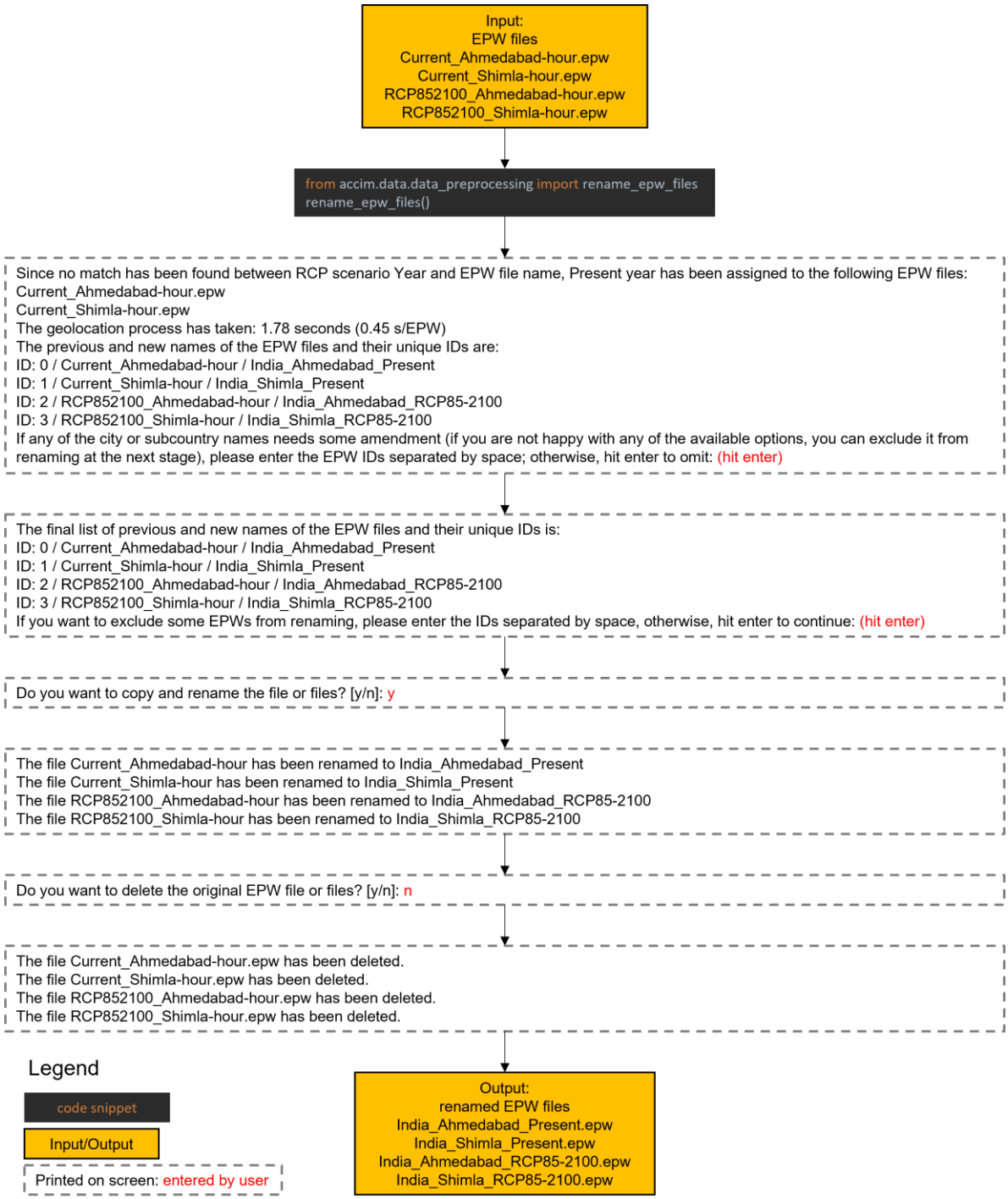

Figure B1. Flowchart for EPW formatting procedure using the class `rename_epw_files`

B.1.2. ACCIS implementation

The next step is applying the adaptive setpoint temperatures. To do so, the class `addAccis` needs to be used. Again, the recommended process, suitable for users with no programming background, is:

a. open a CMD dialog pointing at the path where the IDF files to be handled are located.
b. execute Python by entering “py” or “python”.
c. import the package by entering “`from accim.sim import accis`”.
d. instantiate the class by entering “`accis.addAccis()`”.
e. enter the required information on CMD dialog (related to the arguments that have not been specified)

In this case study, all arguments are specified as shown in Figure B2. As a result, 12 output IDFs are generated, although not all of them are necessary. The IDFs that are needed are listed in Table 2. Thus, the remaining should be removed to avoid unnecessary computational effort.

