

Edouard Walther

Building Physics Applications in Python

First Edition



Spring 2021

Front page background image copyright Séverine Huet ©, front page layout Yannick Schindele ©.

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported” license.



Preface

By Professor Christian Inard
Vice Rector for Research – La Rochelle University

In many sectors, engineers are facing very complex problems. The difficulty of the situations encountered is further complicated by the fact that unsteady or coupled systems, or both, are to be managed.

For the purpose of sizing but foremost for the unsteady simulation of complex systems, the use of numerical methods remains essential and efficient.

Obviously, this is the case in the field of building physics. Indeed, the building associated with its heating and cooling system remains a complex field of study both by its dimensions and by the physical phenomena encountered. This requires considering, in particular, spatial and temporal scales of very dissimilar levels.

Furthermore, numerical computing has evolved considerably over the last few decades to reach remarkable levels of performance. It is thus becoming easier by the day to solve unsteady problems, whether coupled or not, by using both powerful numerical methods and a modern programming language.

In the field of building physics, this book deals with basic and applied aspects of coupled heat and mass transfers and system control in a harmonious and educational way. Of particular interest are applications such as phase change materials, HVAC control or indoor air quality.

This volume will therefore be very fruitful for acquiring fundamental notions as well as applying numerical solutions to complex problems using state-of-the-art methods in python, a programming tool fully recognized by the scientific community.

To conclude, the content of the present work is just like the curriculum and the skills of its author *i.e.* dealing with education, research and development.



About the author

Edouard Walther has been working as a practitioner in applied research at AREP since 2015. In this frame, he leads a group in charge of building physics modelling for semi-open spaces, such as train stations.

Graduated engineer from the INSA Strasbourg (2010), he was awarded both the "*Teacher training in higher education*" Master degree from ENS Cachan and the Civil Engineering Teacher training degree (Agrégation) in 2011. He obtained his PhD. entitled "*Contribution of the Lattice Boltzmann method to the study of the building envelope*" from Paris-Saclay University in 2016.

He teaches on a regular basis at the École Normale Supérieure Paris-Saclay and INSA Strasbourg. His research activities are focused on indoor thermal comfort and air quality in large spaces, air quality in the underground environment, applied numerical methods and modelling of the urban microclimate.



Acknowledgments

If it were only for its author, this volume would not exist: let us take a few lines to acknowledge each contribution.

Forewording this book, Professor Christian Inard of La Rochelle University provides his internationally acknowledged expertise in the field of building physics: may he be warmly thanked for his confidence and for the credibility he gives to this humble volume.

The original idea and implementation of the interactive Jupyter Notebooks is Mr. Yannick Schindele's. I would like to express my gratitude to him not only for his attentive proof reading, crash-testing of the code and patient explanations about distributed version control, but more particularly for two decades of lasting friendship: merci infiniment Yannick !

Antoine Hubert and Mateusz Bogdan wore the heavy "typo-correction" burden and gave their insight about the scientific contents, providing nifty suggestions to help the understanding.

Michele Peou made priceless corrections of my written English throughout the manuscript: an appreciated and salutary input!

Professor Clément Desodt of Université Paris-Saclay provided a version of the geothermal heat pump code and is actually at the very origin of this project. Several years ago, upon his recommendation I contacted a well known editor that rejected my draft manuscript, invoking the prospect of possibly low profits. Lessons learnt, today with Clément's encouragement this book is freely available and open source!

Séverine Huet made the cover's background illustration.

Julien Berger from USMB/LASIE shed light on my understanding of coupled heat and mass transfer.

David Néron from LMT-ENSPS made the original LaTeX .cls file.

Jean-Baptiste Bouvenot notified me a tragic mistake in the Dirac function of the PCM section.

I would be ungrateful not to mention the *Badische Staatsbrauerei Rothaus* was a steady source of inspiration during the long, locked, nights of April and May 2020. Later on, *Domaine Pierre Weber* took over for the November and December 2020 writing support, offering a much appreciated shelter in a quiet environment.

Introduction

Written during the 2020 lock-downs, these pages present an overview of building physics applications in Python language for students, teachers and engineers.

What this book is about

On one end of the scientific literature, profusion of works about solving partial differential equations exist, however often with a somewhat unfamiliar mathematical formalism. On the other end, volumes dealing with building physics may be either technical about HVAC or rather generic about the equations to be used.

Having struggled quite a bit in the past years with the practical implementation of numerical methods in this field, it appeared that putting together the recipes used in a modern programming language could be of interest.

Hence the *parti pris* in this book is to show the link between the governing equations and how to solve them, aiming at a practical use (*i.e.* "how to make things work"). It can be seen as a toolbox for simulation engineering, a basis for the illustration of theory or a kick-start for the study of more complex problems.

This manual is composed of three chapters, with gradual increase in difficulty:

- Chapter 1 succinctly explains the fundamentals of the numerical methods used.
- Chapter 2 shows applications of these methods to heat transfer in phase change materials, PID control, indoor air quality and geothermal heat pumps.
- Chapter 3 deals with coupled problems and minimisation. Applications to polydispersed aerosols in enclosures, heat and mass transfer in walls and parameter fitting for transient problems are proposed.

What this book is not about

Before starting, let us mention a few important things that we do not pretend to investigate here: numerical performance or scheme order, thorough and stringent mathematical developments, code and memory optimisation, or the exact adequacy of the solving procedure to the problem (as far as it does the job!).

What about this book?

For all cases examined in the following chapters, source code or supplementary material such as pollution/weather data are available. In each section, specific links pointing to the corresponding procedures are provided. Two types of users may be interested:

- If you want to play with the parameters of the models described, a simple web browser is sufficient, as interactive plots are at your disposal for many of the examples, thanks to the generous implication of Mr. Yannick Schindele who proposed and set up the corresponding Jupyter Notebooks.
- If you intend to make your own tests or applications, the code can be downloaded from this repository: <https://github.com/eddes/buildingphysics>.

As a few animations are presented throughout the sections, you may find it enjoyable to read these pages with a compatible PDF viewer (such as Foxit or Adobe Reader).

