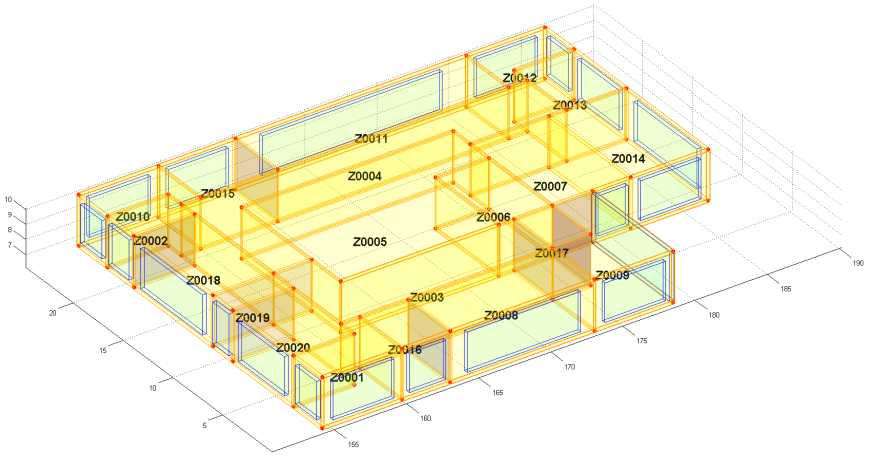
BRCM Toolbox

Documentation

Version 0.95 (beta)
December 10, 2013



David Sturzenegger and Vito Semeraro

www.brcm.ethz.ch

Contents

1 Introduction

Model predictive control (MPC) is a promising alternative in building control with the potential to improve energy efficiency and comfort and to enable demand response capabilities. Creating an accurate building model that is simple enough to allow the resulting MPC problem to be tractable is a challenging but crucial task in the control development.

This Building Resistance-Capacitance Modeling (BRCM) Matlab Toolbox facilitates the physical modeling of buildings for MPC. The Toolbox provides a means for the fast generation of (bi-)linear resistance-capacitance type models from basic geometry, construction and building systems data. Moreover, it supports the generation of the corresponding potentially

time-varying costs and constraints.

The applied modeling principles base on the already successfully validated approaches [1] and [2]. The input data are retrieved from a set of Excel or CSV files which to a large part can also be generated directly from EnergyPlus input data files.

The starting point for model development is always a *thermal model* of the building structure (walls, floors, ceilings, room air volumes) that models the heat exchanges among the elements of the building structure. This model can then be augmented by the addition of any number of *external heat flux* (EHF) models that represent heat fluxes from and to the building structure. The Toolbox currently supports the following EHF models: convective/conductive/solar building envelope heat fluxes, internal gains, conditioned ventilation, radiators and heating/cooling systems within constructions. The software architecture is such that further EHF models can be easily developed and integrated into the Toolbox at a later point in time also by other parties.

The Toolbox further provides basic functionality for the three-dimensional display of the target building's geometry, and for the simulation of a discrete-time representation of the model in open- and closed-loop modes. The BRCM Toolbox is open source and can be used under the GPLv3 license.

For more background information we recommend in this order [3] (description, discussion and some validation of the Toolbox) and [1],[2] (description and validation of the underlying modeling principles).

1.1 General Workflow

Below the general workflow is outlined for using the Toolbox. As mentioned in Section 2, it is recommended that first-time users use the example files coming with the installation.

1. Model the target building by specifying in Excel or CSV files its geometry and construction and its building systems and models of other influences. This is described in Section 3.

2. Generate the model. This step involves actually running the Toolbox functions. It is described in detail in Section 4.

3. Use the model for control or simulation. After generating the model, the system matrices can be readily extracted and functions can be called that generate all constraints and cost matrices as a function of time-varying parameters. To use the model in MPC predictions of the disturbance variables must be made available which is currently not supported by the Toolbox. Even though the Toolbox includes simulation functionality, we expect most users to use the resulting model and costs/constraints in an external simulation setup.

1.2 Model Structure

The BRCM Toolbox aims at providing all necessary matrices/vectors of the following MPC problem

$$\min_{u_0 \dots u_{N-1}} \sum_{k=0}^{N-1} c_k^T \cdot u_k \tag{1}$$

$$\text{s.t.} \quad \text{system dynamics (3)}$$

$$F_{x,k} x_k + F_{u,k} u_k + F_{v,k} v_k \leq f_k \tag{2}$$

$$\forall k = 0, 1, ..., N-1$$

where x_k denotes the states (temperatures of rooms or wall/floor/ceiling layers), u_k the inputs (e.g. heating power or blinds position) and v_k the predicted disturbances (e.g. solar radiation or ambient temperature) at prediction time step k. $F_{x,k}$, $F_{u,k}$, $F_{v,k}$, f_k and c_k denote the potentially time-varying constraint matrices and vectors (actuator and comfort constraints) as well as the cost vector (typically energy or monetary costs) of appropriate dimensions at prediction time step k. The system dynamics are given by

$$\begin{aligned} x_{k+1} =& Ax_k + B_u u_k + B_v v_k + ... \\ & \sum_{i=1}^{n_u} (B_{vu,i} v_k + B_{xu,i} x_k) u_{k,i} \\ \forall k =& 0, 1, ..., N-1 \\ x_0 =& x, \end{aligned} \tag{3}$$

with x denoting the estimated state of the system at the beginning of the MPC horizon and n_u the number of control inputs. A, B_u, B_v and $B_{vu,i}$, $B_{xu,i}$ $i = 1, ..., n_u$ are matrices of appropriate sizes. The BRCM Toolbox is concerned with creating these matrices and providing functions to generate $F_{x,k}$, $F_{u,k}$, $F_{v,k}$, f_k and c_k as a function of potentially time-varying parameters (e.g. minimum supply air temperature or electricity costs).

In [1] we introduced the distinction between the dynamic thermal model and static EHF models. It allows the separation of the rather general modeling of the building's thermal dynamics from the much more building specific modeling of building systems and other external influences. The building's thermal model[1] as a function of EHFs $q(x(t), u(t), v(t))$ is given by

$$\dot{x}(t) = A_t x(t) + B_t q(x(t), u(t), v(t)). \tag{4}$$

The EHF models as functions of control inputs and predictable disturbances

$$\begin{aligned} q(x(t), u(t), v(t)) =& A_q x(t) + B_{q,u} u(t) + B_{q,v} v(t) + ... \\ & \sum_{i=1}^{n_u} (B_{q,vu,i} v(t) + B_{q,xu,i} x(t)) u_i(t) \end{aligned} \tag{5}$$

have in general to be modeled bilinearly to account for mass-temperature products, see [3]. Note that there are no input-input bilinearities. The resulting MPC model is typically solved with good results using sequential linear programming. Model (4) is then combined with (5) and subsequently discretized to obtain (3), which in the following we will refer to as the *full model*.

1.3 Documentation Content

Section 2 describes the installation procedure and recommended first steps. Section 3 outlines how to specify a building in the Toolbox' input data files. In Section 4 the functionality of the Toolbox involved with the actual model generation from the input data is described by usage examples. In Section 5 the modeling principles of the thermal model and the required input data are reported, while Section 6 covers the modeling principles of the external heat flux models and the structure of the required input data. In Section 7 finally, the conversion of EnergyPlus input data files to the Toolbox' thermal model input data is described.

[1] We use the subscript "t" to denote matrices of the thermal model (e.g. A_t) and subscript "q" for matrices of the EHF models

2 Installation and First Steps

The installation of the BRCM Toolbox is managed via tbxManager[2], a very convenient single Matlab m-file that installs and updates Matlab toolboxes. The following steps will install the BRCM Toolbox.

1. Install tbxmanager as described on `www.tbxmanager.com`.
2. Install the BRCM Toolbox by typing in Matlab

```
tbxmanager install brcm
```

3. Run the setup file (this is not strictly necessary but highly recommended[3])

```
BRCM_Setup
```

As a first step it is recommended to have a look at the heavily commented `BRCM_DemoFile.m` which guides step by step through the Toolbox' functionality. To update the BRCM Toolbox the following command can be used

```
tbxmanager update brcm
```

3 Modeling - Preparation of Input Data Files

In this context, we mean by modeling the specification of the Toolbox' input data files which comprise the thermal model and the EHF model input data files. These files can either be in Excel format (it is possible to use OpenOffice Calc and save as Excel) or semicolon-separated CSV files. All files must adhere to a specific structure. This brief section is intended to give a general overview of the modeling process. For more detail on the structure and the meaning of the files we refer to Sections 5 and 6. In any case we recommend reusing the example files to avoid having to rewrite all headers etc.

3.1 Thermal Model Data

The thermal model data is organized around zones (air volumes with assumed uniform temperature) and building elements which each are described in a separate input data file. A building element represents a wall/floor/ceiling/roof and either connects two zones or one zone and a

[2] `http://www.tbxmanager.com`

[3] The Toolbox was mainly developed in object-oriented Matlab with release version R2013a. Using R2013a, the Toolbox makes use of the `meta.class` class member access. However, this feature is only supported since release version R2012a. In order to support older versions, the `meta.class` usage is as a default turned off (making many properties and methods public). When executing the `BRCM_Setup` function in R2012a or more recent, the relevant class files are modified such that `meta.class` is used.

boundary condition to the model (e.g. ambient air, adiabatic, prescribed temperature profile etc.). A building element has a construction type which consists of layers of different materials. Constructions and materials are specified in separate input data files. A building element can also have a so-called no-mass construction which is modeled as a static thermal resistance. These no-mass constructions are specified in another input data file. Building elements can also contain a window which themselves are specified in an input data file.

One option for modeling is to specify the thermal model by hand by first defining the zones, then the connecting building elements and the according constructions/materials/windows. An alternative is to use a parser to generate the thermal model data from an EnergyPlus input data file. This alternative is described in detail in Section 7.

3.2 EHF model Data

The specification of the EHF models must happen after the thermal model has been modeled since specific zones and building elements must be addressed. The exact structure of the EHF model are defined in 6. Usually at least the building hull EHF model is included to define the building's heat exchange with the environment.