Figure B2. Flowchart for the implementation of the ACCIS using the class `addAccis`.

Table 2. Output IDFs from accim for the case study.

ComfStand	ComfMod	HVACmode	Definition	Acronym
CS_IND IMAC C NV	CM_3	HM_2	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in MM operation	IND_Adap_MM
		HM_1	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in NV operation	Ind_Adap_NV
		HM_0	adaptive setpoints based on IMAC-C horizontally extended beyond applicability limits, in AC operation	IND_Adap_AC
	CM_0	HM_0	PMV-based or nearly static setpoint temperatures based on Indian Building Code, in AC operation	IND_Stat_AC
CS_INT ASHRAE55	CM_3	HM_0	adaptive setpoints based on ASHRAE 55 horizontally extended beyond applicability limits, in AC operation	ASH_Adap_AC

B.2. Running simulations

Once EPWs have been formatted and adaptive setpoint temperatures have been implemented in the output IDFs, simulations can be automatically run using the function `runEp`. Again, the recommended process, suitable for users with no programming background, is:

a. open a CMD dialog pointing at the path where the renamed EPWs and IDFs are located.
b. execute Python by entering “py” or “python”.
c. import the package by entering “`from accim.run import run`”.
d. call the function by entering “`run.runEp()`”.
e. enter the required information on CMD dialog (related to the arguments that have not been specified).

Once the function has been called, the user might need to interact with accim to provide any missing argument. In this case, no interaction is needed, since all arguments have been specified as shown in Figure B3. The simulation output files have been saved in the same path. In this case study 20 simulations have been run (5 IDFs x 4 EPWs), therefore there must be 20 CSV files ready to be analysed.