Table of contents

1	A quick reminder about numerical integration	11
1	Integration over time and space	11
1.1	Time integration	11
1.2	Space integration	12
1.3	Formulation: <i>finite</i> what?	13
2	Numerical formulation of equations	14
2.1	Inside homogeneous media	14
2.2	Two-dimensional formulation	16
2.3	Boundary conditions and interfaces	18
3	Overcoming stability issues	20
3.1	Stability for Euler's explicit scheme – A cookbook condition	20
3.2	Crank-Nicolson's scheme	21
4	A word about computer programming	25
4.1	Debugging	25
4.2	Before starting	25
2	Transient problems	26
1	Phase Change Materials	26
1.1	Modelling phase change in a wall	26
1.2	Numerical model	28
1.3	Application – Comparison of temperature profiles	29
2	Indoor Air Quality	32
2.1	Modelling filter clogging	32
2.2	Numerical model	33
2.3	Simulation results for a given critical mass	34
2.4	Cost or air quality?	35
3	PID Controllers in HVAC	40
3.1	Mathematical model of a PID controller	40
3.2	Application - Gravity drainage of a tank	41
3.3	Application - Three-way-valve controller for space heating	46
4	Geothermal heat pump	51
4.1	Mathematical model of ground heat pump	51
4.2	Numerical model in 2D	56
4.3	Application - Geothermal heat pump	59
3	Coupled problems & minimisation	63
1	Indoor Air Quality: <i>two-compartments</i> models	63
1.1	What is an aerosol?	63
1.2	Physical model	64
1.3	Numerical implementation for one size-class	67
1.4	Extension to an n size-class aerosol	69
2	Heat and mass transfer in walls	74
2.1	Phenomenon & Governing equations	74

2.2	Modelling heat & mass transfer through walls	76
2.3	Observations on the methods	80
3	Parameter fitting on differential equations	83
3.1	Minimisation in practice	83
3.2	Automatic tuning of a PID	83
3.3	Air Quality in Underground Stations	88
4	Appendix	93
1	Demonstrations & complements	93
1.1	Matrix formulation in 2D	93
1.2	The K_v value	94
1.3	Demonstration of the relation between Q_v , K_v and a	95
2	Code	96
2.1	Solving (systems of) Equations	96
2.2	Data resampling	96
2.3	Data interpolation	97
2.4	Data fitting	97
2.5	Pareto front computation	98
	Bibliography	99

Nomenclature

Heat transfer

α thermal diffusivity [m^2/s]

Δt time discretisation [s]

ΔT_f half fusion temperature interval [$^{\circ}\text{C}$]

Δx space discretisation [m]

λ material conductivity [$\text{W}/\text{m}/\text{K}$]

ρ density [kg/m^3]

C_p specific heat capacity [$\text{J}/\text{kg}/\text{K}$]

f liquid fraction in the PCM [-]

F_o Fourier number [-]

F_o^{eq} equivalent Fourier number including a convection term [-]

h superficial heat transfer coefficient [$\text{W}/\text{m}^2/\text{K}$]

K conductivity matrix

R_c conduction heat resistance [$\text{m}^2\cdot\text{K}/\text{W}$]

R_s superficial heat resistance [$\text{m}^2\cdot\text{K}/\text{W}$]

T temperature field at time t

T^+ temperature field at time $t + \Delta t$

T_f fusion temperature [$^{\circ}\text{C}$]

HVAC Control

$(mCp)_r$ room indoor thermal mass [J/K]

Δp_r pressure drop of the controlled element [Pa]

Δp_v valve pressure drop [Pa]

ΔT_{LM} mean logarithmic temperature difference [K]

\dot{m} mass flow rate of water in the heater [kg/s]

a valve authority, ratio of $\Delta p_{valve}/\Delta p_{circuit}$ [-]

e error, difference between actual and set value [*unit of the measured quantity*]

K_p gain [*inverse of the unit of the controlled quantity*]

K_v flow rate through the valve [$\text{m}^3 \cdot \text{s}^{-1} \cdot \text{bar}^{-0.5}$]
 K_{vs} flow rate through the valve for 1 bar Δp [$\text{m}^3 \cdot \text{s}^{-1} \cdot \text{bar}^{-0.5}$]
 P heater power [W]
 PB proportional band [*in the unit of the controlled quantity*]
 Q outlet flow rate [m^3/s]
 Q_s, Q_{\max} supply and maximum supply flow rate [m^3/s]
 T_{set} set heating temperature [$^{\circ}\text{C}$]
 T_e, T_a external outdoor and ambient indoor temperature [$^{\circ}\text{C}$]
 T_i, T_d integration and derivation time [s]
 T_i, T_o temperature at the heater inlet and outlet [$^{\circ}\text{C}$]
 US, K heat transfer coefficient of the room envelope and of the heater [W/K]
 y control signal sent to the actuator 0 to 1 [-]

Air Quality

α apparent emission term [$\mu\text{g}/\text{m}^3/\text{train}^2$]
 β fraction of the enclosure's volume replaced by outside air owing to piston effect [-]
 δ deposition coefficient [s^{-1}]
 ρ resuspension coefficient [s^{-1}]
 τ air change rate [s^{-1}]
 C particle concentration in the air compartment [$\mu\text{g}/\text{m}^3$]
 C_e outdoor air concentration [$\mu\text{g}/\text{m}^3$]
 L quantity of particles deposited on surface [$\mu\text{g}/\text{m}^2$]
 Q_v air flowrate [m^3/s]
 S surface [m^2]
 S_v, S_h, S_c deposition surfaces over walls, floors and ceilings [m^2]
 V air volume [m^3]
 v_v, v_h, v_c particle deposition velocities over walls, floors and ceilings [m/s]

Heat & Mass Transfer

δ_v vapour diffusivity [m^2/s]
 φ vapour pressure [%]
 a, b, c parameters for the equation $w_v = w_v(\varphi)$
 C_m moisture storage capacity [$\text{kg}_w/\text{kg}/\text{Pa}$]
 $C_p(w)$ specific heat capacity depending on water content w [J/kg/K]
 F_{ow} Fourier number for vapour transfer [-]