4 Model Generation - Toolbox Functionality and Usage

This section describes the steps that turn the input data files into an MPC applicable model. It is organized around the following workflow.

1. Create a `Building` object (Section 4.1).
2. Load the thermal model data (Section 4.2).
3. Visualize the building (Section 4.3) (optional).
4. Modify thermal model data (Section 4.4) (optional).
5. Declare the EHF models which are to be included (Section 4.5).
6. Generate the full model (Section 4.6).
7. Access the model matrices and generate costs and constraints (Section 4.7).
8. Simulation (Section 4.8) (optional).

4.1 Creating a Building Object

A building instance of the class `Building` is created with the following command

```
B = Building('MyBuilding');
```

This will result in an object with the following properties

```
Building with properties:

            identifier : 'MyBuilding'
        building_model : [1x1 BuildingModel]
    thermal_model_data : [1x1 ThermalModelData]
EHF_model_declarations : [0x0 struct]
```

The object `B` has the identifier [4] `'MyBuilding'`. The property `thermal_model_data` of type `ThermalModelData` is a (still empty) container for all the geometry, construction and materials data of a building, see Section 4.2. The property `building_model` of type `BuildingModel` is a (still empty) container for the state-space matrices of the full model, the thermal sub-model, EHF sub-models and state/input/output/heatflux/constraint identifiers[5] as well as some other model information, see Section 4.6. The (still empty) `EHF_model_declarations` property stores the list of EHF models which are to be included, see Section 4.5.

4.2 Thermal Model Data Input/Output

The thermal model data is specified in the form of seven strictly structured Excel files or CSV files, see Section 5.2. To load the thermal model data the following command can be used

```
MyBuilding.loadThermalModelData(pathToThermalModelDataDir);
```

The above method can be also executed without arguments, in which case a pop-up window will ask for a directory. All required files must be available in the specified directory and they must be conform with the structure defined in Section 5.2, otherwise an error will be thrown. The data is loaded into the `thermal_model_data` property of the `Building` object, e.g.

```
ThermalModelData with properties:

               zones : [1x20 Zone]
   building_elements : [1x124 BuildingElement]
       constructions : [1x10 Construction]
           materials : [1x18 Material]
             windows : [1x20 Window]
          parameters : [1x12 Parameter]
nomass_constructions : [1x1 NoMassConstruction]
        source_files : [1x1 struct]
            is_dirty : 1
```

The `zones`, `building_elements`, `constructions`, `materials`, `windows`, `parameters`, `nomass_constructions` properties are arrays of objects containing the data from the files. The `source_files` stores the paths of the individual files. The `is_dirty` flag finally indicates that the data may not be consistent and is set to 1 whenever any manipulation of the above data objects takes place. Consistency of the data means that all cross-referenced data identifiers exist (e.g. the zone to which a certain building element is adjacent must appear in the `zones` property), that all string values are according to our conventions and that all numerical values lie in their specific allowed set, see Section 5 details on those conventions. The consistence can be checked by executing

```
B.checkThermalModelDataConsistency;
```

[4] In the BRCM Toolbox context, identifier strings must start with a letter and are only allowed to contain alphanumeric characters and '_' (i.e. the same syntax as for Matlab variables).

[5] These identifiers are strings having a specific structure that are used to bookkeep the meaning of matrix entries, e.g., the entry on row 5 and column 3 of the B_u matrix in (3) is the influence of the input with name at third position in the input identifiers on the state with name at fifth position in the state identifiers.

If the model is consistent, the `is_dirty` property is set to 0 which allows the generation of the thermal model. If `is_dirty` is 1 and the user requests the thermal model to be generated, the consistency check is performed automatically. Note however that when the thermal model data is loaded, only the files' existence and structure are checked but not their consistency. The thermal model data can also be written back to files

```
B.writeThermalModelData(pathToModifiedThermalModelDataDir,forceFlag);
```

The `forceFlag` indicates if data should be overwritten if it exists. The flag is optional and `false` by default. The default file format for writing is .xls, but if no Excel is installed on the machine, the data is written into .cvs files with semicolons as delimiter (since commas are used in the data).

4.3 Visualization

The Toolbox supports the visualization of the building. The visualization is based on the vertices specified in the `building_elements` property of the thermal model data object. The building can also be visualized when the thermal model data is inconsistent. Building elements without specified vertices are not drawn (a building element's vertex data of can currently be replaced by its area without consequences for the model generation, see Section 5). The basic command without arguments

```
B.drawBuilding;
```

will produce a 3D figure including every building element and labels for every zone and building element, as shown for instance in Figure 1. Options `'NoLabels'`, `'NoBELabels'`, `'NoZoneLabels'` can be used to control which labels should be plotted. The option `'floorplan'` fixes the view to top. It is possible to control which building elements are plotted by passing a cell array of zone and/or zone group identifiers, see Section 5. Any combinations of these options can be specified. For instance

```
B.drawBuilding({'Z0001','ZoneGrpWest'},'NoBELabels');
```

will plot all building elements of zone `'Z0001'` and all zones in the `'ZoneGrpWest'` group without building element labels.

4.4 Thermal Model Data Modification

Once thermal model data is available the Toolbox provides a generic interface to get and set the values of most of its data objects' properties. The following restrictions apply: it is not allowed to change the identifier of any particular data object. Moreover it is not allowed to change a zone's area or volume or a building element's vertices or area.

Getting a data object's value can be done with the following command

```
B.thermal_model_data.getValue(identifier,propertyName);
```

The command requires the string arguments `identifier`, which must be a valid data object identifier and `propertyName`, which is required to be a valid property for the particular data object component, e.g. for a accessing the property 'spec_heat_capacity' of a material 'M0012' `B.thermal_model_data.getValue('M0012','spec_heat_capacity')`. See Section 5 for a description of all properties.

The complementary set method works the following way

```
B.thermal_model_data.setValue(identifier,propertyname,value);
```

The third argument `value` can be either a numeric or a string according to the property the user wants to address. For an overview of the thermal model data it is possible to use

```
B.printThermalModelData;
```

which prints all the data in tabular format to the command line.

4.5 EHF Model Declaration

The following command specifies which EHF models are to be considered in the full model

```
B.declareEHFModel(EHFClassFile,EHFDataPath,EHFIdentifier);
```

The arguments are the class' m-file name, `EHFClassFile`, the path to this EHF model's data file[6], `EHFDataPath`, and an identifier string, `EHFIdentifier` (first letter alphabetic, the rest alphanumeric or '_'). For a successful declaration, the class file must be on the Matlab path, the data file must exist and the identifier must be valid and unique among the list of EHF identifiers. Table 1 lists the currently available EHF models. They are described in more detail in Section 6. Table 1 provides an overview of the current EHF model. Due to the modular structure of the EHF interface, any user can implement further EHF classes, as long as certain

[6]Every EHF model is parameterized by one Excel file.

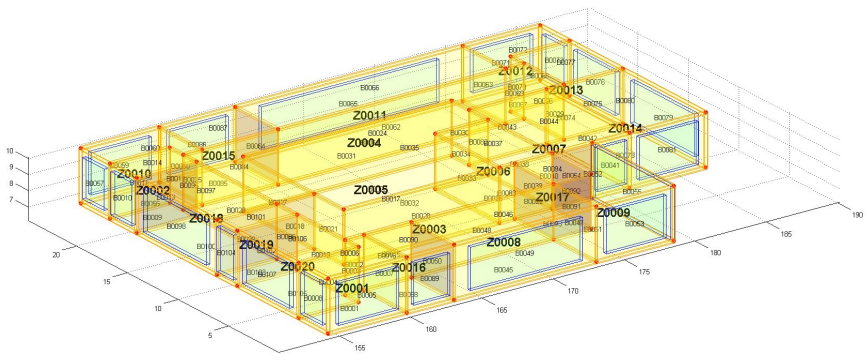


Figure 1: Visualization of a building produced by the BRCM Toolbox.

Class file name	Short description
`BuildingHull.m`	Considers convective heat transfer to all opaque facade parts and convective/conductive heat transfer through statically modeled windows to the zones as well as solar gains through the windows and onto the opaque facade parts.
`AHU.m`	Considers the effect of an air handling unit (AHU) and the heat exchange from air flow between zones induced by having air supply and air return in different zones.
`BEHeatfluxes.m`	Considers heating and cooling from building systems located within certain layers of building elements.
`Radiators.m`	Considers heat gains from radiators.
`InternalGains.m`	Considers internal gains due to occupants, lighting and appliances.

Table 1: EHF models.

conventions are respected. Details about user-specific EHF model class implementations can be found in Section 6.6.

The complementary method, which undeclares a EHF model is

```
B.undeclareEHFModel(EHFClassFile,EHFDataPath,EHFIdentifier);
```

4.6 Building Model Generation

After the thermal model data has been loaded and the EHF models have been declared, the following commands will generate the continuous-time building model, set a discretization timestep and discretize it.

```
B.generateBuildingModel;
B.building_model.setDiscretizationStep(Ts_hrs);
B.building_model.discretize;
```

In the process of `B.generateBuildingModel` first the thermal model (4) is generated (including the data consistency checks), then the individual EHF models are created and compiled into (5) and finally both parts are combined. The result is a full `building_model` object within `B`, for instance

```
BuildingModel with properties:

          identifiers : [1x1 Identifier]
     thermal_submodel : [1x1 ThermalModel]
        EHF_submodels : {[1x1 BuildingHull] [1x1 AHU]}
continuous_time_model : [1x1 struct]
  discrete_time_model : [1x1 struct]
              is_dirty : 0
                Ts_hrs : 0.2500
```

The property `identifiers` is a container for all identifiers of the states x, heat fluxes q, inputs u, disturbances v and constraints. The Toolbox generates for each state x of the thermal model an associated heat flux variable q. The generation of the input, disturbance and constraint identifiers depend on the included EHF models and their specifications, see Section 6. The command `B.building_model.printIdentifiers` will print all available identifiers to the command line. The property `thermal_submodel` contains the state-space matrices as in (4) and the property `EHF_submodels` contains a cell array of EHF model objects, themselves containing the state-space matrices of those submodels, see Section 6.6 for more detail on the implementation of EHF models. Similarly, as in the case of thermal model data, the flag `is_dirty` is used to indicate if the current state-space representation of full building model does not correspond to the current state of the data. Finally, the property `Ts_hrs` is the discretization time step in hours.