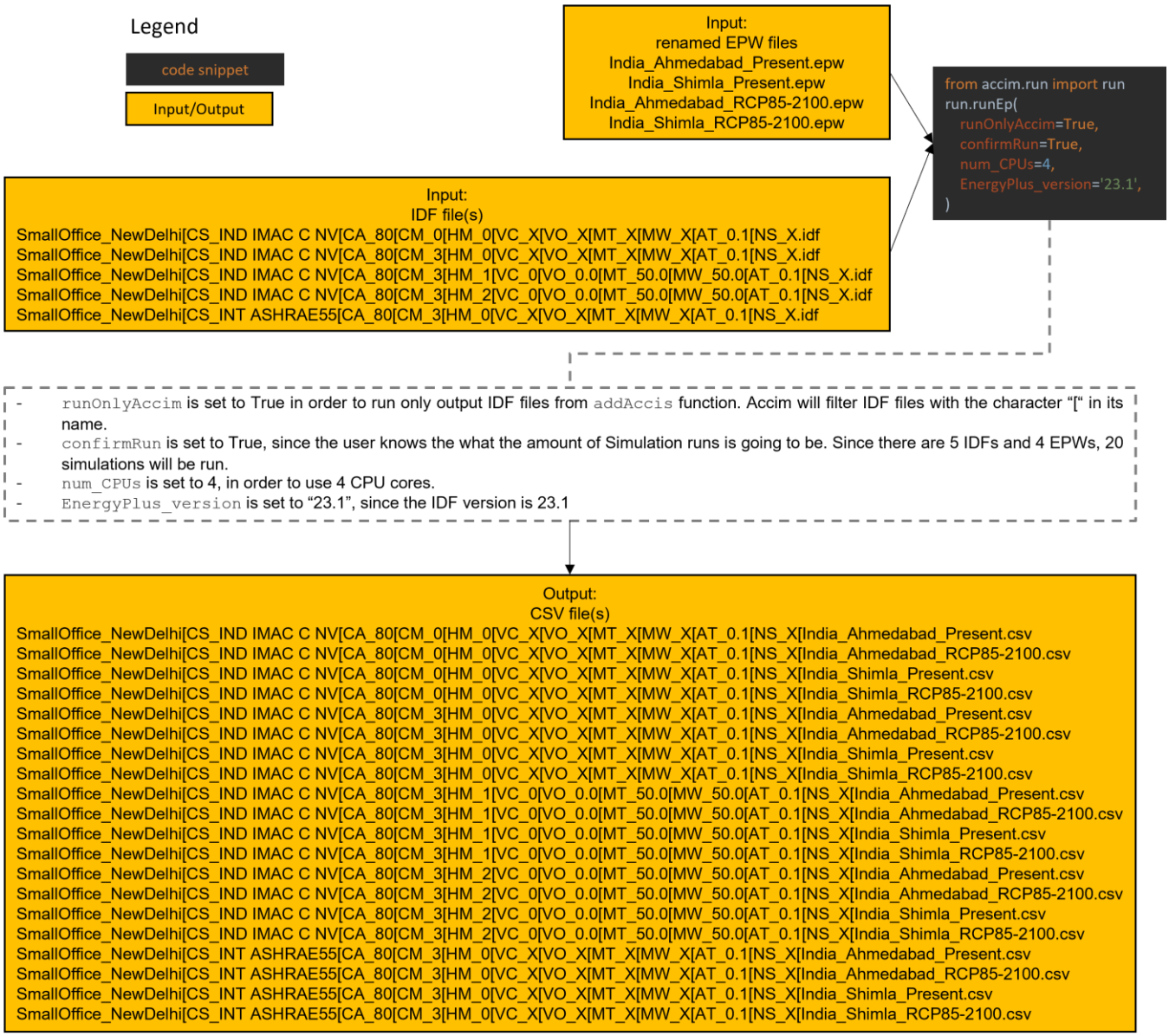

Figure B3. Flowchart for the simulation runs using the function `runEp`.

B.3. Data analysis

In the following subsections, the CSV files resulting from the simulation runs will be analysed, to provide a table to assess the energy demand differences, and to visualize the data in multiple figures. In this case, it is recommended to use an IDE instead of a CMD dialog.

B.3.1. Tables

The aim of this subsection is to create a table to understand how much energy is demanded in the different settings. To do so, the first step would be to generate a DataFrame instance using the class `Table`, which is going to be stored in variable `dataset_runperiod` as shown in Figure B4. Then, the method `format_table` needs to be executed, to filter only energy demand columns at building level, as shown in Figure B5.

Figure B4. Flowchart for the instantiation of the class `Table`

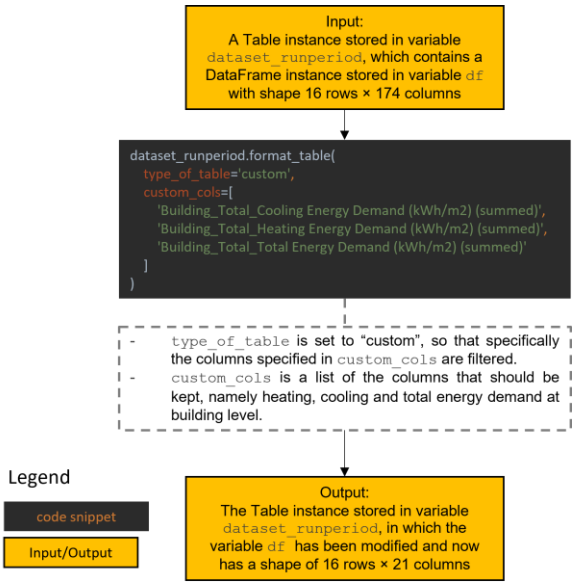

Figure B5. Flowchart for the use of `format_table`

At this point, the user might have a vague idea of what table can be done with the available data, especially if there is a large number of categorical variables. In order to get a clearer idea of this, the user can call the method named `gather_vars_query`. This method has not been previously explained in the methodology since the aim is merely assist the user to clarify the analysis possibilities. In this case, when IDFs with adaptive setpoint were generated, the arguments where more than one value were requested were columns `ComfStand`, `ComfMod` and `HVACmode` (ComfStand=[2, 7], ComfMod=[0, 3], HVACmode=[0, 1, 2]). Thus, these are all the categorical variables that change regarding the IDFs and the possibilities that might be interesting to study, and therefore the variables that have been entered in `vars_to_gather`. Then, accim prints on screen the categorical variables that contains more than one different value (i.e. `ComfMod`, since values are "CM_0" and "CM_3"), and the different combinations based on the combined variables, joined by character "[":