F_{ow}^{eq} Equivalent Fourier number for vapour transfer including surface mass transfer [-]

h_a, h_b surface heat transfer coefficients [W/m²/K]

h_v vapour transfer coefficient [s/m]

L_v latent heat of vaporisation [J/kg]

p vapour pressure [Pa]

p_a, p_b vapour pressures on both sides of the wall [Pa]

p_s saturated vapour pressure [Pa]

T_a, T_b temperatures on both sides of the wall [°C]

w mass water content [kg_w/kg]

Geothermal heat pump

\dot{m} refrigerant mass flow rate [kg/s]

η_C efficiency compared to Carnot's ideal cycle [-]

λ ground conductivity [W/m/K]

ρ ground density [kg/m³]

ρ_a air density [kg/m³]

ρ_w water density [kg/m³]

C_o Courant number [-]

C_{pa} air heat capacity [J/kg/K]

C_{pw} water heat capacity [J/kg/K]

C_p ground heat capacity [J/kg/K]

COP_c, COP_h coefficient of performance in cooling and heating mode [W]

P_c, P_h cooling and heating power of the heat pump [W]

P_{e-} electrical power for the heat pump's compressor [W]

P_l linear power density at the geothermal probe [W/m]

Q_v zone air change [m³/s]

T_{in}, T_{out} indoor and outdoor temperatures [°C]

T_c, T_h evaporator (cold) and condenser (hot) temperatures [°C]

T_h^w, T_c^w heating departure and return water temperatures [°C]

$US, \rho_a Q_v C_{pa}$ envelope conduction and air change heat losses coefficient [W/K]

v_w underground water velocity [m/s]

Chapter 1

A quick reminder about numerical integration

In this chapter, we will briefly review the basics of finite volumes, using simple examples. As the topic has been thoroughly exposed in other, better works such as the famous [22], we will not go too deep into mathematical details such as proofs and Taylor series. Opening this book, we assume you roughly know what discretisation may be. Code snippets are positioned along the sections to allow the reader to have a hands-on approach of the presented numerical method.

Whatever happens, keep in mind that in the end, the trick relies merely in additions and subtractions over little cells!

1 Integration over time and space

Solving partial differential equations in time and space requires integration over those dimensions. In the following paragraphs, we will briefly describe the fundamentals of such operations.

1.1 Time integration

In many cases, a reasonable approximation of derivatives can be made with a simple slope calculation, also named Taylor's series. Consider the formal variation of temperature T over a small time increment dt :

$$\frac{dT}{dt} = \frac{T(t + dt) - T(t)}{dt} \quad (1.1)$$

Let us call T^+ the temperature at time $t + dt$ and T at time t , Equation (1.1) writes numerically, with $\Delta t \sim dt$ the approximation of the infinitesimal time increment:

$$\frac{dT}{dt} \simeq \frac{T^+ - T}{\Delta t} \quad (1.2)$$

Supposing we know the temperature variation f in [K/s] such that:

$$\frac{dT}{dt} = f \quad (1.3)$$

... we can calculate T^+ *explicitly* with the relation below, knowing T :

$$T^+ = T + f\Delta t \quad (1.4)$$

T in the previous equation could well be our initial condition such that $T(t = 0) = T_0$. Calculating $T(t = \Delta t)$ is straightforward with Equation (1.4). Using the obtained result, it is possible to determine $T(t = 2\Delta t)$ *etc. ad libitum*. If we carry on this way until the required number of time increments Δt is reached, the time integration of the equation is performed, as illustrated on Figure 1.1.

Figure 1.1 • Animation – Integration over time.

1.2 Space integration

Many problems in building physics exhibit a second order derivative in space such as the famous heat equation (e.g. " $\frac{\partial^2 T}{\partial x^2}$ "). For the sake of simplicity we will take a look at the one dimensional case. Imagine the 1D space is divided in regular small slices of size Δx , as presented on Figure 1.2.

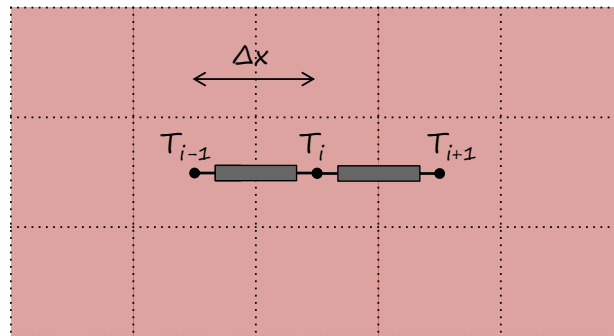


Figure 1.2 • Space discretisation of temperature T in a homogeneous material with a regular space step Δx .

A common way to obtain a second order derivative is to derive... twice. Let us first write the derivatives on the left (superscript \leftarrow) and right hand side of point a (superscript \rightarrow) :

$$\frac{\partial T^{\rightarrow}}{\partial x} \simeq \frac{T(a + \Delta x) - T(a)}{\Delta x} \quad (1.5)$$

$$\frac{\partial T^{\leftarrow}}{\partial x} \simeq \frac{T(a) - T(a - \Delta x)}{\Delta x} \quad (1.6)$$

As mentioned, the second derivative being the derivative of the first derivative* we can calculate $\frac{\partial^2 T}{\partial x^2}$ such that:

*If mathematical derivation causes you allergic outbreaks, try to get over it by repeating this sentence twice in a row.

$$\frac{\partial^2 T}{\partial x^2} = \frac{\frac{\partial T^{\rightarrow}}{\partial x} - \frac{\partial T^{\leftarrow}}{\partial x}}{\Delta x} = \frac{T(a + \Delta x) + T(a - \Delta x) - 2T(a)}{\Delta x^2} \quad (1.7)$$

Note – Equation (1.7) is called a *central* finite difference, as it takes the values on the *left* and *right* of the point considered.