The generation of the thermal model can also be individually issued by using

```
B.generateThermalModel;
```

However, in contrast to the thermal model generation, the user cannot explicitly trigger the generation of an external heat flux model. The EHF model generation is performed implicitly with the generation of the full model. This is mostly due to the fact that a thermal model has to be present in order that the EHF models can be generated.

4.7 Accessing Model Matrices and Generation of Costs and Constraints

The resulting model matrices as in (3) can directly be accessed using the `B.building_model.discrete_time_model` object. Similarly, for the continuous-time model `B.building_model.continuous_time_model` can be accessed.

The Toolbox provides methods for generating the building's model constraints and cost matrices and vectors as in (2) and (1) as a function of potentially time-varying parameters. The constraints matrices F_x, F_u, F_v, g and the cost vector c_u can be obtained as follows

```
[Fx,Fu,Fv,g] = B.building_model.getConstraintsMatrices(constraintsParameters);
cu = B.building_model.getCostVector(costParameters);
```

The arguments `constraintsParameters` and `costParameters` are structures with fields according to every EHF model's identifier. These fields themselves are structures with a set of parameters according to the individual EHF models, for instance for the ventilation mass flow

rate constraints (`AHU` being the identifier of the air handling unit EHF model)

```
constraintsParameters.AHU.m_dot_min = 0;
constraintsParameters.AHU.m_dot_min = 1;
```

See Section 6 for the required cost and constraints parameters for every EHF model.

4.8 Simulation

The class `SimulationExperiment` provides a simulation environment. An instantiation requires a `Building` object that at least contains a thermal model and the sampling time (this will be also the simulation time step). Once the `SimulationExperiment` object is instantiated, its `Building` object cannot be manipulated anymore.

```
SimExp = SimulationExperiment(B);
```

The following commands print and access the x,q,u,v identifiers and get the discretization/simulation time step

```
SimExp.printIdentifiers();
identifiers = SimExp.getIdentifiers();
Ts_hrs = SimExp.getSamplingTime();
```

Before a simulation, the number of simulation time steps and the initial condition must be set as follows

```
SimExp.setNumberOfSimulationTimeSteps(n_timeSteps);
SimExp.setInitialState(x0);
```

Either the full building model of just the thermal model can be simulated.

```
SimExp.simulateThermalModel(simulationMode,arguments);
SimExp.simulateBuildingModel(simulationMode,arguments);
```

Two different modes defined by `simulationMode` are supported, each requiring different arguments. The modes are described in Table 2.

simMode	description
'inputTrajectory'	In this mode the user needs to provide the input trajectories for the specific model as ***arguments***, i.e., either $Q \in \mathbb{R}^{n_q x n_t}$ or $U \in \mathbb{R}^{n_u \text{ x } n_t}$ and $V \in \mathbb{R}^{n_v \text{ x } n_t}$ with n_t, n_t, n_t, n_t being the number of time steps, heat fluxes, control inputs and disturbances, respectively.
'function_handle'	In this mode the user has to provide a handle to a function as ***arguments***, having either the form `qk = f(xk,tk,identifiers)` in the case of thermal model simulation or `[uk,vk] = f(xk,tk,identifiers)` in the case of full model simulation.

Table 2: Description of the simulation modes of a simulation object.

5 Thermal Model Details

The thermal model describes the temperature dynamics of the zones and building elements of a building as a function of external heat fluxes, see (4). This section describes the modeling principles and the input data structure.

5.1 Modeling Principles

The core model generation uses the algorithms described in [1] which in turn are mainly based on the one-zone model from [2]. In [1] we individually parameterized and connected multiple of these one-zone models to build a full building model and validated it against a complex building model in EnergyPlus. The approach makes the following main assumptions for the thermal model of the building: i) the air volume of each zone has uniform temperature; ii) temperatures within building elements vary only along the direction of the surface normal; iii) there is no conductive heat transfer between different building elements; iv) the temperature within a layer of a building element is constant; v) all model parameters are constant over time; vi) long-wave (i.e. thermal) radiation is considered in a combined convective heat transfer coefficient (this is likely to be changed in the future).

Assumption i) and iv) directly lead to modeling the temperature (or thermal energy) of every zone's air mass and every building element's layer using one state. Assumption ii) leads to modeling the (conductive) heat transfer within a building element linearly proportional to the temperature difference between neighboring layers. Assumptions iii) and vi) imply that direct heat transfer among different building elements is neglected. Finally, based on assumptions i) and ii), heat transfer between zone air and surface building element layers is modeled to be proportional to the respective temperature difference.

Several boundary conditions are supported: *adiabatic*, *ambient* (convective heat transfer and solar radiation), *ground* or *prescribed temperature* (conductive heat transfer related to a disturbance input that models the ground or some other temperature). Since (apart from *adiabatic*) these boundary conditions define heat exchanges with the environment, they are implemented in a EHF model, see Section 6.3. Additionally, it is possible to connect the first and the last layer of a building element to the same zone air state in order to approximate a situation in which many identical zones are adjacent to each other.

5.2 Input Data

The thermal model input data consists of seven Excel (.xls, .xlsx) or semicolon-separated CSV files. When loading the thermal model data (see Section 4.2) a path to a directory must be specified. All seven input data files must be in this directory and be named according to the conventions in Table 3.

component	file name
zones	`zones.(xls/xlsx/csv)`
building elements	`buildingelements.(xls/xlsx/csv)`
constructions	`constructions.(xls/xlsx/csv)`
materials	`materials.(xls/xlsx/csv)`
windows	`windows.(xls/xlsx/csv)`
no-mass constructions	`nomassconstructions.(xls/xlsx/csv)`
parameters	`parameters.(xls/xlsx/csv)`

Table 3: BRCM toolbox thermal model data files name convention.

Figure 2 depicts the data model and shows the relation among the components.

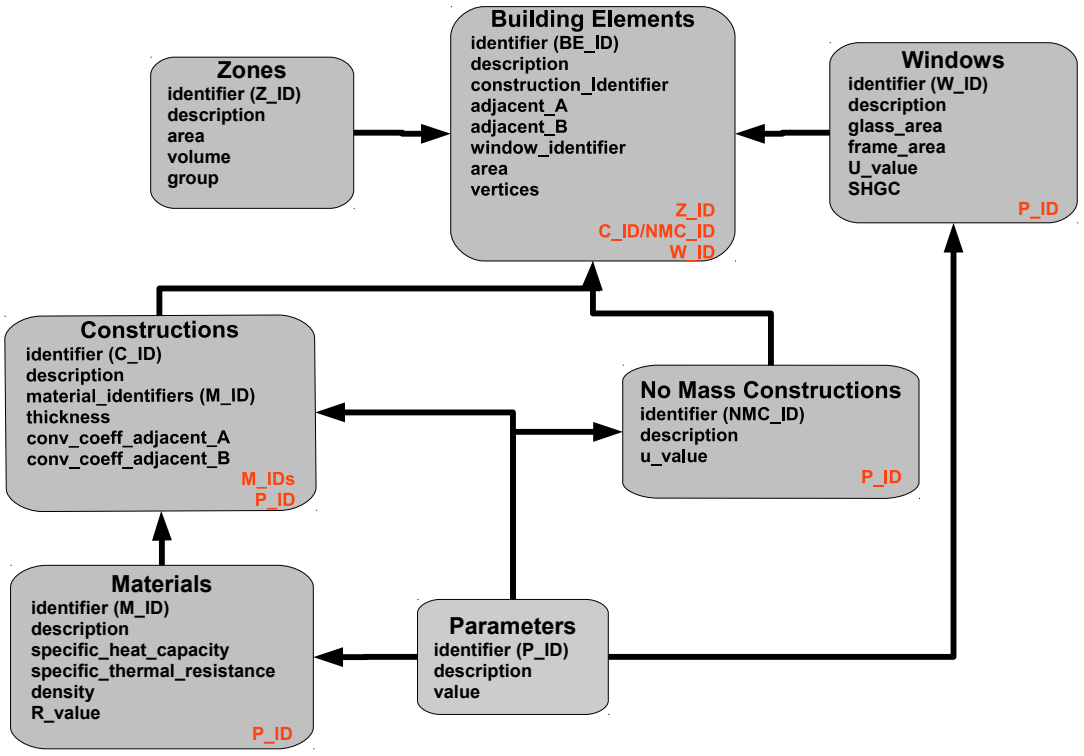


Figure 2: The toolbox data model of a building. An arrow between two elements means that the "arrowhead element" references elements of the "arrow tail element".

Even though three out of seven thermal data component specifications are optional (windows, no-mass constructions, parameters), their associated files must be available for the toolbox depending on the data load command, but their content can be empty except for its header.

In the following we describe the structures and conventions of the thermal model input data files of the specific building components separately. Every file consists of one header row and an arbitrary number of data rows. The header identifiers must appear next to each other in the same order as listed in Table 4 in the header row of the data sheet.

5.2.1 Zones

In the `zones.(xls/xlsx/csv)` data sheet the building's zones are specified. The required properties' header identifier, the valid format and description are shown in Table 4.

Header	Format	Description
identifier	'*Zdddd*'. Must be unique among all zone identifiers.	Uniquely identifies a zone.
description	Any string.	Used for labelling in legends.
area	Positive numerical value.	Floor area of the zone. [m^2].
volume	Positive numerical value.	Volume of the zone. [m^3].
group	Empty or comma separated list of identifiers with the usual conventions (see Section 4.5).	Can be used to address groups of zones. A zone can be part of multiple groups.

Table 4: Property header identifiers, format and description of the `zones.(xls/xlsx/csv)` file.

The header identifiers must appear next to each other in the same order as listed in Table 4 in the header row of the data sheet. Table 5 shows as an example the first few lines of the `zones.(xls/xlsx/csv)` file.

identifier	description	area	volume	group
Z0001	office_west_1	16.02	48.06	offices_west
Z0002	office_west_2	20.05	60.15	offices_west
Z0003	office_east_1	18.04	54.12	offices_east
...	...	...	...	...