- CS_IND IMAC C NV[CM_0[HM_0.
- CS_IND IMAC C NV[CM_3[HM_0.
- CS_IND IMAC C NV[CM_3[HM_2.
- CS_INT ASHRAE55[CM_3[HM_0.

Then, the wrangled table can be generated following Figure B6.

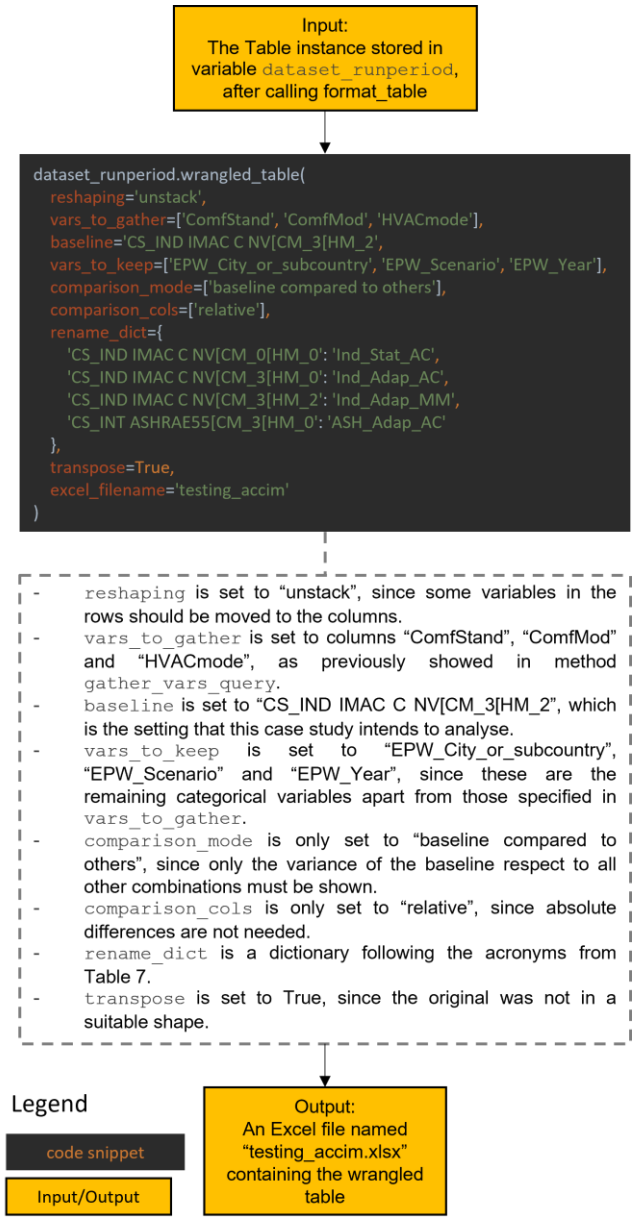

Figure B6. Flowchart for the use of method `wrangled_table`

Finally, an Excel spreadsheet named "testing_accim.xlsx" is created and saved in the same path where the method was called. This output has been broken down into 2 tables for clarity purposes: Table 5 shows the energy demand values obtained and Table 6 the percentage obtained with the use of MM. For instance, the latter shows that IND_Adap_MM provides savings in thermal energy demand ranging from 24 to 65% with respect to IND_Adap_AC depending on the location and climate scenario.