Let i be the point at which the second order partial derivative is to be evaluated and $i - 1, i + 1$, its closest neighbours on the grid. Numerically Equation (1.7) translates to:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} + T_{i-1} - 2T_i}{\Delta x^2} \quad (1.8)$$

$$T^+ = T_i + \Delta t \frac{T_{i+1} + T_{i-1} - 2T_i}{\Delta x^2} \quad (1.9)$$

Equation (1.7) provides an explicit formulation of second-order derivatives depending on the values of the function in a close vicinity of the point considered, also named Euler's scheme.

With this method, starting from known initial conditions at $t = 0$, the combination of space and time integration is a simple series of sums, as illustrated on Figure 1.3.

Figure 1.3 • Animation – Integration over space and time with Euler's explicit scheme.

1.3 Formulation: *finite* what?

Finite volumes or finite difference are similar and often mistaken. A simplified definition would be the following [22]:

- **Finite differences** are obtained by directly replacing the Taylor series approximate of derivatives in the differential equation,
- whereas for **finite volumes** the domain is divided in non-overlapping points such that every node is surrounded by a so called *control volume*. The heat balance is written on each node and yields a finite volume formulation. In other words: the differential equation is integrated.

To summarise, the difference is about where we write the heat balance for each cell. Look at the left hand side of Figure 1.4, representing the boundary of a model, at the interface between air and solid. In the upper grid scheme the heat balance is written on the node directly at the interface whereas for the lower scheme the first solid node is within the material (finite volume). The same principle can be seen on the right hand side of Figure 1.4 at the interface between two materials: for finite volumes the nodes are on each side of the interface.

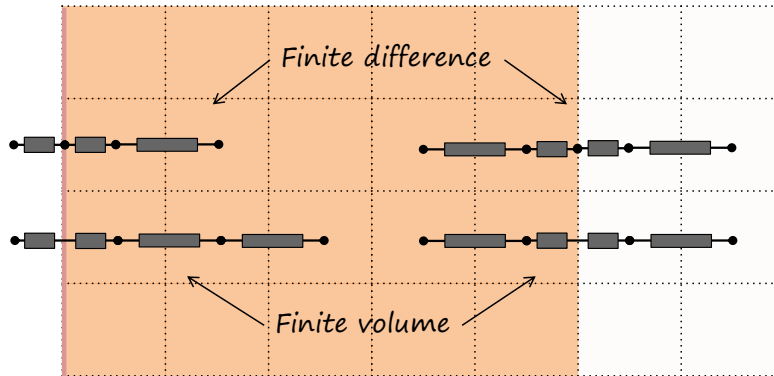


Figure 1.4 • Finite volume and finite difference at the air boundary (left) and at the interface between two materials(right).

As we will see in section 2.3, finite volumes offer a convenient method for the modelling of multilayer materials, as they allow not to significantly rewrite the equations at their interface.

However, if we stick to the previous illustration, there is no direct access to the temperature at the interface between air and solid. The latter can be computed by calculating the flux between two cells and isolating the interface temperature. Interestingly, the choice of finite difference or finite volume does not affect the accuracy of the method [22].

2 Numerical formulation of equations

In the previous sections we have introduced the numerical integration of partial differential equations and introduced the term *explicit*: let us examine how this translates on the well known transient heat equation in one dimension, governing heat transfer in a solid medium:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \quad (1.10)$$

2.1 Inside homogeneous media

In the vicinity of point i we can write the explicit formulation using the derivatives introduced above, and obtain the numerical expression of the heat equation, where the subscript i relates to the point position in the discrete space and the superscript "+" relates to the value of the field at the next time step:

$$\frac{T_i^+ - T_i}{\Delta t} = \alpha \frac{T_{i+1} + T_{i-1} - 2T_i}{\Delta x^2} \quad (1.11)$$

Introducing Fourier's non-dimensional number $F_o = \frac{\alpha \Delta t}{\Delta x^2}$, the ratio of conduction heat transfer to thermal storage in the material slice, we can develop the previous expression such that:

$$T_i^+ = T_i(1 - 2F_o) + F_o(T_{i+1} + T_{i-1}) \quad (1.12)$$

Suppose we set $T_{(x=0)} = 0$ [°C] and $T_{(x=L)} = 10$ [°C], with infinite convective coefficients at each extremity of a bar of size L , respectively at $i = 0$ and $i = n - 1$ at the other end*. Putting Equation (1.12) in a time loop for integration with an "array" implementation is as simple as the code lines below:

*For the sake of coherence with the python notation, the last index is chosen as $n-1$ instead of n .

```

# boundary conditions
T[0]=0
T[n-1]=10
# time loop
while t < sim_time:
    # inside the physical domain (i.e. not the first and last nodes)
    for i in range(1,n-1):
        T_plus[i]=T[i]*(1-2*Fo)+Fo*(T[i+1]+T[i-1])
    # replace and increment time
    T=T_plus
    t+=dt

```

Have a try: [Euler explicit \(vector version\)](#)

Some of us may be more familiar with equations written as matrices. The temperatures at both extremities $i = 0$ and $i = n$ are T_i, T_e and the heat resistances R_c, R_s defined in Equations (1.22) are equal for the sake of simplicity. Each point of the internal domain, that is $0 < i < n$, depends on its two nearest neighbours, hence the shape of the matrix is tridiagonal, as follows:

$$F_o[K][T] = F_o \times \begin{bmatrix} 0 & 0 & 0 & (...) \\ 1 & -2 & 1 & 0 & (...) \\ 0 & 1 & -2 & 1 & 0 & (...) \\ & (...) & 0 & 1 & -2 & 1 & 0 \\ & & (...) & 0 & 1 & -2 & 1 \\ & & & (...) & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} T_i \\ T_1 \\ \dots \\ T_{n-1} \\ T_e \end{bmatrix} \quad (1.13)$$

Note – One can see that the first and last lines of $[K]$ are empty: They allow us to have free hands with boundary conditions T_e, T_i .

