Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Computational approach to extend the air-conditioning usage to adaptive comfort: Adaptive-Comfort-Control-Implementation Script

Daniel Sánchez-García [*], David Bienvenido-Huertas, Carlos Rubio-Bellido

Department of Building Construction II, University of Seville. Seville, Spain

ARTICLE INFO

Keywords:
Building performance simulation
Building energy modelling
Adaptive thermal comfort
Energy efficiency
Adaptive setpoint temperatures
Energy management system
Python

ABSTRACT

Recently, the energy saving potential from using setpoint temperatures based on adaptive comfort has been studied. This study proposes a computational approach, the Adaptive-Comfort-Control-Implementation Script (ACCIS), to extend the air-conditioning usage to adaptive comfort. ACCIS transforms PMV-based into adaptive setpoint building energy models according to both an Input Data File (IDF) and the setup specified by the user. Originally, ACCIS was an Energy Management System (EMS) script, but available functions have been extended, and ACCIS has been nested in an ease-to-use Python package called Adaptive Comfort Control Implemented Model ("*accim*"). A case study has been tested, whose results showed that adaptive setpoint temperatures could achieve an 83% of energy savings. However, its most powerful attribute is that it allows many simulations to be run with no limit because of both its high customisation properties and the fact that it allows the same IDF to be run with various EPW files.

1. Introduction

For years, climate change is not a future problem, but a current one. The effects resulting from climate change, which is originated by the decisions taken by the human being without considering the environmental impact, are worsening year by year [1]. The temperature increase could oscillate between values close to 2.5 °C in 2050, 4.5 °C in 2100, 7.5 °C in 2200, and 8 °C in 2300 [2], taking as reference the mean surface temperature from the preindustrial period. To regulate both pollutant gas emissions and resource depletion, the building sector is required to reduce greenhouse gas emissions by 90% by 2050 [3]. According to the Fifth Assessment Report (AR5) of the Intergovernmental Panel on Climate Change (IPCC), 32% of the global primary energy was used in buildings in 2010, thus producing 19% of global emissions. The 2020 climate and energy package was developed in 2008 to reduce both the global temperature increase and climate change effects [4], and in 2013 the action period was extended to 2030 [5]. One of the main reasons of these emissions is the high energy consumption produced by the building sector [6]. The deficient energy performance of most building stock influences its high energy consumption [7,8]. The Directive 2018/844 [20] has recently established the need of European countries to develop energy renovation strategies for the existing building stock to have energy efficient buildings before 2050. To achieve this goal, building energy consumption should be reduced, mainly the use of air conditioning systems [9]. In this regard, energy consumption measures should be applied to reduce the energy consumption of air conditioning systems.

On the other hand, buildings are refuges of the outdoor conditions. Air-conditioning systems are used to maintain the indoor temperature within acceptable limits. However, the use of these systems, generally with restrictive setpoint temperatures that are very comfortable for the human body, could decrease people's capacity to adapt to ambient temperature variations physiologically, psychologically and in a behavioural way [10], apart from producing high energy consumption. The impact of the variation of setpoint temperatures on both energy consumption and thermal comfort has been widely studied: (i) Giorgos N. Spyropoulos and Constantinos A. Balaras [11] studied the effect of readjusting setpoint temperatures in bank branches in Greece. To reduce energy consumption, these heating and cooling setpoints were adjusted to 20 °C and 26 °C respectively, thus implying a mean energy saving of 18% of the total consumption (56 kWh/m^2); (ii) Tyler Hoyt et al. [12] studied the energy saving obtained by increasing heating and cooling setpoint temperatures through a parametric analysis. The cooling setpoint of 22.2 °C was increased to 25 °C, and the heating setpoint of 21.1 °C was decreased to 20 °C, so a mean reduction of energy consumption of 27 and 34% was respectively obtained; (iii) Parry et al. [13] obtained an annual energy saving of 66% by increasing the cooling setpoint

[*] Corresponding author at: Higher Technical School of Building Engineering, Ave. Reina Mercedes 4A, Seville, Spain.
E-mail address: dsanchez7@us.es (D. Sánchez-García).

https://doi.org/10.1016/j.autcon.2021.103900
Received 2 June 2021; Received in revised form 7 August 2021; Accepted 14 August 2021
Available online 27 August 2021

Nomenclature

ACCIM	Adaptive-Comfort-Control-Implemented Model
ACCIS	Adaptive-Comfort-Control-Implementation Script
ACST	Adaptive Cooling Setpoint Temperature
AHST	Adaptive Heating Setpoint Temperature
AR5	Fifth Assessment Report
AST	Adaptive Setpoint Temperature
BES	Building Energy Simulation
COP	Coefficient of Performance
CV(RMSE)	Coefficient of Variation of the Root Mean Square Error
ECM	Energy Conservation Measure
EER	Energy Efficiency Ratio
EMS	Energy Management System
EPW	EnergyPlus Weather
ERL	EnergyPlus Runtime Language
GUI	Graphical User Interface
IDF	Input Data File
IPCC	Intergovernmental Panel on Climate Change
MBE	Mean Bias Error
PMOT	Prevailing Mean Outdoor Temperature
PMV	Predicted Mean Vote
RMOT	Running Mean Outdoor Temperature
VST	Ventilation Setpoint Temperature

temperature between 2 and 4 °C in an office building in Zurich; (iv) Wan et al. [14] determined that cooling setpoint temperatures greater than 25.5 °C obtained savings in the cooling demand, both in current and future scenarios; (v) Lakeridou et al. [15] observed that thermal comfort levels were not reduced by increasing the cooling setpoint temperature by 2 °C in a field study carried out in an administrative building; and (vi) Fernández et al. [16] obtained an energy saving that oscillated between 12 and 20% by applying some energy conservation measures (ECMs), including the decrease of heating setpoint temperatures and the increase of cooling setpoint temperatures by 1.1 °C.

As previously mentioned, energy consumption was reduced by varying the setpoint temperatures, although these temperatures were static (i.e., based on the PMV index). Recently, many studies have been focused on mitigating this problem by extending the air-conditioning usage to adaptive comfort, i.e., by using adaptive setpoint temperatures (AST). These AST are setpoint temperatures based on adaptive thermal comfort algorithms, and their values correspond to those from upper and lower comfort limits (see Fig. 1). Therefore, since the application of AST implies that operative temperature would fluctuate within the adaptive thermal comfort zone, it is expected to reduce the energy consumption because of the use of less-restricted setpoint temperatures while keeping acceptable thermal human thermal sensation: (i) R.P. Kramer et al. [17] studied the energy saving in a museum in Amsterdam by using various strategies with setpoint temperatures and their respective adaptive thermal comfort level. The value of the lower comfort limit was assigned to the heating setpoint temperature, and HVAC systems were only used in the opening hours of the museum, thus implying an energy saving of 74%; (ii) Sánchez-Guevara Sánchez et al. [18] applied setpoint temperatures based on the comfort limits of the simplified model from the ASHRAE Standard 55–2013 [19], which monthly varies, in three residential buildings located in various cities of Spain. Energy consumption was reduced by both 20% and 80% in heating and cooling, respectively Therefore, adaptive comfort processes are significant for the occupants of all buildings, including those buildings that are air conditioned [20]. Table 1 summarises some of the relevant research articles related to this computational approach. Considering a great energy saving potential relies in this energy conservation measure, it could help many buildings, specially offices [21], to reduce the energy consumption [22].

To develop the previous studies, building energy simulations (BES) were mainly carried out with DesignBuilder [26], which uses the simulation engine EnergyPlus [27]. Moreover, various methods were used to determine the setpoint temperatures: on the one hand, very rudimentary methods, such as the monthly simulation by changing setpoint temperatures and the subsequent annual calculation by summing the results of the 12 simulations, and on the other hand, more effective methods, such as Schedule:Compact, which are objects of EnergyPlus that define in detail setpoint temperatures and operation schedules. However, a long, tedious and time-consuming process was manually conducted, thus constituting error-prone. For example, with Schedule:Compact objects, setpoint temperatures, mainly calculated with an Excel spreadsheet, were copied and pasted in the Schedule: Compact object, the respective EPW file was then assigned, and the combination of setpoint temperature and climate zone was simulated. Other file management tasks were also required. These tasks were carefully performed as there were many files because the studies were based on many combinations of setpoint temperatures and climate zone.

This paper therefore proposes a new computational approach called Adaptive-Comfort-Control-Implementation Script (ACCIS) to extend the air-conditioning usage to adaptive thermal comfort. ACCIS has been developed in the framework established by some research studies, the Adaptive-Comfort-Control-Implemented Model (ACCIM). Some parameters have been studied, constituting the input arguments related to Comfort Mode [23], Category [24] and HVAC mode [25]. Moreover, some previous versions of ACCIS have been used, thus making possible to perform the studies based on many simulations in different locations and AST such as 780 [28] or 48,786 [29] different location-AST combinations. Currently, the software or procedure presented in this paper is the only one. Other methods achieve the same results; however, these methods are manual and therefore time-consuming, tedious and error-prone. Besides, there is no restriction regarding the complexity of the building energy simulation models. That means any materials, constructive features and human activities and building functions can be considered, as long as these are correctly modelled. The study is structured as follows: Section 1 includes the introduction and literature review; Section 2 includes the concepts related to ACCIS and its development; Section 3 describes the case studies, their results and discussion; Section 4 includes the discussion; and finally, Section 5 draws the conclusions.

2. Methodology

2.1. Previous considerations

2.1.1. Adaptive thermal comfort models

Adaptive thermal comfort models have been developed upon the adaptability of building users to external climate fluctuations. These models are mainly applied in naturally ventilated buildings [30]. ASHRAE 55-2020 [31] and EN 16798-1:2019 [32] are internationally recognized standards considered as key documents in their field. The latter is the European standard, which establishes 3 categories according to users' thermal adaptability (T 1). More specifically, each category is based on users' thermal expectations according to the type of building: Category I is applicable to vulnerable users with limited thermal adaptability (e.g., the elderly), Category II is applicable to new buildings, and Category III to existing buildings. Each category establishes an upper and lower limit for the indoor operative temperature (Eqs. (1)–(7)). These limits are calculated using the running mean outdoor temperature (RMOT) (Eq. (8)), which is determined by the weighted average of daily external temperatures. Thermal adaptability finds application under mild external temperatures, and RMOT should be between 10 and 30 °C according to the European model (EN 16798-1:2019).

$$\textit{Optimal comfort temperature} = 0.33 \cdot \mathrm{RMOT} + 18.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{1}$$

$$\textit{Upper limit}\ (\textit{Category I}) = 0.33 \cdot \mathrm{RMOT} + 20.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{2}$$

$$\textit{Lower limit}\ (\textit{Category I}) = 0.33 \cdot \mathrm{RMOT} + 15.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{3}$$

$$\textit{Upper limit}\ (\textit{Category II}) = 0.33 \cdot \mathrm{RMOT} + 21.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{4}$$

$$\textit{Lower limit}\ (\textit{Category II}) = 0.33 \cdot \mathrm{RMOT} + 14.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{5}$$

$$\textit{Upper limit}\ (\textit{Category III}) = 0.33 \cdot \mathrm{RMOT} + 22.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{6}$$

$$\textit{Lower limit}\ (\textit{Category III}) = 0.33 \cdot \mathrm{RMOT} + 13.8\ [^{\circ}\mathrm{C}]\ (10 \le \mathrm{T_{rm}} \le 30) \tag{7}$$

$$\mathrm{RMOT} = \left(T_{ext,d-1} + 0.8T_{ext,d-2} + 0.6T_{ext,d-3} + 0.5T_{ext,d-4} + 0.4T_{ext,d-5} + 0.3T_{ext,d-6} + 0.2T_{ext,d-7}\right)/3.8\ [^{\circ}C] \tag{8}$$

On the other hand, the adaptive thermal comfort model from ASHRAE Standard 55-2020 is widely applied at an international level. It establishes two types of limits according to the percentage of acceptability in the indoor space: 80 and 90% (Eqs. (9)–(12)). The correlations used to determine upper and lower limits are different from the limits used by the European standards. In this standard, the weighted average of daily external temperatures is called prevailing mean outdoor temperature (PMOT), although the same calculation is made. Likewise, the range in which $t_{pma(out)}$ should oscillate varies in ASHRAE 55-2020: PMOT should oscillate between 10 and 33.5 °C for both the upper and lower limit.

$$\textit{Upper limit}\ (80\%\textit{acceptability}) = 0.31 \cdot \mathrm{PMOT} + 21.3\ [^{\circ}\mathrm{C}]\ \left(10 \le \mathrm{t_{pma(out)}} \le 33.5\right) \tag{9}$$

$$\textit{Lower limit}\ (80\%\textit{acceptability}) = 0.31 \cdot \mathrm{PMOT} + 14.3\ [^{\circ}\mathrm{C}]\ \left(10 \le \mathrm{t_{pma(out)}} \le 33.5\right) \tag{10}$$

$$\textit{Upper limit}\ (90\%\textit{acceptability}) = 0.31 \cdot \mathrm{PMOT} + 20.3\ [^{\circ}\mathrm{C}]\ \left(10 \le \mathrm{t_{pma(out)}} \le 33.5\right) \tag{11}$$

$$\textit{Lower limit}\ (90\%\textit{acceptability}) = 0.31 \cdot \mathrm{PMOT} + 15.3\ [^{\circ}\mathrm{C}]\ \left(10 \le \mathrm{t_{pma(out)}} \le 33.5\right) \tag{12}$$

Adaptive thermal comfort is fully implemented within EnergyPlus, and it is currently an active research field whose importance has been increased because it could be seriously affected by climate change.

