House model FutureFactory

Trung Nguyen
HAN University of Applied Sciences
Arnhem, The Netherlands

September 26, 2022

Contents

1 Introduction

Building energy simulation is a vast field of research that started in the late 50's and that is still highly active nowadays. Building energy simulations are mainly used to help taking design decisions, to analyze current designs and to forecast future building energy use. Building energy modelling methods can mainly be divided into three categories:

- White box models (physics-based)
- Black box models (data-driven)
- Grey box models (hybrid)

White box models are based on the equations related to the fundamental laws of energy and mass balance and heat transfer. White box models can be differentiated in two types, *distributed* parameter models and *lumped* parameter models. Lumped parameter models simplify the description of distributed physical systems into discrete entities that approximate the behavior of a distributed system. The advantage of using lumped models is the decrease in simulation time **(Ramallo-González et al.**[1]**)**. White box models are of special interest for the design phase as they are used to predict and analyse the performance of the building envelope and building systems. Black box models are based on the statistical relation between input and output system values. The statistical relation between input and output is based on actual data. The relation between the parameters can differ based on the amount of data and the method used to analyze the relation. Currently, there is a large and active field of research about statistical models that are used on black box models **(Coacley et al.**[2]**)**. Black box models are of special interest when there is a large amount of actual input and output data available.

Grey box models are hybrid models that aim to combine the advantages of both approaches. In order to use them it is necessary to implement some equations and it is also required to have actual data of inputs and outputs.

2 White box lumped model: RC network

2.1 White box lumped model

The objective of the house model for this project is to serve as test environment for a heat pump model, which means that the house model is intended as a tool to help taking building systems design decisions. The house heating demand calculation model implemented for this project is a white box *lumped* model. Specifically, it is a RC network model consisting of resistances (R) and capacities (C). The RC network model is based on the analogy with electrical circuits. The simulation of thermodynamic systems characterizing building elements as resistances or capacities allows to simplify the model while maintaining a high simulation results accuracy **(Bagheri et al.[3], Bacher et al[4].)**.

There are several types of RC models, the most common being 3R4C models and 3R2C models which are applied on the outer and internal wall. For the simulation of simple house buildings 3R2C models perform as accurate as more complex 3R4C models **(Fraisse et al.[5])**. Considering that one of the objectives for this project is to obtain a fast but accurate simulation of a simple dwelling the 3R2C network model appeared a good starting point. In the 3R2C model two indoor temperature nodes are present in the dwelling. **with capacities (usually an air and a wall temperature) and a well-known outdoor temperature** . **Between these 3 temperature nodes 3 heat transfer resistances are present. However, the direct heat transfer between the inner walls and the outdoor air is low. Moreover, uncertainties are present about heat transfer coefficients between walls and indoor air, different indoor temperatures in the house rooms and the ground temperature which deviates from the outdoor temperature. In addition, occupancy behaviour varies strongly.** For that reason, we have made a further simplification to a 2R2C model. In section 4 it is shown that this dwelling model delivers a reliable annual energy consumption.

2.2 House Model R and C Values

This section presents the basic information for calculating a house model based on an RC network. This category of house models, analogous to electrical impedance networks, may have different numbers of R and C components and may have various component topologies. For the specific model properties, references will be given.

In heat transfer theory the basic thermal circuit contains thermal resistances. Heat transfer occurs via conduction, convection and radiation. In analogy with Ohm's Law for electricity, expressions can be derived for the heat transfer rate (analogous to electrical current) and the thermal resistances (analogous to ohmic resistances) in these three modes of heat transfer. The temperature difference plays a role analogous to the electrical voltage difference. These expressions are shown in Fig.1.

Equations for different heat transfer modes and their thermal resistances.

Transfer Mode	Rate of Heat Transfer	Thermal Resistance
Conduction	$\dot{Q} = \dfrac{T_1 - T_2}{\left(\frac{L}{kA}\right)}$	$\dfrac{L}{kA}$
Convection	$\dot{Q} = \dfrac{T_{\text{surf}} - T_{\text{envr}}}{\left(\frac{1}{h_{\text{conv}} A_{\text{surf}}}\right)}$	$\dfrac{1}{h_{\text{conv}} A_{\text{surf}}}$
Radiation	$\dot{Q} = \dfrac{T_{\text{surf}} - T_{\text{surr}}}{\left(\frac{1}{h_r A_{\text{surf}}}\right)}$	$\dfrac{1}{h_r A}$, where $h_r = \epsilon\sigma(T_{\text{surf}}^2 + T_{\text{surr}}^2)(T_{\text{surf}} + T_{\text{surr}})$

Figure 1: Heat transfer modes[6]

In [7] and [8] the expressions in Fig.1 are derived. For conduction, the expression for absolute thermal resistance is:

$$R = \frac{L}{kA} \qquad \left[\frac{K}{W}\right] \tag{1}$$

- L is the distance over which heat transfer takes place, or the thickness of the material $[m]$.
- k (also denoted with λ) is the thermal conductivity of the material. $[\frac{W}{mK}]$.
- A is the conductive surface area $[m^2]$.
- Thermal resistivity is the reciprocal of thermal conductivity and can be expressed as $r = \frac{1}{k}$ in $[\frac{mK}{W}]$

For convection and radiation the expression for thermal resistance is: $R = \frac{1}{h \cdot A}$ $[\frac{K}{W}]$.

- A is the surface area where the heat transfer takes place $[m^2]$.
- h is the heat transfer coefficient $[\frac{W}{m^2K}]$

The R-value (in Dutch: R-waarde or R_d-waarde) of a building material [9] is the thermal resistance of a square meter surface. It can be calculated by multiplying the thermal *resistivity* with the thickness of the material in m. Alternatively it is calculated by dividing the material thickness by the thermal *conductivity* k or λ.

$$\text{R-value} = r \cdot L \qquad \text{or} \qquad \text{R-value} = \frac{L}{k} \qquad \text{or} \qquad \text{R-value} = \frac{L}{\lambda} \qquad \left[m \cdot \frac{m \cdot K}{W}\right] = \left[\frac{m^2 \cdot K}{W}\right] \tag{2}$$

Some typical heat transfer R-values are: [10]:

- Static layer of air, 40 mm thickness (1.57 in) : R = 0.18 $[\frac{m^2K}{W}]$.
- Inside heat transfer resistance, horizontal current : R = 0.13 $[\frac{m^2K}{W}]$.
- Outside heat transfer resistance, horizontal current : R = 0.04 $[\frac{m^2K}{W}]$.
- Inside heat transfer resistance, heat current from down upwards : R = 0.10 $[\frac{m^2K}{W}]$.
- Outside heat transfer resistance, heat current from above downwards : R = 0.17 $[\frac{m^2K}{W}]$.

Note: in Dutch building physics, R-values with subscripts are used:

- R_d-waarde is used for the R-value of a homogeneous building material. $R = \frac{L}{\lambda}$
- R_c-waarde (compound, construction) is used for the R-value of a surface consisting of several building materials. R_c-waarden are calculated as the surface-area weighted sum of R_d-waarden of the building materials. For the simplest roof surface, R_c is a linear combination of the R-values of the wooden joists and girders (spanten en gordingen) and the areas in between with a certain insulation material sandwich. The R-value of the insulation sandwich, in its turn, is the sum of the R-values of the materials in the sandwich. From inside out, this sandwich may consist of *e.g.* a 9.5 mm plaster board, a PIR/PUR insulation panel, an air gap and a wooden roof deck. All types of R-value have the dimension $[\frac{m^2 \cdot K}{W}]$.

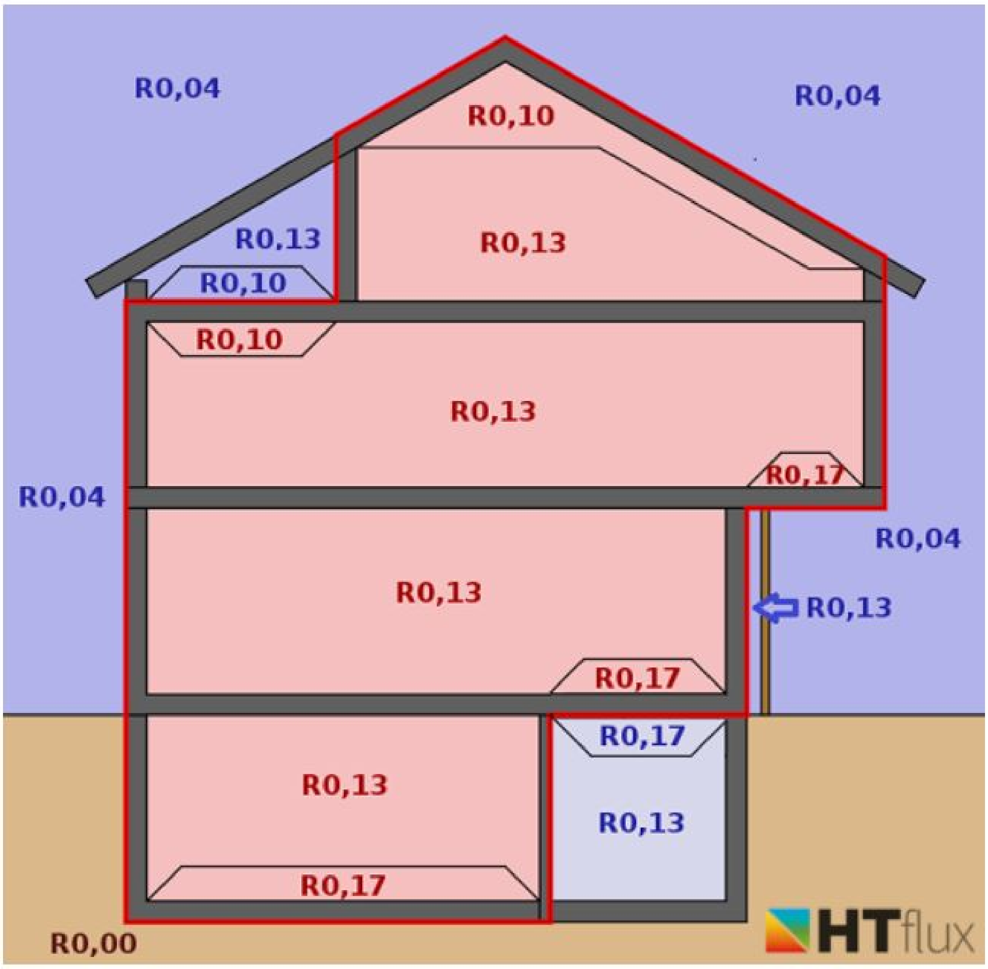

Figure 2: An overview of R-values for heat transfer [11].

The standard R_c-values that have been used for facades, roof and floor until 2020 are summarized in Fig.3:

Construction	New construction	Renovation
Facades [1]	Rc 4.5 m2K / W	Rc 1.3 m2K / W
Roofs [2]	Rc 6.0 m2K / W	Rc 2.0 m2K / W
Floors [3]	Rc 3.5 m2K / W	Rc 2.5 m2K / W

Figure 3: R_c Values [12]

New standard values will be used from 1-1-2021, since the building standard NEN 1068 will be replaced by the NTA 8800 standard. The old and new situation is described in "EnergieVademecum Energiebewust ontwerpen van nieuwbouwwoningen", Hoofdstuk 5: Thermische isolatie, thermische bruggen en luchtdichtheid. [13].

From 2015, the following RC values apply to new construction in the Netherlands:

Location	*RC value (NEN 1068, until 1-1-2021) [m2K / W]*	*Rc value (NTA 8800, from 1-1-2021) [m2K / W]*
floor	**> = 3.5**	**> = 3.7**
facade	**> = 4.5**	**> = 4.7**
roof	**> = 6.0**	**> = 6.3**

Figure 4: R_c Values [14]

The values used for different types of houses such as: row houses, detached houses and apartments can be found in the document "Voorbeeldwoningen 2011" [15]. An example with values for a common type of row house, built in the period from 1975 to 1991 is shown in Fig. 5:

Bouwdelen	Huidig			Besparingspakket			Investeringskosten	
	Opp. (m²)	Rc-Waarde (m² K/W)	U-Waarde (W/m² K)	Opp. (m²)	Rc-Waarde (m² K/W)	U-Waarde (W/m² K)	Per m²	Totaal
Begane grondvloer 3	51,0	0,52	1,28	51,0	2,53	0,36	€ 20	€ 1.020
Plat dak 3	-	-	-	-	-	-	-	€ 0
Hellend dak 3	68,6	1,30	0,64	68,6	2,53	0,36	€ 53	€ 3.640
Achter- en voorgevel								
– Gesloten 3	40,6	1,30	0,64	40,6	2,53	0,36	€ 21	€ 850
– Enkelglas 3	3,1		5,20	-		-	€ 139	€ 430
– Dubbelglas 3	16,2		2,90	-		-	€ 142	€ 2.300
– HR++ glas	-		-	19,3		1,80		
Zijgevel								
– Gesloten	58,4	1,30	0,64	58,4	2,53	0,36	€ 21	€ 1.230
– Enkelglas	-		-	-		-	-	€ 0
– Dubbelglas	1,8		2,90	-		-	€ 142	€ 260
– HR++ glas	-		-	1,8		1,80		

Figure 5: R_c-values for a row house type built between 1975-1991 [15]

2.3 Dwelling (envelope) model analogous to a 2R-2C network

The heat flow will be modelled by analogy to an electrical circuit where heat transfer rate is analogous to by current, temperature difference is analogous to potential difference, heat sources are represented by constant current sources, absolute thermal resistances are represented by resistors and **thermal capacitance** heat capacity ? by capacitors [16]. Figure 6 summarizes the similar term use in different fields.

type	structural analogy[1]	hydraulic analogy	thermal	electrical analogy[2]
quantity	impulse J [N·s]	volume V [m^3]	heat Q [J]	charge q [C]
potential	displacement X [m]	pressure P [N/m^2]	temperature T [K]	potential V [V = J/C]
flux	load or force F [N]	flow rate Q [m^3/s]	heat transfer rate $\dot{Q}$ [W = J/s]	current I [A = C/s]
flux density	stress σ [Pa = N/m^2]	velocity $\mathbf{v}$ [m/s]	heat flux $\mathbf{q}$ [W/m^2]	current density $\mathbf{j}$ [C/(m^2·s) = A/m^2]
resistance	flexibility (rheology defined) [1/Pa]	fluid resistance R [...]	**thermal resistance** R [K/W]	electrical resistance R [Ω]
conductance	... . . . [Pa]	fluid conductance G [...]	thermal conductance G [W/K]	electrical conductance G [S]
resistivity	flexibility $1/k$ [m/N]	fluid resistivity	thermal resistivity [(m·K)/W]	electrical resistivity ρ [Ω·m]
conductivity	stiffness k [N/m]	fluid conductivity	thermal conductivity k [W/(m·K)]	electrical conductivity σ [S/m]
lumped element linear model	Hooke's law $\Delta X = F/k$	Hagen–Poiseuille equation $\Delta P = QR$	Newton's law of cooling $\Delta T = \dot{Q}R$	Ohm's law $\Delta V = IR$
distributed linear model	... . . .	... . . .	Fourier's law $\mathbf{q} = -k\nabla T$	Ohm's law $\mathbf{J} = \sigma\mathbf{E} = -\sigma\nabla V$

Figure 6: Table of Analogies [16]

The 2R-2C house model structure is implemented as described below. The schematic of an envelope house model has been shown in figure 9 and the equivalent electrical 2R-2C network with components and topology is given in fig 10.

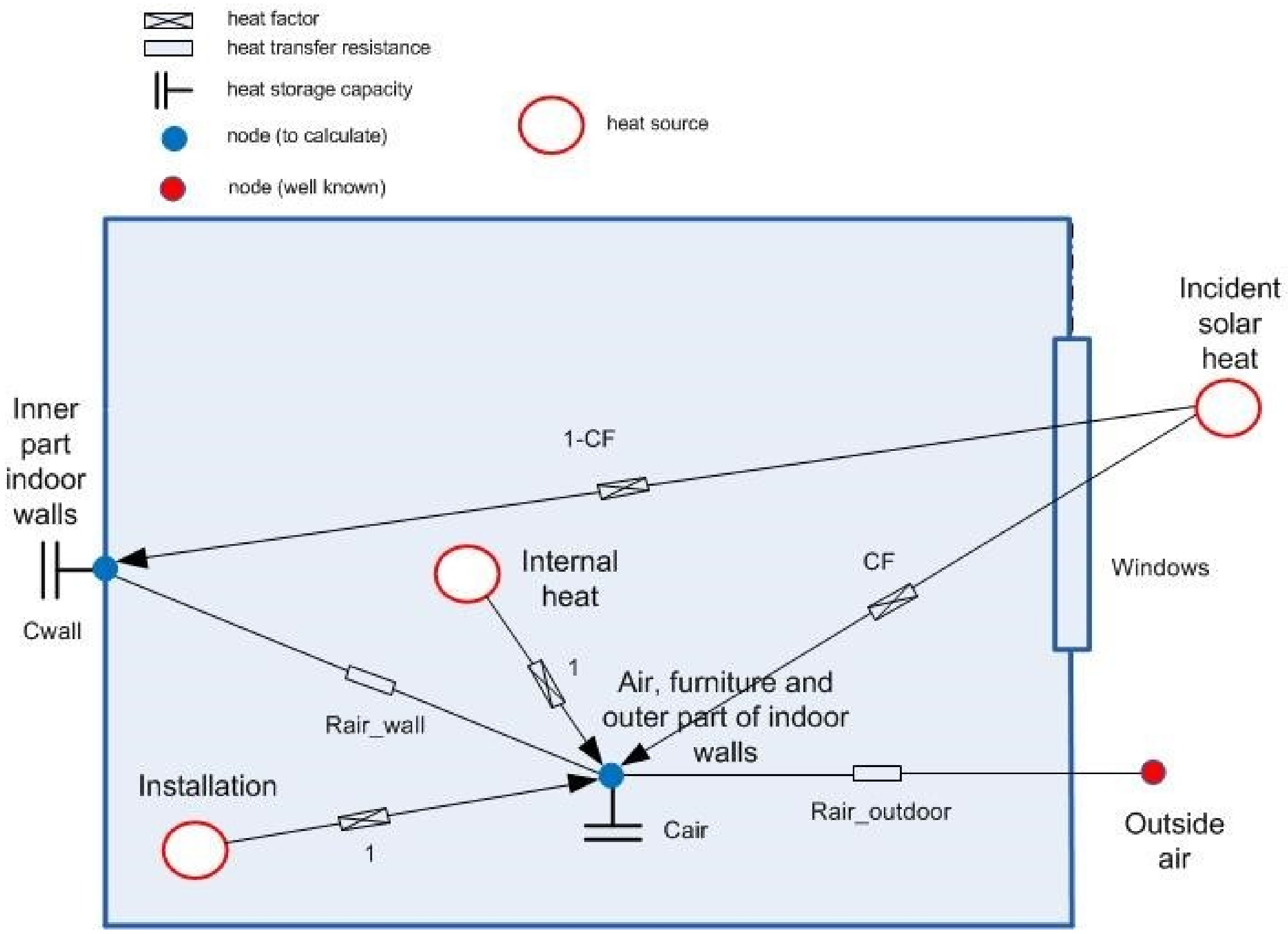

Figure 7: Schematic of envelope model

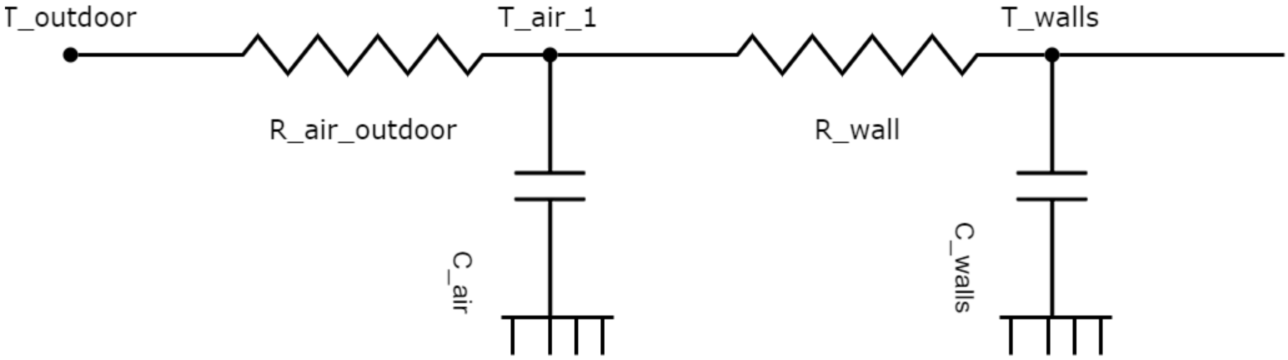

Figure 8: 2R-2C house model

The model consists of two heat capacities $C_{air,\ indoor}$ and C_{wall} and two resistances R_{wall} and $R_{air,\ outdoor}$. The incident solar energy is divided between C_{wall} and C_{air} through the convection factor CF. It is assumed that both internal heat (lighting, occupancy and electric devices) and supplied heat (installation) initially heat up the indoor air. In Fig. 9, they are fully released at the T_{air} node.

