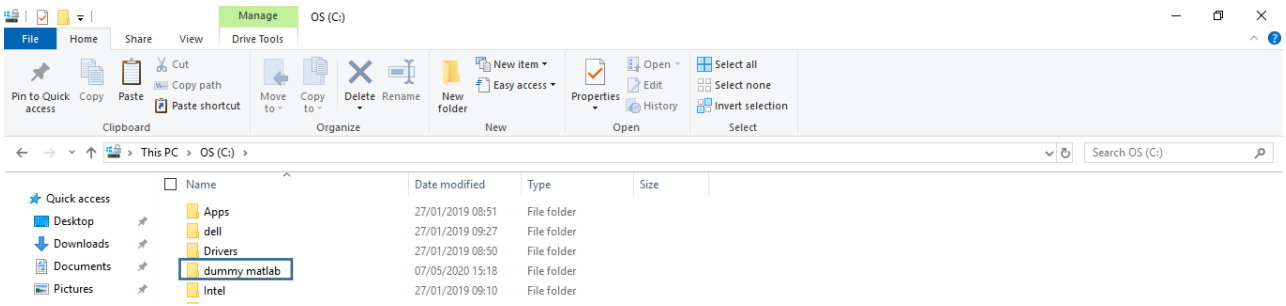
EWASP Instruction Manual

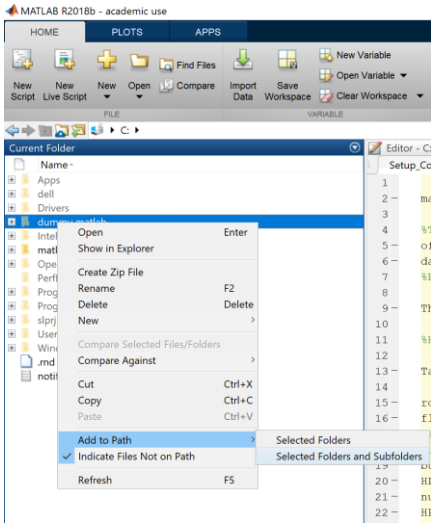
Table of Contents

1 Installing and Running the Model

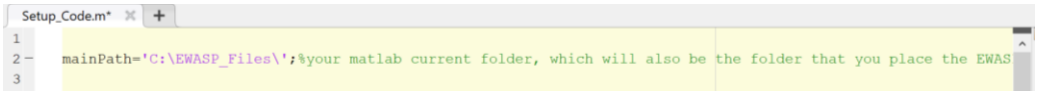
- In order to run, a full working copy of MATLAB, Simulink, and the Simscape Electrical add-on, is required (all of these are usually included in a full installation of MATLAB with an academic license). The model was built using MATLAB 2018, and so this version or later is recommended (though it **may** be somewhat backwards compatible). Ensure these requirements are met before moving to the next stage.
- Download the 'EWASP' folder
- Open the 'EWASP' folder, and place the subfolder 'dummy matlab' directly into the directory 'C:',



- Rename the subfolder as desired, open MATLAB and add the folder and subfolders to path,



- Open the MATLAB code 'dummy matlab/EWASP/Setup_Code.m' – this is the code that input parameters are controlled by, and it contains the only lines that the user must interact with.
- If you renamed the 'dummy matlab' folder, this should be accounted for on line 2 of 'Setup_Code.m'. For example, if 'dummy matlab' has been renamed 'EWASP_Files', then line 2 should be,



- Open the Simulink code 'dummy matlab/EWASP/Final_Model_Building_2_AirflowAdded.slx' – this is the Simulink code that performs the dynamic building simulation.

The model should now run – clicking the Run icon whilst the ‘Setup_code’ window is active should now cause EWASP to simulate the electrical and thermal profiles of 100 mid 00’s detached properties with GSHPs for space heating, and an 80% chance of using GSHPs for DHW storage heating, on a typical winters day. All of these parameters can be altered to suit the user – the method for which is described in section 2.

2 Operational Modes

The EWASP model has 2 distinct operating modes:

- **Automatic Structure Mode:** The layout of the building is automatically assigned based on the chosen building age and type
- **Manual Structure Mode:** The layout of the building is assigned by the user

Regardless of which operational mode is chosen, u-values, building fabric, thermostat settings, occupancy, appliance demand, and user behavior, are randomly assigned either using pre-generated databases or ONS statistical distributions. Weather time series are derived from meteorological datasets.