2.1.2. *EnergyPlus and Energy Management System*

EnergyPlus, the simulation engine used within the ACCIS framework, is an open-source BES software developed by both the Lawrence Berkeley National Laboratory and the U.S Department of Energy. It is used by engineers, architects, and researchers to model both energy consumption and water use in buildings. This software is a console-based program which reads inputs and writes outputs to text files. It was originally written in FORTRAN programming language; however, it is written in C++ since version 8.2.0. This software and its programming language are difficult to understand, so some graphical user interfaces, such as DesignBuilder or OpenStudio [33], have been developed to be used by people with no programming experience .

EnergyPlus is a powerful program but limited in terms of customisation. Nevertheless, there is a built-in feature called Energy Management System (EMS) that overcomes that weakness. This is a powerful tool that develops custom control and modelling routines, and provides high-level, supervisory control to override some aspects of EnergyPlus modelling. However, it is difficult to use it because it needs writing computer programs in a small programming language called EnergyPlus Runtime Language (ERL), which is used to describe the control algorithms. EnergyPlus interprets and executes the ERL program when the

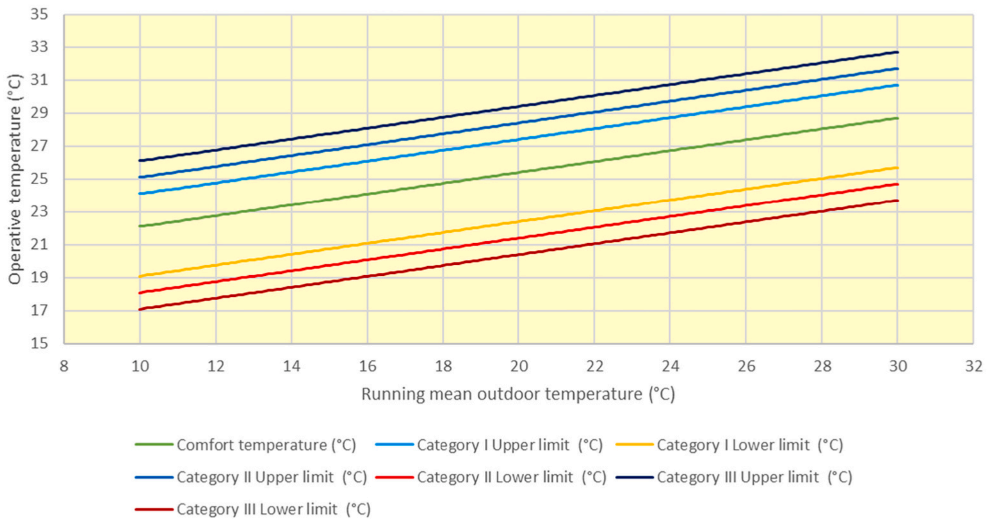


Fig. 1. Upper and lower comfort limits established in EN16798–1 for each category and comfort temperature.

simulation is being run.

In this study, DesignBuilder was used to model the geometry of the case study. Moreover, all the remaining tasks were carried out with Python (namely, *accim* package [34], which is based on the *eppy* package [35], and EP-Launch [36], a simple graphical user interface to run simulations).

2.1.3. *Eppy*

Eppy is an open-source Python package developed by Philip Santosh. It allows IDF files to be programmatically navigated, searched, and modified. Moreover, it is possible to access any field within any object within the IDF file and modify it, and add or delete IDF objects, among other functions. Therefore, the workflow of the *eppy* process consists in taking an input IDF, transforming it into an adaptive-setpoint model based on the user customisation, and generating an output IDF.

The proposed computational approach completely relies on *eppy* because it is used within the *accim* package to perform most tasks. Some of them are as follows:

- To add thermal comfort fields to people objects.
- To check if there is an operative temperature thermostat for each zone; if not, it should be added.
- To check if some Schedule:Compact objects needed to work are in the model; if not, they should be added.
- To check if the values for window fields and properties are correct; if not, they should be amended.
- To check if some EMS objects needed to work are in the model; if not, they should be added.

Table 1
Relevant research articles related to the computational approach.

Authors	Citation	Setpoints	Energy Saving (%)	Level of automation
Giorgos N. Spyropoulos and Constantinos A. Balaras	[11]	Static setpoints	18	Not specified
Tyler Hoyt et al.	[12]	Static setpoints	27 (Cooling); 34 (Heating)	Not specified
Parry et al.	[13]	Static setpoints	66	Not specified
Fernández et al.	[16]	Static setpoints	between 12 and 20	Not specified
R.P. Kramer et al.	[17]	Adaptive setpoints (based on Adaptive Thermal Guideline)	74	Not specified
Sánchez-Guevara Sánchez et al.	[18]	Adaptive setpoints (ASHRAE Standard 55–2013)	80 (Cooling); 20 (Heating)	Low (based on several monthly simulations and following merge)
Sánchez-García et al.	[23]	Adaptive setpoints (EN 15251:2007)	between 23 and 46	Medium (based on Schedule: Compact objects)
Sánchez-García et al.	[24]	Adaptive setpoints (EN 15251:2007)	between 31 and 70	Medium (based on Schedule: Compact objects)
Sánchez-García et al.	[25]	Adaptive setpoints (EN 15251:2007)	between 40 and 62	Medium (based on Schedule: Compact objects)

2.2. ACCIS

2.2.1. Development: From EMS to Python

Originally, ACCIS was an Energy Management System script written in EnergyPlus Runtime Language (ERL) for EnergyPlus. Appendix A includes the most basic version of ACCIS, which applies the upper and lower comfort limits of EN 16798-1-Category II to the adaptive cooling setpoint temperature (ACST) and to the adaptive heating setpoint temperature (AHST), respectively. This version of ACCIS first declares the ACST and AHST global variables, and then some objects, which are called sensors, are added. These objects allow some variables to be monitored. In this case, the variables are RMOT (by means of the sensor called RMOT) and the operative temperature (by means of the sensor called OpTemp). Afterwards, the Program object called SetAST and its ProgramCallingManager object are added. The Program object contains all the statements defining the behaviour that the model should have in the simulation run, and the ProgramCallingManager object specifies the simulation stage when the program should be called. The statements within the SetAST program consist of two conditional blocks (one for ACST and another for AHST) and should be read as follows: if RMOT is greater than or equal to 15 °C (which is the lower applicability limit) and RMOT is lower than or equal to 30 °C (which is the upper applicability limit), then ACST should be equal to RMOT*0.33 + 18.8 + 2 (which is the upper comfort limit). If RMOT is lower than 10 °C, then the ACST should be equal to 10*0.33 + 18.8 + 2 (which is the minimum lower comfort limit when RMOT is equal to 10 °C). In case any of the previous conditions are met, if RMOT is greater than 30 °C, then ACST should be equal to 30*0.33 + 18.8 + 2 (which is the maximum upper comfort limit when RMOT is equal to 30 °C). Afterwards, a similar conditional block is added for AHST. So far, the script would monitor the RMOT and specify how should the ASTs behave, but it has not been overridden any behaviour in the model. This is the purpose of the Actuator objects, which are added in the following step. Therefore, an Actuator object should be added for each AST. In addition, setpoint temperatures are driven by Schedule:Compact objects, so the Actuators should override the behaviour of these objects. Finally, the Program object called ApplyAST and its ProgramCallingManager object are added. The previous SetAST program specifies the values that the ASTs should have (as the program name indicates). However, in ApplyAST, the AST values are assigned to the Actuator objects previously added (FORSCRIPT_ACST_Schedule and FORSCRIPT_AHST_Schedule), thus overriding these Schedule:Compact objects' behaviour.

However, adding that script to the IDF does not transform the fixed-setpoint into adaptive-setpoint behaviour because a Schedule:Compact object should be added for each setpoint (FORSCRIPT_AHST and FORSCRIPT_ACST); moreover, FORSCRIPT_AHST and FORSCRIPT_ACST should be assigned to the heating and cooling setpoints of the HVAC system, among other tasks. These inherent tasks cannot be automated by ACCIS, but they are needed to work with. Therefore, a Python package (called *accim*, as the previous framework on which it has been based) should be developed to nest ACCIS on it. This package is distributed using PyPI (The Python Package Index). For ease-use purposes, both a module called *accis* and a function within it called addAccis()are developed to run this computational approach, so only the addAccis() function is required to be called. When users call the addAccis() function, *accim* carries out many tasks that could not be previously automated and allows ACCIS to be added to any IDF.

2.2.2. Development: Two different approaches

The version of ACCIS included in Appendix A only works with single-zone building energy models (after carrying out all its inherent tasks). To work with multiple-zone models, each zone should have a different HVAC system, so each HVAC system could work independently at different times. Afterwards, ACCIS is split into two different approaches: SingleZone and MultipleZone branches. In the SingleZone branch, the building energy model should have an HVAC system to feed the single zone, while in the MultipleZone branch, no HVAC system is required because *accim* automatically adds a VRF system for each zone.

This bifurcation takes place after developing ACCIS and the *accim* functionality, thus leading to some common features, such as the input arguments included, so users could customise the output IDFs: Adaptive Standard, Category, Comfort Mode, and Tolerances. However, the MultipleZone branch is developed as it is more likely to be used, thus having some features that SingleZone does not have. These features are the additional input arguments: HVAC Mode, Ventilation control, Ventilation setpoint temperature offset, Minimum operative temperature offset, and Maximum wind speed.