Table 5: Exemplary first few lines of the `zones.(xls/xlsx/csv)` file. The header must match exactly and no empty header fields are allowed in horizontal direction.

5.2.2 Building Elements

The `buildingelements.(xls/xlsx/csv)` specifies the building's building elements. The required properties' header identifier, the valid format and description are shown in Table 4. For brevity an examples analogous to Table 5 are omitted for the rest of the section.

Header	Format	Description
identifier	'B*dddd*'. Must be unique among all building element identifiers.	Uniquely identifies a building element.
description	Any string.	Description.
construction_identifier	Construction or no-mass construction identifier, 'C*dddd*' or 'NMC*dddd*'.	Defines the construction type of the building element. A no-mass construction identifier results in a building element that is modeled as a stateless thermal resistance, e.g. to model openings.
adjacent_A	Zone identifier or ambient/ground/adiabatic boundary condition identifier ('AMB', 'GND', 'ADB'), see Section 5.1.	Denotes the building elements boundary condition on side A.
adjacent_B	Zone identifier or ambient/ground/adiabatic boundary condition identifier ('AMB', 'GND', 'ADB'), see Section 5.1.	Denotes the building elements boundary condition on side B.
window_identifier	Empty or window identifier 'W*dddd*'.	Defines the building element's window type. No window is modeled if left empty. Only building elements with 'AMB' value in **adjacent_A** or **adjacent_B** are currently allowed to have windows.
area	Positive numerical value. Can be empty if vertices are specified.	Building element's area. [m^2].
vertices	Comma separated list of curly-bracketed 3D world coordinates, e.g. "{1,2,0},{3,2,0},...." Can be empty if **area** is specified. At least three vertices are required. They must be entered clockwise or counter-clockwise and lie all in a plane.	Corner coordinates of the building element. Used to visualize the building element and compute the building element's area if **area** is emtpy. [m].

Table 6: Property header identifiers, format and description of the `buildingelements.(xls/xlsx/csv)` file.

Note that there is no special convention which side of the building element is considered as **adjacent_A** or **adjacent_B**. The distinction is used to determine the order of the material layers as defined in a construction. In our convention, the first material appearing in a construction's material list (see Table 7) is assumed to be on the **adjacent_A** side while the last is on **adjacent_B** side.

5.2.3 Constructions

The `constructions.(xls/xlsx/csv)` file specifies the building elements' construction. The required properties' header identifier, the valid format and description are shown in Table 7.

Header	Format	Description
identifier	C*dddd*'. Must be unique among all construction identifiers.	Uniquely identifies a construction.
description	Any string.	Description.
material_identifiers	Comma separated list of material identifiers 'M*dddd*'.	Defines the materials of the construction's layers. Convention: First element is on the **adjacent_A** side of the building element, the last on **adjacent_B** side, see Table 6.
thickness	Comma separated list of parametric expressions or non-negative numerical values. The number of values must coincide with the number of material identifiers.	Specifies the thickness of the according layer. Same convention as with the **material_identifiers**. If 0, the layer is modeled as a stateless thermal resistance and the according material must have a specified **R_value**, see Table 8. [m].
conv_coeff_adjacent_A	Parametric expression or positive numerical value.	Convective heat transfer coefficient on the **adjacent_A** side of the building element. [W/m^{2}K].
conv_coeff_adjacent_B	Parametric expression or positive numerical value.	Same as **conv_coeff_adjacent_A** for side B.

Table 7: Property header identifiers, format and description of the `constructions.(xls/xlsx/csv)` file.

5.2.4 Materials

The `materials.(xls/xlsx/csv)` file specifies the constructions' materials. The required properties' header identifier, the valid format and description are shown in Table 8.

Header	Format	Description
identifier	M*dddd*'. Must be unique among all material identifiers.	Uniquely identifies a material.
description	Any string.	Description.
specific_heat_capacity	Parametric expression or numeric value greater zero. [J/kgK]. Must be empty if R-value is specified.	Specific heat capacity of the material.
specific_thermal_resistance	Parametric expression or numeric value greater zero. [mK/W]. Must be empty if R-value is specified.	Specific thermal resistance of the material.
density	Parametric expression or numeric value greater zero. [kg/m^3]. Must be empty if R-value is specified.	Density of the material.
R_value	Parametric expression or numeric value greater zero. [m^2K/W]. Must be empty if specific_heat_capacity, specific_thermal_resistance and density are specified.	Total thermal resistance of the material. Used if the layer having this material should be modeled as pure thermal resistance without a state.

Table 8: Property header identifiers, format and description of the `materials.(xls/xlsx/csv)` file.

5.2.5 Windows

The `windows.(xls/xlsx/csv)` file specifies the building elements' window. The required properties' header identifier, the valid format and description are shown in Table 9.

Header	Format	Description
identifier	W*dddd*'. Must be unique among all window identifiers.	Uniquely identifies a construction.
description	Any string.	Description.
glass_area	Parametric expression or numeric value greater or equal zero. [mˆ2].	Glass area of the window.
frame_area	Parametric expression or numeric value greater or equal zero. [mˆ2].	Frame area (not including glass area) of the window.
U_value	Parametric expression or numeric value greater zero. [W/mˆ2K].	Combined thermal conductivity of the window and frame.
SHGC	Parametric expression or numeric value in [0,1]. [-].	Fraction of the total incident solar radiation that gets converted to primary (transmitted radiation) and secondary (reemitted thermal radiation) heat-gains.

Table 9: Property header identifiers, format and description of the `windows.(xls/xlsx/csv)` file.

5.2.6 No-Mass Constructions

The `nomassconstructions.(xls/xlsx/csv)` file specifies the building elements' no-mass constructions. The required properties' header identifier, the valid format and description are shown in Table 10.

Header	Format	Description
identifier	NMC*dddd*'. Must be unique among all no-mass construction identifiers.	Uniquely identifies a construction.
description	Any string.	Description.
U_value [W/mˆ2K]	Parametric expression or numeric value greater zero, units [W/mˆ2K].	Thermal conductivity of the no-mass construction. Thermal resistance due to convective effects will be added to the overall resistance.

Table 10: Property header identifiers, format and description of the `nomassconstruction.(xls/xlsx/csv)` file.

5.2.7 Parameters

The `parameters.(xls/xlsx/csv)` file specifies the building's parameters. The required properties' header identifier, the valid format and description are shown in Table 11.

Header	Format	Description
identifier	String identifier with the usual notational conventions (see Section 4.5). Must be unique among all construction identifiers.	Uniquely identifies a parameter.
description	Any string.	Description.
value	Numerical value.	

Table 11: Property header identifiers, format and description of the `parameters.(xls/xlsx/csv)` file.

6 EHF Model Details

In this section the currently implemented EHF models are described in detail. Moreover, the interface for including additional custom EHF models is outlined in Section 6.6.

6.1 Internal Gains

This EHF model considers internal gains due to occupants, lighting and appliances. Currently those heat gains are modeled to be purely convective (i.e. their influence is directly added to the zone states). The internal gains are considered as disturbances in [W/m^2].
This EHF model can be included only once to avoid problems with identical identifiers. This is not restrictive since any desired internal gains structure can be specified by one copy.

6.1.1 Input Data

Table 12 shows the structure of the file specifying the internal gains EHF model.

Table 12: File structure of the internal gains EHF model.

zone_identifier[7]	**disturbance_identifier**[8]
Z0001	IGOffices
Z0002	IGCoreFacilities
Z0004	IGOffices
...	...

6.1.2 Model

Table 13 shows the nomenclature used in this section.

Table 13: Nomenclature.

Disturbances	
$v_{IG,<disturbance\_identifier>}$	Internal gains due to equipment and occupants corresponding to $<$*disturbance_identifier*$>$. [W/m^2].

[7] Must be an existing zone identifier. Not all zones need to be specified. The same zone can be influenced by multiple distinct $<$*disturbance_identifier*$>$.

[8] An identifier string subject to the usual notational convention, see the footnote in Section 4.1. For every identifier an according disturbance is created. Same identifiers result in the same disturbance to all of the corresponding zones.

Model parameters	
$a_{zone,i}$	Area of zone i. [m^2]. Taken from thermal model data.
Other nomenclature	
$\mathcal{Z}_{IG}$	Set of all zones for which internal gains have been specified.
$q_{zone,i}$	Heat flux to zone i. [W].

As a first step in the model generation, the identifiers are set up.

Step 1: Generation of identifiers

disturbance identifiers

v_IG_<*disturbance_identifier*> (for every distinct <*disturbance_identifier*>specified in the data sheet (Table 12))

control input identifiers

constraint identifiers

In the second step the matrices of this EHF submodel are set up.

Step 2: Internal gains heat flux to the zones.

for $i \in \mathcal{Z}_{IG}$

Find the <*disturbance_identifier*> corresponding to zone i

$q_{zone,i} \mathrel{+}= a_{zone,i} \cdot v_{IG,<disturbance\_identifier>}$

end

6.1.3 Constraints

This EHFM has no inputs and hence no constraints.

6.1.4 Costs

This EHFM has no inputs and hence no costs.

6.2 Radiators

This EHF model considers heat gains from radiators. Currently those heat gains are added directly to the zone states. The radiator heat gains are considered as control inputs in [W/m^2]. This EHF model can be included only once to avoid problems with identical identifiers. This is not restrictive since any desired radiator influence can be specified in one copy.

6.2.1 Input Data

Table 14 shows the structure of the file specifying the radiators EHF model.

Table 14: File structure for the radiator EHF model.

zone_identifier[9]	**control_identifier**[10]
Z0001	RadOffices
Z0002	RadOffices
Z0004	RadCornerZone
...	...

6.2.2 Model

Table 15 shows the nomenclature used in this section.

Table 15: Nomenclature.

Control inputs	
$u_{Rad,<control\_identifier>}$	Heating power of the radiator system corresponding to $<control\_identifier>$. [W/m^2].
Model parameters	
$a_{zone,i}$	Area of zone i. [m^2]. Taken from thermal model data.
Other nomenclature	
$\mathcal{Z}_{Rad}$	Set of all zones for which radiators have been specified.
$q_{zone,i}$	Heat flux to zone i. [W].
T_s	Sampling time of the model. [s].
J_{Rad}	Total costs related to the use of radiators. [-].
$<Rad\_identifier>$	Identifier of this EHF model, specified in the EHF model declaration.