The calculation of vector $[T_+]$ is straightforward: performing the matrix operations in Equation (1.14) allow to calculate explicitly one time step (in the previous equation, known terms are in green):

$$[T^+] = [T] + F_o[K][T] \quad (1.14)$$

We first need to build this tridiagonal matrix by assembling three diagonal matrices together*:

```

import numpy as np
# size of the square matrix
n_solid=8 # solid nodes
n=n_solid+2 # ... plus boundary conditions
#   row below diag      diag itself      row above diag
K = np.eye(n,n,k=-1)*1 + np.eye(n,n)*-2 + np.eye(n,n,k=1)*1
# put zeros in the first and last line of the tridiag. matrix
# (will be used for boundary conditions > we have n-2=8 solid nodes)
K[0,0],K[0,1]=0,0
K[-1,-1],K[-1,-2]=0,0

```

Technically, the implementation of a time and space integration is pretty simple, described in the following code lines (note that for this time we are keeping F_o out of matrix K as we will make it vary in another section):

*Please note that the resulting matrix is really sparse, roughly in the ratio of $\sim 3/n!$ Empty matrix multiplication is costly: you may use adapted libraries for computations with sparse matrix (for example `scipy's sparse lib`).

```
# explicit time loop
while t < sim_time:
    # matrix multiplication K*T
    T_plus=Fo*np.dot(K,T) + T # Fourier kept as factor of [K]x[T]
    T=T_plus # replace
    t+=dt
```

Have a try: [Euler explicit \(matrix version\)](#)

2.2 Two-dimensional formulation

In two dimensions, the heat equation writes:

$$\rho C_p \frac{dT}{dt} = \lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \quad (1.15)$$

The discretisation of the solid domain along the horizontal and vertical axis is illustrated on Figure 1.5, using respectively the subscripts i, j to localise the cells.

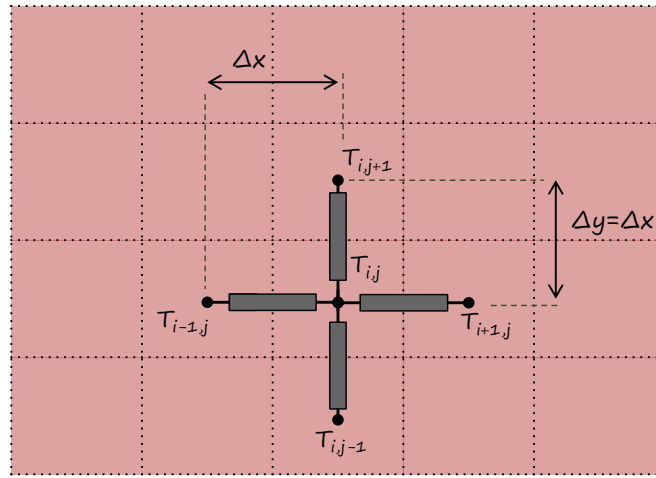


Figure 1.5 • Spatial discretisation scheme in two dimensions.

For the sake of simplicity, the space discretisation is chosen such that $\Delta y = \Delta x$. A handy means of obtaining the discretised version of equation (1.15) is to establish the heat balance at node i , consisting here in four conduction fluxes:

$$\begin{aligned} \Delta V \rho C_p \frac{T_{i,j}^+ - T_{i,j}}{\Delta t} &= \frac{\lambda \Delta S}{\Delta x} (T_{i+1,j} - T_{i,j}) \\ &+ \frac{\lambda \Delta S}{\Delta x} (T_{i-1,j} - T_{i,j}) \\ &+ \frac{\lambda \Delta S}{\Delta x} (T_{i,j+1} - T_{i,j}) \\ &+ \frac{\lambda \Delta S}{\Delta x} (T_{i,j-1} - T_{i,j}) \end{aligned} \quad (1.16)$$

In Equation (1.16), the left hand side corresponds to heat accumulation in the cell, whereas the right hand side represents the conduction heat flux from the neighbouring cells. The elementary cell volume writes $\Delta V = \Delta x^2 L$ [m³], the thickness of the discretised solid being $L = 1$ [m]. The elementary heat flux exchange surface across cells ΔS is such that $\Delta S = \Delta x L$ [m²].

Simplifying (1.16) and introducing the Fourier number where possible in Equation (1.16), one obtains:

$$T_{i,j}^+ - T_{i,j} = \frac{\lambda \Delta t}{\rho C_p \Delta x^2} ((T_{i+1,j} + T_{i-1,j} - 2T_{i,j}) + (T_{i,j+1} + T_{i,j-1} - 2T_{i,j})) \quad (1.17)$$

$$T_{i,j}^+ - T_{i,j} = F_o (T_{i+1,j} + T_{i-1,j} - 2T_{i,j} + T_{i,j+1} + T_{i,j-1} - 2T_{i,j}) \quad (1.18)$$

$$T_{i,j}^+ = (1 - 4F_o)T_{i,j} + F_o(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}) \quad (1.19)$$

Let us imagine a rectangular domain of size $n \times m$ where the temperature is imposed on the upper and lower borders, whereas adiabatic boundary conditions are set on the left and right borders, as per Figure 1.6.

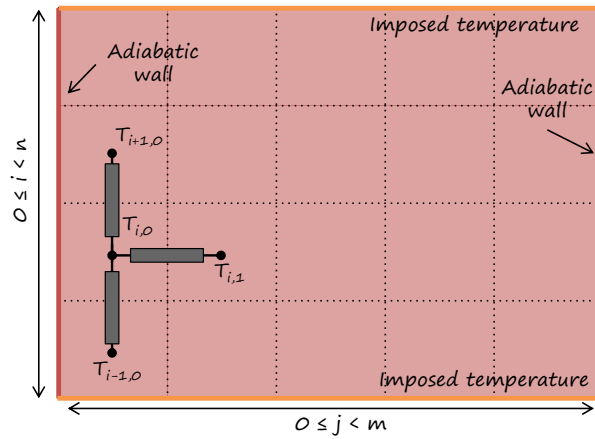


Figure 1.6 • Boundary conditions of the domain.