It is also assumed that furniture and the **surface part** of the walls have the same temperature as the air **and the wall mass is divided between the air and wall mass**. Thus, the heat capacity of the air node consists of the air heat capacity, furniture heat capacity and the heat capacity **of a part of the walls**. **Appendix A** presents the coefficients in the dwelling model. In the resistance $R_{air,\ outdoor}$ the influence of heat transmission through the outdoor walls and natural ventilation is considered.

For the air and wall nodes the following power balances can be set up:

$$C_{air}\frac{dT_{air}}{dt} = \frac{T_{outdoor} - T_{air}}{R_{air\_outdoor}} + \frac{T_{wall} - T_{air}}{R_{air\_wall}} + \dot{Q}_{inst} + \dot{Q}_{internal} + CF \cdot \dot{Q}_{solar} \tag{3}$$

$$C_{wall}\frac{dT_{wall}}{dt} = \frac{T_{air} - T_{wall}}{R_{air\_wall}} + (1 - CF) \cdot \dot{Q}_{solar} \tag{4}$$

- CF: convection factor (solar radiation): the convection factor is the part of the solar radiation that enters the room and is released directly convectively into the room.
- $\dot{Q}_{inst}$: delivered heat from heating system (radiator) [W].
- $\dot{Q}_{inernal}$: internal heat [W].
- $\dot{Q}_{solar}$: heat from solar irradiation [W].
- T_{air}: indoor air temperature ^{o}C.
- $T_{outdoor}$: outdoor temperature ^{o}C.
- T_{wall}: wall temperature ^{o}C.
- R_{air_wall}: walls surface resistance $[\frac{K}{W}]$.
- $R_{air\_outdoor}$: outdoor surface resistance $[\frac{K}{W}]$.
- C_{air}: air thermal capacitance (heat capacity) $[\frac{J}{K}]$[17].
- C_{wall}: wall thermal capacitance (heat capacity) $[\frac{J}{K}]$[17].

Total heat transfer of solar irradiation through the glass windows.

$$\dot{Q}_{solar} = g. \sum (A_{glass}.\dot{q}_{solar}) \tag{5}$$

- $\dot{q}_{solar}$: solar radiation on the outdoor walls $[\frac{W}{m^2}]$.
- g: g value of the glass (ZTA in dutch) [0..1][18]
- A: glass surface $[m^2]$.

3 Dwelling (envelope) model analogous to a 2R-2C network

The 2R-2C house model structure is implemented as described below:

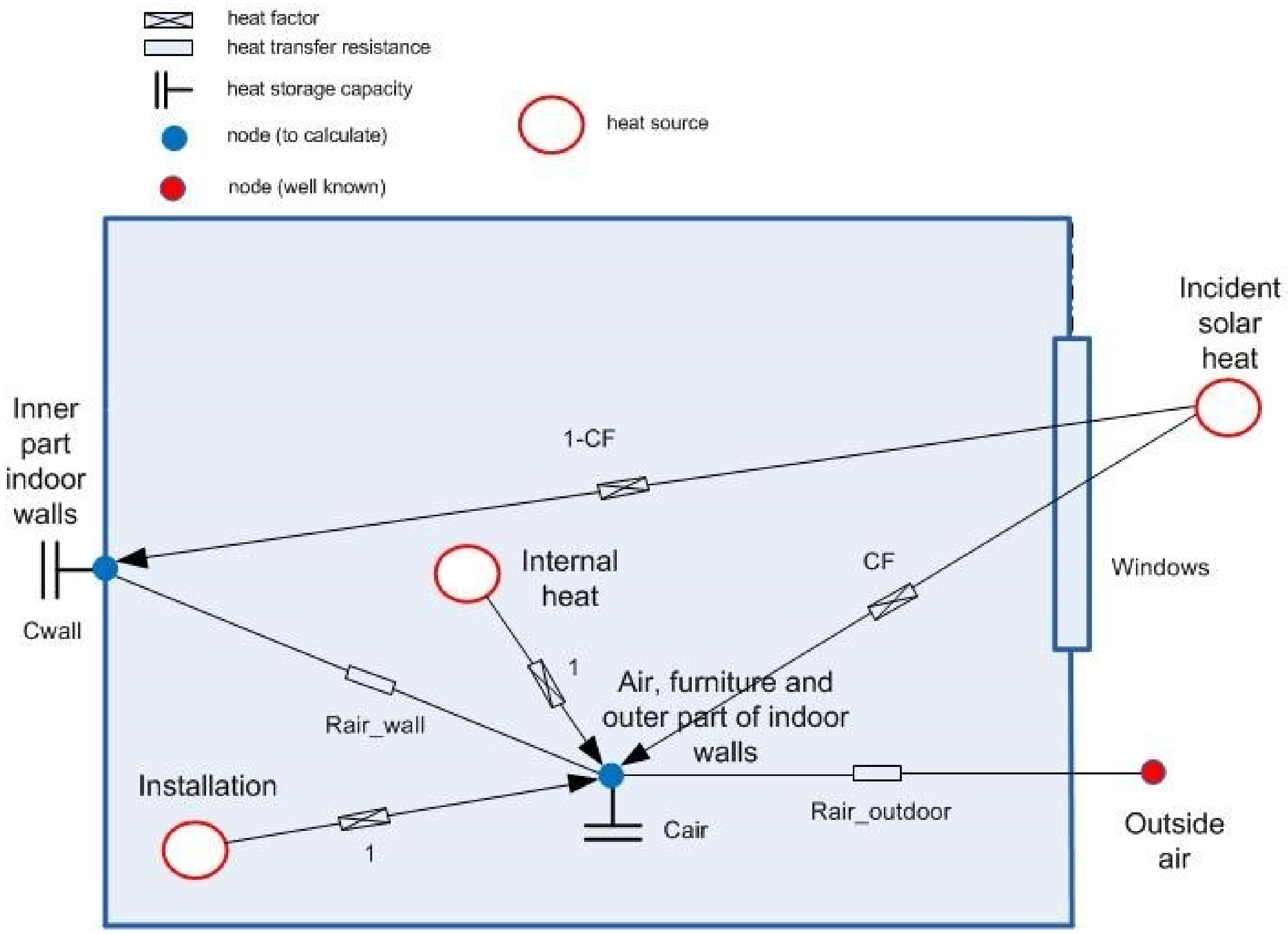

Figure 9: Schematic of envelope model

The equivalent electrical 2R-2C network with components and topology is given in Fig. 10.

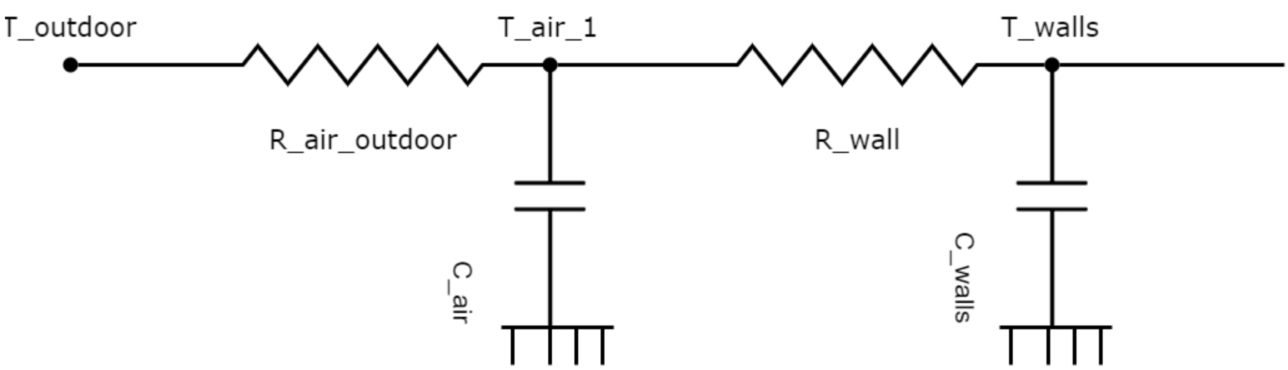

Figure 10: 2R-2C house model

The model consists of two capacitances $C_{air,\ indoor}$ and C_{wall} and two resistances R_{wall} and $R_{air,\ outdoor}$. The incident solar energy is divided between C_{wall} and C_{air} through the convection factor CF. It is assumed that both internal heat (lighting, occupancy and electric devices) and supplied heat (installation) initially heat up the indoor air. In Fig. 10, they are fully released at the T_{air} node.

It is also assumed that furniture and the surface part of the walls have the same temperature as the air and the wall mass is divided between the air and wall mass. Thus, the capacity of the air node consists of the air capacity, furniture capacity and capacity of a part of the walls. **Appendix A** presents the coefficients in the dwelling model. In the resistance $\mathrm{R}_{\text{air, outdoor}}$ the influence of heat transmission through the outdoor walls and natural ventilation is considered.

For the air and wall nodes the following energy balances can be set up:

$$C_{air}\frac{dT_{air}}{dt} = \frac{T_{outdoor} - T_{air}}{R_{air\_outdoor}} + \frac{T_{wall} - T_{air}}{R_{air\_wall}} + \dot{Q}_{inst} + \dot{Q}_{internal} + CF \cdot \dot{Q}_{solar} \tag{6}$$

$$C_{wall}\frac{dT_{wall}}{dt} = \frac{T_{air} - T_{wall}}{R_{air\_wall}} + (1 - CF) \cdot \dot{Q}_{solar} \tag{7}$$

- CF: Convection factor (solar radiation): the convection factor is the part of the solar radiation that enters the room and is released directly convectively into the room
- $\dot{Q}_{inst}$: delivered heat from heating system (radiator) [W].
- $\dot{Q}_{solar}$: heat from solar irradiation [W].
- T_{air}: indoor air temperature ^{o}C.
- $T_{outdoor}$: outdoor temperature ^{o}C.
- T_{wall}: wall temperature ^{o}C.
- R_{air_wall}: walls surface resistance $[\frac{K}{W}]$.
- $R_{air\_outdoor}$: outdoor surface resistance $[\frac{K}{W}]$.
- C_{air}: air capacity $[\frac{J}{K}]$.
- C_{wall}: wall capacity $[\frac{J}{K}]$.

Total heat transfer of solar irradiation through the glass windows.

$$Q_{solar} = g.\sum(A_{glass}.q_{solar}) \tag{8}$$

- q_{solar}: solar radiation on the outdoor walls $[\frac{W}{m^2}]$.
- g: g value of the glass (ZTA in dutch) [0..1][18]
- A: glass surface $[m^2]$.

4 2 Zones house model 7R4C network

The 4R-7C house model structure is implemented as described below:

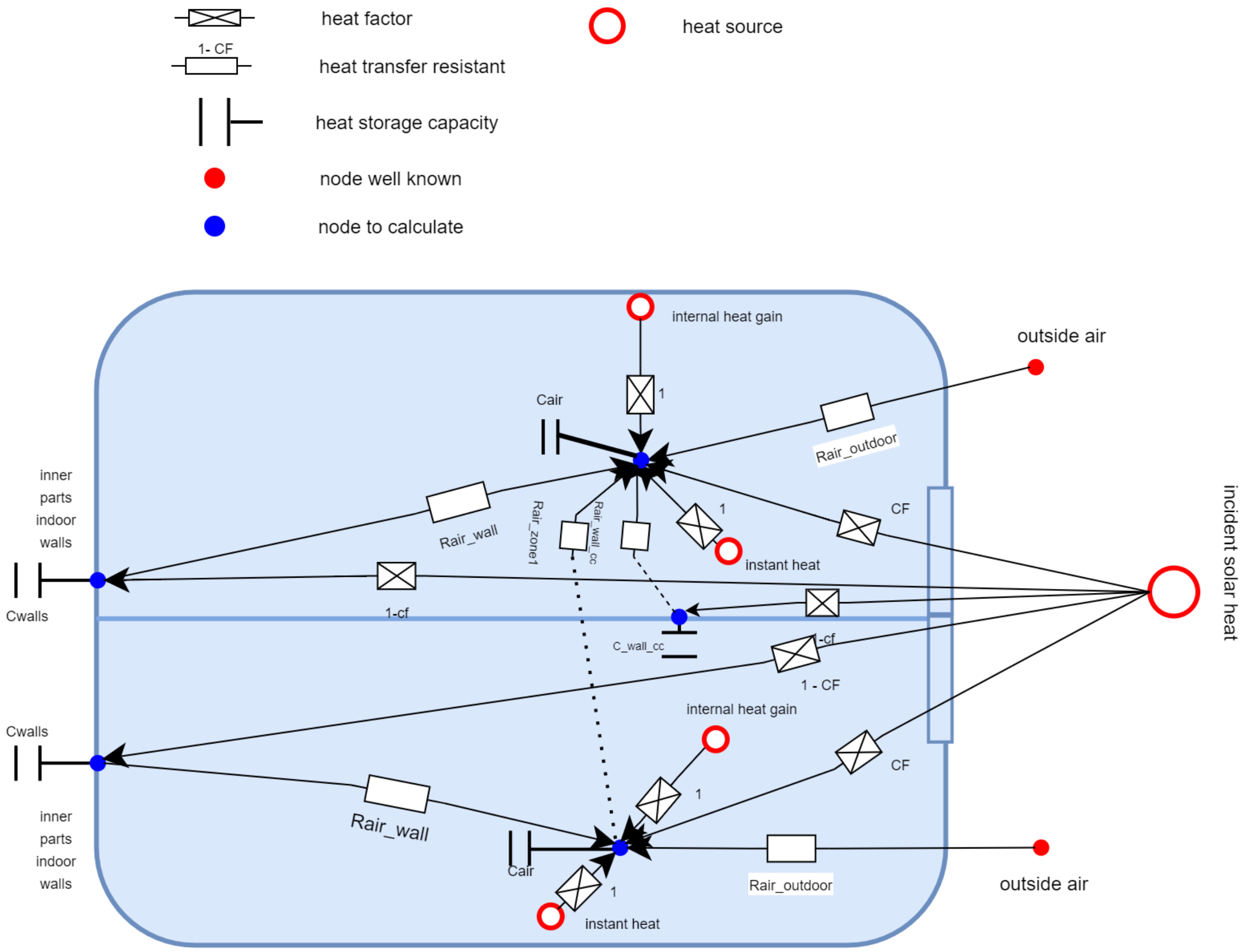

Figure 11: Schematic of a 2 zones house model

The equivalent electrical 7R-4C network with components and topology is given in Fig. 12.

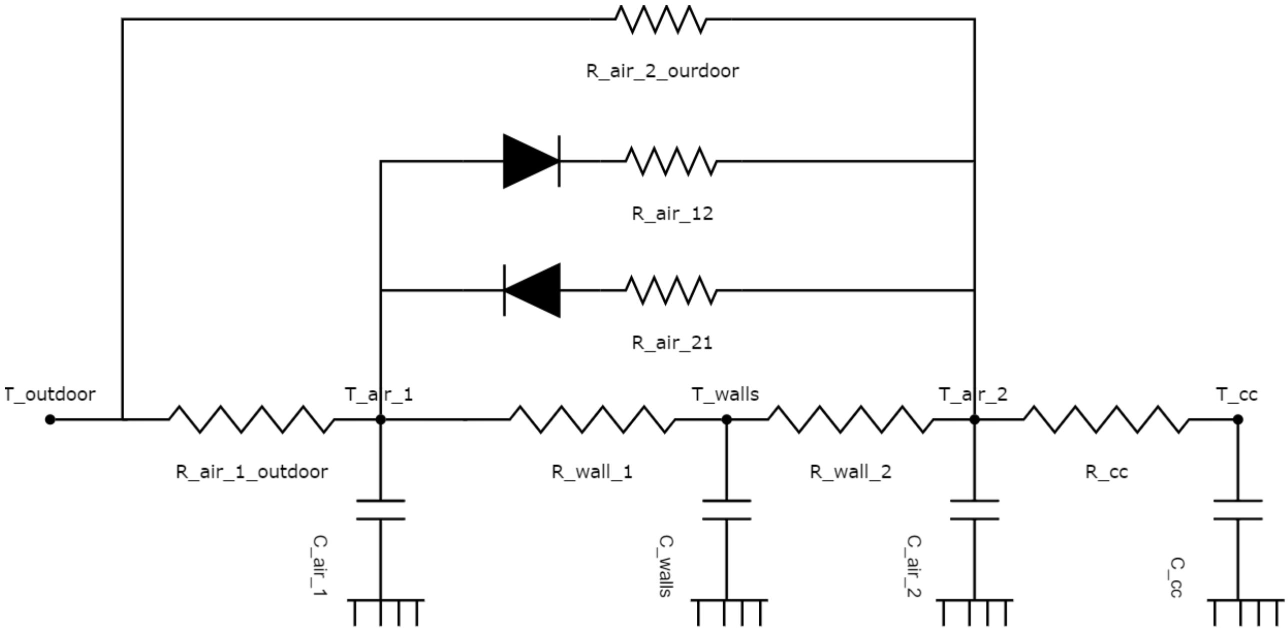

Figure 12: R-C circuits of 2 zones house model

with:

- `T_outdoor` : outdoor temperature [$^{\circ}C$]
- `T_air_1` : zone 1 air temperature [$^{\circ}C$]
- `T_walls` : wall temperature [$^{\circ}C$]
- `T_air_2` : zone 2 air temperature [$^{\circ}C$]
- `T_cc` : temperature of the concrete layer between zone 1 and zone 2 [$^{\circ}C$]
- `R_air_1_outdoor` : outdoor resistance valus.
- `R_wall_1` : walls resistance value.
- `R_wall_2` : walls resistance value.
- `R_cc` : concrete resistance value.
- `R_air_12` : resistance value of air flow from zone 1 to zone 2.
- `R_air_21` : resistance value of air flow from zone 2 to zone 1.

5 Lumped-element thermal model of a building

abels:lumped-element

Heat generation and transport inside a building, with heat loss to the surrounding outdoor environment is governed by the same laws of conduction, convection and radiation as elsewhere. A number of approximations is made, however, which will be treated below:

5.1 Heat Conduction: Fourier's Law

Heat transport *within* a solid material is governed by conduction, according to Fourier's Law, illustrated in Figure 13. One side of a rectangular solid is held at temperature T_1, while the opposite side is held at a lower temperature, T_2. The other four sides are insulated so that heat can flow only in the x-direction. For a given material, it is found that the rate, $\dot{Q}_x$, at which heat (thermal energy) is transferred from the hot side to the cold side (the *heat transfer rate*) is proportional to the cross-sectional area, A, across which the heat flows; the temperature difference, $T_1 - T_2$; and inversely proportional to the thickness, Δx, of the material. That is:

$$\dot{Q}_x = -kA\frac{\Delta T}{\Delta x} \tag{9}$$

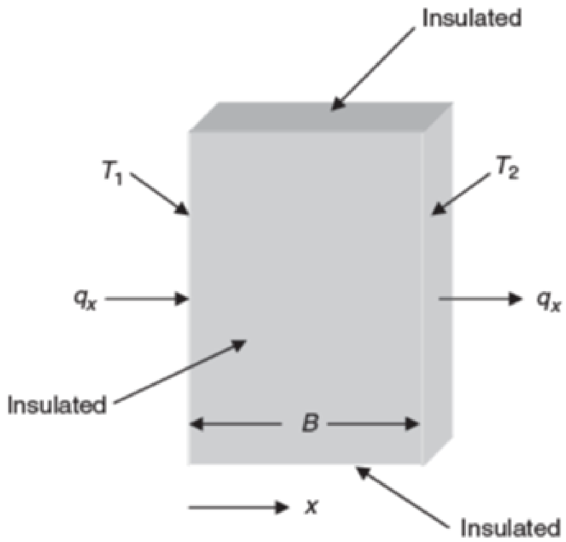

Figure 13: One-dimensional heat conduction in a solid

The constant of proportionality, k, is called the *thermal conductivity.* Equation (9) is also applicable to heat conduction in liquids and gases. However, when temperature differences exist in fluids, convection currents tend to be set up, so that heat is generally not transferred solely by the mechanism of conduction. The thermal conductivity is a property of the material. Values may be found in various handbooks and compendiums of physical property data.