As a first step in the model generation, the identifiers are set up.

[9] Must be an existing zone identifier. Not all zones need to be specified.

[10] An identifier string subject to the usual notational restrictions, see the footnote in Section 4.1. For every identifier an according control input is created. Same identifiers yield the same control input to the corresponding zones.

Step 1: Generation of identifiers

disturbance identifiers
control input identifiers
u_Rad_<*control_identifier*> (for every distinct <*control_identifier*>specified in the data sheet (Table 14))
constraint identifiers
Q_Rad_<*control_identifier*>_min,max (for every distinct <*control_identifier*>specified in the data sheet (Table 14))

In the second step the matrices of this EHF model are set up.

Step 2: Radiator heat flux to the zones

Conductive heat transfer from ground.
for $i \in \mathcal{Z}_{rad}$
Find the <*control_identifier*> corresponding to zone i
$q_{zone,i} += a_{zone,i} \cdot u_{Rad,<control\_identifier>}$
end

6.2.3 Constraints

As described in Section 4.7, the `constraintsParameters` structure must be appropriately set up to generate the constraints for every EHF model. For the radiators EHF model, the fields `constraintsParameters`.<*Rad_identifier*>.<*parameter_name*> must be set for every parameter with name <*parameter_name*> shown in Table 16.

Table 16: Parameters related to the constraint generation which have to be specified whenever constraints are to be generated.

Parameter name	Description
Q_Rad_<*control_identifier*>_max	Maximum heating power for every distinct <*control_identifier*> specified in the input data file (Table 14). [W/m^2].
Q_Rad_<*control_identifier*>_min	Minimum heating power for every distinct <*control_identifier*> specified in the input data file (Table 14). [W/m^2].

Constraint generation.

for every <*control_identifier*>
Q_Rad_<*control_identifier*>_min $\leq$ $u_{Rad,<control\_identifier>}$ $\leq$ Q_Rad_<*control_identifier*>_max
end

6.2.4 Costs

As described in Section 4.7, the `costParameters` structure must be appropriately set up to generate the cost vector. For the radiators EHF model, the fields `costParameters`.<*Rad_identifier*>.<*parameter_name*> must be set for every parameter with name <*parameter_name*> shown in Table 17.

Table 17: Parameters related to the cost generation which have to be specified whenever the cost is to be generated.

Parameter name	Description
costPerJouleHeated	Costs per Joule heating power [1/J].

Cost generation.

for every <*control_identifier*>
$J_{Rad}+=$ costPerJouleHeated $\cdot$ $u_{Rad,<control\_identifier>}$ $\cdot$ T_s
end

6.3 Building Hull

This EHF model considers convective heat transfer to all opaque facade parts and convective/conductive heat transfer through statically modeled windows to the zones (modeled by a U-value). Global solar radiation onto the facades is modeled as disturbance inputs in [W/m^2]. Moreover, it considers air infiltration through the building hull. Note that this implies that those disturbance inputs have to be externally calculated from global solar radiation on a horizontal surface (which is typically what is available). The fact that this calculation is not internalized in the Toolbox is a major drawback. The next version of the Toolbox is planned to incorporate those algorithms.
Solar heat gains to the opaque facade parts are modeled by an absorption coefficient scaling the solar radiation. Thermal radiation exchange of the opaque facade parts is considered in the convective coefficients. Solar heat gains through windows are considered by scaling the incident solar radiation with a constant solar heat gain coefficient (SHGC) and the current blinds

position (if present, modeled as a controllable gain between 0 and 1). The primary solar heat gains (the transmitted radiative power) are distributed among the innermost building element layers proportionally to their surface area while the secondary heat gains (due to heating up of the windows and frames) are added to the zone air. Infiltration is modeled by a fixed air change rate to the zones.
Note that at most one window per building element is supported and currently only building elements with an *ambient* boundary condition can contain a window.
This EHF model can be included only once to avoid problems with identical identifiers. This is not restrictive since any desired building hull heat exchange can be specified by one copy.

6.3.1 Input Data

Table 18 shows the file structure of the building hull model input data. It contains two parts identified by facade_solar_group and window_solar_group, respectively.

Table 18: File structure for the building hull EHF model.

facade_solar_group	**buildingelement_identifier**[11]	**disturbance_identifier**[12]	**absorptance**[13]	
	B0005	Facade_South	0.5	
	B0120	Facade_South	0.5	
	B0113	Facade_North	0.6	
	...	...	...	
window_solar_group	**buildingelement_identifier**[14]	**disturbance_identifier**[15]	**control_identifier**[16]	**secondary_gains_fraction** [17]
	B0005	Facade_South	Facade_South_1	0.1
	B0068	Facade_North		0.15
	B0113	Facade_North	Facade_North	0.1
	...	...	...	...
infiltration_specification	**zone_identifier**[18]	**airchangerate**[19]		
	Z0001	0.5		
	Z0003	0.5		
	...	...	...	

6.3.2 Model

Table 19 shows the nomenclature used in this section.

Table 19: Parameter table of the air handling unit EHF model.

Control inputs	
$u_{blinds,<control\_identifier>}$	Blinds position control input corresponding to $<$*control_identifier*$>$. [-].
Disturbances	
v_{amb}	Ambient temperature. [°C].
v_{gnd}	Ground temperature. [°C].
$v_{solGlobFac,<disturbance\_identifier>}$	Global solar radiation disturbance input corresponding to $<$*disturbance_identifier*$>$. [W/m^2].
Model parameters	
$\rho_{BEo,j}$	Specific thermal resistance of the j-th building element's outermost layer. [mK/W]. From thermal model data.
$d_{BEo,j}$	Thickness of the j-th building element's outermost layer. [m]. From thermal model data.
$f_{secHG,j}$	secondary heat gain fraction of the j-th building element's window. [-]. From EHF model input data.
$f_{SHGC,j}$	solar heat gain coefficient of the j-th building element's window. [-]. From thermal model data.
$a_{BE,j}$	Net area of the j-th building element (i.e. discounting window and frame area). [m^2]. From thermal model data.
$\alpha_{BEo,j}$	Convective coefficient between the j-th building element's outermost layer and ambient air. [W/m^{2}K]. From thermal model data.
$a_{win,j}$	(Glass) area of the j-th building element's window. [m^2]. From thermal model data.

[11] Must be an identifier of an existing building element that has an *ambient* boundary condition. No identifier is allowed to be repeated. The union of all building element identifiers must be identical to the set of all building elements that have an *ambient* boundary condition.

[12] An identifier string subject to the usual notational restrictions. For every identifier an according solar radiation disturbance is created. Same identifiers yield the same disturbance on the corresponding building elements/windows.

[13] Numerical value in [0,1]. Models the fraction of the solar radiation absorbed by the outermost layer of the corresponding building element.

[14] Must be an identifier of an existing building element that has a window. No identifier is allowed to be repeated. The union of all building element identifiers must be identical to the set of all building elements that have a window.

[15] An identifier string subject to the usual notational restrictions. For every identifier an according solar radiation disturbance is created. Same identifiers yield the same disturbance on the corresponding building elements/windows.

[16] An identifier string subject to the usual notational restrictions. For every identifier an according blinds control input is created. Same identifiers yield the same control input for the corresponding windows.

[17] Numerical value in $[0, 1]$. Modeling how much of the transmitted solar radiation is added to the zone node as secondary heat gains due to heating up of windows and frames.

[18] Must be an identifier of an existing zone. No identifier is allowed to be repeated.

[19] Air change rate per hour. The volumetric infiltration flow in $[m^3/s]$ is calculated as the product of the air change rate and the zone's volume divided by 3600.

$a_{frame,j}$	Area of the j-th building element's window frame. [m^2]. From thermal model data.
$\gamma_{BEo,j}$	Solar absorptance of the j-th building element's outermost layer. [-]. From EHF model input data.
$a_{BE,tot,i}$	Total area of all building elements of zone i (not including no-mass constructions). [m^2]. From thermal model data.
acr_i	Air change rate (zone volumes per hour) of zone i. [1/h]. From EHF model input data.
$U_{win,j}$	Combined frame and window heat transfer coefficient. [W/m^2K]. From thermal model data.
$V_{zone,i}$	Volume of zone i. [m^3]. From thermal model data.
C_{air}	Heat capacity of air (assumed independent of the temperature). [J/kgK]. From standard gas properties.
ρ_{air}	Heat capacity of air (assumed independent of the temperature). [kg/m^3]. From standard gas properties.
Other nomenclature	
$\mathcal{B}_{amb}$	Set of all building elements that are connected to ambient temperature
$\mathcal{B}_{win}$	Set of all building elements that are connected to ambient temperature and have a window
$\mathcal{B}_{gnd}$	Set of all building elements connected to the ground
$\mathcal{B}_{zone,i}$	Set of all building elements of zone i
$\mathcal{Z}_{inf}$	Set of all zones for which infiltration has been specified.
$x_{BEo,j}$	State describing the temperature of the j-th building element's outmost layer. [°C].
$q_{zone,i}$	Heat flux to zone i. [W].
$x_{zone,i}$	Temperature of zone i. [°C].
$q_{BEo,j}$	Heat flux to the j-th building element's outermost layer. [W].
$q_{BEi,j}$	Heat flux to the j-th building element's innermost layer. [W].
$<$*BuildingHull_identifier*$>$	Identifier of this EHF model, specified in the EHF model declaration.

As a first step in the model generation, the identifiers are set up. Then, the convective/conductive heat fluxes to outside walls/roofs, ground floors and through windows are considered. Finally, the solar radiation heat fluxes are modeled.