B.3.2. Data visualization

The aim of this subsection is to plot figures that represent the results of the building performance simulations with adaptive setpoint temperatures. Therefore, all 3 methods previously explained at the methodology section are applied. These need to be called after method `format_table`. However, the class `Table` instance created at the previous section had runperiod frequency, which would be translated into very simple and not interesting figures, and besides NV mode was not considered. Therefore, a new instance will be created in this case using hourly frequency, as shown on Figure B7. In this case, `format_table` will be called filtering the columns for the mean operative temperature of all air-conditioned zones, the adaptive setpoint temperatures, a value of running mean outdoor temperature of a single zone (since it is the same for all of them), and the building total heating and cooling energy demands.

Figure B7. Flowchart for the instantiation of Table using hourly frequency, and use of method format_table.

Once `format_table` has been called, the methods to create the figures can follow. The user might expect to obtain figures composed of multiple subplots. First, `scatter_plot` method will be used, in which the typical plot to show the linear regression of adaptive comfort models will be generated. As specified in the arguments in Figure B8, the variables `ComfStand`, `ComfMod` and `HVACmode` were gathered and the different combinations are plotted in rows, while `EPW_City_or_subcountry` and `EPW_Scenario-Year` combinations are plotted in the columns. Data on main y-axis (left) was heating and cooling energy demand with best fit lines, while setpoint temperatures and operative temperature were plotted on the secondary y-axis (right), following the colours specified in the arguments. Considering the `frequency` input argument was set to “hourly” when `Table` class was instantiated, all values refer to hourly basis. Finally, these were renamed following the dictionaries entered. The output is shown in Figure 3, and allows the user to compare indoor temperature values from the use of HVAC systems with adaptive setpoint temperatures based on whether IMAC-C or ASHRAE 55 with roughly static setpoints from the Indian Building Code (in top row of subplots) or no HVAC system at all (i.e. NV and free-running mode, in middle row of subplots). In the latter, the comfortable hours ranged from 37% in Ahmedabad in RCP8.5 2100 to 84% in Shimla Present, while in all other air-conditioned cases, the comfortable hours ranged from 99.88% to 100%.

Figure B8. Flowchart for the use of method `scatter_plot`

Using a similar code snippet but changing method to `time_plot` and omitting the argument `data_on_x_axis`, a figure with time on the x-axis can be generated (Figure B9). In this case, the argument `sharex` is also omitted, since the simulation period is the same for all cases. The output is shown in Figure 4.

Figure B9. Flowchart for the use of method `time_plot`

Finally, using `scatter_plot_with_baseline` as shown in Figure B10, a different scatter plot can be generated. In this case, it is mainly used to compare a baseline combination of variables, plotted on x-axis, with the remaining variants, plotted on y-axis. In the example below, the total heating and cooling energy demands will be compared using the combination “CS_IND IMAC C NV[CM_3[HM_2” (or IND_Adap_MM) as the baseline, since it is expected to provide the greater energy savings. The same `Table` instance will be used, however, to avoid plotting an empty energy demand figure for the combination “CS_IND IMAC C NV[CM_3[HM_1” (or IND_Adap_NV) since it is on naturally ventilated mode, the argument `detailed_cols` is used. In this argument, all the combinations to be plotted are specified, that is, all of them except the NV and the baseline. The output is represented in Figure 5, which compares the hourly energy demand values of the baseline combination “CS_IND IMAC C NV[CM_3[HM_2” (or IND_Adap_MM) against all other combinations. For instance, in this case, the figure shows that there is an important number of hourly cooling energy demand values moving away from the diagonal (or 100% line, in which values are similar for the baseline and other combinations) and gathering around the 25% line, in which baseline values are roughly a quarter of the other combinations.

Figure B10. Flowchart for the use of the method `scatter_plot_with_baseline`

References

Allouhi, A., Y. El Fouih, T. Kousksou, A. Jamil, Y. Zeraouli, and Y. Mourad. 2015. “Energy Consumption and Efficiency in Buildings: Current Status and Future Trends.” *Journal of Cleaner Production* 109. Elsevier Ltd: 118–130. doi:10.1016/j.jclepro.2015.05.139.

ANSI/ASHRAE. 2020. *ASHRAE Standard 55-2020 Thermal Environmental Conditions for Human Occupancy*. Edited by ASHRAE Inc. *ASHRAE Standard*. Atlanta, GA, United States.