The form of Fourier's law given by Equation (9) is valid only when the thermal conductivity can be assumed constant. A more general result can be obtained by writing the equation for an element of differential thickness. in the limit as Δx approaches zero, $\frac{\Delta T}{\Delta x} \rightarrow \frac{dT}{dx}$. Thus, substituting in Equation (9) gives:

$$\dot{Q}_x = -kA\frac{dT}{dx} \tag{10}$$

Equation (10) is not subject to the restriction of constant k. Furthermore, when k is constant, it can be integrated to yield Equation (9). Hence, Equation (10) is the general one-dimensional form of Fourier's law. The negative sign is necessary because heat flows in the positive x-direction when the temperature decreases in the x-direction. Thus, according to the standard sign convention that $\dot{Q}_x$ is positive when the heat flow is in the positive x-direction, $\dot{Q}_x$ must be positive when dT/dx is negative.

5.1.1 More than one dimension

It is often convenient to formulate Fourier's Law in the original phrasing: the *heat flux* $\dot{\varphi}$ is proportional to the *temperature gradient*. We divide (10) by the area to give:

$$\dot{\varphi}_x \equiv \frac{\dot{Q}_x}{A} - k\frac{dT}{dx} \tag{11}$$

where $\dot{\varphi}_x$ is the heat flux. It has units of $\frac{J}{s \cdot m^2} = \frac{W}{m^2}$. Thus, the units of k are $\frac{W}{m \cdot K}$.

Equation (11) is restricted to the situation in which heat flows in the x-direction only. In the general case in which heat flows in all three coordinate directions, the total heat flux is obtained by vector addition of adding the fluxes in the coordinate directions. Thus,

$$\dot{\boldsymbol{\varphi}} = \dot{\varphi}_x\mathbf{i} + \dot{\varphi}_y\mathbf{j} + \dot{\varphi}_z\mathbf{k} \tag{12}$$

where $\dot{\boldsymbol{\varphi}}$ is the heat flux vector and $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are unit vectors in the x-, y-, z-directions, respectively.

Each of the component fluxes is given by a one-dimensional Fourier expression as follows:

$$\dot{\varphi}_x = -k\frac{\partial T}{\partial x} \qquad \dot{\varphi}_y = -k\frac{\partial T}{\partial y} \qquad \dot{\varphi}_z = -k\frac{\partial T}{\partial z} \tag{13}$$

Partial derivatives are used here since the temperature now varies in all three directions. Substituting the above expressions for the fluxes into Equation (12) gives:

$$\dot{\boldsymbol{\varphi}} = -k\left(\frac{\partial T}{\partial x}\mathbf{i} + \frac{\partial T}{\partial y}\mathbf{j} + \frac{\partial T}{\partial z}\mathbf{k}\right) \tag{14}$$

The vector in parenthesis is the temperature gradient vector, and is denoted by ∇T. Hence,

$$\dot{\boldsymbol{\varphi}} = -k\nabla T \tag{15}$$

Equation (15) is the three-dimensional form of Fourier's law. It is valid for homogeneous, isotropic materials for which the thermal conductivity is the same in all directions. Fourier's law states that heat flows in the direction of greatest temperature decrease.

5.1.2 The Heat Conduction Equation

The solution of problems involving heat conduction in solids can, in principle, be reduced to the solution of a single differential equation, the *heat conduction equation*. The equation can be derived by making a thermal power balance on a differential volume element in the solid. For the case of conduction in the x-direction only, such a volume element is illustrated in Figure 14.

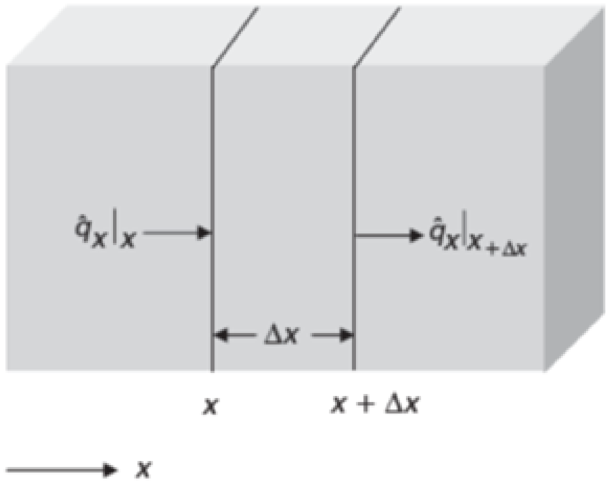

Figure 14: Differential element for 1D heat conduction

The rate at which thermal energy enters the volume element across the face at x is given by the product of the heat flux and the cross-sectional area, $\dot{\varphi_x}|_x \cdot A$. Similarly, the rate at which thermal energy leaves the element across the face at $x + \Delta x$ is $\dot{\varphi_x}|_{x+\Delta x} \cdot A$.

A heat generation term appears in the equation because the balance is made on thermal energy, not total energy. For example, thermal energy may be generated within a solid by an electric current or by decay of a radioactive material.

For a homogeneous heat source of strength $\dot{q}$ *per unit volume*, the net rate of generation is $\dot{q}A\Delta x$. Finally, the rate of accumulation of heat in the material is given by the time derivative of the thermal energy content of the volume element, which is $\rho c(T - T_{ref})A\Delta x$, where T_{ref} is an arbitrary reference temperature. Thus, the balance equation becomes:

$$\left(\dot{\varphi_x}|_x - \dot{\varphi_x}|_{x+\Delta x}\right) A + \dot{q}A\Delta x = \rho c \frac{\partial T}{\partial t} A\Delta x \tag{16}$$

It has been assumed here that the density, ρ, and heat capacity, c, are constant.

Dividing by $A\Delta x$ and taking the limit as $\Delta x \to 0$ yields:

$$\rho c \frac{\partial T}{\partial t} = -\frac{\partial \dot{\varphi_x}}{\partial x} + \dot{q} \tag{17}$$

Using Fourier's law as given by Equation (11), the balance equation becomes:

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\frac{k\partial T}{\partial x}\right) + \dot{q} \tag{18}$$

When conduction occurs in all three coordinate directions, the balance equation contains y- and z-derivatives analogous to the x-derivative. The balance equation then becomes:

$$\rho c \frac{\partial T}{\partial x} = \frac{\partial}{\partial x}\left(\frac{k\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{k\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{k\partial T}{\partial z}\right) + \dot{q} \tag{19}$$

When k is constant, it can be taken outside the derivatives and Equation (19) can be written as:

$$\frac{\rho c}{k}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{q}}{k} \tag{20}$$

or

$$\frac{1}{\alpha}\frac{\partial T}{\partial t} = \nabla^2 T + \frac{\dot{q}}{k} \tag{21}$$

where $\alpha \equiv k/\rho c$ is the *thermal diffusivity* and ∇^2 is the Laplacian operator. The thermal diffusivity has units of m^2/s.

5.2 Convection: Newton's Law of cooling

When a solid is *immersed* in a fluid or atmospheric gas, heat transfer on the interface occurs by convection. This phenomenon is governed by Newton's Law of cooling:

"The rate of heat lost by a body is directly proportional to the temperature difference of a body and its surroundings"

$$\dot{Q}_x = -hA\Delta T \tag{22}$$

5.3 Radiation

5.4 Approximations: A Simplified Model

In building physics, it is often assumed that Fourier's Law is valid in the form of Eq. (9). This can be done under the condition that

$$\nabla^2 T \equiv 0 \rightarrow \quad \frac{\partial T}{\partial \mathbf{r}} = constant \tag{23}$$

5.5 Lumped-element matrix representation

We take the 2R-2C lumped-element model from Section 2:

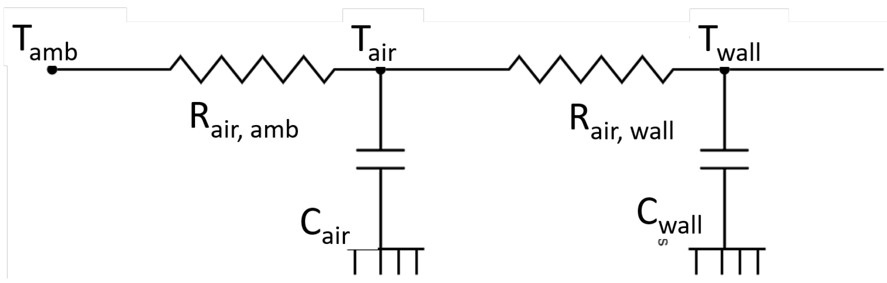

Figure 15: 2R-2C house model revisited

The differential equations are:

$$\begin{aligned} C_{air}\frac{dT_{air}}{dt} &= \frac{T_{amb} - T_{air}}{R_{air,amb}} + \frac{T_{wall} - T_{air}}{R_{air,wall}} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ C_{wall}\frac{dT_{wall}}{dt} &= \frac{T_{air} - T_{wall}}{R_{air,wall}} + \dot{Q}_{solar,wall} \end{aligned} \tag{24}$$

Writing out the differential equations in the classical notation:

$$C_{air}\frac{dT_{air}}{dt} = \left[\frac{-1}{R_{air,amb}} + \frac{-1}{R_{air,wall}}\right] \cdot T_{air} + \frac{1}{R_{air,wall}} \cdot T_{wall} + \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air}$$

$$C_{wall}\frac{dT_{wall}}{dt} = \frac{1}{R_{air,wall}} \cdot T_{air} + \frac{-1}{R_{air,wall}} \cdot T_{wall} + \dot{Q}_{solar,wall} \tag{25}$$

The differential equations can be written in matrix notation as:

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = -\mathbf{K} \cdot \boldsymbol{\theta} + \dot{\mathbf{q}} \tag{26a}$$

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} + \mathbf{K} \cdot \boldsymbol{\theta} = \dot{\mathbf{q}} \tag{26b}$$

with:

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} \tag{27}$$

$$\mathbf{K} \cdot \boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{air,amb}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} \tag{28}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \dot{Q}_{solar,wall} \end{bmatrix} \tag{29}$$

Written out, the differential equation according to (94) becomes:

$$\begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} = \begin{bmatrix} \frac{-1}{R_{air,amb}} + \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \\ \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} + \begin{bmatrix} \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \dot{Q}_{solar,wall} \end{bmatrix} \tag{30}$$

In the alternative notation:

$$\begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} + \begin{bmatrix} \frac{1}{R_{air,amb}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \dot{Q}_{solar,wall} \end{bmatrix} \tag{31}$$

The lumped-element equations above are systems of *first-order ordinary differential equations* (ODE). The first order derivative is with respect to *time*. The (silent) assumption that heat conduction within the air and the wall of the previous 2R-2C model is *faster* than the exchange of heat at the *interfaces* between air and wall and air and ambient surroundings has replaced all spatial information from the *second-order partial differential equations* (PDE) that govern conductive heat transport *within* materials.

Therefore, the lumped-element equations can be solved by:

- the odexxx in Matlab., preferrably ode45.
- the state-space module in Simulink, after conversion to a state-space representation.
- the scipy.integrate.solve_ivp function in Python. In older code, scipy.integrate.odeint is still encountered.
- in C++ several options exist, similar to the options in Python.

The routines in Matlab, Simulink and Python need a *model function* that provides the vector $\dot{\boldsymbol{\theta}}$ for evaluation at any time instance chosen by the algorithm. The equations (94) then should be cast in the following form by left multiplication with $\mathbf{C^{-1}}$.

$$\mathbf{C}^{-1} \cdot \mathbf{C} \cdot \dot{\boldsymbol{\theta}} = -\mathbf{C}^{-1} \cdot \mathbf{K} \cdot \boldsymbol{\theta} + \mathbf{C}^{-1} \cdot \dot{\mathbf{q}} \tag{32a}$$

$$\dot{\boldsymbol{\theta}} = -\mathbf{C}^{-1} \cdot \mathbf{K} \cdot \boldsymbol{\theta} + \mathbf{C}^{-1} \cdot \dot{\mathbf{q}} \tag{32b}$$

Since $\mathbf{C}$ is a *diagonal* matrix with positive elements only, its inverse exists and contains the reciprocal elements on its diagonal:

$$\mathbf{C^{-1}} = \begin{bmatrix} \frac{1}{C_{air}} & 0 \\ 0 & \frac{1}{C_{wall}} \end{bmatrix} \tag{33}$$

This provides the division by the lumped thermal capacitances of the air and wall compartments in the model, necessary for the calculating the derivative vector $\dot{\boldsymbol{\theta}}$ in the model functions.

5.6 Extension of the method to larger lumped-element networks

Take a house model with two stories. Each level in the building is described with a 2R-2C model. Heat transfer occurs between the ground floor and the 1st floor.

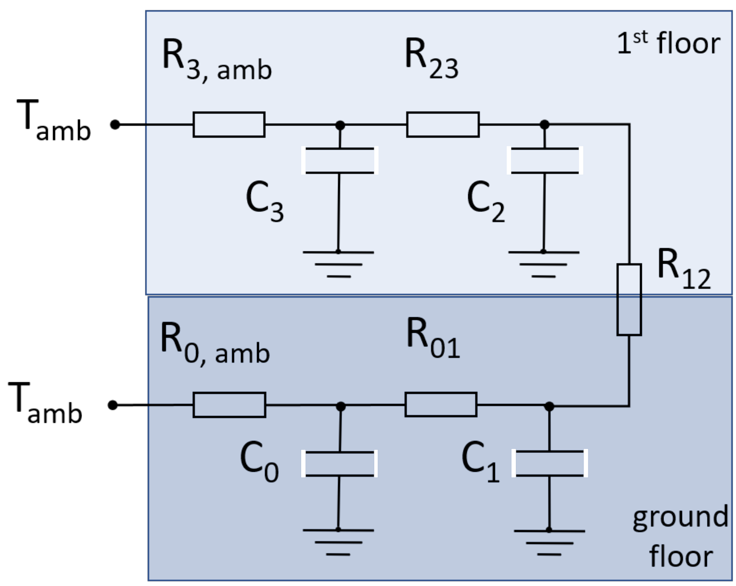

Figure 16: 5R-4C house model

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 & 0 \\ 0 & C_1 & 0 & 0 \\ 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & C_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \\ \frac{dT_3}{dt} \end{bmatrix} \tag{34}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} & \frac{-1}{R_{01}} & 0 & 0 \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} + \frac{1}{R_{12}} & \frac{-1}{R_{12}} & 0 \\ 0 & \frac{-1}{R_{12}} & \frac{1}{R_{12}} + \frac{1}{R_{23}} & \frac{-1}{R_{23}} \\ 0 & 0 & \frac{-1}{R_{23}} & \frac{1}{R_{3,amb}} + \frac{1}{R_{23}} \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{35}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{heat,0} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \\ \dot{Q}_{solar,2} \\ \frac{1}{R_{3,amb}} \cdot T_{amb} + \dot{Q}_{heat,3} + \dot{Q}_{int,3} + \dot{Q}_{solar,3} \end{bmatrix} \tag{36}$$

5.7 Alternative representation of 5R-4C model

The 5R4C model of the previous section can be built from two 2R2C models, one for the ground floor and one for the first floor. The thermal resistance between the construction nodes of the ground and first floor is then added, R_{13} in the figure:

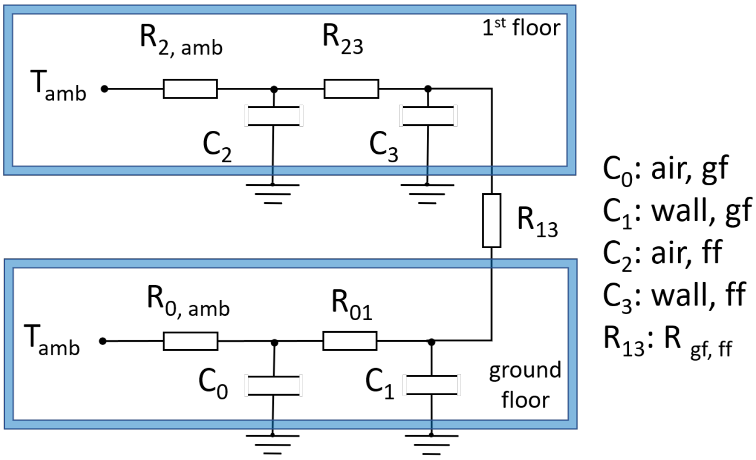

Figure 17: 5R-4C house model, alternative representation

As can be seen in the matrices below, adding R_{13} to the ground floor and first floor "chains" results in a non-symmetric matrix. It has to determined if this disadvantage outweighs the benefit of adding "chains".

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 & 0 \\ 0 & C_1 & 0 & 0 \\ 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & C_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \\ \frac{dT_3}{dt} \end{bmatrix} \tag{37}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} & \frac{-1}{R_{01}} & 0 & 0 \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} + \frac{1}{R_{13}} & 0 & \frac{-1}{R_{13}} \\ 0 & 0 & \frac{1}{R_{2,amb}} + \frac{1}{R_{23}} & \frac{-1}{R_{23}} \\ 0 & \frac{-1}{R_{13}} & \frac{-1}{R_{23}} & \frac{1}{R_{23}} + \frac{1}{R_{13}} \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{38}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{heat,0} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \\ \frac{1}{R_{2,amb}} \cdot T_{amb} + \dot{Q}_{heat,2} + \dot{Q}_{int,2} + \dot{Q}_{solar,2} \\ \dot{Q}_{solar,3} \end{bmatrix} \tag{39}$$

Renumbering restores the matrices to a symmetric representation:

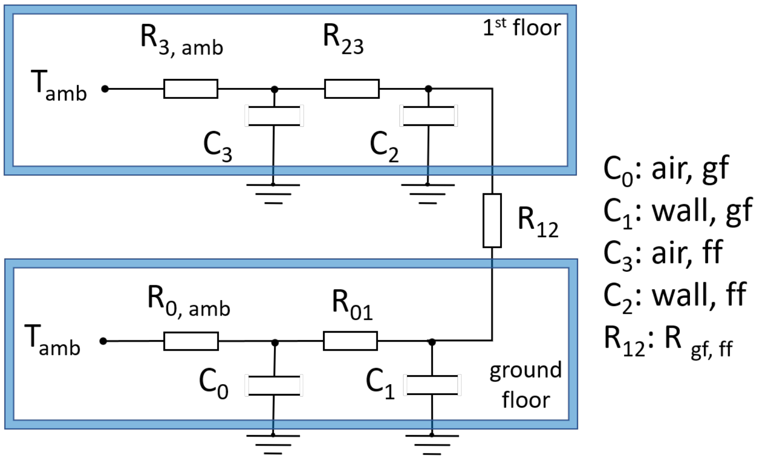

Figure 18: 5R-4C house model, alternative representation, renumbered

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 & 0 \\ 0 & C_1 & 0 & 0 \\ 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & C_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \\ \frac{dT_3}{dt} \end{bmatrix} \tag{40}$$

$$\mathbf{K} \cdot \boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} & \frac{-1}{R_{01}} & 0 & 0 \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} + \frac{1}{R_{12}} & \frac{-1}{R_{12}} & 0 \\ 0 & \frac{-1}{R_{12}} & \frac{1}{R_{23}} + \frac{1}{R_{12}} & \frac{-1}{R_{23}} \\ 0 & 0 & \frac{-1}{R_{23}} & \frac{1}{R_{3,amb}} + \frac{1}{R_{23}} \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{41}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{heat,0} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \\ \dot{Q}_{solar,2} \\ \frac{1}{R_{3,amb}} \cdot T_{amb} + \dot{Q}_{heat,3} + \dot{Q}_{int,3} + \dot{Q}_{solar,3} \end{bmatrix} \tag{42}$$

5.8 2R-2C model with buffervessel

The "air" and "wall" nodes of the 2R2C model can be extended with "radiator" node. The radiator has a finite heat capacity of itself. Instead of a thermal resistance, the radiator heat exchange in W/K is entered in the model. The radiator emits heat to the "air" node only. In its turn, the radiator is fed from a "buffervessel" node. The buffervessel loses heat to the radiator and is heated up by a gas boiler or alternatively a heat pump. The gas boiler does not heat the house directly, as was the case in the simplest model. A schematic view is given in Fig. ??.