2.2.3. Main functionality: Input arguments and easy use

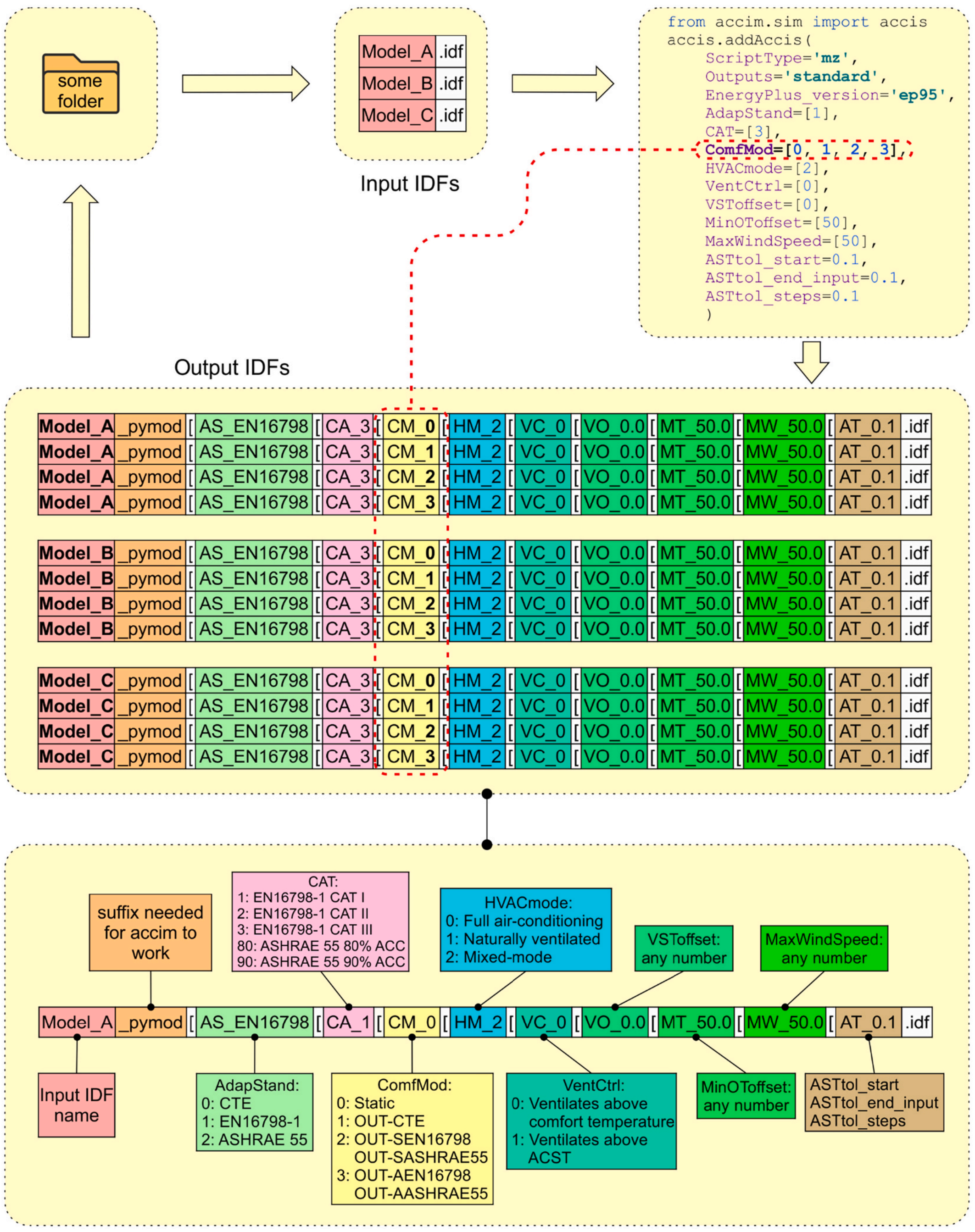
The addAccis() function has been developed to be used as simple as possible, but keeping its high customisation properties. An usage example of the addAccis() function is shown in Appendix B. Fig. 2 includes the workflow from an end-user point of view: there is a folder with some input IDF files, so the user just needs to set the folder as working directory, import the *accis* module from the *accim* package, and then call the addAccis() function. Some arguments may be included, otherwise the user is asked to enter the required information in the command prompt or Python interpreter. Afterwards, the output IDFs are generated in the same folder for each input IDF, based on the input parameters specified by the user. The name of each output IDF consists of each argument used to set up the output IDF file, separated by the character '['. The separator is used to easily separate the columns in other data analytics software, as R or Python itself, by means of the *pandas* package [37].

First, three arguments are used for setting up the script to be used:

- ScriptType: It is used to choose between MultipleZone or SingleZone approaches.
- Outputs: They refer to the desired outputs to be included in the simulation results. 'Standard' means that results will contain the full selection; 'Simplified' means that results are both the hourly operative temperature and the energy demand of each zone; and 'Timestep' means that results are the full selection in timestep frequency, so this is only recommended for tests or for few simulations.
- EnergyPlus_version: It is used to choose EnergyPlus 9.1.0, 9.2.0, 9.3.0, 9.4.0, or 9.5.0.

The remaining arguments are used for setting up the output IDFs:

- AdapStand: It refers to the thermal comfort standard to be applied. Available options are CTE, EN 16798-1, and ASHRAE Standard 55.
- CAT: It refers to the category of the applied adaptive thermal comfort standard to be used. Available options are Categories I, II and III from EN16798-1, and 80% and 90% acceptability from ASHRAE Standard 55.
- ComfMod: It refers to the comfort mode applied. In these comfort modes, adaptive setpoints are mainly used when the adaptive model is applicable; if not, static setpoints are used. Therefore, the difference between the available options is the static model applied, which could be CTE (the Spanish Building Technical Code), the static setpoints specified in the selected adaptive standard, or a horizontal extension of the comfort limits. These options are respectively OUT-CTE, OUT-SEN16798 or OUT-AEN16798 if the EN16798-1 standard is previously chosen, or OUT-CTE, OUT-SASHRAE55 or OUT-AASHRAE55 if ASHRAE Standard 55 is chosen.
- HVACmode: It refers to the HVAC mode applied. Available options are Full air-conditioning, Naturally Ventilated, or Mixed Mode.
- VentCtrl: It refers to the ventilation control. Available options are as follows: the ventilation to be allowed if the operative temperature exceeds the neutral temperature (also known as comfort temperature), or the ventilation to be allowed if the operative temperature exceeds the upper comfort limit. In other words, it sets the value of the neutral temperature or of the upper comfort limit to the Ventilation Setpoint Temperature (VST).
- VSToffset: It applies the entered values (in Celsius degrees) as an offset to the VST. These values could be positive or negative decimal numbers, so that the offset is set above or below the original VST.
- MinOToffset: It stands for the Minimum Outdoor Temperature offset and sets the minimum outdoor temperature offset to the heating setpoint temperature. For example, if the user enters '1', ventilation will not be allowed if outdoor temperature falls below 1 °C below the heating setpoint to avoid excessive cold.
- MaxWindSpeed: It stands for the maximum wind speed and sets the maximum wind speed in which ventilation is allowed (in m/s).
- ASTtol: It stands for the Adaptive Setpoint Temperature tolerance. It applies the number that the user enters as a tolerance for adaptive heating and cooling setpoint temperatures. The original problem was



```
from accim.sim import accis
accis.addAccis(
    ScriptType='mz',
    Outputs='standard',
    EnergyPlus_version='ep95',
    AdapStand=[1],
    CAT=[3],
    ComfMod=[0, 1, 2, 3],
    HVACmode=[2],
    VentCtrl=[0],
    VSToffset=[0],
    MinOToffset=[50],
    MaxWindSpeed=[50],
    ASTtol_start=0.1,
    ASTtol_end_input=0.1,
    ASTtol_steps=0.1
    )
```

some folder
Model_A.idf
Model_B.idf
Model_C.idf
Input IDFs
Output IDFs
Model_A_pymod[AS_EN16798[CA_3[CM_0[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_A_pymod[AS_EN16798[CA_3[CM_1[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_A_pymod[AS_EN16798[CA_3[CM_2[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_A_pymod[AS_EN16798[CA_3[CM_3[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_B_pymod[AS_EN16798[CA_3[CM_0[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_B_pymod[AS_EN16798[CA_3[CM_1[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_B_pymod[AS_EN16798[CA_3[CM_2[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_B_pymod[AS_EN16798[CA_3[CM_3[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_C_pymod[AS_EN16798[CA_3[CM_0[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_C_pymod[AS_EN16798[CA_3[CM_1[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_C_pymod[AS_EN16798[CA_3[CM_2[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Model_C_pymod[AS_EN16798[CA_3[CM_3[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
suffix needed for accim to work
CAT: 1: EN16798-1 CAT I 2: EN16798-1 CAT II 3: EN16798-1 CAT III 80: ASHRAE 55 80% ACC 90: ASHRAE 55 90% ACC
HVACmode: 0: Full air-conditioning 1: Naturally ventilated 2: Mixed-mode
VSToffset: any number
MaxWindSpeed: any number
Model_A_pymod[AS_EN16798[CA_1[CM_0[HM_2[VC_0[VO_0.0[MT_50.0[MW_50.0[AT_0.1.idf
Input IDF name
AdapStand: 0: CTE 1: EN16798-1 2: ASHRAE 55
ComfMod: 0: Static 1: OUT-CTE 2: OUT-SEN16798 OUT-SASHRAE55 3: OUT-AEN16798 OUT-AASHRAE55
VentCtrl: 0: Ventilates above comfort temperature 1: Ventilates above ACST
MinOToffset: any number
ASTtol_start ASTtol_end_input ASTtol_steps

Fig. 2. Workflow to generate output IDFs from an end-user point of view.

that, if the adaptive setpoint is assigned to the comfort limit without any tolerance, then a few hours are not within the comfort zone because of the error in some decimals in the simulation of the operative temperature. Therefore, the original purpose of this argument is to control that all hours are comfortable hours, and this can be assured by considering a small tolerance of 0.10 °C.

For further information, the user can consult the *accim*'s documentation website [38], which contains a more detailed explanation of the input arguments, a tutorial, and some examples, as well as additional information on the installation.

2.2.4. Main functionality: Inside addAccis()

To understand the procedure of addAccis(), it is recommended to have an overview of the *accim* structure. Within the directory accim, a directory called sim can be found. It contains all the modules needed for addAccis() to work: *accis.py*, *accim_Main.py*, *accim_Base.py*, *accim_Base_EMS.py*, *accim_MultipleZone.py*, *accim_MultipleZone_EMS.py*, *accim_SingleZone.py*, *accim_SingleZone_EMS.py*, and *accim_IDFgeneration.py*. The *accis* module contains the addAccis() function, which is the only function that can be called from an end-user point of view; the *accim_Main* module contains the class where all the other modules are imported; modules from *accim_Base* to *accim_SingleZone_EMS* contain the functions that modify the input static-setpoint IDF file to make it a generic adaptive-setpoint IDF; and finally, the *accim_IDFgeneration* module contains the functions that transform the adaptive-setpoint IDF into the output IDFs based on the different various user arguments, and save copies of them. Methods within modules have been organised according to the following criteria: the accim_Base module contains methods not related to EMS, which are used in both MultipleZone and SingleZone

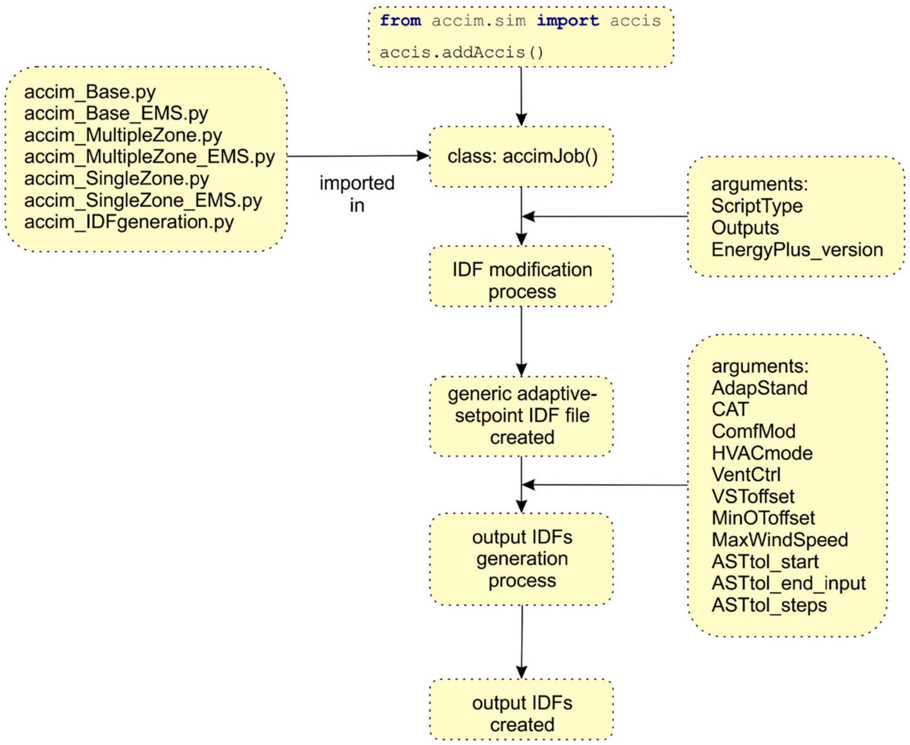

from accim.sim import accis
accis.addAccis()
accim_Base.py
accim_Base_EMS.py
accim_MultipleZone.py
accim_MultipleZone_EMS.py
accim_SingleZone.py
accim_SingleZone_EMS.py
accim_IDFgeneration.py
imported in
class: accimJob()
arguments: ScriptType Outputs EnergyPlus_version
IDF modification process
generic adaptive-setpoint IDF file created
arguments: AdapStand CAT ComfMod HVACmode VentCtrl VSToffset MinOToffset MaxWindSpeed ASTtol_start ASTtol_end_input ASTtol_steps
output IDFs generation process
output IDFs created

Fig. 3. Workflow within the addAccis() function.

branches, while EMS-related methods common to both branches are contained in accim_Base_EMS; same criteria are applied to *accim_MultipleZone*, *accim_MultipleZone_EMS*, *accim_SingleZone*, and *accim_SingleZone_EMS*.