Beckman, William A., Lars Broman, Alex Fiksel, Sanford A. Klein, Eva Lindberg, Mattias Schuler, and Jeff Thornton. 1994. “TRNSYS The Most Complete Solar Energy System Modeling and Simulation Software.” *Renewable Energy* 5 (1–4). Pergamon: 486–488. doi:10.1016/0960-1481(94)90420-0.

Bernal, Willy, Madhur Behl, Truong X. Nghiem, and Rahul Mangharam. 2012. “MLE+: A Tool for Integrated Design and Deployment of Energy Efficient Building Controls.” *BuildSys 2012 - Proceedings of the 4th ACM Workshop on Embedded Systems for Energy Efficiency in Buildings*, 123–130. doi:10.1145/2422531.2422553.

Bienvenido-Huertas, David, Daniel Sánchez-García, and Carlos Rubio-Bellido. 2021. “Adaptive Setpoint Temperatures to Reduce the Risk of Energy Poverty? A Local Case Study in Seville.” *Energy and Buildings* 231 (January). Elsevier B.V.: 110571. doi:10.1016/j.enbuild.2020.110571.

Bienvenido-Huertas, David, Daniel Sánchez-García, Carlos Rubio-Bellido, and David Marín-García. 2021. “Potential of Applying Adaptive Strategies in Buildings to Reduce the Severity of Fuel Poverty According to the Climate Zone and Climate Change: The Case of Andalusia.” *Sustainable Cities and Society* 73 (October): 103088. doi:10.1016/j.scs.2021.103088.

Bienvenido-Huertas, David, Daniel Sánchez-García, Carlos Rubio-Bellido, and Jesús A. Pulido-Arcas. 2021. “Applying the Mixed-Mode with an Adaptive Approach to Reduce the Energy Poverty in Social Dwellings: The Case of Spain.” *Energy* 237 (December): 121636. doi:10.1016/j.energy.2021.121636.

Big Ladder Software Ltd. “Modelkit.” https://bigladdersoftware.com/projects/modelkit/.

Bureau of Indian Standards. 2016. *National Building Code of India Volume 2*. India.

Crawley, Drury B., Linda K. Lawrie, Frederick C. Winkelmann, W. F. Buhl, Y. Joe Huang, Curtis O. Pedersen, Richard K. Strand, et al. 2001. “EnergyPlus: Creating a New-Generation Building Energy Simulation Program.” *Energy and Buildings* 33 (4): 319–331. doi:10.1016/S0378-7788(00)00114-6.

de Dear, Richard J., and Gail Schiller Brager. 1998. “Developing an Adaptive Model of Thermal Comfort and Preference.” *ASHRAE Transactions* 104 (Pt 1A): 145–167. http://www.scopus.com/inward/record.url?scp=0031624196&partnerID=8YFLogxK.

DesignBuilder Software Ltd. 2021. “DesignBuilder.” https://designbuilder.co.uk.

Dhaka, Shivraj, Jyotirmay Mathur, and Vishal Garg. 2012. “Combined Effect of Energy Efficiency Measures and Thermal Adaptation on Air Conditioned Building in Warm Climatic Conditions of India.” *Energy and Buildings* 55. Elsevier B.V.: 351–360. doi:10.1016/j.enbuild.2012.09.038.

European Commission. 2002. *Directive 2002/91/EC of the European Parliament and of the Council of 16 December 2002 on the Energy Performance of Buildings*. *Official Journal of the European Communities*. Vol. 1. Brussels, Belgium.

European Commission. 2006. *Action Plan for Energy Efficiency: Realising the Potential*. Brussels, Belgium.

European Commission. 2011. “A Roadmap for Moving to a Competitive Low Carbon Economy in 2050.” Brussels, Belgium. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52011DC0112.

European committee for standardization. 2019. “EN 16798-1:2019 Energy Performance of Buildings. Ventilation for Buildings. Indoor Environmental Input Parameters for Design and Assessment of Energy Performance of Buildings Addressing Indoor Air Quality, Thermal Environment, Lighting and Acoustics.” https://en.tienda.aenor.com/norma-bsi-bs-en-16798-1-2019-000000000030297474.

European Environment Agency. 2017. *Final Energy Consumption by Sector and Fuel (2017)*. Copenhagen, Denmark.