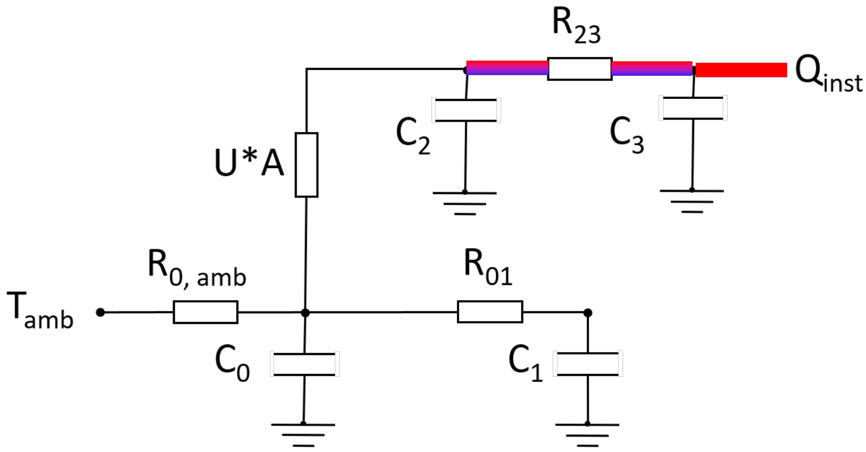

Figure 19: 2R-2C house model with radiator and buffer vessel

The differential equations for heat transport in the model of Fig. **??** are:

$$\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \frac{T_{outdoor}-T_{air}}{R_{air_outdoor}}+\frac{T_{wall}-T_{air}}{R_{air_wall}}+U_{rad}\cdot A_{rad}\cdot(T_{return}-T_{air})+\dot{Q}_{internal}+\dot{Q}_{solar,0}\\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{T_{air}-T_{wall}}{R_{air_wall}}+\dot{Q}_{solar,1}\\
C_{rad}\frac{dT_{return}}{dt} &= \dot{m}\cdot c_{p,water}\cdot(T_{buffer}-T_{return})+U_{rad}\cdot A_{rad}\cdot(T_{air}-T_{return})\\
C_{buffer}\frac{dT_{buffer}}{dt} &= \dot{m}\cdot c_{p,water}\cdot(T_{return}-T_{buffer})+\dot{Q}_{inst}\\
\frac{dE}{dt} &= \dot{Q}_{inst}
\end{aligned} \tag{43}$$

A fifth equation, integrating the heat source energy is sometimes added. Re-arranging the terms in the equation gives:

$$\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \left[\frac{-1}{R_{air_outdoor}}+\frac{-1}{R_{air_wall}}+-1\cdot U_{rad}\cdot A_{rad}\right]\cdot T_{air}+\frac{T_{wall}}{R_{air_wall}}+U_{rad}\cdot A_{rad}\cdot T_{return}+\\
&\quad \frac{T_{outdoor}}{R_{air_outdoor}}+\dot{Q}_{internal}+\dot{Q}_{solar,0}\\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{1}{R_{air_wall}}\cdot T_{air}+\frac{-1}{R_{air_wall}}\cdot T_{wall}+\dot{Q}_{solar,1}\\
C_{rad}\frac{dT_{return}}{dt} &= U_{rad}\cdot A_{rad}\cdot T_{air}+[-U_{rad}\cdot A_{rad}-\dot{m}\cdot c_{p,water}]\cdot T_{return}+\dot{m}\cdot c_{p,water}\cdot T_{buffer}\\
C_{buffer}\frac{dT_{buffer}}{dt} &= \dot{m}\cdot c_{p,water}\cdot T_{return}-\dot{m}\cdot c_{p,water}\cdot T_{buffer}+\dot{Q}_{heat,3}\\
\frac{dE}{dt} &= \dot{Q}_{inst}
\end{aligned} \tag{44}$$

Conversion of the equations to a matrix equation yields:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 & 0\\ 0 & C_1 & 0 & 0\\ 0 & 0 & C_2 & 0\\ 0 & 0 & 0 & C_3 \end{bmatrix}\cdot\begin{bmatrix}\frac{dT_0}{dt}\\ \frac{dT_1}{dt}\\ \frac{dT_2}{dt}\\ \frac{dT_3}{dt}\end{bmatrix} \tag{45}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} + U\cdot A & \frac{-1}{R_{01}} & -U\cdot A & 0 \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} & 0 & 0 \\ -U\cdot A & 0 & U\cdot A + \frac{1}{R_{23}} & \frac{-1}{R_{23}} \\ 0 & 0 & \frac{-1}{R_{23}} & \frac{1}{R_{23}} \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{46}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,0} \\ 0 \\ \dot{Q}_{heat,3} \end{bmatrix} \tag{47}$$

5.9 2R-2C model with radiator only

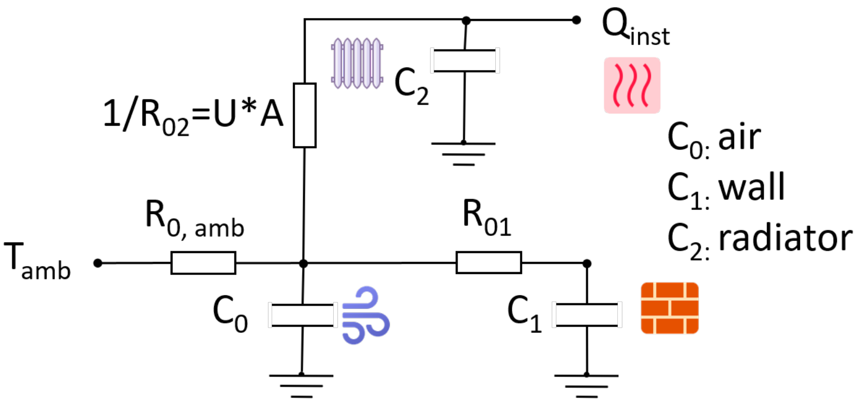

Figure 20: 2R-2C house model with radiator only

The rate of heat transfer from a radiator to the ambient(room) air can be calculated as follows [19]:

$$\begin{aligned} P &= P_{50} \cdot \left[\Delta T_{LMTD} \cdot \frac{1}{49.32} \right]^n \\ \Delta T_{LMTD} &= \frac{T_{inlet} - T_{return}}{\ln \frac{T_{inlet} - T_{ambient}}{T_{return} - T_{ambient}}} \\ n &= 1.33 \end{aligned} \tag{48}$$

This is sometimes simplified to:

$$\begin{aligned} P &= U \cdot A \cdot \Delta T_{LMTD} \\ \Delta T_{LMTD} &= \frac{T_{inlet} - T_{return}}{\ln \frac{T_{inlet} - T_{ambient}}{T_{return} - T_{ambient}}} \end{aligned} \tag{49}$$

or simplified to [20, 21]:

$$\begin{aligned} P &= K_m \cdot \Delta T^n \\ \Delta T &= \frac{T_{inlet} + T_{return}}{2} - T_{ambient} \end{aligned} \tag{50}$$

The differential equations for heat transport in the model of Fig. 20 are:

$$\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \frac{T_{outdoor}-T_{air}}{R_{air_outdoor}} + \frac{T_{wall}-T_{air}}{R_{air_wall}} + U_{rad}\cdot A_{rad}\cdot (T_{rad}-T_{air}) + \dot{Q}_{internal} + \dot{Q}_{solar,0} \\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{T_{air}-T_{wall}}{R_{air_wall}} + \dot{Q}_{solar,1} \\
C_{rad}\frac{dT_{rad}}{dt} &= \dot{Q}_{inst} + U_{rad}\cdot A_{rad}\cdot (T_{air}-T_{rad})
\end{aligned} \tag{51}$$

Re-arranging the terms in the equation gives:

$$\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \left[\frac{-1}{R_{air_outdoor}} + \frac{-1}{R_{air_wall}} + -1\cdot U_{rad}\cdot A_{rad}\right]\cdot T_{air} + \frac{T_{wall}}{R_{air_wall}} + U_{rad}\cdot A_{rad}\cdot T_{rad} + \\
&\quad \frac{T_{outdoor}}{R_{air_outdoor}} + \dot{Q}_{internal} + \dot{Q}_{solar,0} \\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{1}{R_{air_wall}}\cdot T_{air} + \frac{-1}{R_{air_wall}}\cdot T_{wall} + \dot{Q}_{solar,1} \\
C_{rad}\frac{dT_{rad}}{dt} &= U_{rad}\cdot A_{rad}\cdot T_{air} - U_{rad}\cdot A_{rad}\cdot T_{rad} + \dot{Q}_{heat,2}
\end{aligned} \tag{52}$$

Conversion of the equations to a matrix equation yields:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 \\ 0 & C_1 & 0 \\ 0 & 0 & C_2 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \end{bmatrix} \tag{53}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} + U\cdot A & \frac{-1}{R_{01}} & -U\cdot A \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} & 0 \\ -U\cdot A & 0 & U\cdot A \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \end{bmatrix} \tag{54}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}}\cdot T_{amb} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \\ \dot{Q}_{heat,2} \end{bmatrix} \tag{55}$$

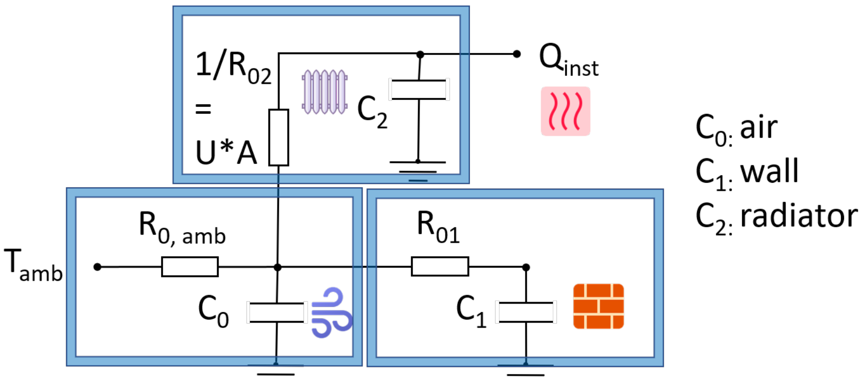

Figure 21: 2R-2C house model with radiator in 3 chains

Starting with the basic 2R2C model we write down the matrices. Note that the heat source for the house is omitted at first. Solar energy entering the house is partitioned between air and wall, Heat generated due to the presence and activities of inhabitants is added to the air node:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 \\ 0 & C_1 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \end{bmatrix} \tag{56}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} & \frac{-1}{R_{01}} \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \tag{57}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \end{bmatrix} \tag{58}$$

As a third link in the chain, a radiator is added, with a heat capacity C_{rad} and a heat delivery $U \cdot A \cdot (T_{rad} - T_{air})$ to the air node. The heat source $\dot{Q}_{inst}$ is now connected to the radiator.

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_0 & 0 & 0 \\ 0 & C_1 & 0 \\ 0 & 0 & C_2 \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_0}{dt} \\ \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \end{bmatrix} \tag{59}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{0,amb}} + \frac{1}{R_{01}} + U \cdot A & \frac{-1}{R_{01}} & -U \cdot A \\ \frac{-1}{R_{01}} & \frac{1}{R_{01}} & 0 \\ -U \cdot A & 0 & U \cdot A \end{bmatrix} \cdot \begin{bmatrix} T_0 \\ T_1 \\ T_2 \end{bmatrix} \tag{60}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{0,amb}} \cdot T_{amb} + \dot{Q}_{int,0} + \dot{Q}_{solar,0} \\ \dot{Q}_{solar,1} \\ \dot{Q}_{heat,2} \end{bmatrix} \tag{61}$$

In this example, it becomes visible (in red) that the rank of the C- and K-matrix, and the $\dot{q}$-vector is extended by 1. The heat capacity of the radiator is included as an extra *diagonal* element in the C-matrix. The heat delivery from the radiator to the indoor air is added to or subtracted from the 00, 22, 02 and 20 elements of th K-matrix, so that it remains a *symmetric* matrix. The heater is connected to the radiator, represented by element 2 of the $\dot{q}$-vector.

5.10 2R2C revisited: 2R3C

The 2R2C model as represented in 10 treats the node of the outside temperature (T_{amb}) differently from the other nodes, T_{air} and T_{walls}. This representation is inconsistent, and actually incomplete. Implicitly, the model links a source/sink to the node that controls the outdoor temperature. In literature, one can find models in which this source has been made explicit, such as in [22]. It seems that this representation has been lost over time.

In order to complete the analogy with the other nodes in the model we can connect an additional capacitor (C_{amb}). The capacity will be tending to infinity, as we assume the outside temperature does not change due to heat exchange with the house.

o:figure uding a rce and acitor

Adding the capacitor and the source also will change the equations. Actually, it results in a more structured set of equations. The equations will be as follows:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{amb} & 0 & 0 \\ 0 & C_{air} & 0 \\ 0 & 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{amb}}{dt} \\ \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} \tag{62}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{amb,air}} & \frac{-1}{R_{amb,air}} & 0 \\ \frac{-1}{R_{amb,air}} & \frac{1}{R_{amb,air}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ 0 & \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{amb} \\ T_{air} \\ T_{wall} \end{bmatrix} \tag{63}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{Q}_{amb} \\ \dot{Q}_{air} \\ \dot{Q}_{wall} \end{bmatrix} \tag{64}$$

In the equations we now see that the matrix K represents the interaction between the different heat capacities. The off-diagonal elements are equal to (minus) conductance factor $\frac{-1}{R}$ between the respective connected nodes. The structure of K is such that the sum over the rows will always be zero, where the diagonal elements equal the negative sum of the off-diagonal elements.

The vector $\dot{q}$ contains all heat sources (and sinks).

Generalizing the idea above, alternative models can be easily constructed using an underlying graph. In the graph each node is labeled with a heat capacity C_i, and temperature T_i. Nodes i and j can be connected by an edge labeled with $R_{i,j}$, where $\frac{1}{R_{i,j}}$ represents the heat conductance between the two nodes. The K-matrix is the connectivity matrix of the graph, where $K_{i,j} = \frac{-1}{R_{i,j}}$. The diagonal elements, Ki, i are set such that the sums over the rows will be equal to zero.

Additionally, each node can be connected to a source (or sink). Two types of sources are available. A "temperature source" represents a source that will keep the temperature of the connected node constant. This source type can be used to set the ambient temperature.

A heat source represents a source that will provide a continuous constant energy flow into the node. This source type can be used to represent the inflow of energy by for example the sun.

5.10.1 example: 2R-2C house with buffer

5.11 3R2C model

For an apartment building, the simplest model has two nodes with a finite heat capacity, the interior and the building construction. Both nodes have a finite thermal resistance to the ambient environment. Finally there is a thermal resistance between the nodes. Graphically, the model is represented by Figure 22

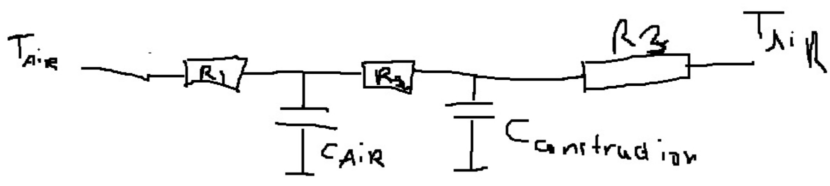

Figure 22: 3R-2C house model

The differential equations are:

$$
\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \frac{T_{amb}-T_{air}}{R_{air,amb}} + \frac{T_{wall}-T_{air}}{R_{air,wall}} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{T_{air}-T_{wall}}{R_{air,wall}} + \frac{T_{amb}-T_{wall}}{R_{wall,amb}} + \dot{Q}_{solar,wall}
\end{aligned}
\tag{65}
$$

Writing out the differential equations in the classical notation:

$$
\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \left[\frac{-1}{R_{air,amb}} + \frac{-1}{R_{air,wall}}\right]\cdot T_{air} + \frac{1}{R_{air,wall}}\cdot T_{wall} + \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{1}{R_{air,wall}}\cdot T_{air} + \left[\frac{-1}{R_{wall,amb}} + \frac{-1}{R_{air,wall}}\right]\cdot T_{wall} + \frac{1}{R_{wall,amb}}\cdot T_{amb} + \dot{Q}_{solar,wall}
\end{aligned}
\tag{66}
$$

The differential equations can be written in matrix notation as:

$$
\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = -\mathbf{K}\cdot\boldsymbol{\theta} + \dot{\mathbf{q}} \tag{67a}
$$

$$
\mathbf{C}\cdot\dot{\boldsymbol{\theta}} + \mathbf{K}\cdot\boldsymbol{\theta} = \dot{\mathbf{q}} \tag{67b}
$$

with:

$$
\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} \tag{68}
$$

$$
\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{air,amb}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{air,wall}} & \frac{1}{R_{wall,amb}} + \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} \tag{69}
$$

$$
\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \frac{1}{R_{wall,amb}}\cdot T_{amb} + \dot{Q}_{solar,wall} \end{bmatrix} \tag{70}
$$

Written out, the differential equation according to (94) becomes:

$$
\begin{aligned}
\begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} = \begin{bmatrix} \frac{-1}{R_{air,amb}} + \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \\ \frac{1}{R_{air,wall}} & \frac{-1}{R_{wall,amb}} + \frac{-1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} + \\
\begin{bmatrix} \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{solar,wall} \end{bmatrix}
\end{aligned}
\tag{71}
$$

In the alternative notation:

$$\begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} + \begin{bmatrix} \frac{1}{R_{air,amb}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{air,wall}} & \frac{1}{R_{wall,amb}} + \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} + \dot{Q}_{int,air} + \dot{Q}_{solar,air} \\ \frac{1}{R_{wall,amb}} \cdot T_{amb} + \dot{Q}_{solar,wall} \end{bmatrix} \tag{72}$$

it is clear that in this example, where multiple nodes in the thermal network are connected to the ambient surroundings, the approach of Section 5.10 becomes more adventageous:

5.12 3R3C model

The previous 3R2C model representation necessitates an *ad hoc* term in the heat supply vector $\dot{\mathbf{q}}$. Analogous to Section 5.10, we can include the ambient surroundings as a (large) heat capacity into the model. This will change the 3R2C model into a 3R3C model. The equations become:

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{amb} & 0 & 0 \\ 0 & C_{air} & 0 \\ 0 & 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{amb}}{dt} \\ \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} \tag{73}$$

For $\mathbf{K}$ we can start with filling out the non-diagonal symmetic matrix elements:

$$\mathbf{K} = \begin{bmatrix} 0 & \frac{-1}{R_{amb,air}} & \frac{-1}{R_{amb,wall}} \\ \frac{-1}{R_{amb,air}} & 0 & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{amb,wall}} & \frac{-1}{R_{air,wall}} & 0 \end{bmatrix} \tag{74}$$

Then we can complete the diagonal elements, so that the sum over each row becomes zero:

$$\mathbf{K} \cdot \boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{amb,air}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{amb,air}} & \frac{-1}{R_{amb,wall}} \\ \frac{-1}{R_{amb,air}} & \frac{1}{R_{amb,air}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{amb,wall}} & \frac{-1}{R_{air,wall}} & \frac{1}{R_{amb,wall}} + \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{amb} \\ T_{air} \\ T_{wall} \end{bmatrix} \tag{75}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{Q}_{amb} \\ \dot{Q}_{air} \\ \dot{Q}_{wall} \end{bmatrix} \tag{76}$$

5.13 Coupling the housemodel elements

The housemodel is to be extended with modular elements representing the installations that supply the heat demanded by the building. Each subsystem contributes its own set of differential equations to the total system. In Fig. 23, the subsystems are indicated with a color code.