Fig. 3 shows the procedure carried out when addAccis() is called, and Appendix C shows the actual Python code; (i) Lines 62 to 69 show the imports needed and list the input IDFs; (ii) lines 70 to 144 evaluate whether the arguments used to set up the script are properly included; if not, the user is asked to include them in command prompt or python interpreter; finally, the arguments values are stored; (iii) lines 145 to 209 consist of the procedure to create the generic adaptive-setpoint IDF. In these lines, a loop, which iterates over the input IDF, is created. Within this loop, the following actions are carried out as follows:

- (lines 150-156) The class accimJob is instantiated according to the previously stored arguments.
- (lines 156-159) Methods related to the accim_Base module are called:
  - o (line 156, method setComfFieldsPeople()) existing People objects are amended to include thermal comfort fields.
  - o (line 157, method addOpTempTherm()) checks if ZoneControl: Thermostat:OperativeTemperature objects for each zone are in the model; if not, they should be added.
  - o (line 158, method addBaseSchedules()) checks if 'On' Schedule: Compact object is in the model; if not, it should be added.
  - o (line 159, method setAvailSchOn()) sets the heating and cooling availability schedules of ZoneHVAC:IdealLoadsAirSystem objects to on, so that heating and cooling are always allowed to work.
- (lines 162-173) Depending on the branch, i.e., SingleZone or MultipleZone, some methods are called as follows:
  - o In case of MultipleZone,
    - ▪ (line 163, method addMultipleZoneSch()) some Schedule: Compact objects are added if needed;
    - ▪ (line 164, method addCurveObj()) an extensive list of Curve objects are added;
    - ▪ (lines 165, methods addDetHVACobj()) afterwards, detailed HVAC objects are added, i.e., the VRF system and related components;
    - ▪ (line 166, method checkVentIsOn()) checks that settings related to calculated natural ventilation are as needed;
    - ▪ (line 167, method addForscriptSchMultipleZone()) adds the aforementioned FORSCRIPT_AHST and FORSCRIPT_ACST Schedule:Compact objects for each zone.
  - o (line 169, method addForscriptSchSingleZone()) In case of SingleZone, the code is significatively shorter because no HVAC system is required. Thus, it only adds the FORSCRIPT_AHST and FORSCRIPT_ACST Schedule:Compact objects.
- (lines 171-173) Methods related to accim_Base_EMS, which is common to both SingleZone and MultipleZone branches, are called as follows:
  - o (line 171, method addEMSProgramsBase()) adds some EnergyManagementSystem:Program objects.
  - o (line 172, method addEMSOutputVariableBase ()) adds some EMS OutputVariable objects.
- (lines 174-184) Depending on the branch, i.e., SingleZone or MultipleZone, some methods related to EMS only for each branch are called as follows:
  - o In case of MultipleZone:

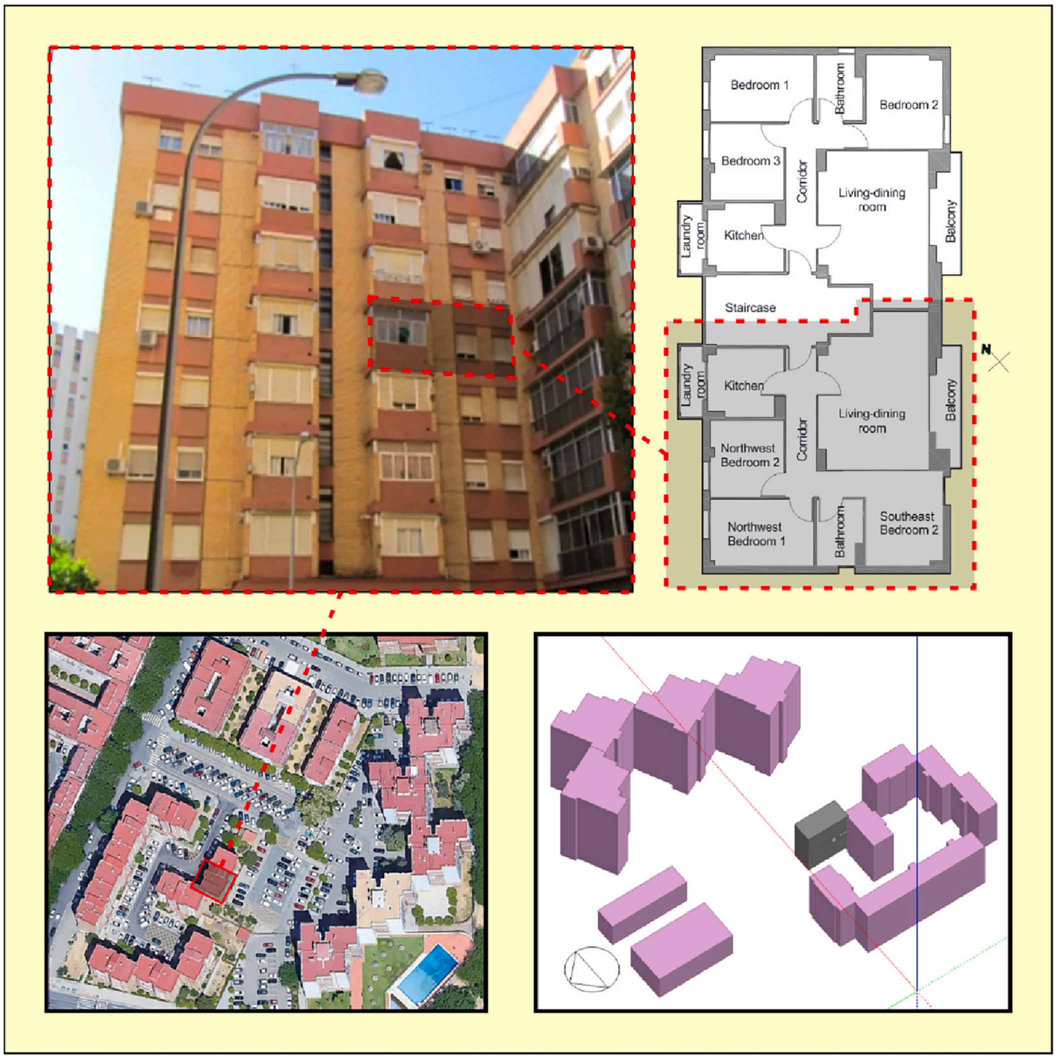


Fig. 4. Photographs of the building location, the northwest-facing façade, the plan view of the dwelling, and the building energy model.

Table 2
Technical characteristics and thermophysical properties of the case study.

Component	Description	Thickness [m]	U-Value [W/(m^{2}K)]	Internal Heat Capacity [kJ/(m^{2}K)]
External wall	Cement plaster Hollow brick Air gap Cement plaster Brick facing	0.26	1.35	80.35
Internal wall	Cement plaster Double hollow brick masonry Cement plaster	0.10	2.74	39.00
Windows	Aluminium frame; simple glazing 3 mm	–	5.89	–
Floor and paving	Terrazzo paving Sand Lightweight floor slab, cast in place, with a depth of 25 cm	0.28	1.76	147.63

Table 3
Sensors' specifications.

HOBO Pendant data logger 8K-UA-002-08	
Measurement Range	Temperature: −20° to 70 °C (−4° to 158 °F)
Accuracy	Temperature: ± 0.53 °C from 0° to 50 °C (± 0.95 °F from 32° to 122 °F)
Resolution	Temperature: 0.14 °C at 25 °C (0.25 °F at 77 °F) Drift: Less than 0.1 °C/year (0.2 °F/year)
HOBO U12- 012	
Measurement Range	Temperature: −20° to 70 °C (−4° to 158 °F)
Accuracy	Temperature: ±0,35 °C from 0 °C to 50 °C (±0,63 °F from 32 °F to 122 °F)
Resolution	Temperature: 0,03 °C at 25 °C (0,05 °F at 77 °F) Drift: Less than 0.1 °C/year (0,2 °F/year)

Table 4
MBE and CV(RMSE) values for indoor air and outdoor dry-bulb temperatures for each bedroom and monitoring period.

Monitoring period	Bedroom	Indoor air temperature		Outdoor dry-bulb temperature	
		MBE	CV (RMSE)	MBE	CV (RMSE)
14th Jan 2015 - 03rd Feb 2015	Northwest bedroom 1	−4.71%	13.42%	4.47%	25.74%
	Southeast bedroom	−6.30%	16.47%	4.61%	25.35%
14th May 2015 - 12th Jun 2015	Northwest bedroom 1	3.43%	7.36%	5.87%	26.26%
	Southeast bedroom	4.51%	8.03%	5.46%	29.27%
22nd Jun 2015 - 22nd Jul 2015	Northwest bedroom 1	−0.56%	7.55%	5.93%	21.41%
	Southeast bedroom	0.45%	8.42%	6.15%	23.04%

- ▪ (line 175, method addGlobVarListMultipleZone()) removes GlobalVariable objects if existed, and adds needed GlobalVariable objects needed to work.
- ▪ (line 176, method addEMSSensorsMultipleZone()) adds Sensor objects for EMS if they are not in the model.
- ▪ (line 177, method addEMSActuatorsMultipleZone()) adds Actuator objects for EMS if they are not in the model.
- ▪ (line 178, method addEMSProgramsMultipleZone()) adds Program objects for EMS if they are not in the model.

o In case of SingleZone, the methods called are similar, but focused on the SingleZone branch.

- (line 185, method addEMSPCMBase()) adds the ProgramCallingManager objects based on the EMS Programs added.
- (lines 187-199) Depending on both the branch, i.e., MultipleZone or SingleZone, and the outputs, i.e., 'standard', 'simplified' or 'timestep', different methods, which add Output:Variable objects, are called.

Once the generic adaptive-setpoint IDF file has been created, (iv) lines 207-257 check if arguments are given when the addAccis() function is called; if so, the genIDFMultipleZone() or genIDFSingleZone() methods generate the output IDFs according to the branch, MultipleZone or SingleZone, respectively. These methods consist of extensive loops which iterates through each given value of each argument, in which input values of the EMS program called SetInputData are modified to match the input arguments. Therefore, the arguments given to set up the output IDFs are transferred to this EMS program and used in the whole EMS script, so the generic adaptive-setpoint IDF file is no longer generic but customised based on the arguments. If arguments are not given when the addAccis() function is called, then the inputdataMultipleZone() or inputdataSingleZone() methods would ask the user to introduce them on prompt command or python interpreter, and output IDFs are generated following the same process. Finally, the output IDFs are saved in the same folder where input IDFs initially were. For further

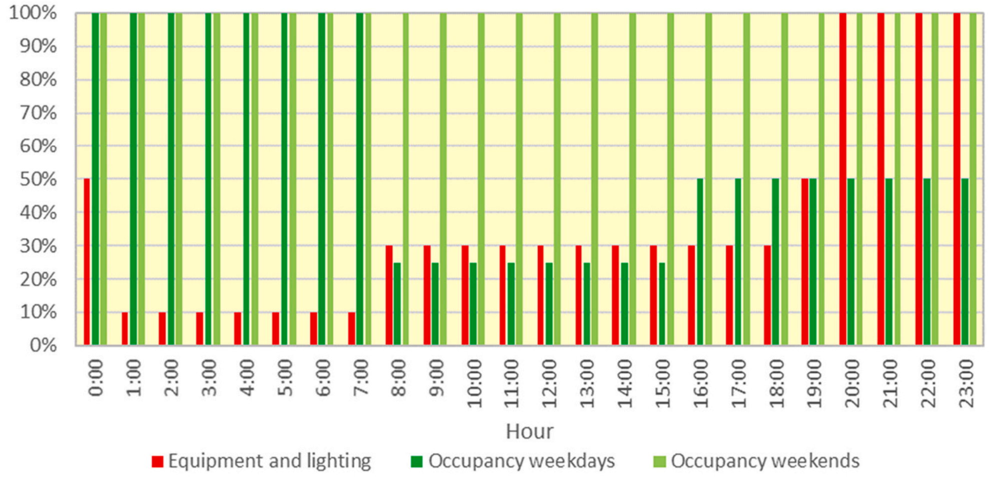


Fig. 5. Bar chart with hourly profiles of equipment and lighting and occupancy on weekdays and weekends.

Table 5
Setpoint temperatures for static and adaptive models.