Determining the imposed temperature boundary condition is straightforward. Let us establish the heat balance equation for $j = 0$ in order to derive the adiabatic boundary condition, illustrated on Figure 1.6:

$$\Delta V \rho C_p \frac{T_{i,0}^+ - T_{i,0}}{\Delta t} = \frac{\lambda \Delta S}{\Delta x} (T_{i+1,0} - T_{i,0}) + \frac{\lambda \Delta S}{\Delta x} (T_{i-1,0} - T_{i,0}) + \frac{\lambda \Delta S}{\Delta x} (T_{i,1} - T_{i,0}) \quad (1.20)$$

As expected, one observes the heat flux on the left hand-side vanishes in equation (1.20). The left boundary condition hence writes:

$$T_{i,0}^+ = (1 - 3F_o)T_{i,0} + F_o(T_{i+1,0} + T_{i-1,0} + T_{i,1}) \quad (1.21)$$

A similar expression is found when writing the heat balance for $j = m - 1$. The simplest implementation of the algorithm is to use an array formulation as proposed below.

```
# time loop
while t < sim_time:
    # upper/lower boundary conditions
    for j in range(0,m):
        # inlet
        T[0,j]=T_up
        T_temp[0,j]=T[0,j]
        # outlet
        T[n-1,j]=T_low
        T_temp[n-1,j]=T[n-1,j]
    # adiabatic boundary conditions on sides
    for i in range(1,n-1):
        # j=0
        T[i,0]= T[i,0]*(1-3*Fo) \
            + Fo * (T[i+1,0] + T[i-1,0] + T[i,1])
        T_temp[i,0]=T[i,0]
        # j=m-1
        T[i,m-1] = T[i,m-1]*(1-3*Fo) + Fo*(T[i-1,m-1] + T[i+1,m-1] +T[i,m-2])
        T_temp[i,m-1]=T[i,m-1]
    # inside the domain
    for i in range(1,n-1):
```

```

for j in range(1,m-1):
    T_temp[i,j] = T[i,j]*(1-4*Fo) + Fo*(T[i+1,j] + T[i-1,j]) \
                + Fo*(T[i,j+1] + T[i,j-1])

T=T_temp
T_plus=T_temp
t+=dt

```

The matrix formulation in 2D is less intuitive, as the numbering used to access every element of the matrix must be converted from the 2D array indices to a 1D array. It is exposed in the Appendix 1.1.

2.3 Boundary conditions and interfaces

In this section, the formulations of equations at the interfaces between material for the superficial heat exchange are derived.

Superficial heat exchange

At the air interface, depicted on Figure 1.7 (left), the symbols for electrical equivalent resistance stand for the conduction heat transfer resistance over half a cell R_c and superficial global heat transfer resistance R_s combining convection and linearised radiation:

$$R_c = \frac{\Delta x}{2\lambda} \quad (1.22)$$

$$R_s = \frac{1}{h} \quad (1.23)$$

where h is the superficial heat transfer coefficient combining convection and radiation [$\text{W}/\text{m}^2/\text{K}$].

Using the heat balance method, the temperature evolution over a time increment Δt in the first solid node T_i (here $i = 1$) writes as the sum of the conductive flux to the inside of the material and the flux towards the ambience at temperature T_e :

$$\rho C_p \Delta x \frac{T_i^+ - T_i}{\Delta t} = \frac{\lambda}{\Delta x} (T_{i+1} - T_i) + \frac{T_e - T_i}{R_c + R_s} \quad (1.24)$$

If we develop Equation (1.24) and introduce an equivalent Fourier number $F_o^{eq} = \frac{\Delta t}{\rho C_p \Delta x} (R_c + R_s)^{-1}$, the heat balance reduces to:

$$T_i^+ = T_i(1 - F_o - F_o^{eq}) + F_o T_{i+1} + F_o^{eq} T_e \quad (1.25)$$

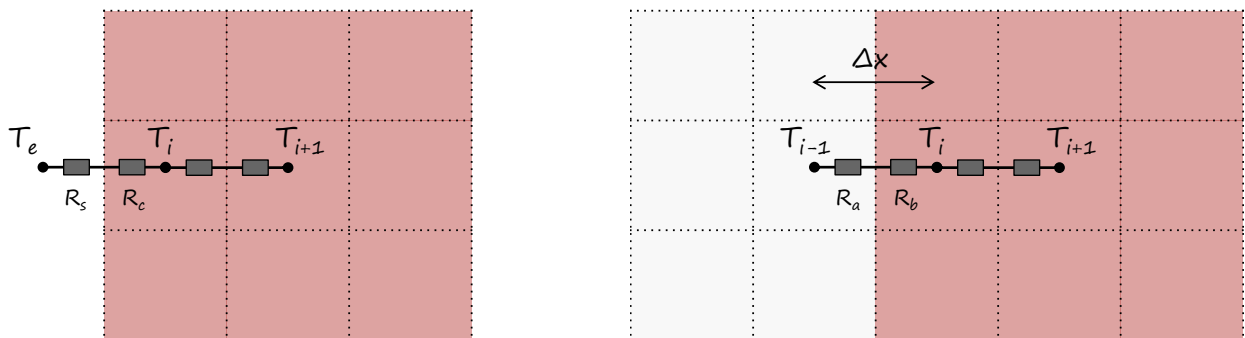


Figure 1.7 • Position of the nodes at the air boundary (left) and at the interface between two materials a and b (right).

The values of matrix' $[K]$ coefficients are to be written as follows, keeping the first line nil in order to impose the boundary condition:

$$[K][T] = \begin{bmatrix} 0 & 0 & 0 & (...) \\ F_o^{eq} & 1 - F_o - F_o^{eq} & F_o & 0 & (...) \\ 0 & F_o & -2F_o & F_o & 0 & (...) \\ (...) & (...) & (...) & (...) & (...) & (...) \end{bmatrix} \begin{bmatrix} T_e \\ T_i \\ T_{i+1} \\ (...) \end{bmatrix} \quad (1.26)$$

Note – The superficial heat transfer h , often combines convection and linearised radiation in building physics: assuming $h_r \simeq 4\sigma\epsilon T_m^3$ which is pretty accurate below 100 [°C]. It is then possible to write the flux at the interface as a linear relation $\varphi = (h_c + h_r)(T_e - T_i)$.