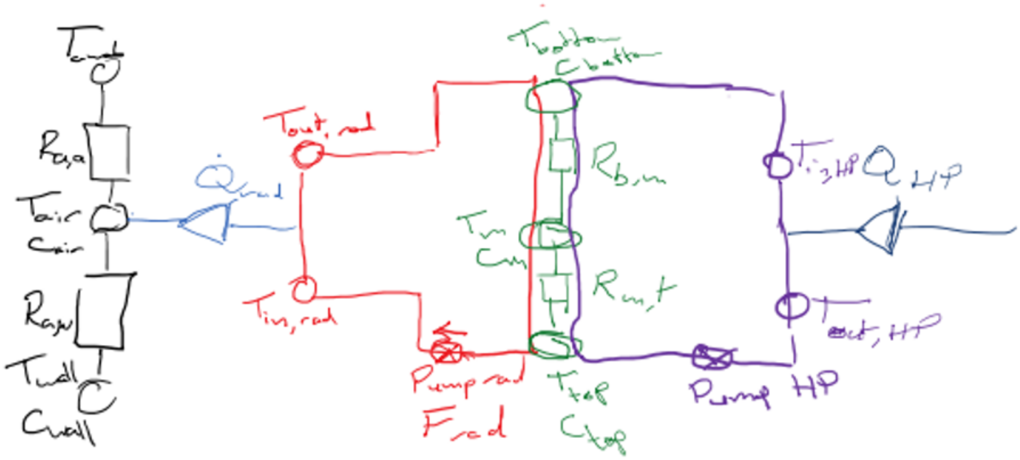

Figure 23: Grand Model

The building itself, in black, generates the differential equations:

$$
\begin{aligned}
T_{air}: \quad C_{air}\frac{dT_{air}}{dt} &= \frac{T_{amb}-T_{air}}{R_{air,amb}} + \frac{T_{wall}-T_{air}}{R_{air,wall}} + \dot{Q}_{rad,air} \\
T_{wall}: \quad C_{wall}\frac{dT_{wall}}{dt} &= \frac{T_{air}-T_{wall}}{R_{air,wall}}
\end{aligned}
\tag{77}
$$

T_{amb}: given as piecewise constant function, in interval $[t_i, t_{i+1}]$, $T_{amb} = T_{amb,i}$

The radiator element, coupled to the node T_{air}, transfers heat to the building at a rate $\dot{Q}_{rad}$. It has a feed temperature T_{feed} and a return temperature T_{return}. The radiator is modeled as a "cross-flow" heat exchanger, obeying the "radiator equation":

$$
\begin{aligned}
\dot{Q}_{rad}: \quad & \dot{Q}_{rad} = C_{rad}\cdot(\Delta T_{LMTD})^n, \text{ with } \Delta T_{LMTD} = \frac{T_{feed}-T_{return}}{\ln\left(\frac{T_{feed}-T_{air}}{T_{return}-T_{air}}\right)} \\
\text{also, :} \quad & \dot{Q}_{rad} = F_{rad}\cdot c_w\cdot(T_{feed}-T_{return})
\end{aligned}
\tag{78}
$$

when T_{feed}, T_{air} and F_{rad} are known, we have two equations with two unknowns $\dot{Q}_{rad}$ and T_{return}.

Question: when do you solve this system? Do you need to solve this within the time interval $[t_i, t_{i+1}]$?

Further equations:

$$
\begin{aligned}
T_{feed} &= T_{top} \\
T_{return} &\text{ should follow from equations above.}
\end{aligned}
\tag{79}
$$

$$T_{top}\colon \quad C_{top}\frac{dT_{top}}{dt} = \frac{T_{top}-T_{mid}}{-R_{mid,top}} + F_{HP}\cdot c_w\cdot(T_{HP,out}-T_{top}) + \max(F_{rad}-F_{HP},0)\cdot c_w\cdot(T_{mid}-T_{top})$$

$$\begin{aligned}
T_{mid}\colon \quad C_{mid}\frac{dT_{mid}}{dt} &= \frac{T_{top}-T_{mid}}{R_{mid,top}} + \frac{T_{mid}-T_{bot}}{-R_{bot,mid}} + \\
&\max(F_{HP}-F_{rad},0)\cdot c_w\cdot(T_{top}-T_{mid}) + \max(F_{rad}-F_{HP},0)\cdot c_w\cdot(T_{mid}-T_{bot})
\end{aligned}$$

$$\begin{aligned}
T_{bot}\colon \quad C_{bot}\frac{dT_{bot}}{dt} &= \frac{T_{mid}-T_{bot}}{R_{bot,mid}} + F_{rad}\cdot c_w\cdot(T_{return}-T_{bot}) + \\
&\max(F_{HP}-F_{rad},0)\cdot c_w\cdot(T_{mid}-T_{bot})
\end{aligned} \tag{80}$$

$$\begin{aligned}
T_{HP,in} &= T_{bot} \\
T_{HP,out}\colon \quad \dot{Q}_{HP} &= F_{HP}\cdot c_w\cdot(T_{HP,out}-T_{HP,in}\dot{Q}_{HP} \quad = f(T_{HP,in},T_{HP,out},T_{src,in},T_{src,out}))
\end{aligned} \tag{81}$$

heat pump function? Also here the question is: when to solve this equation?

Writing out the differential equations in the classical notation:

$$\begin{aligned}
C_{air}\frac{dT_{air}}{dt} &= \left[\frac{-1}{R_{air,amb}} + \frac{-1}{R_{air,wall}}\right]\cdot T_{air} + \frac{1}{R_{air,wall}}\cdot T_{wall} + \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{heat,air} \\
C_{wall}\frac{dT_{wall}}{dt} &= \frac{1}{R_{air,wall}}\cdot T_{air} + \frac{-1}{R_{air,wall}}\cdot T_{wall}
\end{aligned} \tag{82}$$

The differential equations of the 2R-2C house model (in black) be written in matrix notation as:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = -\mathbf{K}\cdot\boldsymbol{\theta} + \dot{\mathbf{q}} \tag{83a}$$

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} + \mathbf{K}\cdot\boldsymbol{\theta} = \dot{\mathbf{q}} \tag{83b}$$

with:

$$\mathbf{C}\cdot\dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{air} & 0 \\ 0 & C_{wall} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{air}}{dt} \\ \frac{dT_{wall}}{dt} \end{bmatrix} \tag{84}$$

$$\mathbf{K}\cdot\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{air,amb}} + \frac{1}{R_{air,wall}} & \frac{-1}{R_{air,wall}} \\ \frac{-1}{R_{air,wall}} & \frac{1}{R_{air,wall}} \end{bmatrix} \cdot \begin{bmatrix} T_{air} \\ T_{wall} \end{bmatrix} \tag{85}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{air,amb}}\cdot T_{amb} + \dot{Q}_{heat,air} \\ 0 \end{bmatrix} \tag{86}$$

The routines in Matlab, Simulink and Python need a *model function* that provides the vector $\dot{\boldsymbol{\theta}}$ for evaluation at any time instance chosen by the algorithm. The equations (94) then should be cast in the following form by left multiplication with $\mathbf{C^{-1}}$.

$$\mathbf{C}^{-1} \cdot \mathbf{C} \cdot \dot{\boldsymbol{\theta}} = -\mathbf{C}^{-1} \cdot \mathbf{K} \cdot \boldsymbol{\theta} + \mathbf{C}^{-1} \cdot \dot{\mathbf{q}} \tag{87a}$$

$$\dot{\boldsymbol{\theta}} = -\mathbf{C}^{-1} \cdot \mathbf{K} \cdot \boldsymbol{\theta} + \mathbf{C}^{-1} \cdot \dot{\mathbf{q}} \tag{87b}$$

Since **C** is a *diagonal* matrix with positive elements only, its inverse exists and contains the reciprocal elements on its diagonal:

$$\mathbf{C^{-1}} = \begin{bmatrix} \frac{1}{C_{air}} & 0 \\ 0 & \frac{1}{C_{wall}} \end{bmatrix} \tag{88}$$

This provides the division by the lumped thermal capacitances of the air and wall compartments in the model, necessary for the calculating the derivative vector $\dot{\boldsymbol{\theta}}$ in the model functions.

Radiator

$$\begin{aligned} C_{feed}\frac{dT_{feed}}{dt} &= F_{rad} \cdot c_w \cdot (T_{top} - T_{feed}) \\ C_{return}\frac{dT_{feed}}{dt} &= \end{aligned} \tag{89}$$

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{feed} & 0 \\ 0 & C_{return} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{feed}}{dt} \\ \frac{dT_{return}}{dt} \end{bmatrix} \tag{90}$$

$$\mathbf{K} \cdot \boldsymbol{\theta} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} T_{feed} \\ T_{return} \end{bmatrix} \tag{91}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} 0 \\ \dot{Q}_{rad,air} \end{bmatrix} \tag{92}$$

5.14 Buffer vessel

The buffer vessel model is the general model for a ”stratified (layered) tank”. In order to avoid extensive convection in the tank, the model assumes that addition of hot water from the heat source *and* extraction of hot water to the sink (demand) occurs in the *top* layer. Return flow from the sink is to the *bottom* layer of the vessel.

The differential equations [80] can be rewritten as:

$$C_{top}\frac{dT_{top}}{dt} = \frac{-1}{R_{mid,top}}(T_{top} - T_{mid}) + \max(F_{rad} - F_{HP}, 0) \cdot (T_{mid} - T_{top})$$
$$+ F_{HP} \cdot (T_{HP,out} - T_{top})$$

$$C_{mid}\frac{dT_{mid}}{dt} = \frac{1}{R_{mid,top}}(T_{top} - T_{mid}) + \frac{-1}{R_{bot,mid}}(T_{mid} - T_{bot})$$
$$+ \max(F_{HP} - F_{rad}, 0) \cdot (T_{top} - T_{mid}) + \max(F_{rad} - F_{HP}, 0) \cdot (T_{mid} - T_{bot}) \tag{93}$$

$$C_{bot}\frac{dT_{bot}}{dt} = \frac{1}{R_{bot,mid}}(T_{mid} - T_{bot}) + \max(F_{HP} - F_{rad}, 0) \cdot (T_{mid} - T_{bot})$$
$$+ F_{rad} \cdot (T_{return} - T_{bot})$$

These differential equations can be written in matrix notation as previously, but a *convection* matrix **F** is added:

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} + \mathbf{K} \cdot \boldsymbol{\theta} + \mathbf{F} \cdot \boldsymbol{\theta} = \dot{\mathbf{q}} \tag{94a}$$

with:

$$\mathbf{C} \cdot \dot{\boldsymbol{\theta}} = \begin{bmatrix} C_{top} & 0 & 0 \\ 0 & C_{mid} & 0 \\ 0 & 0 & C_{bot} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_{top}}{dt} \\ \frac{dT_{mid}}{dt} \\ \frac{dT_{bot}}{dt} \end{bmatrix} \tag{95}$$

$$\mathbf{K} \cdot \boldsymbol{\theta} = \begin{bmatrix} \frac{1}{R_{mid,top}} & \frac{-1}{R_{mid,top}} & 0 \\ \frac{-1}{R_{mid,top}} & \frac{1}{R_{mid,top}} + \frac{1}{R_{bot,mid}} & \frac{-1}{R_{bot,mid}} \\ 0 & \frac{-1}{R_{bot,mid}} & \frac{1}{R_{bot,mid}} \end{bmatrix} \cdot \begin{bmatrix} T_{top} \\ T_{mid} \\ T_{bot} \end{bmatrix} \tag{96}$$

$$\mathbf{F} \cdot \boldsymbol{\theta} = \begin{bmatrix} F_{HP,out} + F_{rad} & F_{rad} & 0 \\ F_{rad} & 0 & F_{rad} \\ 0 & F_{rad} & F_{HP} + F_{rad} \end{bmatrix} \cdot \begin{bmatrix} T_{top} \\ T_{mid} \\ T_{bot} \end{bmatrix} \tag{97}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{R_{air,amb}} \cdot T_{amb} + \dot{Q}_{heat,air} \\ 0 \end{bmatrix} \tag{98}$$

F1: [0 1 2 0]

F2: [2 1 0 2]

$$\mathbf{DF_{F1}} = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \tag{99}$$

$$\mathbf{DF_{F2}} = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \tag{100}$$

In each time step, when the flow sizes have been determined by the control algorithms, each directed-flow-matrix is multiplied by its respected flow size in $[\frac{\mathrm{m}^3}{\mathrm{s}}]$. All resulting matrices can then be added together. Assuming a flow of size f_1 and f_2 for the flows $F1$ and $F2$, respectively we now get the matrix **SF**:

$$\mathbf{SF} = f_1 \cdot \mathbf{DF_{F1}} + f_2 \cdot \mathbf{DF_{F2}} = \begin{bmatrix} 0 & f_1 - f_2 & f_2 - f_1 \\ f_2 - f_1 & 0 & f_1 - f_2 \\ f_1 - f_2 & f_2 - f_1 & 0 \end{bmatrix} \tag{101}$$

The heat transfer induced by the flows is only in the direction of the water flow. The correct elements are obtained by taking the $\min(\mathbf{SF}, 0)$, here we mean for each element in **SF** we take the minimum of the respective element and 0. Thus, in the case $f_1 > f_2$ the matrix **SF** will become:

$$\min(\mathbf{SF}, 0) = \begin{bmatrix} 0 & f_1 - f_2 & 0 \\ 0 & 0 & f_1 - f_2 \\ f_1 - f_2 & 0 & 0 \end{bmatrix} \tag{102}$$

Now, the diagonal elements can be computed. The diagonal elements are equal to minus the sum of the off-diagonal elements in its respective row. For the matrix given in equation 127 this results in the flow matrix **F**:

$$\mathbf{F} = \begin{bmatrix} -(f_1 - f_2) & f_1 - f_2 & 0 \\ 0 & -(f_1 - f_2) & f_1 - f_2 \\ f_1 - f_2 & 0 & -(f_1 - f_2) \end{bmatrix} \tag{103}$$

Finally, we need to multiply the resulting flow matrix with the density (ρ_{water}) and the specific heat ($c_{p,water}$), in order to obtain the heat transferred by the water due to the water flows. The resulting matrix can be added to the K matrix as given in equation 123.

$$v_{pump} \cdot A_{pipe} \cdot \rho_{water} \cdot c_{p,water} = \qquad \left[\frac{m}{s} \cdot m^2 \cdot \frac{kg}{m^3} \cdot \frac{J}{kg \cdot K} = \frac{J}{K \cdot s} = \frac{W}{K}\right] \tag{104}$$

5.14.1 Radiator

In a radiator, the heat transport is in good approximation only due to *convection* of a gas (steam) or liquid (water, glycol, brine) For a liquid, the following points of view may be taken: The equations for the radiator element are:

$$\begin{aligned} F_{rad} &= \dot{f} \cdot \rho \cdot c_w \\ F_{rad} &= \dot{m} \cdot c_w \end{aligned} \tag{105}$$

where:

ρ is the density of the liquid in $[kg/m^3]$

c_w is the specific heat capacity of water: $4.2 \cdot 10^3 J/(kg \cdot K)$

$\dot{f}$ is the liquid volume flow in $[m^3/s]$

$\dot{m}$ is the liquid mass flow in $[kg/s]$

The equations for the heat transfer of the radiator are:

$$
\begin{aligned}
\dot{Q}_{rad} - C_{rad} \cdot (\Delta T_{LMTD})^{n} &= 0 \\
\dot{Q}_{rad} - F_{rad} \cdot (T_{feed} - T_{return}) &= 0 \\
\text{with } \Delta T_{LMTD} = \frac{T_{feed} - T_{return}}{\ln\left(\frac{T_{feed}-T_{air}}{T_{return}-T_{air}}\right)}
\end{aligned} \tag{106}
$$

This nonlinear system of equations can be solved for the two unknowns $[\dot{Q}_{rad} \quad T_{return}]$, if input data T_{feed}, T_{amb}, C_{rad} and F_{rad} are provided. A solver needs the function template in Eq. 106, with the unknowns vector as input variable. Furthermore, the Jacobian of the function template has to be calculated. Evaluation of an analytical expression of the partial derivatives in the Jacobian always outperforms numerical derivative calculations. Thus, for the upper equation (function) in set 106:

$$
\begin{aligned}
\frac{\partial f}{\partial \dot{Q}_{rad}} &= 1 \\
\frac{\partial f}{\partial T_{return}} &= -Cn \cdot \frac{\left(\frac{T_1-T_2}{\ln\left(\frac{T_1-T_3}{T_2-T_3}\right)}\right)^{n-1}\left(\frac{T_1-T_2}{T_2-T_3} - \ln\left(\frac{T_1-T_3}{T_2-T_3}\right)\right)}{\ln^2\left(\frac{T_1-T_3}{T_2-T_3}\right)}
\end{aligned} \tag{107}
$$

for the second equation:

$$
\begin{aligned}
\frac{\partial f}{\partial \dot{Q}_{rad}} &= 1 \\
\frac{\partial f}{\partial T_{return}} &= -F_{rad}
\end{aligned} \tag{108}
$$

See: https://www.derivative-calculator.net/.

The Jacobian matrix

$$
\mathbf{J}_{i,j} = \frac{\partial f_i(\mathbf{x})}{\partial x_j}
$$

becomes:

$$
\begin{bmatrix}
\frac{\partial f_1}{\partial \dot{Q}_{rad}} & \frac{\partial f_1}{\partial T_{return}} \\
\frac{\partial f_2}{\partial \dot{Q}_{rad}} & \frac{\partial f_2}{\partial T_{return}}
\end{bmatrix} \tag{109}
$$

with:

$$
\mathbf{x} = \begin{bmatrix} \dot{Q}_{rad} \\ T_{return} \end{bmatrix} \tag{110}
$$

The properties and the calculation of $\mathbf{x}$ is implemented in Python in the Radiator class. The methods of this class are:

- __init__: initializes the class members. Members starting with __* are private members.
- get_lmtd: returns the value of private member __lmtd.
- func_rad: calculates radiator equations and partial derivatives.
- update: uses scipy.optimize.root() to find the roots $\mathbf{x}$ of the radiator equations.

Listing 1: Radiator class

```python
class Radiator:
    """ class for general Radiator object."""
    def __init__(self, exp_rad):
        self.T_feed = None
        self.c_rad = None
        self.exp_rad = exp_rad
        self.c_w = 4.2e3      # [J/kg K]
        self.rho = 1000       # [kg/m^3]
        self.flow = None      # [m^3/s]
        self.F_rad = None      # heat flow in [W/K] = flow * rho * c_w
        self.T_amb = 20.0

        self.q_dot = 0.0
        self.T_ret = None

        self.__denominator = None
        self.__lmtd = None

    def get_lmtd(self):
        return self.__lmtd

    def func_rad(self, x):
        """model function for scipy.optimize.root().

        Args:
            x: vector with unknowns [self.q_dot, self.T_ret]

        Returns:
            f : vector with model functions evaluated at x
            df : Jacobian (partial derivatives of model functions wrt x
        """
        self.__lmtd = LMTD_radiator(T_feed=self.T_feed, T_return=x[1], T_amb=20.0,
            corrfact=1.0)
        # set of nonlinear functions for root finding
        f = [x[0] - (self.c_rad * self.__lmtd ** self.exp_rad),
             x[0] - self.F_rad * (self.T_feed - x[1])]

        h1 = self.c_rad * self.exp_rad
        h1 *= self.__lmtd ** (self.exp_rad - 1.0)

        h2 = (self.T_feed - x[1]) / (x[1] - self.T_amb)
        denominator = np.log(self.T_feed - self.T_amb) - np.log(x[1] - self.T_amb)
        h2 -= denominator

        dTdt = -(h1 * h2) / (denominator * denominator)

        df = np.array([[1.0, dTdt],
                       [1.0, self.F_rad]])
        return f, df

    def update(self):
        """update roots [self.q_dot, self.T_ret] with model function
           using scipy.optimize.root().

        Returns:
            None
        """
        opt_res = root(self.func_rad, [self.q_dot, self.T_ret], jac=True, method='hybr')
        self.q_dot = opt_res.x[0]
        self.T_ret = opt_res.x[1]
```

The Radiator class uses a helper function LMTD_radiator to determine the effective temperature drop:

Listing 2: LMTD_radiator function

```
def LMTD_radiator(T_feed, T_return, T_amb, corrfact=1.0):
    """calculates log mean temperature difference

    representative value in case of varying temperature difference along heat exchanger
    https://checalc.com/solved/LMTD_Chart.html
    Args:
        T_feed:     entry temperature hot fluid or gas
        T_return:    exit temperature hot fluid or gas
        T_amb:    entry temperature cold fluid or gas
        corrfact:    see:     https://checalc.com/solved/LMTD_Chart.html
                              https://cheguide.com/lmtd_charts.html
                              https://excelcalculations.blogspot.com/2011/06/lmtd-correction-factor.html
                              http://fchart.com/ees/heat_transfer_library/heat_exchangers/hs2000.htm
                              https://yjresources.files.wordpress.com/2009/05/4-3-lmtd-with-tutorial.pdf
                              https://www.engineeringtoolbox.com/arithmetic-logarithmic-mean-temperature-d
    Returns:
        LMTD temperature
        corr_fact * ( Delta T 1 - Delta T 2 ) / ln (Delta t 1 / Delta T 2)
    """
    eps = 1e-9
    DeltaT_fr = T_feed - T_return
    DeltaT_feed = T_feed - T_amb
    DeltaT_ret = T_return - T_amb
    # assert (DeltaT_fr > 0), "Output temperature difference $\Delta T_1$ is negative"
    # assert DeltaT_in > DeltaT_out, "Input temperature difference $\Delta T_1$ is smaller
        than output "

    denominator = np.log(DeltaT_feed) - np.log(DeltaT_ret)
    nominator = DeltaT_fr
    # assert denominator > eps, "Ratio of input/output temperature difference too large"
    log_mean_diff_temp = corrfact * nominator / denominator
    return log_mean_diff_temp
```

5.15 Package "housemodel"

The repository "twozone_housemodel-git" contains the modules for the house model. The customary way to organize the modules is to make a *Python package* with *subpackages*. This opens up the possibility of publishing the package on PyPi, so that it can be imported.