Step 1: Generation of identifiers

disturbance identifiers

v_amb	
v_gnd	(if any building element is modeled to be in contact with the ground)
v_solGlobFac_<*disturbance_identifier*>	(for every distinct <*disturbance_identifier*> specified in the data sheet (Table 18))

control input identifiers

u_blinds_<*control_identifier*>	(for every distinct <*control_identifier*> specified in the data sheet (Table 18))

constraint identifiers

BPos_<*control_identifier*>_{min,max}	(for every distinct <*control_identifier*> specified in the data sheet (Table 18))

Step 2: Convective/conductive heat flux

Conductive heat transfer from ground.

for $j \in \mathcal{B}_{gnd}$

compute thermal resistance $R \coloneqq \rho_{BEo,j} \cdot \frac{d_{BEo,j}}{2}$

$q_{BEo,j} \mathrel{+}= \frac{a_{BE,j}}{R}(v_{gnd} - x_{BEo,j})$

end

Convective/conductive heat transfer to opaque facade parts

for $j \in \mathcal{B}_{amb}$

compute total thermal resistance $R \coloneqq \frac{1}{\alpha_{BEo,j}} + \rho_{BEo,j} \cdot \frac{d_{BEo,j}}{2}$

$q_{BEo,j} \mathrel{+}= \frac{a_{BE,j}}{R}(v_{amb} - x_{BEo,j})$

end

Convective/conductive heat transfer through windows to the zones

for $j \in \mathcal{B}_{win}$

find zone i associated with building element j

$q_{zone,i} \mathrel{+}= U_{win,j} \cdot (a_{win,j} + a_{frame,j}) \cdot \left(v_{amb} - x_{zone,i}\right)$

end

Step 3: Heat flux due to solar radiation

for $j \in \mathcal{B}_{amb}$
$q_{BEo,j} \mathrel{+}= a_{BE,j} \cdot \gamma_{BEo,j} \cdot v_{solGlobFac,j}$
end

for $j \in \mathcal{B}_{win}$
find zone i associated with building element j
get the set $\mathcal{B}_{zone,i}$ of all building elements in contact with zone i
compute total area of building elements (not including no-mass constructions) in zone i: $a_{BE,tot,i} \coloneqq \sum_{k \in \mathcal{B}_{zone,i}} a_k$
if zone i has a <*control_identifier*>
$q_{zone,i} \mathrel{+}= f_{secHG,j} \cdot a_{win,j} \cdot f_{SHGC,j} \cdot v_{solGlobFac,<disturbance\_identifier>} \cdot u_{blinds,<control\_identifier>}$
for $k \in \mathcal{B}_{zone,i}$
$q_{BEi,k} \mathrel{+}= \frac{a_k}{a_{BE,tot,i}}(1 - f_{secHG,j}) \cdot a_{win,j} \cdot f_{SHGC,j} \cdot v_{solGlobFac,<disturbance\_identifier>} \cdot u_{blinds,<control\_identifier>}$
end
else
$q_{zone,i} \mathrel{+}= f_{secHG,j} \cdot a_{win,j} \cdot f_{SHGC,j} \cdot v_{solGlobFac,<disturbance\_identifier>}$
for $k \in \mathcal{B}_{zone,i}$
$q_{BEi,k} \mathrel{+}= \frac{a_k}{a_{BE,tot,i}}(1 - f_{secHG,j}) \cdot a_{win,j} \cdot f_{SHGC,j} \cdot v_{solGlobFac,<disturbance\_identifier>}$
end
end
end

Step 4: Infiltration

for $i \in \mathcal{Z}_{inf}$
$q_{zone,i} \mathrel{+}= \rho_{air} \cdot C_{air} \cdot \frac{\text{acr}_i}{3600} \cdot V_{zone,i} \cdot (v_{amb} - x_{zone,i})$
end

6.3.3 Constraints

As described in Section 4.7, the `constraintsParameters` structure must be appropriately set up to generate the constraints for every EHF model. For the building hull EHF model, the fields `constraintsParameters`.<*BuildingHull_identifier*>.<*parameter_name*> must be set for every parameter with name <*parameter_name*> shown in Table 20.

Table 20: Parameters related to the constraint generation which have to be specified whenever constraints are to be generated.

Parameter name	Description
BPos_<*control_identifier*>_max	Maximum blind position for every distinct <*control_identifier*>specified in the input data file (Table 18). Must lie in [0,1].
BPos_<*control_identifier*>_min	Minimum blind position for every distinct <*control_identifier*>specified in the input data file (Table 18). Must lie in [0,1].

The pseudo-code below shows the constraint generation.

Constraint generation.

for every <*control_identifier*>
BPos_<*control_identifier*>_min $\leq u_{blinds,<control\_identifier>} \leq$ BPos_<*control_identifier*>_max
end

6.3.4 Costs

No costs are associated with the movement of blinds.

6.4 Building Element Heat Fluxes

This EHF model considers heating and cooling from building systems located within certain layers of building elements. The heating/cooling systems are considered as control inputs in [W/m^2] and act directly on the states representing the according layers.
This EHF model can be included only once to avoid problems with identical identifiers. This is not restrictive since any desired heating/cooling influence on building elements can be specified by one copy.

6.4.1 Input data

Table 21 shows the file structure for the building element heat flux model input data.

Table 21: File structure for the building element heat flux model.

buildingelement_identifier[20]	**layer_number**[21]	**control_identifier**[22]	**heating_cooling_selection**[23]
B0001	2	cTABS	c
B0006	2	cTABS	c
B0047	3	hTABS	h
...	...	...	

6.4.2 Model

Table 22 shows the nomenclature used in this section.

Table 22: Nomenclature.

Control inputs	
$u_{BEH,<control\_identifier>}$	Heating/cooling power corresponding to <*control_identifier*>. [W/m^2].
Model parameters	
$a_{BE,j}$	Area of building element j. [m^2]. From thermal model data.
Other nomenclature	
$\mathcal{B}_{BEH,<control\_identifier>}$	Set of all building elements that are influenced by <*control_identifier*>.
$q_{BE,j}^{L,l}$	Heat flux to layer l of building element j. [W].
$J_{<BEH\_identifier>}$	Total costs related to the use of heating/cooling systems located within building elements. [-].

Step 1: Generation of identifiers

disturbance identifiers	
control input identifiers	
u_BEH_<*control_identifier*>_{heat,cool}	(for every distinct <*control_identifier*> specified in the data sheet (Table 21).{heat,cool} is selected according to the corresponding heating_cooling_selection value.)
constraint identifiers	
Q_BEH_<*control_identifier*>_{heat,cool}_min Q_BEH_<*control_identifier*>_{heat,cool}_max	(for every distinct <*control_identifier*> specified in the data sheet (Table 21). {heat,cool} is selected according to the corresponding heating_cooling_selection value.)

[20] Must be an identifier of an existing building element.
[21] Positive integer. Number of the layer to which the heating/cooling is applied.
[22] An identifier string subject to the usual notational restrictions. For every identifier an according heating/-cooling control input is created. Same identifiers yield the same control input for the corresponding building element layers.
[23] Either 'h' or 'c'. Determines whether the input is cooling or heating. Must be the same for a particular <*control_identifier*>.

Table 23: Parameters related to the constraint generation which have to be specified whenever constraints are to be generated.

Parameter name	Description
Q_BEH_<*control_identifier*>_max	Maximum heating power for every distinct <*control_identifier*>specified in the input data file (Table 21). [W/m^2].
Q_BEH_<*control_identifier*>_min	Minimum heating power for every distinct <*control_identifier*>specified in the input data file (Table 21). [W/m^2].

Step 2: Heat flux to building elements

for all <*control_identifier*>
Find the set $\mathcal{B}_{BEH,<control\_identifier>}$ of all building elements influenced by <*control_identifier*>.
for $j \in \mathcal{B}_{BEH,<control\_identifier>}$
find layer l of building element j on which the heat flux is acting
if <*control_identifier*> is a heating input
$q_{BE,j}^{L,l} += a_{BE,j} \cdot u_{BEH,<control\_identifier>}$
else
$q_{BE,j}^{L,l} -= a_{BE,j} \cdot u_{BEH,<control\_identifier>}$
end
end
end

6.4.3 Constraints

As described in Section 4.7, the `constraintsParameters` structure must be appropriately set up to generate the constraints of all the EHF models. For the building element heat flux, the fields
`constraintsParameters`.<*BEH_identifier*>.<*parameter_name*> must be set for every parameter shown in Table 23.

Constraint generation.

for every <*control_identifier*>
Q_BEH_<*control_identifier*>_min $\leq$ $u_{BEH,<control\_identifier>}$ $\leq$ Q_BEH_<*control_identifier*>_max
end

6.4.4 Costs

As described in Section 4.7, the `costParameters` structure must be appropriately set up to generate the cost vector. For the building element heat fluxes EHF model, the fields `costParameters.`<*BEH_identifier*>`.`<*parameter_name*> must be set for every parameter with name <*parameter_name*> shown in Table 24.

Table 24: Parameters related to the cost generation which have to be specified whenever the cost is to be generated.

Parameter name	Description
costPerJouleHeated	Costs per Joule heating power [1/J].
costPerJouleCooled	Costs per Joule cooling power [1/J].

Cost generation.

for every <*control_identifier*>
  if <*control_identifier*> is a heating input
    $J_{<BEH\_identifier>} += \text{costPerJouleHeated} \cdot u_{BEH,<control\_identifier>} \cdot T_s$
  else
    $J_{<BEH\_identifier>} += \text{costPerJouleCooled} \cdot u_{BEH,<control\_identifier>} \cdot T_s$
  end
end

6.5 Air Handling Unit

This EHF model considers the effect of an air handling unit (AHU) and the heat exchange from air flow between zones induced by having air supply and air return in different zones. This model considers only the thermal aspects of the ventilation and does not account for humidity, CO_2 levels and other air quality measures. One such EHF model describes a single AHU, i.e. all supply air inlets have the same air temperature. If multiple AHUs have to be modeled, this EHF model can be included multiple times (with different EHF model identifiers as specified when the EHF is declared).

The individual components of the AHU are depicted in Figure 3. At a minimum, the model includes a controlled unconditioned air mass flow from either the ambient air with temperature T_{amb} or some other pre-specified temperature T_{AHUin}. Additionally to that minimum model, it is possible to include a combination of the following: A heater and/or cooler in the supply air (modeled as simple positive/negative heat fluxes), an energy recovery (ERC) system (i.e. a heat exchanger between return and supply air) and an evaporative cooler[24].