Model	Standard	Limit	Range (°C)	Setpoint Temperature (°C)		
				January–May	June–September	October–December
Static model	EN16798-1 Cat. III	Upper limit	all	25	27	25
		Lower limit	all	18	22	18
Adaptive model	OUT-AEN16798 Cat. III	Upper limit (ACST)	< 10	$0.33 \cdot 10 + 22.8$		
			$10 \leq T_{rm} < 30$	Eq. (6)		
			> 30	$0.33 \cdot 30 + 22.8$		
		Lower limit (AHST)	< 10	$0.33 \cdot 10 + 13.8$		
			$10 \leq T_{rm} \leq 30$	Eq. (7)		
			> 30	$0.33 \cdot 30 + 13.8$		

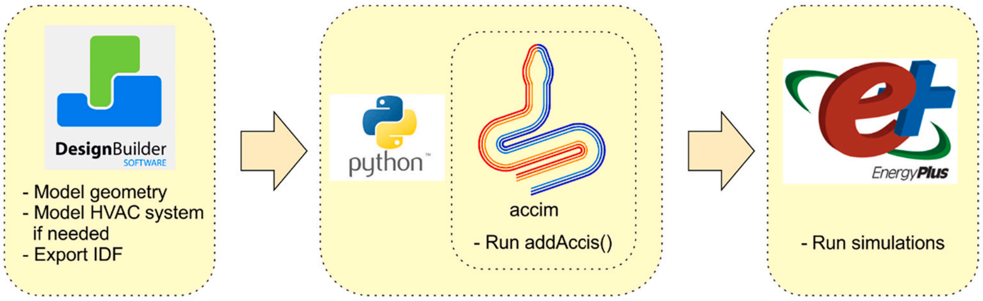


Fig. 6. Workflow process for any case study.

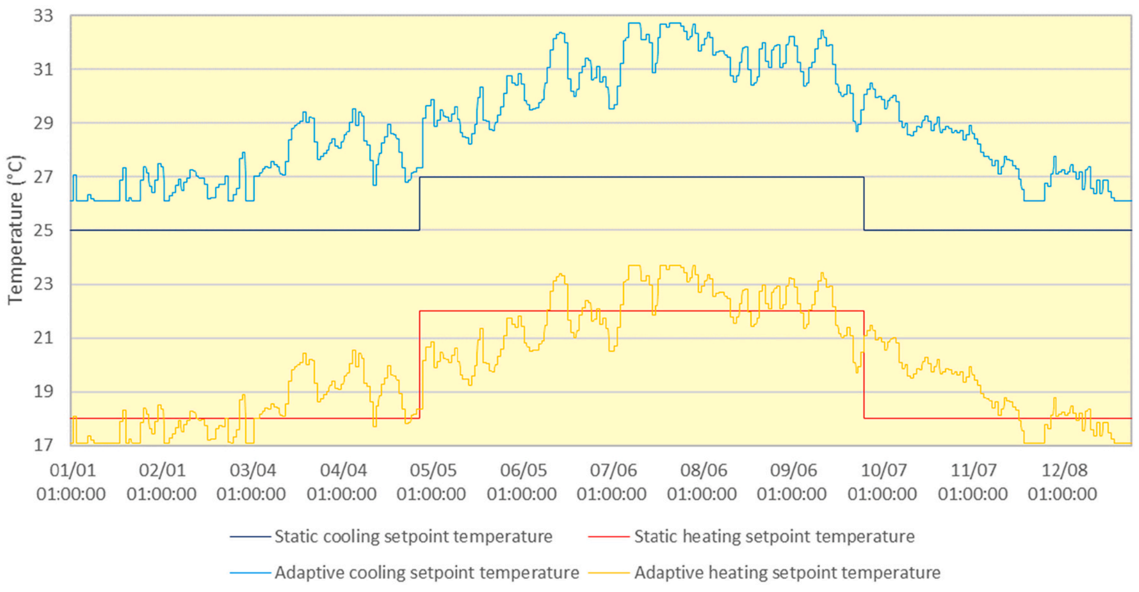


Fig. 7. Comparison of static and adaptive setpoint temperatures.

information, the user can consult the web repository [34] which includes everything related to the *accim* code.

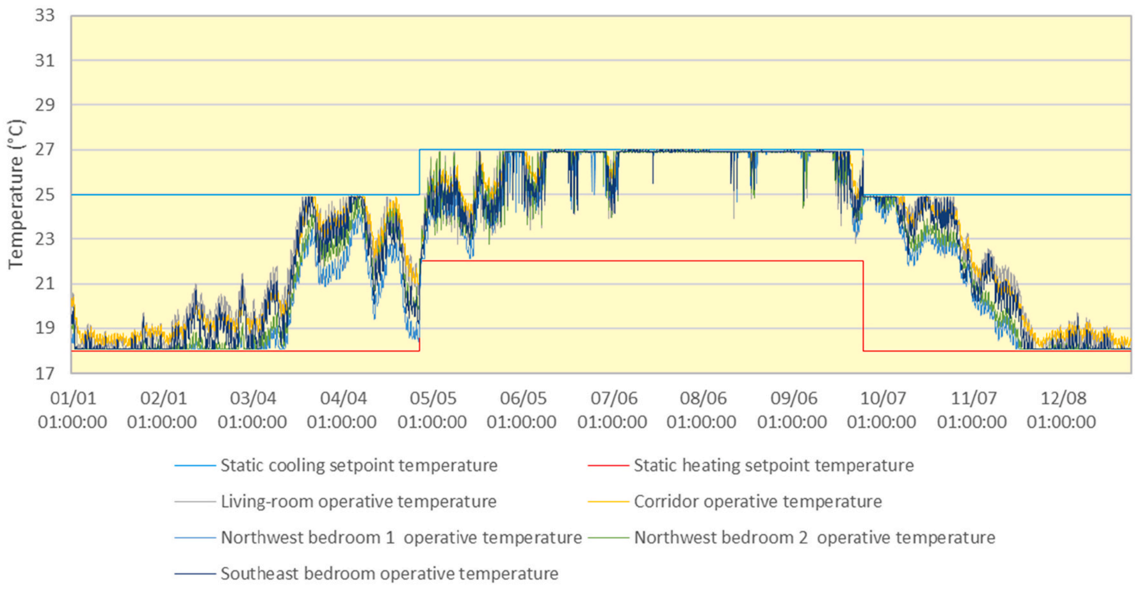


Fig. 8. Fluctuation of the operative temperature within the static setpoints.

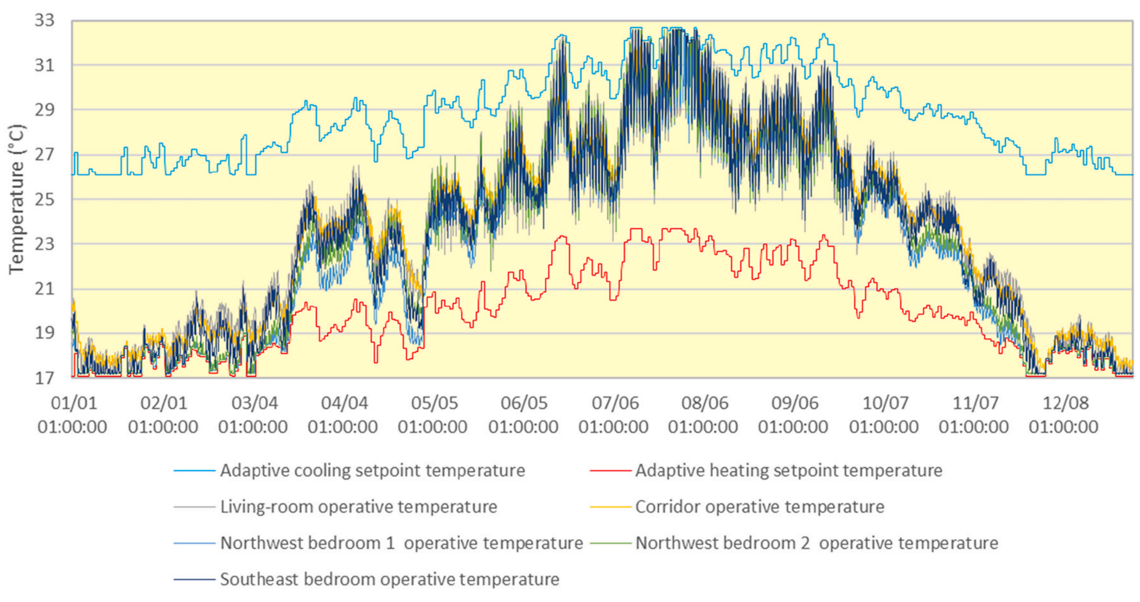


Fig. 9. Fluctuation of the operative temperature within the adaptive setpoints.

Table 6
Energy savings obtained by applying adaptive setpoint temperatures.

Room	Operation mode	Static setpoints	Adaptive setpoints	Energy saving (%)
Living-room	Cooling consumption (kWh)	505.9	23.5	95.4
	Heating consumption (kWh)	31.0	14.4	53.6
Corridor	Cooling consumption (kWh)	102.2	3.2	96.9
	Heating consumption (kWh)	0.4	0.4	−0.2
Southeast bedroom	Cooling consumption (kWh)	215.5	6.5	97.0
	Heating consumption (kWh)	47.8	32.8	31.3
Northwest bedroom 1	Cooling consumption (kWh)	125.8	0.1	99.9
	Heating consumption (kWh)	124.0	106.6	14.0
Northwest bedroom 2	Cooling consumption (kWh)	155.8	5.9	96.2
	Heating consumption (kWh)	53.1	40.1	24.5
Dwelling	Total (kWh)	1361.4	233.5	82.8
	Total (kWh/m^2)	17.7	3.0	82.8

As a general overview, (i) an instance of the accimJob class is created based on the user's input arguments related to the script setup; (ii) several functions, which were previously imported into the class, are executed to generate a generic adaptive-setpoint IDF file: these functions are the main body of the process, since all tasks needed to transform static into adaptive energy models are performed; once the IDF modification process is finished, the generic adaptive-setpoint IDF file is created based on the user's input arguments needed to set up the output

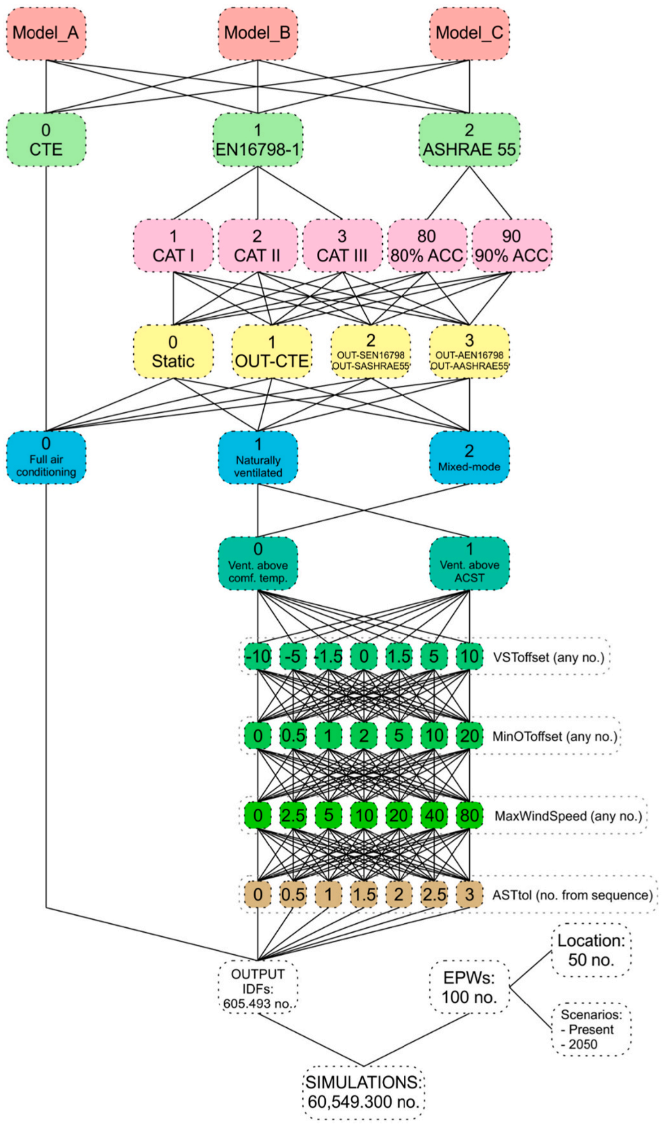


Fig. 10. Example of extensive simulation runs using addAccis().

IDFs, and (iii) the output IDFs generation process takes place: in this process, the functions modify the EMS programs input data based on the input arguments and save copies, thus creating the output IDFs.

3. Applications

3.1. Case study and energy saving potential

The case study is a residential building built in 1978. Before NBE-CT 79 [39], the first Spanish building energy regulation, became effective in 1979, envelopes have been built without insulation. It is an 8-storey building, and the dwelling has a living-room, a bathroom, a kitchen and three bedrooms, with a surface area of 77 m^2 (Fig. 4). The building is in Seville, a city in the south of Spain, where climate is Mediterranean, with hot summers and mild winters, and belongs to the Csa class according to Köppen–Geiger's classification [40]. The envelope is the typical external wall used in this building period, which is a double-leaf brick wall with air gap. The properties of the external walls and the remaining elements modelled in the case study are included in Table 2, and the case study has a Variable Refrigerant Flow system with Coefficient of Performance and Energy Efficiency Ratio values of 2.10 and 2.00, respectively.