See: https://pypi.org/

From commit e74ce58 the files in the twozone_housemodel-git repository are organized as a package. The proposed structure, implemented in this commit, is:

```
twozone_housemodel-git
└── housemodel
    ├── __init__.py
    ├── controls
    │   └── __init__.py
    ├── solvers
    │   └── __init__.py
    ├── sourcesink
    │   └── __init__.py
    └── tools
        └── __init__.py
```

```
└── tests
    ├── __init__.py
    ├── context1.py
    └── test_*.py
```

- the *repository root* twozone_housemodel-git contains the simulation scripts and configuration files (for now)
- the *package root* housemodel contains the complete package. This can be seen since it contains an (empty) __init__.py module.
- the *subpackage* folders contain the modules with common functions and classes for all simulations. They each contain an (empty) __init__.py module.
- a tests folder is placed carefully as a subfolder of the *repository root*. See: https://docs.python-guide.org/writing/structure/ for the underlying philosophy. Here, testing modules (scripts) can be placed. If the names of the test scripts start with test_, they can be automatically run with the pytest Python package.

Note: Running the simulations and tests is best done from the *repository root*. All simulations and tests have been updated to find the package, subpackages and configuration files from this directory.

6 Finite-element discretization

6.1 Heat pump discretization

Het algemene model van de warmtepomp en airco wordt afgeleid met behulp van het volgende geschematiseerde black box model:

6.2 Buffer vessel discretization

Als voorbeeld van een model met warmtestromen en warmtediffusie wordt het volgende model van een buffervat beschouwd:

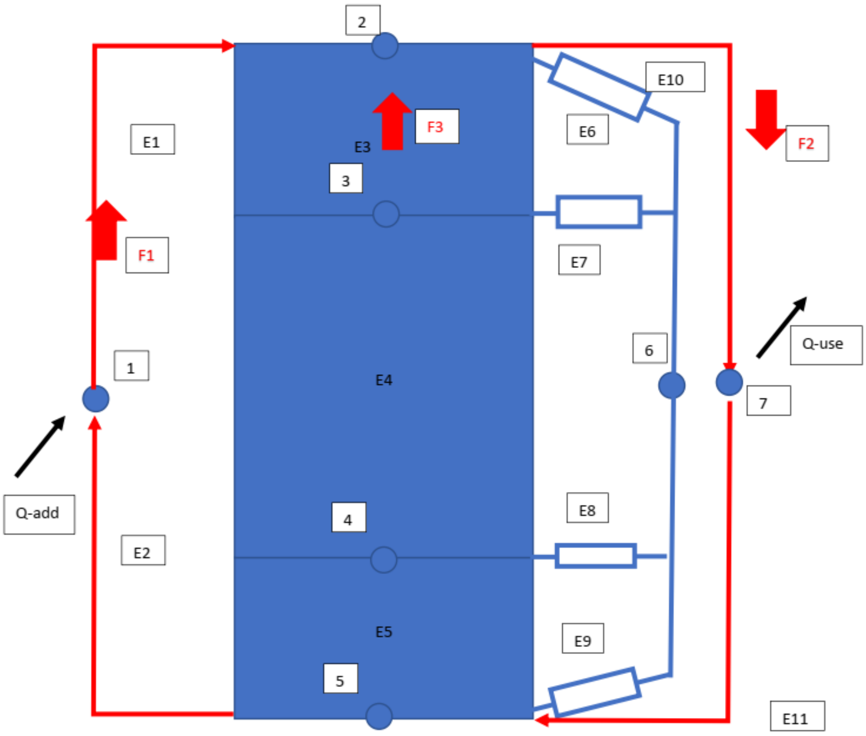

Figure 24: Finite-element buffer vessel

Dit is een buffervat in een verwarmingsinstallatie.

Voor het modelleren hiervan wordt gebruik gemaakt van een tweetal universele elementen:

1. Exchange element. Dit element beschrijft warmtetransportt via een vloeistofstroom F door het element en warmtetransport door geleiding. Het element kan een warmtecapaciteit hebben.

2. Een puntbron. Deze bron beschrijft warmteontwikkeling of warmte-onttrekking in een punt.

Het vat is verdeeld in 3 niveaus over de hoogte:

1. Bovenzijde vat (element 3)

2. Midden vat(element 4)

3. Onderzijde vat (element 5)

In het vat is sprake van:

- Warmteverlies naar andere temperatuurniveau 's en naar de omgeving. De omgevingstemperatuur in dit model is gegeven door de temperatuur in punt 7. De elementen die het volume van het vat beschrijven (E3,E4 en E5) wisselen warmte uit naar elkaar en naar punt 6.

- Warmtetransport door waterstromen. In waterstromen buiten het vat wordt warmte onttrokken of toegevoegd. In het vat is een waterstroom die zorgt voor het kortsluiten van de kringlopen.

Deze beide mechanismen worden meegenomen in het model.

Er wordt verondersteld dat gebruik wordt gemaakt van gelaagdheid. Boven in het vat heerst een hogere temperatuur dan onderin het vat. Daarnaast is er sprake van een tweetal leidingen waarin warmte wordt uitgewisseld met de omgeving:

- Een leiding waarin warmte wordt toegevoegd aan het vat $\dot{Q}_{add}$. Deze leiding neemt vloeistof (water) onder uit het vat (punt 5), verhoogt de temperatuur door warmte-inbreng (punt 1) en brengt het water boven in het vat weer in. Deze leiding bestaat uit de elementen E1 (boven) en E2 (onder). In deze leiding is een vloeistofstroom $F_1(kJ/(Ks))$ aanwezig die de warmte transporteert.

- Een leiding waarin warmte wordt onttrokken aan het vat $\dot{Q}_{use}$. Deze leiding neemt vloeistof (water) boven uit het vat (punt 2), verlaagt de temperatuur door warmteonttrekking (punt 7) en brengt het water boven in het vat weer in. Deze leiding bestaat uit de elementen E10 (boven) en E11 (onder). In deze leiding is een vloeistofstroom $F_2(kJ/(Ks))$ aanwezig die de warmte transporteert.

6.3 Matrixvergelijking

Voor het oplossen van de temperatuurverdeling wordt per knooppunt een energiebalans opgesteld. Deze energiebalans resulteert in een matrixvergelijking.

$$\mathbf{K}\theta + \mathbf{C}\dot{\theta} = \dot{\mathbf{q}} \tag{111}$$

K: de warmtegeleidingsmatrix (W/K) C: de warmtecapaciteitsmatrix (J/K) θ: de temperatuursvector (K) $\dot{\theta}$: de tijdsafgeleide van de temperatuursvector (K/s) $\dot{q}$: de vector met thermische brontermen (W)

6.4 Elementen in de stroming

Om te komen tot het matrixmodel wordt begonnen met één exchange element zoals hierboven geïntroduceerd (E1,E2,E3,E4,E5,E10,E11). Dit element bevat 2 knooppunten. De volgende veronderstellingen worden gedaan:

- Het element bevat 2 knooppunten waarmee deze verbonden is met de omgeving. De nummering bedraagt: n_1 en n_2.

- Binnen het element heerst een lineair verlopende temperatuur, van knooppunt naar knooppunt.

- Binnen het element is een vloeistofstroom F die zorgt voor additioneel warmtetransport. De stroomrichting is van knooppunt 1 naar knooppunt 2.

- De warmtecapaciteit van het element wordt evenredig verdeeld over de knooppunten.

De warmtestroom vanuit het element naar knoopunten 1 en 2 moet in balans zijn met de andere warmtestromen en de warmtegeneratie in de betreffende knooppunten.

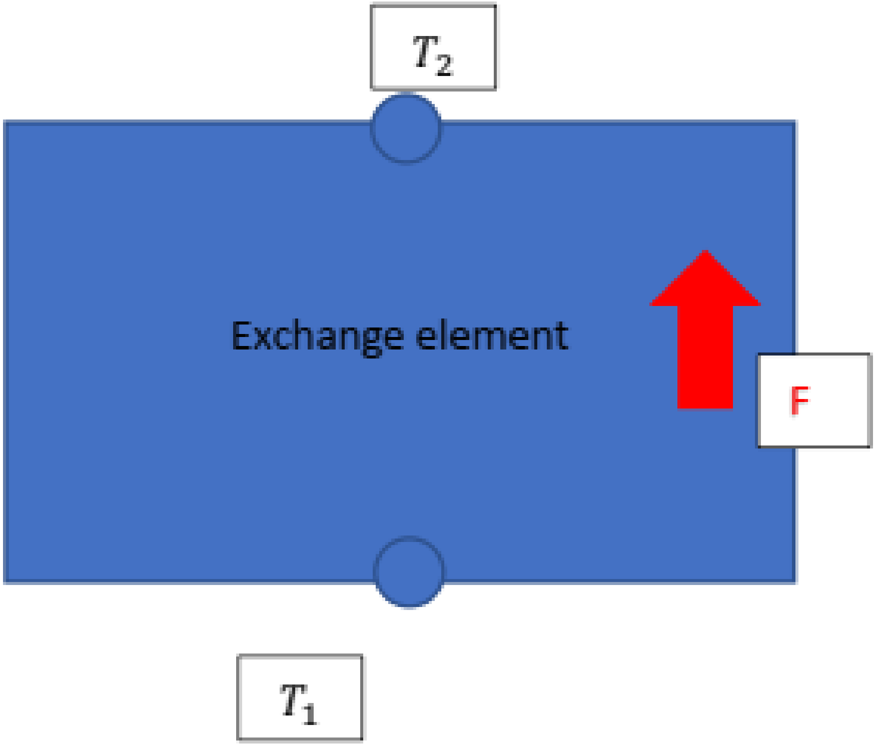

Figure 25: Exchange element buffer vessel

Voor de warmtestromen vanaf knooppunt 1 geldt:

$$(T_1 - T_2) \cdot \frac{1}{R_e} + C_{e1,1} \cdot \frac{dT_1}{dT} = \dot{Q}_{ext,1} \tag{112}$$

T_1: temperatuur in knooppunt 1
T_2: temperatuur in knooppunt 2
R_e: warmteweerstand voor geleiding tussen knooppunt 1 en 2
$C_{e1,1}$: warmtecapaciteit in knooppunt 1 van element 1
$\frac{dT_1}{dT}$: temperatuursverandering in de tijd in knooppunt 1
$\dot{Q}_{ext,1}$: externe warmtetoevoer in knooppunt 1

De vloeistofstroom F vanuit knooppunt T_1 heeft dezelfde temperatuur als T_1 en heeft dus geen invloed op de temperatuur in T_1.

Voor de warmtestromen vanaf punt 2 geldt:

$$(T_2 - T_1) \cdot \frac{1}{R_e} + F \cdot (T_2 - T_1) + C_{e1,2} \cdot \frac{dT_2}{dT} = \dot{Q}_{ext,2} \tag{113}$$

$C_{e1,2}$: warmtecapaciteit in knooppunt 2 van element 1
$\frac{dT_2}{dT}$: temperatuursverandering in de tijd in knooppunt 2
$\dot{Q}_{ext,2}$: externe warmtetoevoer in knooppunt 2

In matrixnotatie:

$$\begin{bmatrix} 1/R_e & -1/R_e \\ -1/R_e - F & 1/R_e + F \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} C_{e1,1} & 0 \\ 0 & C_{e1,2} \end{bmatrix} \cdot \begin{bmatrix} \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \end{bmatrix} = \begin{bmatrix} \dot{Q}_{ext,1} \\ \dot{Q}_{ext,2} \end{bmatrix} \tag{114}$$

Merk op dat door de aanwezigheid van een vloeistofstroom F, de geleidingsmatrix niet langer symmetrisch is.

Daarnaast wordt gebruik gemaakt van elementen die alleen een warmteweerstand weergeven. Deze elementen (E6,E7,E8 en E9) kunnen worden gerepresenteerd met de volgende matrixvergelijking

$$\begin{bmatrix} 1/R_e & -1/R_e \\ -1/R_e & 1/R_e \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} \dot{Q}_{ext,1} \\ \dot{Q}_{ext,2} \end{bmatrix} \tag{115}$$

6.5 Systemmatrices van het buffervat

Er wordt nu een systeemmatrix opgesteld voor het schematisch weergegeven model. Deze systeemmatrix wordt opgebouwd uit de verschillende elementmatrices. De noodzakelijke rang van deze systeemmatrix is het aantal knooppunten in het warmtestroomschema minus het aantal voorgeschreven knooppunten in dit schema. Er zijn 7 knooppunten in het model. In dit geval wordt in punt 6 de temperatuur voorgeschreven. Er resteren dan 6 onafhankelijke vrijheidsgraden.

6.5.1 Capaciteitsmatrix C

There are 7 nodes in the system. The heat capacities are:

$$\begin{aligned}
&\text{node 1} \quad C_{e1,1} + C_{e2,2} = 0.5 \cdot (C_{e1} + C_{e2}) \\
&\text{node 2} \quad C_{e1,2} + C_{e3,1} + C_{e10,2} + C_{e6,2} = 0.5 \cdot (C_{e1} + C_{e3} + C_{e10} + C_{e6}) \\
&\text{node 3} \quad C_{e3,2} + C_{e4,1} + C_{e7,2} = 0.5 \cdot (C_{e3} + C_{e4} + C_{e7}) \\
&\text{node 4} \quad C_{e4,2} + C_{e5,1} + C_{e8,2} = 0.5 \cdot (C_{e4} + C_{e5} + C_{e8}) \\
&\text{node 5} \quad C_{e5,2} + C_{e2,1} + C_{e9,2} + C_{11,2} = 0.5 \cdot (C_{e5} + C_{e2} + C_{e9} + C_{11}) \\
&\text{node 6} \quad C_{e6,1} + C_{e7,1} + C_{e8,1} + C_{e9,1} = 0.5 \cdot (C_{e6} + C_{e7} + C_{e8} + C_{e9}) \\
&\text{node 7} \quad C_{e10,2} + C_{e11,1} = 0.5 \cdot (C_{e10} + C_{e11})
\end{aligned} \tag{116}$$

$$0.5 \cdot \begin{bmatrix} C_{e1}+C_{e2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{e1}+C_{e3}+C_{e10}+C_{e6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{e3}+C_{e4}+C_{e7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{e4}+C_{e5}+C_{e8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{e5}+C_{e2}+C_{e9}+C_{e11} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{e6}+C_{e7}+C_{e8}+C_{e9} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{e10}+C_{e11} \end{bmatrix}$$

The heat capacities of element E1, E2, E10 and E11 (pipelines) are very small and can be approximated to be zero. Also, the elements E6, E7, E8 and E9 are thermal leaks, connected to node 6, which may be a boundary condition.

The capacity matrix thus reduces to:

$$0.5 \cdot \begin{bmatrix} \approx 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{e3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{e3}+C_{e4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{e4}+C_{e5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{e5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \approx 0 \end{bmatrix}$$

In the approximation above, the buffer vessel system is not solvable, without adding some heat capacity to nodes N1 and N7. *e.g.* these nodes must be coupled with a house model element with finite heat capacity.

In any case, the node N6 (boundary condition) does not contribute a DOF to the system of equations. Row 6 and column 6 will be removed from the set of equations.

Listing 3: Generation of capacity matrix

```

    A = nx.adjacency_matrix(G, nodelist=list(range(G.order())))
    B = A.toarray()
    # print(B, "\n")

    row_sums = np.sum(B, axis=1).tolist()
    C_matrix = np.diag(np.array(row_sums), k=0)
    return C_matrix


def K_from_elements(df: pd.DataFrame):
    """assemble K-matrix from Dataframe

    Args:
        df: Dataframe from Excel spreadsheett (float)]

    Returns:
        K_matrix (ndarray):  2D matrix with conductances in network
    """
    # convert Dataframe into list of spreadsheet rows, called "rows"
    # rows becomes a list of lists
    rows = []
    for row in range(len(df.index)):
        rows.append(df.iloc[row].values.tolist())

    # extract element 4: and element 2 of each row into "nodelists"
    nodelists = []
    for row in rows:
        nodelist = [x for x in row[4:] if np.isnan(x) == False]
        nodelist.append(row[3])
        nodelists.append(nodelist)

    # nodelists is a list [node, node, weight], suitable for networkx
    G = nx.Graph()
    G.add_weighted_edges_from(nodelists)
```

6.5.2 $\dot{\mathbf{q}}$-vector

De vector $\dot{\mathbf{q}}$ wordt gegeven door:

$$\left(\begin{array}{cc|c} 1 & 1 & \dot{q}_1 = \dot{Q}_{add} \\ 2 & 2 & \dot{q}_2 \\ 3 & 3 & \dot{q}_3 \\ 4 & 4 & \dot{q}_4 \\ 5 & 5 & \dot{q}_5 \\ 6 & 6 & \dot{q}_6 \\ 7 & 7 & \dot{q}_7 = -\dot{Q}_{use} \end{array}\right)$$

$$\left(\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 9 \end{array}\right)$$

In nodes N1 and N7 an external een heat source / heat sink is contributing to the power balance.

Het voorgeschreven knooppunt (randvoorwaarde, boundary condition) 6 is verbonden aan knooppunten 2, 3, 4 en 5. Dit zijn de ook de vrijheidsgraden 2, 3, 4 en 5. Vrijheidsgraden worden aangegeven met DOF (degree of freedom). In de overeenkomstige knooppunten wordt de capaciteits? stiffness matrix **K** en de bronvector (load vector) $\dot{\mathbf{q}}$ aangepast.De aanpassing van de K-matrix wordt verderop toegelicht. De bronvector wordt als volgt aangepast:

$$\begin{bmatrix} 1 & 1 & \dot{Q}_{add} \\ 2 & 2 & 0 - \frac{1}{R_{2,6}} \\ 3 & 3 & 0 - \frac{1}{R_{3,6}} \\ 4 & 4 & 0 - \frac{1}{R_{4,6}} \\ 5 & 5 & 0 - \frac{1}{R_{5,6}} \\ 6 & 7 & -\dot{Q}_{use} \end{bmatrix}$$

6.5.3 Geleidingsmatrix K

elements with heat capacity must have two nodes: E3, E4, E5

elements without heat capacity have two nodes as well: E1, E2, E6, E7, E8, E9, E10, E11

elements without heat conduction: E1, E2, E10, E11

elements with heat conduction: E3, E4, E5, E6, E7, E8, E9

elements with heat convection (flow): E1, E2, E3, E4, E5, E10, E11

The "conduction" matrix is equivalent to the stiffness matrix in a mechanical FE analysis. In terms of the thermal system topology this matrix contains the "edges" between the nodes. The matrix is setup as a 7 x 7 square zero matrix with the nodes as DOF.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{117}$$

The conductive elements in the system are:

$$\begin{aligned} \frac{1}{R_{2,3}} &= \frac{1}{R_{e3}} \\ \frac{1}{R_{3,4}} &= \frac{1}{R_{e4}} \\ \frac{1}{R_{4,5}} &= \frac{1}{R_{e5}} \\ \frac{1}{R_{2,6}} & \quad \frac{1}{R_{3,6}} \quad \frac{1}{R_{4,6}} \quad \frac{1}{R_{5,6}} \end{aligned} \tag{118}$$

The conductive element $\frac{1}{R_{12}} = \frac{1}{R_{e1}} = 0$. This means $R_{12} \to \infty$. We assume, the pipeline is perfectly insulated and creates no heat leak. Thermal energy flowing *through* it is completely delivered to the target node. Likewise, this holds for the elements E2, E10 and E11.