2.1 Automatic Structure Mode

This mode allows the user to load preset structures for their buildings – the exact structure depends on the chosen building type and age, and represents a typical floorplan. In this instance, the user need only define the following parameters.

Manual_Structure_Definition

MUST equal 0 for automatic mode to begin (1 will activate manual mode).

OffsetDays

Day of the year to simulate, where 1st of January = 0 (valid values 0 → 364).

DayType

Weekday or weekend (valid values 'WD' or 'WE').

ThermostatBand

The width in °C of the room thermostat hysteresis band (suggest leaving this at 0.1, as this is typical).

Tank_Exist_Probability

The probability that any given building has a DHW tank. Size is automatically calculated based on number of occupants. For example, if 'numRuns'=100, and 'Tank_Exist_Probability'=0.85, then around 85 of the simulated residences will have hot water tanks.

rotation

Rotates the building – all buildings are defined with the assumption that the face with the front door is facing north. By entering values 0, 90, 180, or 270, the building can be rotated clockwise around compass directions (e.g. if 90 was entered, the front door face would instead be facing east). Valid values are 0, 90, 180, and 270.

flip

Flips the building so that the E wall becomes the W wall, and vice versa. This affects the airflow and irradiance driven components of the model.

buildingAge

Year the residence was built – influences the structural layout, all U values, capacitances, material densities, and Air leakage properties. Valid range 0 → infinite, though all values pre 1900 and post 2010 will produce the same results.

buildingType

Type of house valid values 1 = End Terraced, 2 = Mid Terraced, 3 = Semi-Detached, 4 = Detached.

HPdT

The maximum difference in temperature between inside and outside that the heat pump is sized to achieve (at typical wind speeds). This would typically be around 24 – 30°C, though under most operating conditions this variable has little effect. Valid range 0 →∞ (though vastly exceeding 30°C would be an unusual choice in the UK)

numRuns

Number of simulations to run (range 1+)

HPType

The source type of the heat pump. 2 allowed values:

- 'G' → Ground Source
- 'A'→ Air Source

2.2 Manual Structure Mode

All parameters required in automatic structure mode must be defined as detailed in section 2.1. The following additional parameters must also be defined:

ManualStructureDefinition

Must equal 1

roomUse

1 X 12 vector that details the purpose of rooms 1 through 11. Entry 12 should equal ' ', the remaining 11 entries must equal one of the following:

- 'L' → Living/Dining Room
- 'B' → Bedroom
- 'O' → Other (e.g. kitchen/bathroom/utility)
- ' ' → Room Does not exist/would not be expected to have heating

Example:

```
roomUse=['L' 'O' 'O' 'O' 'O' 'B' 'B' 'B' 'B' 'O' 'O' ' '];
```

frenchDoors

1 X 12 vector that details the presence/absence of French doors for rooms 1 through 11. Element 12 should always equal 'N'. The remaining elements should be assigned as follows:

- 'Y' → French doors are present
- 'N' → No French doors, or room does not exist

Example:

Only room 1 has French doors,

```
frenchDoors=['Y' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N' 'N'];
```

upDown

11 X 1 vector that details the floor that each room is on. Each element must take one of the following values

- 0 → Room is on ground floor
- 1 → Room is on first floor
- 2 → Room is on second floor/is a loft conversion

Example:

```
upDown=[0;0;0;0;1;1;1;1;1;1;1];
```

floorType

Value describing the floor construction. May take only the following values

- 0 → Ground floor is suspended timber (more typical of pre millennium properties, though this is not exclusively true)
- 1 → Ground floor is concrete slab (more typical of post millennium properties, though this is not exclusively true)

floorOrCeilingArea

The area of the floor/ceiling (note: the model assumes rooms are square, and so floor and ceiling areas are equal and interchangeable). 1 X 12 vector where final element always equals 0, and each other element equals the floor/ceiling area of the room.