Interface between materials

At the interface of materials a and b with conductivities λ_a, λ_b and local conduction resistances R_a and R_b shown on Figure 1.7 (right), let us introduce a local mean Fourier number F_o^m . The latter appears between nodes $i - 1$ and i , the interface being halfway at $\Delta x/2$:

$$F_o^m = \frac{\Delta t}{\rho C_p \Delta x^2} \frac{1}{\frac{1}{\lambda_a} + \frac{1}{\lambda_b}} = \frac{\Delta t}{\rho C_p \Delta x^2} \frac{2(\lambda_a + \lambda_b)}{\lambda_a \lambda_b} \quad (1.27)$$

The local Fourier number of material b is :

$$F_o^b = \frac{2\Delta t}{\rho C_p \Delta x^2} \frac{1}{\frac{1}{\lambda_b} + \frac{1}{\lambda_b}} = \frac{\alpha_b \Delta t}{\Delta x^2} \quad (1.28)$$

The balance equation writes as:

$$T_i^+ = T_i(1 - F_o^b - F_o^m) + F_o^b T_{i+1} + F_o^m T_{i-1} \quad (1.29)$$

Under another shape, the matrix $[K]$ for two materials delimited around the position of T_m can be written as follows, imposing T_a, T_e at the external boundaries:

$$[K][T] = \begin{bmatrix} 0 & 0 & 0 & (...) \\ F_o^a & -2F_o^a & F_o^a & 0 & (...) \\ 0 & F_o^a & -2F_o^a & F_o^a & 0 & (...) \\ (...) & F_o^a & -(F_o^a + F_o^m) & F_o^m & -(F_o^m + F_o^b) & F_o^b & 0 \\ (...) & 0 & F_o^m & -(F_o^m + F_o^b) & F_o^b & -2F_o^b & F_o^b \\ (...) & (...) & 0 & F_o^b & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_a \\ T_1 \\ T_m \\ T_{m+1} \\ T_e \end{bmatrix} \quad (1.30)$$

The advantage of finite volumes in such cases is that a generic formulation can be used. Indeed, with an interface located right in the middle of two cells, there is no need for rewriting the equations at each interface between materials, which is particularly interesting for multi-layered materials.

Taking again matrix $[K]$, we can now modify it as follows in order to represent a two-layers material:

```
# let's build the conductivity matrix
K=np.eye(n,n,k=-1)*1 + np.eye(n,n)*-2+ np.eye(n,n,k=1)*1
K[0,0],K[0,1],K[-1,-1],K[-1,-2]=0,0,0,0 # ghost lines for boundary conditions
coeffs_Fo=np.ones(len(T)) # an array to affect the upper/lower half of matrix K
coeffs_Fo[0:i_interface+1]=Fo1 # prepare the upper part of the matrix
coeffs_Fo[i_interface+1:]=Fo2 # ... the lower part of the matrix
K=coeffs_Fo*K
# diagonal terms around interface
```

```

K[i_interface,i_interface]=-(Fo1+Fo_eq)
K[i_interface+1,i_interface+1]=-(Fo2+Fo_eq)
# sub and supra diag terms around interface
K[i_interface,i_interface+1]=Fo_eq
K[i_interface+1,i_interface]=Fo_eq

```

Note – Unfortunately, in real life, the interface seldom falls right in the middle between two nodes. The following formulation, derived from the steady state conduction heat balance, is to be preferred in this case [22]:

$$F_o^m = \frac{\Delta t}{\rho C_p \Delta x^2} \frac{1}{\frac{\Delta x_a}{\lambda_a} + \frac{\Delta x_b}{\lambda_b}} \quad (1.31)$$

where Δx_a and Δx_b are respectively the distance of the interface with the first node in material a and b such that $\Delta x_a + \Delta x_b = \Delta x$.

In this case the local Fourier number needs to be adapted within the matrix as well, as the volume of the two cells around the interface is no more equal.

The code below provides a simple technique to find the nodes between which the interface of two materials is positioned and compute the equivalent Fourier number.

```

# we know the abscissa of the interface L_layer
# find the node left of the interface
i_interface=int(L_layer/dx) # lowest integer ='floor' rounding
# find the distance dx1 between the node and the interface
dx1=x_interface%dx # the leftover of the division
# compute dx2, distance between interface and node i+1
dx2=dx-dx1
# compute lambda_eq at the interface
k_eq=1/(dx1/lambda1+dx2/lambda2)
# equivalent Fourier number Fo_eq
Fo_eq=dt*k_eq/(rho1*Cp1*dx**2)
...
# plot it, it may help!
plt.arrow(L_layer,0,0,10)

```

3 Overcoming stability issues

Although the explicit scheme is very intuitive and relatively easy to implement, it has the drawback of *conditional stability*. As detailed stability analyses of the numerical schemes can be found in an abundant literature, we will focus here on a practical way of getting your model to work, either respecting the explicit stability condition, or changing the numerical scheme for a more stable one (in our case with a higher order as well).

3.1 Stability for Euler's explicit scheme – A cookbook condition

Stability is obviously conditioned by:

- the physical problem,
- the material properties,

- the integration scheme used,
- the discretisation,
- the dimension in space.

A simple and efficient memento is “the term factoring T has to remain positive”. It is actually a condition for the matrix to staying diagonally dominant and hence inversible*. Taking Equation (1.12) as an example translates to the most famous stability condition in numerical methods. In the solid domain the term factoring T is the following:

$$(1 - 2F_o) \geq 0 \quad (1.32)$$

Isolating Fourier’s number yields:

$$F_o \leq \frac{1}{2} \quad (1.33)$$

$$\frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2} \quad (1.34)$$

$$\Delta t \leq \frac{\Delta x^2}{2\alpha} \quad (1.35)$$

In other terms, going back to the definition of Fourier’s number, the condition $F_o \leq 0.5$ for pure 1D conduction means the quantity of heat transferred through the control volume cannot excess half its thermal storage capacity.