Moreover, it is possible to assign to any zone a flow fraction defining how much of the total AHU air mass flow enters it. Since often supply and return ducts are not in the same offices, it is possible to define air flows between zones by additionally specifying flow fractions originating from other zones.

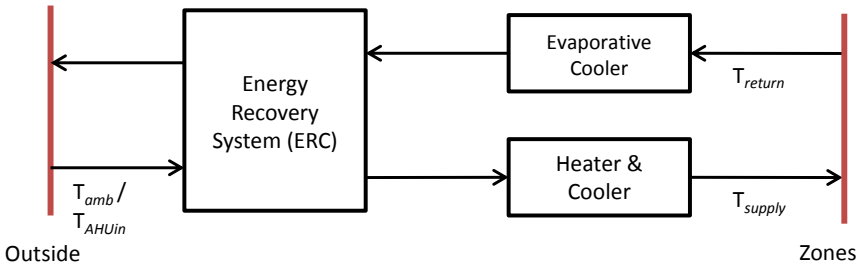


Figure 3: AHU schematic

The derivation of the model is given in [4]. It assumes an optimal actuation of the subcomponents (e.g. the ERC is fully used when the evaporative cooler is active). The restriction to a bilinear model made it necessary to introduce some intermediate variables as control inputs. We briefly describe their meaning here but refer for a detailed description to [4]. The total physical air mass flow through the AHU is given by the sum of the ERC bypassing and non-bypassing (if applicable) massflow control inputs $u_{ERC} + u_{noERC}$. The usage fraction of the ERC is given by $u_{ERC}/(u_{ERC} + u_{noERC})$. The usage fraction of the evaporative cooler is given by $u_{evapCooler}/u_{ERC}$.

6.5.1 Input data

Table 25: File structure according to our convention for the EHF model of a building's air handling unit.

AHU_specification	**key**[25]	**value**	
	hasERC[26]	1	
	ERCefficiency[27]	0.5	
	hasEvapCooler[28]	1	
	EvapCoolerEfficiency[29]	0.8	
	hasHeater[30]	1	
	hasCooler[31]	1	
	has_AHU_Tin[32]	1	
airflow_specification	**zone_identifier**[33]	**flow_fraction**[34]	**from_identifier**[35]
	Z0003	0.5	AHU
	Z0004	0.5	Z0003
	...	...	...

6.5.2 Model

Table 26 shows the nomenclature used in this section.

Table 26: Nomenclature.

Control inputs	
u_{ERC}	Air mass flow through fully active ERC. [kg/s].
u_{noERC}	Air mass flow bypassing ERC. [kg/s].
$u_{evapCooler}$	Virtual air mass flow through fully active evaporative cooler. [kg/s].
u_{heater}	Heating power. [W].
u_{cooler}	Cooling power. [W].
Disturbances	
v_{amb}	Ambient temperature. [°C].
v_{Tin}	Fresh air if different from ambient. [°C].
$v_{\Delta wb}$	(drybulb - wetbulb) temperature of the return air assumed independent of the actual return air temperature. [°C].
Model parameters	
η_{ERC}	Efficiency of the heat exchanger, i.e. the fraction of the maximum possible heat exchange between supply and return air. [-]. From EHF model input data.
$\eta_{evapCooler}$	Efficiency of the evaporative cooler, i.e. the fraction of the maximum possibly achievable temperature difference $v_{\Delta wb}$. [-]. From EHF model input data.
C_{air}	Heat capacity of air (assumed independent of the temperature). [J/kgK]. From standard gas properties.
Other nomenclature	
$x_{zone,i}$	Room temperature of zone i. [°C].
$q_{zone,i}$	Heat flux to zone i. [W].
$\mathcal{Z}_{sup}$	Set of all zones directly supplied by the AHU.

[24]The evaporative cooler cools the return air by spraying water into it. The cold is then added to the supply air via the ERC (this implies that the evaporative cooler cannot be present without the ERC).
[25]This column describes the construction type of the AHU. All fields are mandatory.
[26]Binary (0/1). Determines whether the AHU has an ERC.
[27]Numeric $\in [0, 1]$. Specifies the efficiency of the AHU's ERC.
[28]Binary (0/1). Determines whether the AHU has an evaporative cooler in the return path.
[29]Numeric $\in [0, 1]$. Specifies the efficiency of the evaporative cooler.
[30]Binary (0/1). Determines whether a heater is installed in the supply path.
[31]Binary (0/1). Determines whether a cooler is installed in the supply path.
[32]Binary (0/1). Determines whether the fresh air supply should be modeled by another temperature than ambient temperature.
[33]Zone identifier. Zones for which the air flow defined by the corresponding flow_fraction and from_identifer is considered.
[34]Numeric $\in [0, 1]$. Fraction of the the total AHU air massflow coming from the corresponding from_identifier. Sum of air flow fractions, where from_identifier is AHU must be equal to 1 and for all zones Δf :=inflow$_{tot}$-outflow$_{tot} \geq 0$. The remaining Δf is assumed to be return air.
[35]Zone identifier or 'AHU'. Describes the air flow source into the specific zone.

J_{AHU} Total costs related to the operation of the AHU.	
$\mathcal{Z}_{ret}$	Set of all zones directly returning air to the AHU.
T_{supply}	Temperature of the AHU supply air (see Figure 3). [°C].
T_{return}	Temperature of the AHU return air (see Figure 3). Flow fraction weighted average of the return zone temperatures. [°C].
<*AHU_identifier*>	Identifier of this EHF model, specified in the EHF model declaration.

In a first step, the identifiers are generated. These identifiers are used to bookkeep the meaning of the matrix and vector entries. All except ambient temperature and the drybulb-wetbulb difference (those are assumed to be the same for all models) have an infix <*AHU_identifier*>to distinguish them from other AHU model inputs. Note that only those identifiers are generated which are necessary according to the settings in the input data sheet 6.5.1.

Step 1: Generation of identifiers

disturbance identifiers

v_amb	(if has_AHU_Tin == 0)
v_<*AHU_identifier*>_Tin	(if has_AHU_Tin == 1)
v_deltaWb	(if hasEvapCooler == 1)

control input identifiers

u_<*AHU_identifier*>_ERC	(if hasERC == 1)
u_<*AHU_identifier*>_noERC	
u_<*AHU_identifier*>_cooler	(if hasCooler == 1)
u_<*AHU_identifier*>_heater	(if hasHeater == 1)
u_<*AHU_identifier*>_evapCooler	(if hasEvapCooler == 1)

constraint identifiers

<*AHU_identifier*>_mdot_{min,max}	
<*AHU_identifier*>_noERC_nonneg	
<*AHU_identifier*>_ERC_nonneg	(if hasERC == 1)
<*AHU_identifier*>_T_supply_{min,max}	
<*AHU_identifier*>_Q_heater_{min,max}	(if hasHeater == 1)
<*AHU_identifier*>_Q_cooler_{min,max}	(if hasCooler == 1)
<*AHU_identifier*>_evapCooler_nonneg	(if hasEvapCooler == 1)
<*AHU_identifier*>_evapCooler_max	(if hasEvapCooler == 1)

In a next step the influence on zones that are directly supplied with air is considered as shown by the pseudo-code below. In the formulas below $\{0,1\}$ specifies the availability of the signals according to the specification in the input data sheet. Note this is a very compact notation.

Step 2: AHU influence on directly supplied zones.

Find the set of all zones that are directly supplied by the AHU, $\mathcal{Z}_{sup}$, and the corresponding AHU supply flow fractions $f^i_{sup,AHU}\ \forall i \in \mathcal{Z}_{sup}$.

Find the set of all zones that return air to the AHU, $\mathcal{Z}_{ret}$, and the corresponding AHU return flow fractions $f^i_{ret,AHU}\ \forall i \in \mathcal{Z}_{ret}$.

if has_AHU_Tin == 1
  $v = v_{Tin}$
else
  $v = v_{amb}$
end

for $i \in \mathcal{Z}_{sup}$

$$
\begin{aligned}
q_{zone,i} \mathrel{+}= f^i_{sup,AHU} \cdot \Big(& C_{air} \cdot \big(v - x_{zone,i}\big) \cdot \big(\{0,1\} \cdot u_{ERC} + u_{noERC}\big) + \{0,1\} \cdot u_{heater} \\
& - \{0,1\} \cdot u_{cooler} + \{0,1\} \cdot u_{ERC} \cdot C_{air} \cdot \eta_{ERC} \cdot \Big(\sum_{k \in \mathcal{Z}_{ret}} f^k_{ret,AHU} \cdot x_{zone,k} - v \Big) \\
& - \{0,1\} \cdot u_{evapCooler} \cdot C_{air} \cdot \eta_{ERC} \cdot \eta_{evapCooler} \cdot v_{\Delta wb} \Big)
\end{aligned}
$$

end

In the third step, the air flow between zones is considered. Recall from the input data sheet that flow_fraction and from_identifier define the fraction of the total massflow that goes into the current zone originating from from_identifier. The expression $x_{\text{from\_identifier}(i)}$ denotes the state describing the temperature of the zone from_identifier(i).

Step 3: Air flow model between zones.

for $i \in \mathcal{Z}_{airflow}$
  if from_identifier(i) $\sim=$ 'AHU'
    $q_{zone,i} \mathrel{+}= \text{flow\_fraction}(i) \cdot C_{air} \cdot \big(x_{\text{from\_identifier}(i)} - x_{zone,i}\big) \cdot \big(\{0,1\} \cdot u_{ERC} + u_{noERC}\big)$
  end
end

6.5.3 Constraints

As described in Section 4.7, the `constraintsParameters` structure must be appropriately set up to generate the constraints of all the EHF models. For the building element heat flux, the fields
`constraintsParameters`.<*AHU_identifier*>.<*parameter_name*> must be set for every parameter shown in Table 27.

Table 27: Parameters related to the constraint generation which have to be specified whenever constraints are to be generated.