European Union. 2010. *Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the Energy Performance of Buildings*. *Official Journal Of The European Union*. Vol. 153. Brussels, Belgium.

Földváry Ličina, Veronika, Toby Cheung, Hui Zhang, Richard de Dear, Thomas Parkinson, Edward Arens, Chungyoon Chun, et al. 2018. “Development of the ASHRAE Global Thermal Comfort Database II.” *Building and Environment* 142 (September). Pergamon: 502–512. doi:10.1016/J.BUILDENV.2018.06.022.

Guglielmetti, Rob, Dan Macumber, and Nicholas Long. 2011. “Openstudio: An Open Source Integrated Analysis Platform.” In *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, 442–449. https://www.osti.gov/biblio/1032670.

Hirsch, J.J. “EQUEST: The QUick Energy Simulation Tool.” http://www.doe2.com/equest/.

Hong, Tianzhen, Jared Langevin, and Kaiyu Sun. 2018. “Building Simulation: Ten Challenges.” *Building Simulation*, 871–898. https://doi.org/10.1007/s12273-018-0444-x.

Hoyt, Tyler, Edward Arens, and Hui Zhang. 2015. “Extending Air Temperature Setpoints: Simulated Energy Savings and Design Considerations for New and Retrofit Buildings.” *Building and Environment* 88 (June). Elsevier Ltd: 89–96. doi:10.1016/j.buildenv.2014.09.010.

Intergovernmental Panel on Climate Change. 2007. *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*.

Intergovernmental Panel on Climate Change. 2014. *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Edited by Intergovernmental Panel on Climate Change. *Climate Change 2013 - The Physical Science Basis*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781107415324.004.

Kramer, R. P., M. P.E. Maas, M. H.J. Martens, A. W.M. van Schijndel, and H. L. Schellen. 2015. “Energy Conservation in Museums Using Different Setpoint Strategies: A Case Study for a State-of-the-Art Museum Using Building Simulations.” *Applied Energy* 158. Elsevier Ltd: 446–458. doi:10.1016/j.apenergy.2015.08.044.

Li, Peixian, Thomas Parkinson, Stefano Schiavon, Thomas M. Froese, Richard de Dear, Adam Rysanek, and Sheryl Staub-French. 2020. “Improved Long-Term Thermal Comfort Indices for Continuous Monitoring.” *Energy and Buildings* 224. Elsevier: 110270. doi:10.1016/j.enbuild.2020.110270.

Manu, Sanyogita, Yash Shukla, Rajan Rawal, Leena E. Thomas, and Richard de Dear. 2016. “Field Studies of Thermal Comfort across Multiple Climate Zones for the Subcontinent: India Model for Adaptive Comfort (IMAC).” *Building and Environment* 98 (March). Elsevier Ltd: 55–70. doi:10.1016/j.buildenv.2015.12.019.

Monge Palma, Rafael, José Sánchez Ramos, María del Carmen Guerrero Delgado, Teresa Rocío Palomo Amores, Laura Romero Rodríguez, and Servando Álvarez Domínguez. 2023. “Extending the Thermal Comfort Band in Residential Buildings: A Strategy towards a Less Energy-Intensive Society.” *Applied Sciences* 13 (12): 7020. doi:10.3390/app13127020.

Mui, Kwok Wai Horace, and Wai Tin Daniel Chan. 2003. “Adaptive Comfort Temperature Model of Air-Conditioned Building in Hong Kong.” *Building and Environment* 38 (6). Pergamon: 837–852. doi:10.1016/S0360-1323(03)00020-9.

Parkinson, Thomas, Richard de Dear, and Gail Brager. 2020. “Nudging the Adaptive Thermal Comfort Model.” *Energy and Buildings* 206 (January). Elsevier B.V.: 109559. doi:10.1016/j.enbuild.2019.109559.

Roudsari, Mostapha Sadeghipour. “Ladybug Tools.” https://www.ladybug.tools/.

Sánchez-García, Daniel. 2021. “Accim’s Documentation.” https://accim.readthedocs.io/en/latest/index.html.