Example:

A House is represented by 3 zones with zone 1 floor area = 10m^2, room 2 floor area = 12m^2, room 3 floor area = 9m^2. Therefore the required cod for floorOrCeilingArea is:

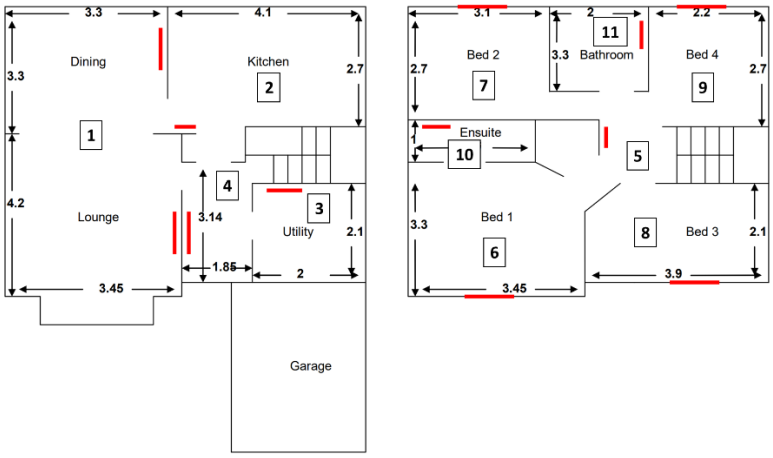
```
floorOrCeilingArea= [10 12 9 0 0 0 0 0 0 0 0 0];
```

NESW_Lengths

The length of external wall (in meters) (**NOT Areas**) for each room (room number equal to column number) that is facing North (row 1), East (row 2), South (row 3), West (row 4).

Example

This is best explained using a floorplan and its equivalent NESW matrix. Assume the bottom face of the building is facing north, and labels represent room numbers (where lounge and dining are approximated as one room)



The required Code is:

```
NESW_Lengths=[
3.4 0 0 2.0 0 3.5 0 3.9   0   0 0
7.5 0 0 0   0 3.3 2.7 0   0   0 1
3.3 4.1 0   0 0 0 3.1 0   2.2 2 0
0   2.7 2.1 1 0 0 0   2.1 2.7 0 0];
```

NESW_Windows

The same logic as with 'NESW_Lengths' but with window **areas** replacing wall lengths in the matrix.

Example

Using the previously shown floorplan, the appropriate matrix would be approximately equal to:

```
NESW_Windows=[
3.6 0   0   0   0   1   0   1   0   0   0
0   0   0   0   0   0   0   0   0   0   0
2.5 2.1 0   0   0   0   1   0   1   0   0
0   0   0   0   0   0   0   0   0   0   0];
```

Note: window areas are approximated from floorplan

verticalConnectMatrix

Describes which rooms/zones are connected by a stairwell, where the row is the current room, the column is each of the other rooms. If room 1 were connected to room 2 by a stairwell, then the matrix entry (1,2) would be 1 if room 2 was above room1, or -1 if room 2 was below room 1. The matrix MUST be skew-symmetric i.e. if (1,2)=1 the (2,1)=-1. The matrix must be consistent with the 'upDown' vector i.e. if room 2 is defined as being above room 1 in this matrix, then the entry for room 2 in 'upDown' must be one greater than the entry for room 1, otherwise the airflow model will give bizarre results or fail to solve.

Example

Downstairs Room 4 is linked to upstairs room 5 via a stairwell, there are no other stairwells in the building. The correct code is therefore:

```
verticalConnectMatrix=[
```

```
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0
 0  0  0 -1  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
];
```

HCM_State_1

11 × 11 matrix that describes how each of the rooms are connected by doorways during normal daytime operation, where 0=not connected via a doorway, 1=connected via an open door, 2=connected via a closed door. If room 2 were connected to room 3 by an open doorway, then the element (2,3) would equal 1. Furthermore, the matrix must be symmetric i.e. if (2,3)=1 then (3,2)=1 (this makes logical sense, for if room 2 is connected to 3, then 3 is connected to 2).