After gathering the necessary data, DesignBuilder was used to create a building energy simulation model. To validate the model based on both the indoor air temperature and the outdoor dry-bulb temperature, three different monitorings were carried out along the year: between 14th January and 3rd February in winter, between 14th May and 12th June in spring, and between 22nd June and 22nd July in summer. Moreover, HOBO Pendant data logger 8K-UA-002-08 sensors were used to measure the outdoor temperature, and HOBO U12- 012 sensors to measure the indoor air temperature (Table 3). A total of four sensors were placed: two measured the temperatures in the southeast bedroom, and the other two were placed in the northwest bedroom 1. Therefore, measured and simulated indoor air and outdoor dry-bulb temperatures were evaluated following the ASHRAE Guideline 14-2014 [41]. The Mean Bias Error (MBE) (Eq. (13)) and the Coefficient of Variation of the Root Mean Square Error (CV(RMSE)) (Eq. (14)) for hourly values did not exceed the range ± 10% in case of MBE, and these were lower than 30% in case of CV(RMSE), so the model was considered validated (Table 4).

$$MBE = \frac{\sum_{i=1}^{n}(y_i - x_i)}{n} \bullet 100\ [\%] \tag{13}$$

$$CV(RMSE) = \frac{1}{\overline{y}}\left(\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}\right)^{1/2} \bullet 100\ [\%] \tag{14}$$

Where y_i is the measured value, x_i is the simulated value, n is the number of measures, and $\overline{y}$ is the average of measured values.

The usage profiles for occupancy and equipment and lighting were defined as per the Spanish Building Technical Code [42]. In this profile, occupancy varied from 25 to 100% in working days, and it was always 100% in weekends. In case of equipment and lighting, both followed the same profile, varying along the day (Fig. 5). Values of full occupancy latent load, sensible loads and both for equipment and lighting were respectively 1.36 W/m^2, 2.15 W/m^2, and 4.40 W/m^2.

The daily setpoint temperatures used in the static and adaptive models are shown in Table 5. In the static model, the static setpoints from EN16798-1 Category III were applied. Heating season was considered to cover from June to September, and cooling season from October to May. In the adaptive model, the EN 16798-1 Category III adaptive standard was applied when RMOT was contained in applicability limits; otherwise, adaptive setpoints would have been horizontally extended (which means that ComfMod OUT-AEN16798 was chosen). In relation to the expectation levels, Category III was chosen because the building exists.

In addition, mixed-mode was applied. Mixed mode prioritises the use of natural ventilation when outdoor conditions are suitable, otherwise air-conditioning is used. Therefore, the following conditions were evaluated in all zones:

- If the operative temperature was greater than VST (obtained from VentCtrl and VSToffset arguments).
- If the outdoor dry-bulb temperature was less than VST.
- If no heating or cooling was active.
- If the operative temperature was less than ACST.
- If the outdoor dry-bulb temperature was greater than the minimum outdoor temperature (obtained from the MinOToffset argument).
- If the wind speed was less than or equal to the maximum wind speed (obtained from the MaxWindSpeed argument).

If all previous conditions are returned true, then windows are opened and ventilation is allowed; otherwise, windows are closed. In this case, no limit restrictions for the minimum outdoor temperature or the maximum wind speed were set, and VST was set without any offset from the comfort temperature.

The aim was to compare the consumption obtained by using adaptive and static setpoint temperatures, so only two no. outputs IDFs were required: one for the static model, and another for the adaptive model. To obtain these by running the addAccis() function, the code included below was used, considering that ComfMod argument 0 refers to the static model, and argument 3 refers to the adaptive model (OUT-AEN16798):

```
from accim.sim import accis
accis.addAccis(
    ScriptType='mz',
    Outputs='standard',
    EnergyPlus_version='ep95',
    AdapStand=[1],
    CAT=[3],
    ComfMod=[0, 3],
    HVACmode=[2],
    VentCtrl=[0],
    VSToffset=[0],
    MinOToffset=[50],
    MaxWindSpeed=[50],
    ASTtol_start=0.1,
    ASTtol_end_input=0.1,
    ASTtol_steps=0.1
    )
```

Fig. 6 shows the overall workflow process for any case study. The model should be validated, so that the model results are reliable; however, it is not required for *accim* to work, so it was not included in the workflow. First, the building geometry was modelled with Designbuilder, and IDF was exported, although this step could have been carried out with another GUI; the HVAC system was not modelled because the model has multiple zones, and the VRF system that *accim* adds is similar to the existing one; then, addAccis() function was run including the desired arguments; and finally, simulations were run by using EP-Launch.

After finishing the overall process, simulation results were obtained. According to previous studies, AST provides a significant energy saving potential. Fig. 7 shows the comparison between adaptive and static setpoint temperatures. Static setpoint temperatures changed from cooling to heating season, while adaptive setpoint temperatures changed every day according to the previous days' climate variations.

Therefore, the operative temperature was within these setpoint temperatures in both static and adaptive cases (Figs. 8-9). However, adaptive setpoints were less restricting than the static ones, achieving an energy saving of 82.8% (Table 6).

3.2. Limitless simulation numbers

What makes the addAccis() function interesting is that it allows customised adaptive setpoint simulations to be run with no limit number, and this is achieved by means of two main features.

The first feature is that addAccis() function allows a high customization of the output IDFs because multiple values can be used for multiple arguments: the user mainly uses EN16798-1 or ASHRAE 55 standards (or CTE in the case of Spain) with any of the categories or acceptability ranges, with any of the proposed Comfort Modes, and with any of the HVAC modes and Ventilation controls. For these arguments, the values represent the various setups that have been developed. However, the remaining arguments can be feed with any values in °C or m/s, depending on the argument. For example, if the addAccis() function was called considering three input IDFs and arguments according to Fig. 10, the number of output IDFs would be 605,493. Besides, computational time related to *accim* was significantly low: it took 20 s to generate 65 output IDFs from a single file, although it could vary depending on the size and the complexity of the IDF.

The second feature is that the addAccis() function adds the ACCIS. When adaptive-setpoints are used with other methods apart from ACCIS, the IDF should be assigned to a specific EPW because these adaptive setpoints would have been previously calculated for that EPW. On the contrary, when ACCIS is used, the IDF could be assigned to any EPW because the adaptive setpoints would be calculated 'on the go' by both the EMS programs and EnergyPlus. Thus, the 605,493 output IDFs could be simulated with, for example, 100 various EPWs and considering different climate zones or future scenarios, thus obtaining 60,549.300 no. simulations. Although computational cost and time related to the simulations would be quite high in that case, time consumed by the user would be minimum because most steps within the process are automated. For example, by using EP-Launch, once the output IDFs are generated, it takes a few clicks to run the simulations: click on the 'Group of Input Files' tab, create a new group, select all the desired output IDFs within the path, select all the desired EPWs within the path, and finally specify the output files' location and hit the 'Simulate Group' button.

4. Discussion

The energy saving potential and the extensive simulation runs allowed by this computational approach provide new possibilities to perform energy simulations not only at a country or continent scale, but a global one. Subsequently, this approach provides new possibilities to extend the scope of different research studies focused on adaptive comfort, such as climate change mitigation strategies [25] or energy poverty studies [43,44]. To the authors knowledge, the proposed computational approach is the first approach that allows this, and besides, it allows other regular adaptive-setpoint simulations to be performed by consuming less time than any other approach. Moreover, this approach has been automated and developed with ease-use purposes, so there are no error-prone, while other approaches include performing file management tasks for dozens or hundreds different EPW files, and the IDF files with adaptive setpoints need to be simulated specifically with the pertinent EPW file.

However, this computational approach has some limitations: it works with EnergyPlus 9.1-2-3-4-5.0 versions. Older versions than 9.1.0 do not work, and not all the results might be included in the .csv file generated for versions 9.1-2-3.0. The reasons are the slight changes in the EnergyPlus IDD file for the different 9.X.0 versions. Input IDFs can be exported from any GUI software, but EnergyPlus should be used as simulation engine. In addition, language should be English such as any user-defined objects within the IDF. Any special character, such as accents or 'ñ', should not be included because they could imply some unexpected error related to 'Latin-1' codec. Likewise, the SingleZone branch only works with single-zone models (as its name indicates), and the MultipleZone branch adds the VRF system, which is a standard object that can be customised to change the EER or CoP, among others, although it requires modifying the source code within *accim* package. Thus, some background knowledge on python programming is highly recommended. One of the future steps will be the strengthening of this weakness by developing the SingleZone branch to work with multiple zones, considering that each zone must be fed by some independent HVAC system.

5. Conclusions

Adaptive setpoint temperatures are a potential solution to enhance building energy efficiency. Its high energy saving potential has been widely proved. However, the procedure carried out to achieve these results have been time-consuming and error-prone because most tasks have been manually performed. The outcome of this research is a novel computational approach, called ACCIS, which has been developed to amend this disadvantage, and currently there is no other method capable of automating the process of applying adaptive setpoint temperatures. ACCIS transforms PMV-based into adaptive setpoint building energy models, according to both an IDF and the setup specified by the user. ACCIS was initially an Energy Management System script written in ERL, which could compute the adaptive setpoint temperature based on the RMOT calculated by EnergyPlus 'on the go', i.e., while the simulation is running, although some actions should still be carried out manually. To amend this aspect, a fully-documented Python package called *accim* was developed to perform these actions and to add the ACCIS to the IDF. This python package was developed with ease-use purposes as it is only necessary to import the *accis* module and to call the addAccis() function, so high knowledge of Python programming language was not required.

To apply this computational approach, a case study was used to evaluate the energy saving potential provided by using adaptive setpoint temperatures instead of PMV-based setpoint temperatures. The model consisted of a residential unit within a residential building built before the current energy code was mandatory; the model was in the city of Seville, in the south of Spain. After modelling the geometry and validating the model, the addAccis() function was called to obtain two output IDFs that were simulated afterwards by using EP-Launch. Simulation results showed that using adaptive setpoint temperatures provides 83% of energy saving specifically for this case study; however, this ECM could be extended to other locations, and could therefore be presented as a relevant climate change mitigation strategy.

Nevertheless, the minimum time-consuming and error-prone were not the most powerful attribute: addAccis() allowed many customised adaptive setpoint simulations to be run with no limit. This was achieved through two main features: first one is that the addAccis() function allows a high customization of the output IDFs because multiple values for multiple arguments can be used, and second one is that the addAccis() function adds the ACCIS. Thus, adaptive setpoints could be calculated as simulation was run by both the EMS programs and EnergyPlus, and therefore the IDF could be simulated with any EPW file. However, in other manual procedures, each IDF file had previously calculated adaptive setpoints for some specific location, and therefore it could be only simulated with the EPW file of that location.

This computational approach is a powerful tool, but it has some limitations related to three aspects: the EnergyPlus versions it works with, the format of the input IDFs, and the computational approach itself. However, this is the only method that performs many adaptive setpoint simulations, so the scope of research studies on adaptive comfort, such as climate change mitigation strategies or energy poverty studies, could be increased.

Declaration of Competing Interest

The authors declared that there is no conflict of interest.