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{-1}{R_{2,3}} & 0 & 0 & \frac{-1}{R_{2,6}} & 0 \\
0 & \frac{-1}{R_{2,3}} & 0 & \frac{-1}{R_{3,4}} & 0 & \frac{-1}{R_{3,6}} & 0 \\
0 & 0 & \frac{-1}{R_{3,4}} & 0 & \frac{-1}{R_{4,5}} & \frac{-1}{R_{4,6}} & 0 \\
0 & 0 & 0 & \frac{-1}{R_{4,5}} & 0 & \frac{-1}{R_{5,6}} & 0 \\
0 & \frac{-1}{R_{2,6}} & \frac{-1}{R_{3,6}} & \frac{-1}{R_{4,6}} & \frac{-1}{R_{5,6}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{119}
$$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{R_{2,3}} + \frac{1}{R_{2,6}} & \frac{-1}{R_{2,3}} & 0 & 0 & \frac{-1}{R_{2,6}} & 0 \\
0 & \frac{-1}{R_{2,3}} & \frac{1}{R_{2,3}} + \frac{1}{R_{3,4}} + \frac{1}{R_{3,6}} & \frac{-1}{R_{3,4}} & 0 & \frac{-1}{R_{3,6}} & 0 \\
0 & 0 & \frac{-1}{R_{3,4}} & \frac{1}{R_{3,4}} + \frac{1}{R_{4,5}} + \frac{1}{R_{4,6}} & \frac{-1}{R_{4,5}} & \frac{-1}{R_{4,6}} & 0 \\
0 & 0 & 0 & \frac{-1}{R_{4,5}} & \frac{1}{R_{4,5}} + \frac{1}{R_{5,6}} & \frac{-1}{R_{5,6}} & 0 \\
0 & \frac{-1}{R_{2,6}} & \frac{-1}{R_{3,6}} & \frac{-1}{R_{4,6}} & \frac{-1}{R_{5,6}} & \frac{1}{R_{2,6}} + \frac{1}{R_{3,6}} + \frac{1}{R_{4,6}} + \frac{1}{R_{5,6}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{120}
$$

If node N6 is a boundary condition *i.e.* T_6 is given as a constant. The corresponding row in the **K** and **C** matrices can be removed. For the nodes that are connected to node N6 (row 2, 3, 4 and 5) the thermal connection can be moved to the $\dot{\mathbf{q}}$ vector. This can be seen by writing down the differential equation for node N2:

$$
0.5\, C_{e3} \cdot \frac{dT_2}{dt} = \frac{1}{R_{2,3}}(T_3 - T_2) + \frac{1}{R_{2,6}}(T_6 - T_2) = (-\frac{1}{R_{2,3}} - \frac{1}{R_{2,6}}) \cdot T_2 + \frac{1}{R_{2,3}} \cdot T_3 + \frac{1}{R_{2,6}} \cdot T_6 \tag{121}
$$

$$
\begin{array}{ccc}
\mathbf{K}\theta & + \quad \mathbf{C}\dot{\theta} = & \dot{\mathbf{q}} \\
(\frac{1}{R_{2,3}} + \frac{1}{R_{2,6}}) \cdot T_2 - \frac{1}{R_{2,3}} \cdot T_3 + & 0.5\, C_{e3} \cdot \frac{dT_2}{dt} = & \frac{1}{R_{2,6}} \cdot T_6
\end{array} \tag{122}
$$

Note: the sum of each row in **K** is not *zero* anymore, but equals the corresponding vector element in $\dot{\mathbf{q}}$.

This reduces the matrices to:

$$
\mathbf{C} = 0.5 \cdot \begin{bmatrix} \approx 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{e3} & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{e3}+C_{e4} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{e4}+C_{e5} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{e5} & 0 \\ 0 & 0 & 0 & 0 & 0 & \approx 0 \end{bmatrix}
$$

$$
\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{R_{2,3}}+\frac{1}{R_{2,6}} & \frac{-1}{R_{2,3}} & 0 & 0 & 0 \\ 0 & \frac{-1}{R_{2,3}} & \frac{1}{R_{2,3}}+\frac{1}{R_{3,4}}+\frac{1}{R_{3,6}} & \frac{-1}{R_{3,4}} & 0 & 0 \\ 0 & 0 & \frac{-1}{R_{3,4}} & \frac{1}{R_{3,4}}+\frac{1}{R_{4,5}}+\frac{1}{R_{4,6}} & \frac{-1}{R_{4,5}} & 0 \\ 0 & 0 & 0 & \frac{-1}{R_{4,5}} & \frac{1}{R_{4,5}}+\frac{1}{R_{5,6}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{123}
$$

$$
\dot{\mathbf{q}} = \begin{bmatrix} \dot{Q}_{add} \\ \frac{1}{R_{2,6}} \cdot T_6 \\ \frac{1}{R_{3,6}} \cdot T_6 \\ \frac{1}{R_{4,6}} \cdot T_6 \\ \frac{1}{R_{5,6}} \cdot T_6 \\ -\dot{Q}_{use} \end{bmatrix}
$$

Listing 4: Generation of stiffness matrix

```python
    A = nx.adjacency_matrix(G, nodelist=list(range(G.order())))
    B = A.toarray()
    # print(B, "\n")

    row_sums = np.sum(B, axis=1).tolist()
    K_matrix = B - np.diag(np.array(row_sums), k=0)
    return K_matrix


def flowlist_to_edges(fl: list):
    # fl = list(range(10))
    # el = [[fl[i], fl[i+1]] for i in range(len(fl)-1)]
    # el2 = [[i, j] for i, j in zip(fl[:-1], fl[1:])]
    el3 = [[i, j] for i, j in zip(fl, fl[1:])]
    # print(el3)
    return el3


def flow_to_F_matrix(flowlist: list, rank: int):
    # flatten
    flattened = [val for f in flowlist for val in f]

    # print("The original list : " + str(flattened))
    # Finding missing elements in List
```

nodes: E1 has 1 and 2

E2 has 1 and 5

E3 has 2 and 3

E4 has 3 and 4

E5 has 4 and 5

E6 has 2 and 6

E7 has 3 and 6
E8 has 4 and 6
E9 has 5 and 6
E10 has 2 and 7
E11 has 5 and 7

edges conductivity R and convection F:

1 and 2
1 and 5
2 and 3
3 and 4
4 and 5
2 and 6
3 and 6
4 and 6
5 and 6
2 and 7
5 and 7

6.6 Water flow heat transfer

In the buffer vessel model, cf. Figure 24, two water flows run through the system. The first, labeled with $F1$, draws cold water from the bottom layer of the tank. This water is heated up in node 1. In the figure this is done using the external heat flow Q_{add}, but in a full system model another model component, for example a heat pump, can be connected here. (*Note*: In order to be able to add heat in a sensible way to the system node 1 has to have some heat capacity. The approximation done above making the capacity of the pipes zero is then not valid.) The heated water flows back into the vessel in the top level. The water flow $F1$ outside the vessel, induces an equal sized flow of water in side the vessel from the top layer towards the bottom, through the elements $E3$, $E4$ and $E5$.

The second water flow $F2$ draws water from the hot top layer. In node 7 a heat flow Q_{use} is extracted. Here, similar to node 1, another model component (for example a radiator) may be connected. (*Note*: the note made for node 1 is valid here is well). The cooled water will flow back into the vessel in the bottom layer. F2 induces a flow in the buffer vessel opposite to $F1$, running though $E5$, $E4$ and $E3$, consecutively.

The water flows will be controlled by pumps, either by a on/off manner (switching between a fixed water volume per second and 0) or a more advanced varying flow rate. Since $F1$ and $F2$ can be controlled separately, the flow in the vessel, labeled with $F3$ can run either direction, from bottom to top, or from top to bottom. The size of the flow $F3$ is given by: $F3 = F1 - F2$. When $F3$ is positive it flows from top to bottom in the vessel. A negative value, implies a water flow from bottom to top.

6.6.1 Setting up the flow matrix

As indicated earlier, the flow rates can be controlled, and thus may change over time. This means that the terms for the heat exchange due to the water flows need to be generated at each time step, or at least after each change in the flow rates. A matrix that represents the flows may be generated in the following process.

- First of all, the flows in the system need to be defined in the input file. For each flow we need to know the order it traverses the elements in the system, and more specifically the nodes it passes. This can be done by considering each flow separately, and listing the nodes you pass in the direction of the flow. For $F1$

this gives $\text{Nodes}_{F1}\,[0,1,2,3,4,0]$, and for $F2$ this gives $\text{Nodes}_{F2}\,[6,4,3,2,1,6]$. (note the labeling of the nodes used here is the number in Figure 24 minus one.) In the list the first element is equal to the last element, which shows that the flow is a closed loop. The depicted flow $F3$ is only the difference between $F1$ and $F2$, and does not need to be defined by itself.

- From the ordered list of nodes, we can create a "'directed-flow-matrix"' (**DF**) for each flow. This matrix should be of the same size as the conductance-matrix (**K**) and the capacity-matrix (**C**). The directed-flow-matrix contains a 1 for each matrix-element that corresponds to a connection between nodes in the direction of the flow, and -1 for a connection between nodes in the opposite direction. Thus for flows $F1$ and $F2$ the matrix will be:

$$\mathbf{DF_{F1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{124}$$

$$\mathbf{DF_{F2}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{125}$$

  These matrices can be build up from the list as defined in the previous step, by looping through the list and taking the elements $\text{Nodes}(i, i+1)$, and filling in a one at the matrix element $(\text{Node}(i), \text{Node}(i+1))$. After looping through all these pairs we have filled in all connections in the direction of the flow, $\mathbf{DF^{+1}}$. The connections against the flow, $\mathbf{DF^{-1}}$, are given by: $\mathbf{DF^{-1}} = -1 \cdot (\mathbf{DF^{+1}})^T$. Finally, $\mathbf{DF} = \mathbf{DF^{+1}} + \mathbf{DF^{-1}}$.

- In each time step, when the flow sizes have been determined by the control algorithms, each directed-flow-matrix is multiplied by its respected flow size in $[\frac{\mathrm{m}^3}{\mathrm{s}}]$. All resulting matrices can then be added together. Assuming a flow of size f_1 and f_2 for the flows $F1$ and $F2$, respectively we now get the matrix **SF**:

$$\mathbf{SF} = f_1 \cdot \mathbf{DF_{F1}} + f_2 \cdot \mathbf{DF_{F2}} = \begin{bmatrix} 0 & f_1 & 0 & 0 & -f_1 & 0 & 0 \\ -f_1 & 0 & f_1 - f_2 & 0 & 0 & 0 & f_2 \\ 0 & f_2 - f_1 & 0 & f_1 - f_2 & 0 & 0 & 0 \\ 0 & 0 & f_2 - f_1 & 0 & f_1 - f_2 & 0 & 0 \\ f_1 & 0 & 0 & f_2 - f_1 & 0 & 0 & -f_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -f_2 & 0 & 0 & f_2 & 0 & 0 \end{bmatrix} \tag{126}$$

- The heat transfer induced by the flows is only in the direction of the water flow. The correct elements are obtained by taking the $\min(\mathbf{SF}, 0)$, here we mean for each element in **SF** we take the minimum of the respective element and 0. Thus, in the case $f_1 > f_2$ the matrix **SF** will become:

$$\min(\mathbf{SF}, 0) = \begin{bmatrix} 0 & 0 & 0 & 0 & -f_1 & 0 & 0 \\ -f_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & f_2 - f_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & f_2 - f_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f_2 - f_1 & 0 & 0 & -f_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -f_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{127}$$

- Now, the diagonal elements can be computed. The diagonal elements are equal to minus the sum of the of diagonal elements in its respective row. For the matrix given in equation 127 this results in the flow matrix **F**:

$$\mathbf{F} = \begin{bmatrix} f_1 & 0 & 0 & 0 & -f_1 & 0 & 0 \\ -f_1 & f_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & f_2 - f_1 & -(f_2 - f_1) & 0 & 0 & 0 & 0 \\ 0 & 0 & f_2 - f_1 & -(f_2 - f_1) & 0 & 0 & 0 \\ 0 & 0 & 0 & f_2 - f_1 & f_1 & 0 & -f_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -f_2 & 0 & 0 & 0 & 0 & f_2 \end{bmatrix} \tag{128}$$

- Finally, we need to multiply the resulting flow matrix with the density (ρ_{water}) and the specific heat ($c_{p,water}$), in order to obtain the heat transferred by the water due to the water flows. The resulting matrix can be added to the K matrix as given in equation 123.

$$v_{pump} \cdot A_{pipe} \cdot \rho_{water} \cdot c_{p,water} = \left[\frac{m}{s} \cdot m^2 \cdot \frac{kg}{m^3} \cdot \frac{J}{kg \cdot K} = \frac{J}{K \cdot s} = \frac{W}{K} \right] \tag{129}$$

Note 1, when the system contains flows of different fluids, the described steps need to be followed for each fluid type separately. Each fluid will have its own matrix which will contribute to the overall system. This also implies the need to define the density and specific heat for each flow.

Note 2, at this moment the process does not deal with splitting and merging of the water flows. Therefore, a system that may control valves to distribute the water over different radiators using one supply pipe, and one pump, is not feasible in this concept, yet.

6.7 House model (2R2C-model) in finite element structure

Although the equations of the house model as presented in Section **??** have a very similar structure, one should not mix the lumped-element approach presented in Section **??** with the finite-element approach of this chapter. Therefore, we revisit the 2R2C-model in order to incorporate the house model in the finite-element approach.

Recall that, in the lumped-element model, the 2R2C model contained two connected elements with a heat capacity, one for the air in the house and one for the building structure. In the lumped element model these heat capacities cold be directly modeled with one capacitor for the air and one for the walls. The heat transfer between the two capacities was modeled with a single heat conductor (resistor). Finally, the air could exchange heat with the node fro the ambient temperature, which has a fixed temperature. This resulted in the model as presented in Figure 26.

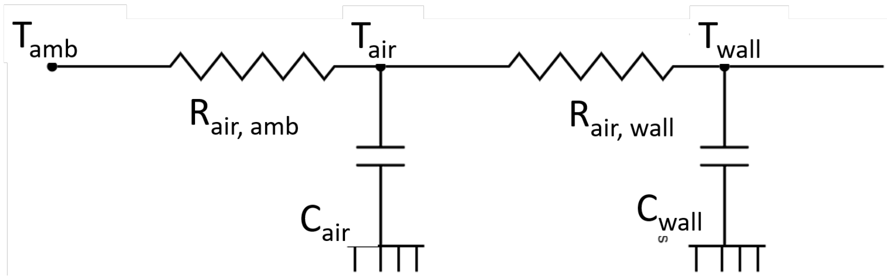

Figure 26: 2R-2C house model revisited

When we want to model this same concept in the finite-element structure, we need four elements: one element that represents the air, one that represents the walls, one element for the interaction between the air and the wall, and one element for the interaction between the air and the ambient surroundings. The model is sketched in Figure **??**.

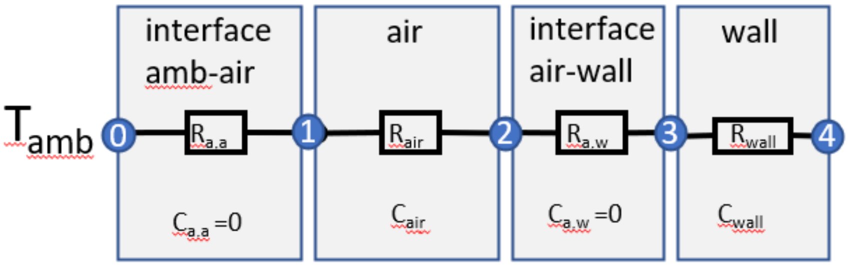

Figure 27: Finite element representation of the 2R2C house model

In this model the heat capacity of the air will be divided over the nodes 1 and 2, with $0.5 \cdot C_{air}$ for each node. In the same manner, the heat capacity of the wall will be divided over the nodes 3 and 4. The elements that are needed for the heat exchange between air and wall, and air and ambient surroundings do not have any heat capacity.

In the lumped-element model the air and wall were considered to have a homogeneous temperature. This is no longer the case in the finite element model. Theoretically, one could make the heat resistance within the air and wall elements zero, which would lead to an instantaneous heat transfer between the nodes 1 and 2, and 3 and 4, respectively. Thus resulting in an equal temperature for nodes 1 and 2, and 3 and 4. However, computationally this will lead to problems, since the corresponding $\mathbf{K}$ matrix will have $\frac{1}{R} = \infty$ entries in the representative positions. Therefore, in practice, an internal heat resistance has to be applied in the air and wall elements. Thus, resulting in a gradient within the elements.

The **C** and **K** matrices of the system will be:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 \cdot C_{air} & 0 & 0 & 0 \\ 0 & 0 & 0.5 \cdot C_{air} & 0 & 0 \\ 0 & 0 & 0 & 0.5 \cdot C_{wall} & 0 \\ 0 & 0 & 0 & 0 & 0.5 \cdot C_{wall} \end{bmatrix}$$
$$\mathbf{K} = \begin{bmatrix} \frac{1}{R_{a,a}} & -\frac{1}{R_{a,a}} & 0 & 0 & 0 \\ -\frac{1}{R_{a,a}} & \frac{1}{R_{a,a}} + \frac{1}{R_{air}} & -\frac{1}{R_{air}} & 0 & 0 \\ 0 & -\frac{1}{R_{air}} & \frac{1}{R_{air}} + \frac{1}{R_{a,w}} & -\frac{1}{R_{a,w}} & 0 \\ 0 & 0 & -\frac{1}{R_{a,w}} & \frac{1}{R_{a,w}} + \frac{1}{R_{wall}} & -\frac{1}{R_{wall}} \\ 0 & 0 & 0 & -\frac{1}{R_{wall}} & \frac{1}{R_{wall}} \end{bmatrix} \tag{130}$$

7 Solar irradiation and PV yield

In the house model, energy supply from solar irradition plays an important role. Firstly, solar energy enters the building through windows and poorly insulated surfaces. In winter, this reduces the cost of heating the building. In summer, however, this leads to an extra energy expenditure for cooling the building, which may attain uncomfortable indoor temperature levels in case of large window surfaces or poor insulation.

A second issue is that the yield of PV and PVT panels, which are often installed nowadays, depends on the solar irradiation. Weather conditions, especially the cloud cover density have a strong influence on the electric power and energy yield of these installations.

Therefore, it is important to be able to calculate the solar irradiation quantity, spectral distribution and spatial properties. Only then, a reliable estimation of the energy demand, and of the useful fraction of solar irradiation can be made.

7.1 Solar software

Software for calculation of solar irradiation on the surface of the earth exists in many shapes and implementations. To achieve the final goal, calculation of the solar (power) falling on a surface with a certain orientation, a number of steps have to be carried out.

1. establish the geolocation of the object (building, PV(T) panel) of interest
2. establish the time instant or time range of interest
3. convert the time instant to local, timezone-aware time or UTC
4. find the apparent position of the sun in the sky (azimuth and inclination)
5. determine the attenuation of the earth's atmosphere for the geolocation and time(s) of interest
6. determine the DNI
7. determine the orientation of the surface of interest (azimuth and inclination)
8. determine the direct, diffuse and global irradiation on the surface
9. determine the fraction of the solar irradiation that is effective as an energy source (window transmittance, PV(T) efficiency)

Among the packages available for solar irradiation calculations, we find:

- PV_LIB Toolbox: available for Matlab and Python [23–26].
- solarenergy: available as Python package [27, 28]
- qsun: available as Matlab function or Python function.

7.2 Geolocation

The location of a building or installation needs to be given in *latitude* and *longitude*, in units of degrees with a decimal point. Division in arcminutes and arcseconds is less common nowadays, since the introduction of GPS. Latitude is positive for the northern hemisphere, negative to the south of the equator. The equator itself is zero latitude. Longitude is positive to the east of the Royal Observatory in Greenwich, London, UK, negative to the west of London. The Meridian of Greenwich runs from the North pole to the South Pole through London and has zero longitude. At the poles, latitude is ± 90 degrees and longitude is undefined.

For **Arnhem**, NL, a **latitude of 52.0 degrees** and a **longitude of 6.0 degrees** may be used as an approximation to the geolocation. In reality this geolocation is found in a field between Velp and Rheden, NL.

- PV_LIB Toolbox has a module location.py. In this module, a class Location is defined, with attributes *latitude* and *longitude*. These attributes are in *decimal degrees i.e.* 52.0 and 6.0.
- solarenergy has a module radiation.py with a function sun_position_from_date_and_time. Input parameters to this function are *longitude* and *latitude* in *radians*. The solarenergy has a conversion constant d2r to convert from decimal degrees to radians.
- qsun: longitude and latitude are not input parameters. They are fixed: the chosen location is for De Bilt, NL (52.1 N, 5.1 E).

7.3 Time and timezones

In many programming languages, a datetime object exists. The basic functionality of such an object includes:

- a convention about time "zero".
- a representation of time, stored in an integer or floating-point value.
- a set of conversion routines from various time strings *e.g.* 2021-11-25 17:28:31:321+01:00 to the storage format, and back.
- timezone awareness and daylight savings options.