Sánchez-García, Daniel, David Bienvenido-Huertas, Jesús A. Pulido-Arcas, and Carlos Rubio-Bellido. 2023. “Extending the Use of Adaptive Thermal Comfort to Air-Conditioning: The Case Study of a Local Japanese Comfort Model in Present and Future Scenarios.” *Energy and Buildings* 285 (April): 112901. doi:10.1016/j.enbuild.2023.112901.

Sánchez-García, Daniel, David Bienvenido-Huertas, and Carlos Rubio-Bellido. 2021. “Computational Approach to Extend the Air-Conditioning Usage to Adaptive Comfort: Adaptive-Comfort-Control-Implementation Script.” *Automation in Construction* 131 (November). Elsevier: 103900. doi:10.1016/j.autcon.2021.103900.

Sánchez-García, Daniel, David Bienvenido-Huertas, Mónica Tristancho-Carvajal, and Carlos Rubio-Bellido. 2019. “Adaptive Comfort Control Implemented Model (ACCIM) for Energy Consumption Predictions in Dwellings under Current and Future Climate Conditions: A Case Study Located in Spain.” *Energies* 12 (8). MDPI AG: 1498. doi:10.3390/en12081498.

Sánchez-García, Daniel, Jorge Martínez-Crespo, Ulpiano Ruiz-Rivas Hernando, and Carmen Alonso. 2023. “A Detailed View of the Adaptive-Comfort-Control-Implementation Script (ACCIS): The Capabilities of the Automation System for Adaptive Setpoint Temperatures in Building Energy Models.” *Energy and Buildings* 288 (June): 113019. doi:10.1016/j.enbuild.2023.113019.

Sánchez-García, Daniel, Carlos Rubio-Bellido, Juan Jesús Martín del Río, and Alexis Pérez-Fargallo. 2019. “Towards the Quantification of Energy Demand and Consumption through the Adaptive Comfort Approach in

Mixed Mode Office Buildings Considering Climate Change." *Energy and Buildings* 187 (March). Elsevier Ltd: 173–185. doi:10.1016/j.enbuild.2019.02.002.

Sánchez-Guevara Sánchez, Carmen, Anna Mavrogianni, and Fco Javier Neila González. 2017. "On the Minimal Thermal Habitability Conditions in Low Income Dwellings in Spain for a New Definition of Fuel Poverty." *Building and Environment* 114: 344–356. doi:10.1016/j.buildenv.2016.12.029.

Santosh, Philip. 2023. "Running EnergyPlus from Eppy." *Eppy Documentation*. Accessed May 3. https://eppy.readthedocs.io/en/latest/runningeplus.html.

Santosh, Philip, Tran Tuan, Eric Allen Youngson, and Jamie Bull. 2004. "Eppy Web Repository." https://github.com/santoshphilip/eppy.

Schild, P.G. "EpXL: EnergyPlus-Excel." https://github.com/SchildCode/EpXL.

Sun, Ruiji, Stefano Schiavon, Gail Brager, Edward Arens, Hui Zhang, Thomas Parkinson, and Chenlu Zhang. 2024. "Causal Thinking: Uncovering Hidden Assumptions and Interpretations of Statistical Analysis in Building Science." *Building and Environment* 259 (July). Elsevier Ltd. doi:10.1016/j.buildenv.2024.111530.

University of Strathclyde. 2002. *The ESP-r System for Building Energy Simulation*.

U.S. Department of Energy. 2023. "Prototype Building Models." Accessed June 3. https://www.energycodes.gov/prototype-building-models#Commercial.

Wang, Chenli, Kaleb Pattawi, and Hohyun Lee. 2020. "Energy Saving Impact of Occupancy-Driven Thermostat for Residential Buildings." *Energy and Buildings* 211 (March). Elsevier B.V.: 109791. doi:10.1016/j.enbuild.2020.109791.

Wetter, Michael. 2001. "GenOpt - A Generic Optimization Program." *Seventh International IBPSA Conference*, no. August: 601–608.

Yi, Z. "JEplus." http://www.jeplus.org/wiki/doku.php.

Yun, Geun Young, Je Hyeon Lee, and Koen Steemers. 2016. "Extending the Applicability of the Adaptive Comfort Model to the Control of Air-Conditioning Systems." *Building and Environment* 105. Elsevier Ltd: 13–23. doi:10.1016/j.buildenv.2016.05.027.