Appendix A. Simplest version of ACCIS (for EN 16798–1, Category 2)

```
EnergyManagementSystem:GlobalVariable,
        ACST,                 !Adaptive Cooling Setpoint Temperature
        AHST;                 !Adaptive Heating Setpoint Temperature

EnergyManagementSystem:Sensor,
        RMOT,                 !Running Mean Outdoor Temperature
        People Block1:Zone1,
        Zone Thermal Comfort CEN 15251 Adaptive Model Running Average Outdoor Air Temperature;

EnergyManagementSystem:Sensor,
        OpTemp,               !Operative Temperature
        Block1:Zone1,
        Zone Thermostat Operative Temperature;

EnergyManagementSystem:ProgramCallingManager,
        SetAST,               !Set Adaptive Setpoint Temperatures
        BeginTimestepBeforePredictor,
        SetAST;

EnergyManagementSystem:Program,
        SetAST,

        if (RMOT >= 15) && (RMOT <= 30),
                set ACST = RMOT*0.33+18.8+3,
        elseif RMOT < 15,
                set ACST = 15*0.33+18.8+3,
        elseif RMOT > 30,
                set ACST = 30*0.33+18.8+3,
        endif,

        if (RMOT >= 10) && (RMOT <= 30),
                set AHST = RMOT*0.33+18.8-4,
        elseif RMOT < 10,
                set AHST = 10*0.33+18.8-4,
        elseif RMOT > 30,
                set AHST = 30*0.33+18.8-4,
        endif;

EnergyManagementSystem:Actuator,
        FORSCRIPT_ACST_Schedule,
        FORSCRIPT_ACST,
        Schedule:Compact,
        Schedule Value;

EnergyManagementSystem:Actuator,
        FORSCRIPT_AHST_Schedule,
        FORSCRIPT_AHST,
        Schedule:Compact,
        Schedule Value;

EnergyManagementSystem:ProgramCallingManager,
        ApplyAST,             !Apply Adaptive Setpoint Temperatures
        BeginTimestepBeforePredictor,
        ApplyAST;

EnergyManagementSystem:Program,
        ApplyAST,
        Set FORSCRIPT_ACST_Schedule = ACST,
        Set FORSCRIPT_AHST_Schedule = AHST;
```

Appendix B. Usage example of addAccis() function within *accim* package version 0.1.14

```
from accim.sim import accis

accis.addAccis(
    # Three first input arguments are related to the type of script to be used
    # ScriptType: 'multiplezone' or 'mz', 'singlezone' or 'sz'
    ScriptType='mz',
    # Outputs: 'simplified', 'standard' or 'timestep'
    # if standard, results will contain the full selection in hourly frequency
    # if simplified, results are going to be hourly operative temperature and energy demand
    # if timestep, results are going to be the full selection in timestep frequency
    Outputs='standard',
    # EnergyPlus_version:
    # 'ep91' for EnergyPlus 9.1.0
    # 'ep92' for EnergyPlus 9.2.0
    # 'ep93' for EnergyPlus 9.3.0
    # 'ep94' for EnergyPlus 9.4.0
    # 'ep95' for EnergyPlus 9.5.0
    EnergyPlus_version='ep95',
    # Following input arguments are related to the setup of the output IDFs
    # It is allowed to enter multiple values, but must be lists
    # AdapStand: 0 for CTE, 1 for EN16798-1, 2 for ASHRAE Standard 55
    AdapStand=[0, 1, 2],
    # CAT:
    # 1, 2 or 3 for EN 16798-1's Categories I, II or III
    # 80 or 90 for ASHRAE Standard 55's 80% and 90% acceptability
    CAT=[1, 2, 3, 80, 90],
    # ComfMod:
    # 0 for Static setpoints
    # 1 for OUT-CTE
    # 2 for OUT-SEN16798 or OUT-SASHRAE55
    # 3 for OUT-AEN16798 or OUT-AASHRAE55
    ComfMod=[0, 1, 2, 3],
    # HVACmode: 0 for full air-conditioning, 1 for naturally ventilated, 2 for mixed-mode
    HVACmode=[0, 1, 2],
    # VentCtrl:
    # 1 allows ventilation if operative temperature exceeds neutral temperature
    # 2 allows ventilation if operative temperature exceeds the upper comfort limit
    VentCtrl=[0, 1],
    # VSToffset: applies the entered values as an offset
    # to the ventilation setpoint temperature, in Celsius degrees
    VSToffset=[0, 1, 2, 3],
    # MinOToffset: sets the minimum outdoor temperature as an offset
    # to the heating setpoint temperature, in Celsius degrees
    MinOToffset=[1, 2, 3],
    # MaxWindSpeed: sets the maximum wind speed in which ventilation is allowed, in m/s.
    MaxWindSpeed=[10, 20, 30],
    # ASTtol: applies tolerances for the adaptive heating and cooling setpoint temperatures.
    # Information must be entered as a sequence of numbers:
    # ASTtol_start specifies the start of the sequence,
    # ASTtol_end_input specifies the end of the sequence,
    # ASTtol_steps specifies the spacing between numbers in the sequence
    # in the example below, the output of the AST sequence would be 0, 0.25, 0.50,0.75, 1.00
    ASTtol_start=0,
    ASTtol_end_input=1,
    ASTtol_steps=0.25
    )
```

Appendix C. *accis* module within *accim* package version 0.1.14.

```
"""
Run the function below to add the ACCIS.

This function transform fixed setpoint temperature
building energy models into adaptive setpoint temperature energy models
by adding the Adaptive Comfort Control Implementation Script (ACCIS)
"""

def addAccis(
        ScriptType: str = None,
        Outputs: str = None,
        EnergyPlus_version: str = None,
        AdapStand: any = None,
        CAT: any = None,
        ComfMod: any = None,
        HVACmode: any = None,
        VentCtrl: any = None,
        VSToffset: any = [0],
        MinOToffset: any = [50],
        MaxWindSpeed: any = [50],
        ASTtol_start: float = 0.1,
        ASTtol_end_input: float = 0.1,
        ASTtol_steps: float = 0.1,
        NameSuffix: str = '',
        verboseMode: bool = True,
        confirmGen: bool = None):
    """
    Parameters
    :param ScriptType: The default is 'MultipleZones'. Can be 'MultipleZones'or
        'mz', or 'SingleZone' or 'sz'.
    :param Outputs: The default is 'Standard'. Can be 'Standard',
        'Simplified' or 'Timestep'.
    :param EnergyPlus_version: The default is 'Ep95'. Can be 'Ep91' or 'Ep95'.
    :param AdapStand: The default is None.
    (0 = CTE; 1 = EN16798-1; 2 = ASHRAE 55)
    :param CAT: The default is None.
    (1 = CAT I; 2 = CAT II; 3 = CAT III; 80 = 80% ACCEPT; 90 = 90% ACCEPT)
    :param ComfMod: The default is None.
    (0 = Static; 1 = OUT-CTE; 2 = OUT-SEN16798/SASHRAE55; 3 = OUT-AEN16798/AASHRAE55)
    :param HVACmode: The default is None.
    (0 = Fully Air-conditioned; 1 = Naturally ventilated; 2 = Mixed Mode)
    :param VentCtrl:
    (0 = Ventilates above neutral temperature; 1 = Ventilates above upper comfort limit)
    :param VSToffset: Please refer to documentation.
    :param MinOToffset: Please refer to documentation.
    :param MaxWindSpeed: Please refer to documentation.
    :param ASTtol_start: Please refer to documentation.
    :param ASTtol_end_input: Please refer to documentation.
    :param ASTtol_steps: Please refer to documentation.
    :param NameSuffix: Please refer to documentation.
    :param verboseMode: True to print the process on screen. Default is True.
    :param confirmGen: True to skip confirmation of output IDF generation. Default is None.

    Exceptions
            DESCRIPTION. EnergyPlus version not supported.
        Only works for versions between EnergyPlus 9.1 (enter Ep91) and
        EnergyPlus 9.5 (enter Ep95).

    :return:
    """
    import accim.sim.accim_Main as accim_Main
    from os import listdir

    filelist = ([file for file in listdir() if file.endswith('.idf')
                 and not '_pymod' in file])

    filelist = ([file.split('.idf')[0] for file in filelist])
```

```
objArgsDef = (
    ScriptType is not None,
    Outputs is not None,
    EnergyPlus_version is not None
)

fullScriptTypeList = ['MultipleZone',
                      'multiplezone',
                      'mz',
                      'SingleZone',
                      'singlezone',
                      'sz'
]

fullOutputsList = [
    'Standard',
    'standard',
    'Simplified',
    'simplified',
    'Timestep',
    'timestep'
]

fullEPversionsList = [
    'Ep91',
    'ep91',
    'Ep92',
    'ep92',
    'Ep93',
    'ep93',
    'Ep94',
    'ep94',
    'Ep95',
    'ep95'
]

if all(objArgsDef):
    pass
else:
    ScriptType = input("Enter the ScriptType (MultipleZone or mz, or SingleZone or sz): ")
    while ScriptType not in fullScriptTypeList:
        ScriptType = input("ScriptType was not correct. "
                           "Please, enter the ScriptType "
                           "(MultipleZone or mz, or SingleZone or sz): ")
    Outputs = input("Enter the Output (Standard, Simplified or Timestep): ")
    while Outputs not in fullOutputsList:
        Outputs = input("Output was not correct. "
                        "Please, enter the Output (Standard, Simplified or Timestep): ")
    EnergyPlus_version = input("Enter the EnergyPlus version (Ep91 to Ep95): ")
    while EnergyPlus_version not in fullEPversionsList:
        EnergyPlus_version = input("EnergyPlus version was not correct. "
                                   "Please, enter the EnergyPlus version (Ep91 to Ep95): ")
if verboseMode:
    print('ScriptType is: '+ScriptType)
if ScriptType not in fullScriptTypeList:
    print('Valid ScriptTypes: ')
    print(fullScriptTypeList)
    raise ValueError(ScriptType + " is not a valid ScriptType. "
                                  "You must choose a ScriptType from the list above.")
if verboseMode:
    print('Outputs are: '+Outputs)
if Outputs not in fullOutputsList:
    print('Valid Outputs: ')
    print(fullOutputsList)
    raise ValueError(Outputs + " is not a valid Output. "
                               "You must choose a Output from the list above.")
if verboseMode:
    print('EnergyPlus version is: '+EnergyPlus_version)
```

. (*continued*).

```
if EnergyPlus_version not in fullEPversionsList:
    print('Valid EnergyPlus_version: ')
    print(fullEPversionsList)
    raise ValueError(EnergyPlus_version + " is not a valid EnergyPlus_version. "
                                          "You must choose a EnergyPlus_version"
                                          "from the list above.")

for file in filelist:
    if verboseMode:
        print('''\n========================START OF PROCESS========================\n''')
        print('Starting with file:')
        print(file)
    z = accim_Main.accimJob(
        filename_temp=file,
        ScriptType=ScriptType,
        EnergyPlus_version=EnergyPlus_version,
        verboseMode=verboseMode
    )

    z.setComfFieldsPeople(verboseMode=verboseMode)
    z.addOpTempTherm(verboseMode=verboseMode)
    z.addBaseSchedules(verboseMode=verboseMode)
    z.setAvailSchOn(verboseMode=verboseMode)

    if ScriptType.lower() == 'MultipleZones'.lower() or ScriptType.lower() == 'mz':
        z.addMultipleZoneSch(verboseMode=verboseMode)
        z.addCurveObj(verboseMode=verboseMode)
        z.addDetHVACobj(verboseMode=verboseMode)
        z.checkVentIsOn(verboseMode=verboseMode)
        z.addForscriptSchMultipleZone(verboseMode=verboseMode)
    elif ScriptType.lower() == 'SingleZone'.lower() or ScriptType.lower() == 'sz':
        z.addForscriptSchSingleZone(verboseMode=verboseMode)

    z.addEMSProgramsBase(verboseMode=verboseMode)
    z.addEMSOutputVariableBase(verboseMode=verboseMode)

    if ScriptType.lower() == 'MultipleZones'.lower() or ScriptType.lower() == 'mz':
        z.addGlobVarListMultipleZone(verboseMode=verboseMode)
        z.addEMSSensorsMultipleZone(verboseMode=verboseMode)
        z.addEMSActuatorsMultipleZone(verboseMode=verboseMode)
        z.addEMSProgramsMultipleZone(verboseMode=verboseMode)
    elif ScriptType.lower() == 'SingleZone'.lower() or ScriptType.lower() == 'sz':
        z.addGlobVarListSingleZone(verboseMode=verboseMode)
        z.addEMSSensorsSingleZone(verboseMode=verboseMode)
        z.addEMSActuatorsSingleZone(verboseMode=verboseMode)
        z.addEMSProgramsSingleZone(verboseMode=verboseMode)

    z.addEMSPCMBase(verboseMode=verboseMode)

    if ScriptType.lower() == 'MultipleZones'.lower() or ScriptType.lower() == 'mz':
        z.addEMSOutputVariableMultipleZone(verboseMode=verboseMode)
        if Outputs.lower() == 'Simplified'.lower():
            z.addSimplifiedOutputVariables(verboseMode=verboseMode)
        elif Outputs.lower() == 'Standard'.lower():
            z.addOutputVariablesMultipleZone(verboseMode=verboseMode)
    elif ScriptType.lower() == 'SingleZone'.lower() or ScriptType.lower() == 'sz':
        if Outputs.lower() == 'Simplified'.lower():
            z.addSimplifiedOutputVariables(verboseMode=verboseMode)
        elif Outputs.lower() == 'Standard'.lower():
            z.addOutputVariablesSingleZone(verboseMode=verboseMode)
    if Outputs.lower() == 'Timestep'.lower():
        z.addOutputVariablesTimestep(verboseMode=verboseMode)

    z.saveaccim(verboseMode=verboseMode)
    if verboseMode:
        print('Ending with file:')
        print(file)
        print('''\n=======================END OF PROCESS=======================\n''')
```

. (*continued*).

```
args_needed_mz = (
    AdapStand is not None,
    CAT is not None,
    ComfMod is not None,
    HVACmode is not None,
    VentCtrl is not None,
)

args_needed_sz = (
    AdapStand is not None,
    CAT is not None,
    ComfMod is not None,
)

if ScriptType.lower() == 'MultipleZones'.lower() or ScriptType.lower() == 'mz':
    if all(args_needed_mz):
        z.genIDFMultipleZone(
            AdapStand=AdapStand,
            CAT=CAT,
            ComfMod=ComfMod,
            HVACmode=HVACmode,
            VentCtrl=VentCtrl,
            VSToffset=VSToffset,
            MinOToffset=MinOToffset,
            MaxWindSpeed=MaxWindSpeed,
            ASTtol_start=ASTtol_start,
            ASTtol_end_input=ASTtol_end_input,
            ASTtol_steps=ASTtol_steps,
            NameSuffix=NameSuffix,
            verboseMode=verboseMode,
            confirmGen=confirmGen
            )
    else:
        z.inputdataMultipleZone()
        z.genIDFMultipleZone()
elif ScriptType.lower() == 'SingleZone'.lower() or ScriptType.lower() == 'sz':
    if all(args_needed_sz):
        z.genIDFSingleZone(
            AdapStand=AdapStand,
            CAT=CAT,
            ComfMod=ComfMod,
            ASTtol_start=ASTtol_start,
            ASTtol_end_input=ASTtol_end_input,
            ASTtol_steps=ASTtol_steps,
            NameSuffix=NameSuffix,
            verboseMode=verboseMode,
            confirmGen=confirmGen
            )
    else:
        z.inputdataSingleZone()
        z.genIDFSingleZone()
```

. (*continued*).