Example

The example floorplan is represented in the following matrix, and it is assumed that all doors are open, except the one between room 6 and room 10:

```
HCM_State_1=[
 0  1  0  1  0  0  0  0  0  0  0
 1  0  0  1  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0  0
 1  1  1  0  0  0  0  0  0  0  0
 0  0  0  0  0  1  1  1  1  0  1
 0  0  0  0  1  0  0  0  0  2  0
 0  0  0  0  1  0  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0
 0  0  0  0  0  2  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0
];
```

HCM_State_2

11 × 11 matrix that describes how each of the rooms are connected by doorways during night operation. The option to define this is included as many households have significantly more closed internal doors at night. The matrix is defined in exactly the same way as HCM_State_1. The usage of this 2nd state can be disabled by setting 'Diurnal_State_Active=0'or enabled by setting 'Diurnal_State_Active=1'. Please note that any element that was set to zero in 'HCM_State_1' must remain zero in 'HCM_State_2'. Failing to follow this rule will not cause a failure to solve, but implies that an internal doorway is being added and removed from the house every day.

conductiveHconnectMatrix

11 × 11 matrix that details the length of wall between rooms e.g. if room 1 shares 3 meters of internal wall with room 2, then the elements (1,2) and (2,1) both equal 3 (note: the matrix must be symmetric).

It should be noted that this matrix is not essential to the running of the model, and if there is no desire to model conductive heat transfer through internal walls, the matrix can be left empty.

Example

The correct matrix for the floor plan detailed in the example floorplan *would be:*

```
conductiveHconnectMatrix=[
0   3.3 0   3.1 0   0   0   0   0   0   0
3.3 0   0   3.8 0   0   0   0   0   0   0
0   0   0   4.1 0   0   0   0   0   0   0
3.1 3.8 4.1 0   0   0   0   0   0   0   0
0   0   0   0   0   4.3 0.6 3.9 2.8 0   2
0   0   0   0   4.3 0   0   1.6 0   2.7 0
0   0   0   0   0.6 0   0   0   0   3.1 3.3
0   0   0   0   3.9 1.6 0   0   0   0   0
0   0   0   0   2.8 0   0   0   0   0   3.3
0   0   0   0   0   2.7 3.1 0   0   0   0
0   0   0   0   2   0   3.3 0   3.3 0   0];
```

conductiveVconnectMatrix

11 × 11 matrix that details which rooms lie directly above/below others, and handles conductive heat transfer between intermediate floors. Each room is allowed to be defined as above no more than 2 rooms, and below no more than 2 rooms, so the user should choose the largest room overlaps). If room 1 is below room 5, then the entry (1,5) should be 1, and if room is above room 5, then (7,5) should equal -1. The matrix DOES NOT have to be skew-symmetric (i.e. (2,1) does not have to equal –(1,2)); this is because whilst a significant portion of room 1 may be below room 2, room 2 may be above far larger areas of room 3 and 4 than room 1.

It should be noted that this matrix is not essential to the running of the model, and if there is no desire to model conductive heat transfer through intermediate floors, the matrix can be left empty.

Example

A sensible matrix for the example floorplan would be:

```
conductiveVConnectMatrix=[
 0  0  0  0  0  1  1  0  0  0  0
 0  0  0  0  0  0  0  0  1  0  0
 0  0  0  0  0  0  0  1  0  0  0
 0  0  0  0  0  0  0  1  0  0  0
 0  0  0 -1  0  0  0  0  0  0  0
 -1  0  0  0  0  0  0  0  0  0  0
 -1  0  0  0  0  0  0  0  0  0  0
 0  0 -1  0  0  0  0  0  0  0  0
 0 -1  0  0  0  0  0  0  0  0  0
 -1  0  0  0  0  0  0  0  0  0  0
 0 -1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
];
```

Diurnal_State_Active

Defines whether the 'doorway state' defined in 'HCM_State_2' is ever activated. It can take the valid values:

0 → 'HCM_State_2' is never activated
1 → 'HCM_State_2' is activated between the hours 'Evening_State_Start_Hour' and 'Evening_State_End_Hour'

'Evening_State_Start_Hour' and 'Evening_State_End_Hour'

Evening_State_Start_Hour is the hour (between 0 and 23) in which the doorway state transitions from 'HCM_State_1' (the daytime state) to 'HCM_State_2' (the night-time state).

Evening_State_End_Hour is the hour (between 0 and 23) in which the doorway state transitions from 'HCM_State_2' (the night-time state) back to 'HCM_State_1' (the daytime state).

It does not matter whether the hour value for 'Evening_State_End_Hour' is larger or smaller than the value for 'Evening_State_Start_Hour' – the model will figure out which hours are between the start and end times e.g. choosing a start time of 23 and end time of 6 will result in the hours 0,1,2,3,4,5,6 and 23 being assigned the night-time doorway state.