Parameter name	Description
T_supply_max	Maximum supply air temperature. [°C].
T_supply_min	Minimum supply air temperature. [°C].
Q_heater_max	Maximum supplied heating power by the heater. [W].
Q_cooler_max	Maximum supplied cooling power by the cooler. [W].
Q_heater_min	Minimum supplied heating power by the heater. [W].
Q_cooler_min	Minimum supplied cooling power by the cooler. [W].
mdot_min	Minimum supplied air flow. [kg/s].
mdot_max	Maximum supplied air flow. [kg/s].
v_fullModel	Disturbance vector of the full model. This includes unneeded disturbances from other EHF models too, but is more convenient.
x	Current state of the system. Necessary to linearize bilinearities in the constraints.

Constraint generation.

Generate constraints according to <*AHU_identifier*>_mdot_{min,max}

mdot_min $\leq \{0,1\} \cdot u_{ERC} + u_{noERC} \leq$ mdot_max

Generate constraints according to <*AHU_identifier*>_noERC_nonneg

$0 \leq u_{noERC}$

Generate constraints according to <*AHU_identifier*>_ERC_nonneg (if hasERC == 1)

$0 \leq u_{ERC}$

Generate constraints according to <*AHU_identifier*>_Q_heater_{min,max} (if hasHeater == 1)

Q_heater_min $\leq u_{heater} \leq$ Q_heater_max

Generate constraints according to <*AHU_identifier*>_Q_cooler_{min,max} (if hasCooler == 1)

Q_cooler_min $\leq u_{cooler} \leq$ Q_cooler_max

Generate constraints according to <*AHU_identifier*>_evapCooler_{nonneg,max}(if hasEvapCooler == 1)

$0 \leq u_{evapCooler} \leq u_{ERC}$

Generate constraints according to <*AHU_identifier*>_T_supply_{min,max}

$$\begin{aligned} C_{air} \cdot \big(\{0,1\} \cdot u_{ERC} + u_{noERC}\big) \cdot \text{T\_supply\_min} \leq\ & C_{air} \cdot v \cdot \big(\{0,1\} \cdot u_{ERC} + u_{noERC}\big) \\ & + \{0,1\} \cdot u_{heater} - \{0,1\} \cdot u_{cooler} + \{0,1\} \cdot C_{air} \cdot \eta_{ERC} \cdot u_{ERC} \cdot \big(T_{Return} - v\big) \\ & - \{0,1\} \cdot C_{air} \cdot \eta_{ERC} \cdot \eta_{evapCooler} \cdot v_{\Delta wb} \cdot u_{evapCooler} \\ & \leq C_{air} \cdot \big(\{0,1\} \cdot u_{ERC} + u_{noERC}\big) \cdot \text{T\_supply\_max}, \end{aligned}$$

where $T_{Return} \coloneqq \sum_{k \in \mathcal{Z}_{ret}} f^{k}_{ret,AHU} \cdot x_{zone,k}$ and v have been calculated from v_fullModel and x.

6.5.4 Costs

As described in Section 4.7, the `costParameters` structure must be appropriately set up to generate the cost vector. For the AHU EHF model, the fields `costParameters.`<*AHU_identifier*>.<*parameter_name*> must be set for every parameter with name <*parameter_name*> shown in Table 28.

Table 28: Parameters related to the cost generation which have to be specified whenever the cost is to be generated.

Parameter name	Description
costPerJouleHeated	Costs per Joule heating power [1/J].
costPerJouleCooled	Costs per Joule cooling power of the cooling coil [1/J].
costPerKgAirTransported	Costs per kg air transported [1/kg].
costPerKgCooledByEvapCooler	Costs per kg air cooled by the evaporative cooler at maximum usage [1/kg].

Cost generation.

$$J_{AHU} += \text{costPerKgAirTransported} \cdot (u_{ERC} + u_{noERC}) \cdot T_s$$
$$J_{AHU} += \text{costPerJouleHeated} \cdot u_{heater} \cdot T_s$$
$$J_{AHU} += \text{costPerJouleCooled} \cdot u_{cooler} \cdot T_s$$
$$J_{AHU} += \text{costPerKgCooledByEvapCooler} \cdot u_{evapCooler} \cdot T_s$$

6.6 User-Defined EHF Models

The toolbox was designed in order to support user defined EHF models by providing a simple interface to add custom EHF models (methods `declareEHFModel` and `undeclareEHFModel`), see Section 4.5. The following architecture ensures not only the provision of the EHF system matrices but also of functions generating corresponding constraints and costs and enables the addition of further EHF models by other parties. Any EHF model is an instance of a class satisfying the following properties: i) it has been derived from the base EHF model class `EHFModelBaseClass`; ii) its constructor takes as input just the `Building` object, an identifier string for the particular EHF model and a path to an input data sheet that parameterizes it; iii) based on the input data sheet, the constructor generates the EHF model matrices and implements the abstract EHF model base class methods `getConstraintsMatrices` and `getCostVector`. Since the EHF model depends only on its "own" control and disturbance inputs, functions are provided in the EHF model base class that generate the full sized model matrices, constraints matrices and cost vector given the identifiers of the full model's control and disturbance inputs.

If these conditions are satisfied, the Toolbox can iterate over all EHF model objects, call these functions and compile the results into the full model and the full model's constraints and costs matrices/vectors.

7 EnergyPlus to BRCM Thermal Model Input Data Converter

The Toolbox allows to convert EnergyPlus input data files (idf) into the BRCM thermal model input data format (see Section 5.2). Note that this only works for the thermal model data and

that the EHF models have to be specified by hand. Not all of the EnergyPlus idf features can be translated into the BRCM format, see Section 7.2.

The benefit this functionality provides is threefold: i) It allows to generate similar models to compare BRCM models to EnergyPlus; ii) many buildings have already been modeled in EnergyPlus and there exist very useful programs to do so; iii) many researchers use EnergyPlus/Matlab co-simulations for demonstrating their MPC controllers' performance.

The conversion consists of four steps. First, the version of the idf-file is identified and the according input data dictionary (idd) file that defines the idf-file format is parsed. Second, the idf-file is read and objects are created with attribute names/values according to the idd/idf-files. Third, these objects are converted into "intermediate" objects that contain only the information needed later for the BRCM thermal model input data generation. This object layer unifies and aggregates the information from all possible ways how building elements and constructions can be specified in EnergyPlus. In the fourth and last step, the intermediate layer is converted into the BRCM thermal model input data and saved.

The converter has been extensively tested with idf-files of the currently latest EnergyPlus version 8.0. Also 7.0, 7.1, 7.2 idf-files have been successfully converted (but not extensively tested).

7.1 Usage

The following command will execute the conversion

```
outputFolderName = convertIDFToBRCM(idfFilename,outputFolderName,forceFlag);
```

In this, `idfFilename` is a mandatory argument defining which idf-file is to be converted and `outputFolderName` and `forceFlag` are optional arguments specifying the output directory (will be created if it does not exist) and whether the function should overwrite existing files without asking.

7.2 Limitations

The conversion algorithm has some limitations. First it must be stressed again that all EHF models have to be defined by hand (it is planned however to add support for this). Also, there exist many features for defining EnergyPlus model geometry and constructions and not all of them are supported. If any of the unsupported features are present in the idf-file, the converter will produce an appropriate error or, if not critical, a warning message. The following list summarizes all currently not supported EnergyPlus features

- No doors and internal windows are considered (conversion will issue a warning, but continue).
- Window properties are summarized by a separate U-Value and solar heat gain coefficient (SHGC) for each window construction type. These parameters will appear in the parameter list and their values have be specified by the user (default values are provided). EnergyPlus window constructions and window materials are currently *not* considered.
- Zone volume/area/height can currently not be calculated in the case of non-vertical walls or non-horizontal floors/ceilings (conversion will issue an error). This can be circumvented by specifying these values in the corresponding EnergyPlus zone objects.

- It is currently not possible to compute the vertices from non-"detailed" EnergyPlus surfaces (in which the surface's location is not specified using vertices). The thermal model can still be generated but the corresponding surfaces cannot be plotted.
- Zone multipliers not equal to 1 cannot be considerd (conversion will issue an error).
- The following values of `Outside Boundary Condition` in EnergyPlus surface descriptions are not supported: `GroundFCfactorMethod`, `OtherSideConditionsModel`, `GroundSlab`*, `GroundBasement`* (conversion will issue an error).
- Additional material properties `MaterialProperty:`* are not supported (conversion will issue a warning, but continue).
- Shading surfaces are not considered.

7.3 Considered idf-Objects

This section summarizes the idf-objects that are considered in the conversion.

7.3.1 Group - Surface Construction Elements

`Material`
`Material:NoMass`
`Material:InfraredTransparent`
`Material:AirGap`
`Construction`
`Construction:InternalSource`

7.3.2 Group - Thermal Zone Description/Geometry

`Zone`	with the above limitation regarding the zone multiplier
`InternalMass`	
`Window`	
`FenestrationSurface:Detailed`	only exterior windows, i.e. `Surface Type: Window`
`Floor:Detailed`	with the above limitations regarding `Outside Boundary Condition`
`Floor:Adiabatic`	
`Floor:GroundContact`	
`Floor:Interzone`	
`Roof`	
`RoofCeiling:Detailed`	with the above limitations regarding `Outside Boundary Condition`
`Ceiling:Adiabatic`	
`Ceiling:Interzone`	
`Wall:Interzone`	
`Wall:Adiabatic`	
`Wall:Detailed`	with the above limitations regarding `Outside Boundary Condition`
`Wall:Exterior`	
`Wall:Underground`	
`BuildingSurface:Detailed`	with the above limitations regarding `Outside Boundary Condition`

References

[1] D. Sturzenegger, D. Gyalistras, M. Morari, and R. Smith, "Semi-automated modular modeling of buildings for model predictive control," *BuildSys'12 Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pp. 99–106, 2012.

[2] B. Lehmann, D. Gyalistras, M. Gwerder, K. Wirth, and S. Carl, "Intermediate complexity model for model predictive control of integrated room automation," *Energy and Buildings*, vol. 58, pp. 250–262, 2013.

[3] D. Sturzenegger, D. Gyalistras, V. Semeraro, M. Morari, and R. S. Smith, "BRCM Matlab Toolbox: Model Generation for Model Predictive Building Control," in *American Control Conference 2014 (submitted)*, 2014.

[4] D. Sturzenegger, "Bilinear modeling for model predictive control of an air handling unit," tech. rep., ETH Zurich, 2012. www.control.ee.ethz.ch.