References

[1] World Wildlife Fund, Living Planet Report 2014: Species and Spaces, People and Places, WWF International, Gland, Switzerland, 2014, https://doi.org/10.1007/s13398-014-0173-7.2.

[2] M. Meinshausen, S.J. Smith, K. Calvin, J.S. Daniel, M.L.T. Kainuma, J. Lamarque, K. Matsumoto, S.A. Montzka, S.C.B. Raper, K. Riahi, A. Thomson, G.J.M. Velders, D.P.P. van Vuuren, The RCP greenhouse gas concentrations and their extensions from 1765 to 2300, Clim. Chang. 109 (2011) 213–241, https://doi.org/10.1007/s10584-011-0156-z.

[3] European Commission, A Roadmap for Moving to a Competitive Low Carbon Economy in 2050, 2011, pp. 1–15. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52011DC0112 (accessed July 18, 2021).

[4] European Commission, 20 20 by 2020 - Europe's Climate Change Opportunity. https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX%3A52008DC0030, 2008.

[5] European Commission, A Policy Framework for Climate and Energy in the Period from 2020 to 2030, Brussels, 2014, https://doi.org/10.1007/s13398-014-0173-7.2.

[6] L. Pérez-Lombard, J. Ortiz, C. Pout, A review on buildings energy consumption information, Energy Build. 40 (2008) 394–398, https://doi.org/10.1016/j.enbuild.2007.03.007.

[7] K. Park, M. Kim, Energy demand reduction in the residential building sector: a case study of Korea, Energies. 10 (2017) 1–11, https://doi.org/10.3390/en10101506.
[8] R. Lowe, Technical options and strategies for decarbonizing UK housing, Build. Res. Inf. 35 (2007) 412–425, https://doi.org/10.1080/09613210701238268.
[9] E.L. Vine, E. Kazakevicius, Residential energy use in Lithuania: the prospects for energy efficiency, Energy. 24 (1999) 591–603, https://doi.org/10.1016/S0360-5442(99)00013-4.
[10] M.A. Humphreys, Thermal comfort temperatures world-wide - the current position, Renew. Energy 8 (1996) 139–144, https://doi.org/10.1016/0960-1481(96)88833-1.
[11] G.N. Spyropoulos, C.A. Balaras, Energy consumption and the potential of energy savings in Hellenic office buildings used as bank branches - a case study, Energy Build. 43 (2011) 770–778, https://doi.org/10.1016/j.enbuild.2010.12.015.
[12] T. Hoyt, E.A. Arens, H. Zhang, Extending air temperature setpoints: simulated energy savings and design considerations for new and retrofit buildings, Build. Environ. 88 (2014) 89–96, https://doi.org/10.1016/j.buildenv.2014.09.010.
[13] IPCC, Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, Cambridge, 2007. ISBN: 978 0521 88010–7.
[14] K.K.W. Wan, D.H.W. Li, J.C. Lam, Assessment of climate change impact on building energy use and mitigation measures in subtropical climates, Energy. 36 (2011) 1404–1414, https://doi.org/10.1016/j.energy.2011.01.033.
[15] M. Lakeridou, M. Ucci, A. Marmot, I. Ridley, The potential of increasing cooling set-points in air-conditioned offices in the UK, Appl. Energy 94 (2012) 338–348, https://doi.org/10.1016/j.apenergy.2012.01.064.
[16] N. Fernandez, S. Katipamula, W. Wang, Y. Huang, G. Liu, Energy savings modelling of re-tuning energy conservation measures in large office buildings, J. Build. Perform. Simul. 8 (2015) 391–407, https://doi.org/10.1080/19401493.2014.961032.
[17] R.P. Kramer, M.P.E. Maas, M.H.J. Martens, A.W.M. van Schijndel, H.L. Schellen, Energy conservation in museums using different setpoint strategies: a case study for a state-of-the-art museum using building simulations, Appl. Energy 158 (2015) 446–458, https://doi.org/10.1016/j.apenergy.2015.08.044.
[18] C. Sánchez-Guevara Sánchez, A. Mavrogianni, F.J. Neila González, On the minimal thermal habitability conditions in low income dwellings in Spain for a new definition of fuel poverty, Build. Environ. 114 (2017) 344–356, https://doi.org/10.1016/j.buildenv.2016.12.029.
[19] ANSI/ASHRAE, ASHRAE Standard 55–2013 Thermal Environmental Conditions for Human Occupancy, GA, United States, 2013. ISSN: 1041–2336.
[20] T. Parkinson, R. de Dear, G. Brager, Nudging the adaptive thermal comfort model, Energy Build. 206 (2020) 109559, https://doi.org/10.1016/j.enbuild.2019.109559.
[21] M. Gangolells, M. Casals, J. Ferré-Bigorra, N. Forcada, M. Macarulla, K. Gaspar, B. Tejedor, Office representatives for cost-optimal energy retrofitting analysis: a novel approach using cluster analysis of energy performance certificate databases, Energy Build. 206 (2020) 109557, https://doi.org/10.1016/j.enbuild.2019.109557.
[22] Casals Gangolells, Ferré-Bigorra, Macarulla Forcada, Tejedor Gaspar, Energy benchmarking of existing office stock in Spain: trends and drivers, Sustainability. 11 (2019) 6356, https://doi.org/10.3390/su11226356.
[23] D. Sánchez-García, D. Bienvenido-Huertas, M. Tristancho-Carvajal, C. Rubio-Bellido, 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 (2019) 1498, https://doi.org/10.3390/en12081498.
[24] D. Sánchez-García, C. Rubio-Bellido, M. Tristancho, M. Marrero, A comparative study on energy demand through the adaptive thermal comfort approach considering climate change in office buildings of Spain, Build. Simul. 13 (2020) 51–63, https://doi.org/10.1007/s12273-019-0560-2.
[25] D. Sánchez-García, C. Rubio-Bellido, J.J.M. del Río, A. Pérez-Fargallo, Towards the quantification of energy demand and consumption through the adaptive comfort approach in mixed mode office buildings considering climate change, Energy Build. 187 (2019) 173–185, https://doi.org/10.1016/j.enbuild.2019.02.002.
[26] DesignBuilder Software Ltd, DesignBuilder. https://designbuilder.co.uk, 2021 (accessed July 18, 2021).
[27] D.B. Crawley, L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, M.J. Witte, J. Glazer, EnergyPlus: creating a new-generation building energy simulation program, Energy Build. 33 (2001) 319–331, https://doi.org/10.1016/S0378-7788(00)00114-6.
[28] D. Bienvenido-Huertas, D. Sánchez-García, C. Rubio-Bellido, M.J. Oliveira, Influence of adaptive energy saving techniques on office buildings located in cities of the Iberian Peninsula, Sustain. Cities Soc. 53 (2020) 101944, https://doi.org/10.1016/j.scs.2019.101944.
[29] D. Bienvenido-Huertas, D. Sánchez-García, C. Rubio-Bellido, J.A. Pulido-Arcas, Analysing the inequitable energy framework for the implementation of nearly zero energy buildings (nZEB) in Spain, J. Build. Eng. 35 (2021) 102011, https://doi.org/10.1016/j.jobe.2020.102011.
[30] Z. Tong, Y. Chen, A. Malkawi, Z. Liu, R.B. Freeman, Energy saving potential of natural ventilation in China: the impact of ambient air pollution, Appl. Energy 179 (2016) 660–668, https://doi.org/10.1016/j.apenergy.2016.07.019.
[31] ANSI/ASHRAE, ASHRAE Standard 55–2020 Thermal Environmental Conditions for Human Occupancy, Atlanta, GA, United States, 2020. ISSN: 1041–2336.
[32] European committee for standardization, EN 16798–1:2019 Energy performance of buildings, in: 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, 2019. https://en.tienda.aenor.com/norma-bsi-bs-en-16798-1-2019-000000000030297474 (accessed August 6, 2021).
[33] R. Guglielmetti, D. Macumber, N. Long, Openstudio: An open source integrated analysis platform, in: Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association, 2011, pp. 442–449. ISSN: 2522-2708, https://www.osti.gov/biblio/1032670 (accessed August 6, 2021).
[34] D. Sánchez-García, accim web repository. https://github.com/dsanchez-garcia/accim, 2021 (accessed July 19, 2021).
[35] P. Santosh, T. Tuan, E.A. Youngson, J. Bull, eppy web repository. https://github.com/santoshphilip/eppy, 2004 (accessed July 18, 2021).
[36] Alliance for Sustainable Energy LLC, GARD Analytics Inc., EP-Launch. https://github.com/NREL/EP-Launch, 2018 (accessed July 18, 2021).
[37] J. Reback, W. McKinney, Jbrockmendel, J. Van den Bossche, T. Augspurger, P. Cloud, S. Hawkins, Gfyoung, M. Roeschke Sinhrks, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, Patrick, M. Garcia, J. Schendel, A. Hayden, D. Saxton, V. Jancauskas, M. Gorelli, R. Shadrach, A. McMaster, P. Battiston, S. Seabold, K. Dong, chris-b1, h-vetinari, pandas-dev/pandas: Pandas 1.2.4, 2021, https://doi.org/10.5281/zenodo.4681666.
[38] D. Sánchez-García, accim's documentation. https://accim.readthedocs.io/en/latest/index.html, 2021.
[39] The Government of Spain, Royal Decree 2429/79. Approving the Basic Building Norm NBE-CT-79, about the Thermal Conditions in Buildings. https://www.boe.es/eli/es/rd/1979/07/06/2429, 1979 (accessed August 6, 2021).
[40] F. Rubel, M. Kottek, Observed and projected climate shifts 1901-2100 depicted by world maps of the Köppen-Geiger climate classification, Meteorol. Z. 19 (2010) 135–141, https://doi.org/10.1127/0941-2948/2010/0430.
[41] ANSI/ASHRAE, ASHRAE Guideline 14–2014: Measurement of Energy, Demand, and Water Savings, GA, United States, 2014. ISSN: 1049-894X.
[42] The Government of Spain, Royal Decree 314/2006. Approving the Spanish Technical Building Code CTE-DB-HE-1, 2013, pp. 1–43. https://www.boe.es/eli/es/rd/2006/03/17/314 (accessed August 6, 2021).
[43] R. Castaño-Rosa, J. Solís-Guzmán, C. Rubio-Bellido, M. Marrero, Towards a multiple-indicator approach to energy poverty in the European Union: a review, Energy Build. 193 (2019) 36–48, https://doi.org/10.1016/j.enbuild.2019.03.039.
[44] A. Pérez-Fargallo, C. Rubio-Bellido, J.A. Pulido-Arcas, F.J. Guevara-García, Fuel poverty potential risk index in the context of climate change in Chile, Energy Policy 113 (2018) 157–170, https://doi.org/10.1016/j.enpol.2017.10.054.