Example

HCM1 → HCM2 start transition occurs at 10:00 pm, and reverts at 6:59 am.

The correct code is therefore:

```
Evening_State_Start_Hour=10;
Evening_State_End_Hour=6;
```

2.3 Advanced Settings Mode

The advanced settings relate to various hot water tank controls, space heating return loop controls that act alongside the room thermostat, defrost controls. It is suggested that these are left alone during normal use of the model, and it should be noted that significant changes may produce unpredictable model behaviors. They can be altered in both Manual and automatic structure definition mode.

Aux_Threshold

Temperature at which the Auxiliary heater takes over from the heat pump when raising the temperature to 60°C for legionella control – this value MUST be equal to or lower than Tank_Upper_Temp or the Aux heater may never activate.

Tank_Upper_Temp

Temperature at which heating stops during normal operation.

Tank_Lower_Temp

Temperature at which tank heating begins during normal operation.

HEX_ReturnLowerLimit

If it is detected that the return side temperature of the indirect heating loop that feeds the tanks heat exchanger falls below this temperature, the heat pump begins to heat this loop.

HEX_ReturnUpperLimit

If it is detected that the return side temperature of the indirect heating loop that feeds the tanks heat exchanger reaches this temperature, the heat pump stops heating this loop.

ReturnUpperLimit

Temperature that the return side of the Space heating loop must reach before a heating of this loop ceases.

ReturnLowerLimit

Temperature that the return side of the Space heating loop must fall to before a request is sent to begin heating the loop (heating will not begin if DHW heating is in progress).

Defrost_After_Time

Cumulative minutes of operation below 1.7°C before a defrost is required (Typical values 30 → 90).

DesiredDeltaT

Desired delta t across radiator flow and return ports - Affects chosen radiator flow rate and size – values outside of range 5 - 10°C not advised.

3 Interactor Model

The interactor model can be used by a MATLAB competent user to program additional logic into the basic model. There are no particular rules as to how logic must be programmed, though there are some general rules and some inputs and outputs with fixed behaviors and function. The inputs and o

3.1 Inputs

Clock

Inputs current simulation time, should not be changed, but can be used in the user's program as an input variable to their logic.

HeatSysIn

A value representing the temperature of a room component of the building model, whose name is contained within a Simulink 'from' block. Can take any of the values:

T1,T2,….,T11 → temperatures of zones 1 through 11. *<Note the default thermostat is in zone 1, so T1 may be of limited use as an input variable here>*

ExternalIn and ExternalIn2

A pair of 1440x1 vectors that can be used to input a value that varies minute by minute (1440 = 60 min × 24 hours). Could be used to represent a request logic (e.g. where 1 means a request to perform an operation is being made, 0 means no request is being made), or a continuous value input such as grid frequency or grid demand – see examples 2 & 3 in the tutorial for ideas on how these inputs may be used.

ModelFeedbackIn1 & ModelFeebackIn2

Return the output from ModelFeebackOut1 or 2. This allows the model to update variables from previous time steps. See examples 1 & 3 in the tutorial for an example of usage.

3.2 Outputs

SystemActive

Binary Variable. Deactivates heating system if set to zero, returns full control to default heating system logic if set to 1. Routed to a scope in tutorial, Routed to space heating control in the real model.

RelaxBand

Binary variable. Changes the thermostat band to 16-16.1oC in the real model if set to 1, (minimum acceptable for thermal comfort) leaves thermostat band alone if set to zero. Routed to a scope in tutorial to show binary output.

ModelFeedbackOut & ModelFeedbackOut2

Used to send variable values to the next timestep - useful when updating a variable such as battery state of charge, or latching a value when programming an additional thermostat (see example 1)

MonitorOut & MonitorOut2

Used to monitor outputs that the user may want to measure, such as battery SoC or discharge power. Routed to scopes in the tutorial file, printed to workspace and stored in variables 'MonitorOut1' and 'MonitorOut2' in the full model. Printout is of dimensions 1440 × numRuns.