7.3.1 Time formats and conventions

Many conventions are currently in use. The most "universal" is the UNIX Timestamp. Its *epoch*, the "zero" time is 1 January 1970, 00:00:00 (UTC). The time is represented by an *integer* which counts the *seconds* elapsed since the epoch. Originally, the representation was an int32, which would mean that the computer time is up in the year 2038. Backwards, the beginning of computer time would be in 1901. Fortunately, 64-bit computer registers now also use an int64 for UNIX timestamp representation, which alleviates this shortcoming for all practical situations.

The int64 representation stretches so far into the future and past, that it makes room for improvement. Microsoft Windows maintains a FILETIME structure, built from two DWORD (uint32) entries, which taken together to a 64-bit value represent the number of 100-ns intervals since January 1, 1601 00:00:00.0000000 (UTC).

typedef struct _FILETIME {
DWORD dwLowDateTime;
DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;

In Python, the original datetime package contains a datetime class which has its epoch at 1 January 1970, just like the UNIX timestamp. The datetime class has members: year (1-9999), month (1-12), day (1- # of days in month), hour (0-23), minute (0-59), second (0-59) and microsecond (0-999999). Moreover, it has an attribute tzinfo, which handles timezone info and an attribute fold (0, 1) to handle the occurrence of two identical wall times when daylight savings time is reset in autumn.

However, the Python package pandas has an alternative Timestamp class, which uses a int64, representing the number of 1-ns intervals since 1 January 1970. This makes it compatible with UNIX timestamps (divide by 1e9) and with classical Python datetime objects. The type is given as datetime64[ns, Europe/Amsterdam]. This reveals that, apart from the timestamp in UTC, a timezone may be stored. This is done with the helper package pytz, which is installed as a dependency of pandas. It is strongly recommended to always use timezone-aware timestamps, even if UTC is meant. The pytz package also handles daylight savings times smoothly in timezone-aware timestamps.

7.3.2 Examples in Python

The standard Python datetime object is defined in the module datetime.py. On import, it is recommended to also include the timedelta object fom the same module. The use of datetime and timedelta objects without setting timezone information is shown in Listing ??.

In combination with geolocation, however, it is recommended to use *timezone-aware* datetime objects. This is demonstrated in Listing ??. Note that the *attribute* of the datetime class is named tzinfo. The input argument for the *method* datetime.now is named tz. The value of this input argument sets datetime.tzinfo from None to a meaningful timezone value.

https://www.alpharithms.com/generating-artificial-time-series-data-with-pandas-in-python-272321/

https://stackoverflow.com/questions/993358/creating-a-range-of-dates-in-python

https://stackoverflow.com/questions/1060279/iterating-through-a-range-of-dates-in-python/1060330#1060330

https://stackoverflow.com/questions/13445174/date-ranges-in-pandas

https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.date_range.html

https://www.w3resource.com/pandas/date_range.php

Voorbeeld timestamp and date_range in Pandas.

- PV_LIB Toolbox has a module location.py. In this module, a class Location is defined, with attributes *latitude* and *longitude*. These attributes are in *decimal degrees i.e.* 52.0 and 6.0.
- solarenergy has a module radiation.py with a function sun_position_from_date_and_time. Input parameters to this function are *longitude* and *latitude* in *radians*. The solarenergy has a conversion constant d2r to convert from decimal degrees to radians.
- qsun: longitude and latitude are not input parameters. They are fixed: the chosen location is for De Bilt, NL (52.1 N, 5.1 E).

7.3.3 Conversion of NEN5060 time information

In the spreadsheet *NEN5060-2018.xlsx*, shown in Figure 28a, the first four colums A:D contain the timestamp information. Since the NEN 5060 data is derived from hourly KNMI weather data, it follows the convention of the KNMI records, where the diurnal HOUR data runs from 1-24. The corresponding record of KNMI weather data is given in Figure 28b. KNMI uses UTC timestamps (https://www.knmidata.nl/data-services/knmi-producten-overzicht) pointing to data from the *previous* hour. These UTC timestamps are coded in the columns YYYYMMDD and H, respectively.

In Listing 5 the conversion from the NEN 5060 spreadsheet colums, read into a Pandas Dataframe, is shown. The function pandas.to_datetime correctly handles an offset of $-1\,h$, thereby changing the hour range to 0-23. Thus, the Pandas Timestamps refer to the *following* hour period. The Pandas Timestamps thus obtained are still *naive*. Conversion to *timezone-aware* UTC Timestamps is done by the tz_localize function, which uses a timezone from the pytz package. The *timezone-aware* UTC Timestamps can be converted to the timezone "Europe/Amsterdam" by calling the tz_localize function again. In the local Dutch Timestamps, the Daylight Savings Time (DST) is automatically included. columns with a Pandas UTC and local timestamp are inserted at the beginning of the NEN 5060 DataFrame.

	A	B	C	D	E	F	G	H	I
1	jaar	MONTH(datum)	DAY(datum)	HOUR(uur)	globale_zonnestraling	diffuse_zonnestraling	directe_zonnestraling	directe_normale_zonnestraling	temperatuur
2					W/m2	W/m2	W/m2	W/m2	0,1 °C
3	2001	1	1	1	0	0	0	0	15
4	2001	1	1	2	0	0	0	0	17
5	2001	1	1	3	0	0	0	0	15
6	2001	1	1	4	0	0	0	0	12
7	2001	1	1	5	0	0	0	0	11
8	2001	1	1	6	0	0	0	0	11
9	2001	1	1	7	0	0	0	0	13
10	2001	1	1	8	0	0	0	0	15
11	2001	1	1	9	0	0	0	0	18
12	2001	1	1	10	6	6	0	0	22
13	2001	1	1	11	19	19	0	0	26
14	2001	1	1	12	39	39	0	1	34
15	2001	1	1	13	36	36	0	1	40
16	2001	1	1	14	19	19	0	1	44
17	2001	1	1	15	8	8	0	0	44
18	2001	1	1	16	3	3	0	0	45
19	2001	1	1	17	0	0	0	0	46
20	2001	1	1	18	0	0	0	0	48
21	2001	1	1	19	0	0	0	0	50
22	2001	1	1	20	0	0	0	0	52
23	2001	1	1	21	0	0	0	0	58
24	2001	1	1	22	0	0	0	0	61
25	2001	1	1	23	0	0	0	0	61
26	2001	1	1	24	0	0	0	0	62
27	2001	1	2	1	0	0	0	0	61
28	2001	1	2	2	0	0	0	0	61
29	2001	1	2	3	0	0	0	0	58

(a) NEN 5060 spreadsheet (detail)

12	# STN	YYYYMMD	H	T	SQ	Q	P
13	260	20010101	1	15	0	0	10036
14	260	20010101	2	17	0	0	10022
15	260	20010101	3	15	0	0	10006
16	260	20010101	4	12	0	0	9996
17	260	20010101	5	11	0	0	9988
18	260	20010101	6	11	0	0	9978
19	260	20010101	7	13	0	0	9970
20	260	20010101	8	15	0	0	9967
21	260	20010101	9	18	0	0	9963
22	260	20010101	10	22	0	2	9965
23	260	20010101	11	26	0	7	9955
24	260	20010101	12	34	0	14	9948
25	260	20010101	13	40	0	13	9949
26	260	20010101	14	44	0	7	9949
27	260	20010101	15	44	0	3	9951
28	260	20010101	16	45	0	1	9950
29	260	20010101	17	46	0	0	9953
30	260	20010101	18	48	0	0	9950
31	260	20010101	19	50	0	0	9951
32	260	20010101	20	52	0	0	9952
33	260	20010101	21	58	0	0	9952
34	260	20010101	22	61	0	0	9949
35	260	20010101	23	61	0	0	9950
36	260	20010101	24	62	0	0	9945
37	260	20010102	1	61	0	0	9941
38	260	20010102	2	61	0	0	9938
39	260	20010102	3	58	0	0	9934

(b) KNMI hourly data in csv format (detail)

Figure 28: NEN 5060 spreadsheet and parent KNMI hourly weather record.

Listing 5: Conversion of NEN5060 timestamp to timezone-aware Pandas Timestamp

```
def NENdatehour2datetime(nen_df: pd.DataFrame):
    # define timezones
    utz = timezone('UTC')
    nltz = timezone('Europe/Amsterdam')

    # convert columns 'jaar', 'MONTH(datum)', 'DAY(datum)', 'HOUR(uur)' into Pandas timestamps
    # subtracting 1 hour from the 'HOUR(uur)' values (works automatically!)
    pdt_naive = pd.to_datetime(dict(year=nen_df['jaar'],
                                    month=nen_df['MONTH(datum)'],
                                    day=nen_df['DAY(datum)'],
                                    hour=nen_df['HOUR(uur)'] - 1))
    # make NAIVE UTC forward-looking timestamp AWARE
    # Note: this cannot be done inplace because Timestamps are IMMUTABLE
    # Note2: since pdt_naive is a pandas Series object, use Series.dt.tz_localize and
        Series.dt.tz_convert
    pdt_utc = pdt_naive.dt.tz_localize(tz=utz)
    # convert AWARE UTC to AWARE local time
    pdt_local = pdt_utc.dt.tz_convert(tz=nltz)

    # insert AWARE UTC and AWARE LOCAL DateTimeIndex as first columns in DataFrame
    nen_df.insert(loc=0, column='utc', value=pdt_utc)
    nen_df.insert(loc=1, column='local_time', value=pdt_local)
    return nen_df
```

7.3.4 Gregorian and Julian time

Today's calendar is the Gregorian calendar, introduced by pope Gregory XIII in 1582. This calendar refines the use of leap years, compared to its predecessor, the Julian calendar, introduced by Julius Caesar in 45 B.C. [29]. In the transition process in October 1582, 10 days had to be skipped. It is clear that this time gap was good for society (finally, Turkey introduced the Gregorian calendar in 1926!), but not for astronomy. That is why astronomers kept using the Julian calendar - between 1582 and 1926 - and ever since. That means they have to define a new epoch every 50 years, to compensate for the imperfections of the Julian calendar. The big advantage is that the planets have kept their undisturbed orbits and that the Harmony of the Spheres is still in sync with ancient times.

7.4 Position of the sun

7.5 Attenuation of the solar radiation

7.6 Direct Normal Incidence (DNI)

7.7 Orientation of the receiving surface

7.8 Direct, diffuse and global irradiation

7.9 Efficiency

8 PV and solar collector modeling

This section presents the (proposed) models that describe the behavior of PV-panels, thermal solar collectors and the combination of the two as PVT panels.

8.1 generic panel properties

PV panels and thermal collectors have a common set of properties. Both are oriented surfaces, which transforms the incoming energy from the solar radiation into useful energy; electrical energy for PV, and heat for thermal collectors. The yield highly depends on the location, orientation with respect to the sun and the total surface area. Below the common properties are listed:

- surface_area: the surface of the panels in m^2.
- longitude: longitude of the location of the panels, given in degrees.
- latitude: latitude of the location of the panels, given in degrees.
- inclination: angle of the panel with the horizontal plane in degrees. The value lies between 0 degrees for horizontal and 90 degrees for vertical.
- azimuth: angle with due south direction in degrees (for the northern hemisphere). The value lies between -180 degrees and 180 degrees, with 90 degrees facing due west and -90 degrees facing due east.

Using these properties one can compute the irradiance level at a given time. Based on the NEN5060 irradiation numbers for the measured global irradiance on the horizontal plane, and the derived diffuse irradiance on the horizontal plane we can find the contributions of the direct and diffuse irradiance.

8.2 splitting global irradiance into direct and diffuse

Most weather data contain only a measurement for the global irradiance on a horizontal plane. In order to make a good estimate for the yield of PV and thermal panels it is important to have an estimate of the direct and diffuse irradiance on the oriented surface of the panels, separately. In literature different experimental models can be found that give a method for making this split. In [30], Dervishi and Mahdavi compare a set of these models that have been published over the years. They conclude that, of the models in their analysis, the model by Erbs et al. [31] gives the best results.

The Erbs model determines a clearness index k_t based on the extraterrestrial solar irradiance (I_o), the sun altitude (α) and the measured global irradiance (I_t):

$$k_t = \frac{I_t}{I_o \cdot \sin(\alpha)}. \tag{131}$$

In the model, I_o is determined with the following equation:

$$I_o = I_{sc} \cdot \left(1 + 0.33 \cdot \cos\frac{360 \cdot n}{365}\right) \cdot \cos(\theta_z), \tag{132}$$

where I_{sc} is the extraterrestrial solar constant irradiance (set to 1367 W/m^2), n is the day number, and θ_z is the zenith angle.

Based on the clearness index k_t the fraction of the diffuse horizontal irradiance (k_d) can be determined:

$$\text{interval:} \quad k_t \leq 0.22 \quad k_d = 1 - 0.09 k_t, \tag{133}$$

$$\text{interval:} \quad 0.22 < k_t \leq 0.8 \quad k_d = 0.9511 - 0.1604 k_t + 4.39 k_t^2 - 16.64 k_t^3 + 12.34 k_t^4, \tag{134}$$

$$\text{interval:} \quad k_t > 0.8 \quad k_d = 0.165. \tag{135}$$

Now, using k_d we can determine the diffuse contribution of the irradiance on the horizontal plane $I_{dif,h} = k_d \cdot I_t$. The direct irradiance onthe horizontal plane is the complementary part, $I_{dir,h} = I_t - I_{diff,h}$.

8.3 irradiation on an inclined surface

In order to be able to compute the output power of the PV-panel we need to compute the contributions of both the diffuse and direct irradiance on the oriented surface of the PV-panel. For the direct irradiance ($I_{dir,p}$) this ca be done by using the location and orientation of the panels and the orientation of the sun.

$$I_{dir,p} = \frac{\cos\theta}{\sin h} \tag{136}$$

In order to transform

8.4 PV-panel efficiency

A PV-panel converts the energy of the incoming solar irradiation to electrical energy. The efficiency of the conversion depends on the temperature of the panels according to the relationship [REF to dictaat Marc]:

$$\eta_{\text{cell}}(T_{\text{cell}}) = \eta_{\text{cell,N}} \left(1 + \gamma_{textT} \left(T_{\text{cell}} - T_{\text{cell,N}}\right)\right), \tag{137}$$

where $\eta_{\text{cell,N}}$ is the nominal efficiency according to the panel specifications, γ_{textT} is temperature coefficient according to the panel specifications, $T_{\text{cell,N}}$ is the reference temperature at which the nominal efficiency is measured, and T_{cell} is the actual temperature of the panel.

The actual temperature can be approximated using the formula:

$$T_{\text{cell}} \approx T_a + \left(43.3\text{exp}l \left[-0.61 \left(\frac{v_w}{\text{m/s}}\right)^{0.63}\right] + 2.1\right) \left(\frac{I_{g,s}}{1000\text{W/m}^2}\right) \tag{138}$$

9 NEN and ISO

The list of NEN and ISO standard used in the calculation:

- NTA 8800
- NEN 1068
- ISO 6946
- ISO 10077-2
- NEN 7120

10 Manual; how to work with the two zone house model

10.1 Voor wie?

Deze manual is bedoeld om een handreiking te geven aan bedrijven die de impact van hun warmtebron willen doorrekenen.

References

[1] Alfonso P.Ramallo-GonzálezaMatthew, E.EamesaDavid, and A.Coleyb. “Positioning and Design Recommendations for Materials of Efficient Thermal Storage Mass in Passive Buildings”. In: *Energy and Buildings Volume 60,Pages 174-184* (2013). DOI: 10.1016/j.enbuild.2013.01.014.

[2] Daniel Coakley, Paul Raftery, and Marcus Keane. “A review of methods to match building energy simulation models to measured data”. In: *Renewable and Sustainable Energy Reviews Volume 37, Pages 123-141* (2014). DOI: 10.1016/j.rser.2014.05.007.

[3] Ali Bagheri, Véronique Feldheim, and Christos S. Ioakimidis. “On the Evolution and Application of the Thermal Network Method for Energy Assessments in Buildings”. In: *Energies* 11.4 (2018). ISSN: 1996-1073. DOI: 10.3390/en11040890. URL: https://www.mdpi.com/1996-1073/11/4/890.

[4] Madsen Henrik and Bacher Peder. *Thermal Performance Characterization using Time Series Data; IEA EBC Annex 58 Guidelines*. Dec. 2015. DOI: 10.13140/RG.2.1.1564.4241.

[5] Fraisse et al. “Lumped parameter models for building thermal modelling: An analytic approach to simplifying complex multi-layered constructions”. In: *Energy and Buildings Volume 34, Issue 10, Pages 1017-1031* (2002). DOI: 10.1016/S0378-7788(02)00019-1.

[6] *Lumped-element model*. URL: https://en.wikipedia.org/wiki/Lumped-element_model.

[7] *Heat-transfer-thermodynamics*. URL: https://heat-transfer-thermodynamics.blogspot.com/2016/06/fundamentals-of-thermal-resistance.html.

[8] *Fundamentals of thermal resistance*. URL: https://celsiainc.com/heat-sink-blog/fundamentals-of-thermal-resistance.

[9] *R-value (insulation)*. URL: https://en.wikipedia.org/wiki/R-value_(insulation)#cite_note-Standardization-4.

[10] *Overall heat transfer coefficient*. URL: https://www.engineeringtoolbox.com/overall-heat-transfer-coefficient-d_434.html.

[11] *Surface heat transfer coefficient*. URL: https://www.htflux.com/en/documentation/boundary-conditions/surface-resistance-heat-transfer-coefficient.

[12] *Het bouwbesluit over isolatie en rc waarde*. URL: https://www.isolatiemateriaal.nl/kenniscentrum/het-bouwbesluit-over-isolatie-en-rc-waarde.

[13] *ISSO*. URL: https://v-1isso-1nl-1y6tawt2z0091.stcproxy.han.nl/q/9d67bdb7.

[14] *R-waarde*. URL: https://www.joostdevree.nl/shtmls/r-waarde.shtml.

[15] *Voorbeeldwoningen 2011*. URL: https://www.rvo.nl/onderwerpen/duurzaam-ondernemen/gebouwen/woningbouw/particuliere-woningen/voorbeeldwoningen.

[16] *Absolute thermal resistance*. URL: https://en.wikipedia.org/wiki/Thermal_resistance.

[17] *Thermal mass*. URL: https://en.wikipedia.org/wiki/Thermal_mass.

[18] ISSO. “Handboek HBz Zonnestraling en zontoetreding”. In: kennisbank, 2010. Chap. ”5.5.1 en 5.2”. ISBN: 978-90-5044-190-2.

[19] Engineering Toolbox. *Heat Emission from Radiators and Heating Panels*. URL: https://www.engineeringtoolbox.com/heat-emission-radiators-d_272.html.

[20] NEN. *NEN-EN 442-2:2014 en*. URL: https://www.nen.nl/nen-en-442-2-2014-en-202612.

[21] OpenEnergyMonitor. *Learn — OpenEnergyMonitor - Radiator Model*. URL: https://learn.openenergymonitor.org/sustainable-energy/building-energy-model/radiatormodel.

[22] G.G.J. Achterbos et al. "The Development of a Convenient Thermal Dynamic Building Model". In: *Energy and Buildings* 8 (1985), pp. 183–196. URL: https://ris.utwente.nl/ws/portalfiles/portal/6737586/Achterbosch85development.pdf.

[23] Sandia Labs. *PV_LIB Toolbox*. URL: https://pvpmc.sandia.gov/applications/pv_lib-toolbox/.

[24] Sandia Labs. *PV_LIB Toolbox for Python*. URL: https://pvpmc.sandia.gov/applications/pv_lib-toolbox/pv_lib-toolbox-for-python/.

[25] Sandia Labs. *ReadTheDocs PV_LIB*. URL: https://pvlib-python.readthedocs.io/en/latest/.

[26] pvlib. *GitHub - pvlib/pvlib-python*. URL: https://github.com/pvlib/pvlib-python.

[27] MarcvdSluys. *ReadTheDocs solarenergy*. URL: https://solarenergy.readthedocs.io/en/latest/.

[28] MarcvdSluys. *MarcvdSluys/SolarEnergy: A Python module to do simple modelling in the field of solar energy*. URL: https://github.com/MarcvdSluys/SolarEnergy.

[29] timeanddate. *Julian to Gregorian calendar: How we lost 10 days*. URL: https://www.timeanddate.com/calendar/julian-gregorian-switch.html.

[30] Sokol Dervishi and Ardeshir Mahdavi. "Computing diffuse fraction of global horizontal solar radiation: A model comparison". In: *Solar energy* 86.6 (2012), pp. 1796–1802.

[31] DG Erbs, SA Klein, and JA Duffie. "Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation". In: *Solar energy* 28.4 (1982), pp. 293–302.