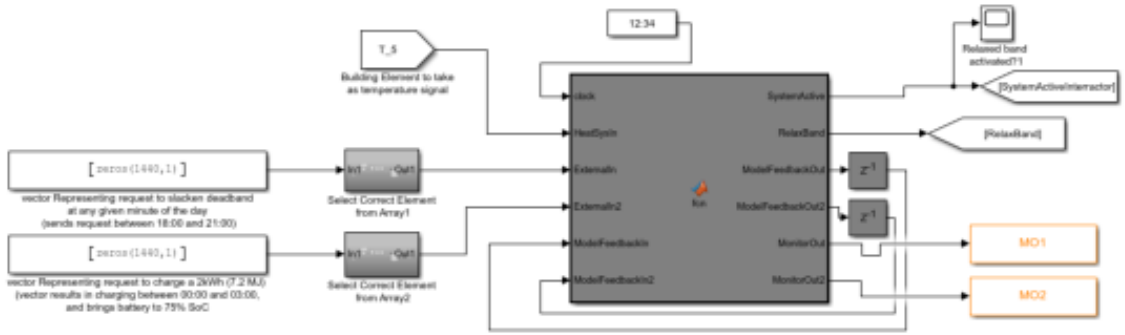
3.3 Finding the Tutorial File

The tutorial file for the model is found in `'C:\x\EWASP\InterractorModelTutorial.slx'`, where `x` is the name you replaced 'dummy matlab' with. It shows 3 examples of how the block may be used to include additional logic, or to add electrical components such as energy storage. Double-clicking the file will open it in Simulink.

3.4 Finding the Interactor Submodel in the full EWASP model

Open the Simulink model (must be open to allow model to run – see section 1), and

You will find a subsystem that looks like this:



Grey Components: Components that are part of the model
White Components: Example Inputs/Outputs

Double clicking on input variables will allow the user to change the values. Vectors attached to ExternalIn & ExternalIn2 are often better defined in the matlab workspace. Double clicking on the larger 'fcn' block in the center will allow the user to edit the algorithms that govern the block behavior. The code contained within the 'fcn' block is near identical to that contained within the tutorial file, but the parameter values are altered such that the block doesn't interfere with the models' basic operation.

3.5 Programming the logic

There are no rules to the logic the user chooses to program, though examples of some common idea may be found in the tutorial file (see section 3.3). It is worth noting that unused output may need to be set to a default value in your code (e.g. if SystemActive is not used, the line SystemActive=1 should be included in the users code) to avoid errors, though it should be stressed that by default, the block is programmed such that it **does not** interfere with the basic operation of the model at all.

It should also be noted that by editing this block, the user is expanding on the basic model, and thus we cannot guarantee that the model will work/run under your additional control schemes.

Use of the interactor model will require some expertise with MATLAB coding, though we endeavor to make this part of the code more accessible to non-MATLAB users in future iterations.

4 Outputs

The model will produce 2 outputs, which will appear in the MATLAB workspace upon completion of the simulation

ElecOut

1440 × numRuns array containing the electricity demand profile (in units W) for each simulation

ThermOut

1440 × numRuns array

5 Errors & Bugs

Because this is an early version of the model, it MAY occasionally fail to solve due to unaddressed bugs. The instance of this we have so far encountered are:

- Undampened oscillations in room temperatures caused by drastic instantaneous changes in the airflow model. This was eliminated by suppressing the rate at which airflow could change within any given second, and we have not yet observed any further errors.
- Undampened oscillations in room temp caused by definition of extremely small downstairs or upstairs landing areas as individual zones. To overcome this error, either avoid defining spaces less than $1m^2$ as their own zone i.e. integrate them into the area of a neighboring room, or approximate that they are slightly larger (note this should affect model results by <1% for even the smallest and leakiest of houses). Alternatively, the model step size can be reduced within Simulink; the default step size is 5s, and any value that is an integer division of this will work.

If an unresolvable error does occur:

- Take note of the number of columns in 'ElecOut' in the workspace column (e.g. if the variable is shown to be '1440x49 double' then the number of columns is 49.
- On line 267 '`i=1:numRuns`' change the value 1 to number of columns+1'
- Position the cursor anywhere on line 264, and press the 'Run Section button'

This will resume the model at the run number where it failed. We will try to remove the need for manual handling of this error in future updates.

If you are able – email the parameters that caused the model to fail to r.c.johnson@sheffield.ac.uk with the error message, and we will endeavor to find the bug and eliminate it.