Note – The stability condition for the 2D scheme presented in Section 2.2 writes similarly as in one dimension:

$$1 - 4F_o \geq 0 \quad (1.36)$$

$$\Delta t \leq \frac{\Delta x^2}{4\alpha} \quad (1.37)$$

The explicit scheme in 2D is hence twice more restrictive than the 1D for mere conduction (noticeably, for the 3D scheme the maximum Δt is to be three times smaller than the 1D limit).

For the model including superficial heat transfer between air and solid, a second equation arises. A more restrictive stability condition appears, as F_o^{eq} is strictly positive:

$$(1 - F_o - F_o^{eq}) \geq 0 \quad (1.38)$$

$$1 - \frac{\Delta t}{\Delta x^2} \left(\alpha + \frac{\Delta x}{\rho C_p} (R_c + R_s)^{-1} \right) \geq 0 \quad (1.39)$$

In the application of the method, we will calculate the stability condition of Euler’s scheme with another physical example.

For Euler’s scheme, increasing Δx may be an option to respect condition (1.39), potentially at the cost of precision. The alternative would be to reduce Δt , at the expense of computational time. Next section exposes another integration to overcome the stability issue.

3.2 Crank-Nicolson’s scheme

Euler’s explicit scheme has the advantage of simplicity and straightforward implementation, however its stability conditions are often detrimental to execution time. A means of increasing stability is to change the explicit numerical scheme for an *implicit scheme*.

*May Mr. A. Triboix be thanked here for this efficient mnemonic.

The principle of implicit numerical schemes is to use a linear combination between the value of the simulated field at the current time step (for instance T) and the field at the next time step (T^+): the formulation is said to be *implicit* because an equation must be solved at each time step, whereas for the explicit scheme it is obtained by mere addition.

Let us have a closer look at Crank-Nicolson's scheme. The central finite difference in space *and* time with a time step $\Delta t/2$ applied to Equation (1.10) yields following result, the notation $T^{+1/2}$ being the value of field T at time $t + \Delta t/2$:

$$\frac{T_i^+ - T_i}{\frac{2\Delta t}{2}} = \alpha \frac{T_{i+1}^{+1/2} + T_{i-1}^{+1/2} - 2T_i^{+1/2}}{\Delta x} \quad (1.40)$$

Proceeding with half time steps is not especially convenient. Let the value of field $T^{+1/2}$ be approximated as the average between the field value at the current time step T and the one at the next time step T^+ :

$$\frac{T_i^+ - T_i}{\Delta t} \simeq \alpha \frac{(T_{i+1}^+ + T_{i-1}^+ - 2T_i^+) + (T_{i+1} + T_{i-1} - 2T_i)}{2\Delta x} = \frac{\alpha}{2} \frac{T_{i+1}^+ + T_{i-1}^+ - 2T_i^+}{\Delta x} + \frac{\alpha}{2} \frac{T_{i+1} + T_{i-1} - 2T_i}{\Delta x} \quad (1.41)$$

A visual interpretation of the differences between the explicit and implicit schemes is provided on Figure 1.8. One can observe the values in time and space involved in the computation of the field for each method.

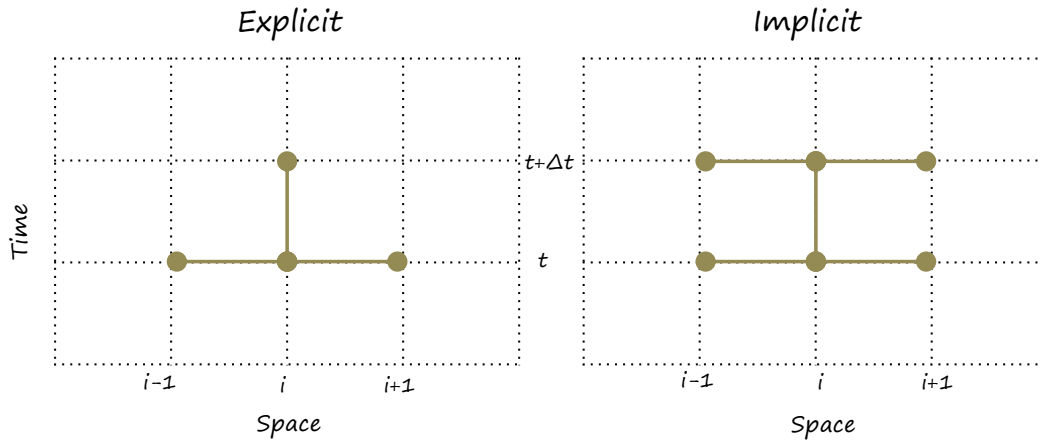


Figure 1.8 • Explicit *versus* implicit, a graphical comparison of the time and space steps involved.

Equation (1.41) means that T^+ is a combination of the known, explicit field T and the unknown, implicit field T^+ . As the contributions of both terms are weighted by $1/2$, it makes for its name *semi-implicit*. Put under matrix form, following equation is Crank-Nicolson's scheme with the known field in green and the unknown in orange:

$$[T^+] = [T] + \frac{1}{2}[K][T] + \frac{1}{2}[K][T^+] \quad (1.42)$$

Note – This method is of order 2 in time and space, stable irrespectively of Δt (this does not mean that the results are accurate with a large time step however), which is an interesting feature for transient problems where the error adds up at every time step. Its formulation imposes to solve for $[T^+]$ at each time increment.

Crank-Nicolson's scheme is actually a particular form of implicit schemes (1.43). The latter are integration schemes that combine a given fraction of fields T and T^+ depending on the so-called *relaxation factor*

β . Mathematically, they write:

$$[T^+] = [T] + (1 - \beta)[K][T] + \beta[K][T^+] \quad (1.43)$$

Where $[K]$ is the matrix defined in the previous section, containing the Fourier numbers and heat transfer coefficients. The relaxation factor β defines the amount of each field in the solution. The value $\beta = 1$ is a fully implicit scheme, whereas $\beta = 0$ is fully explicit. Tuning β may be a means of enabling or accelerating convergence for some problems. Figure 1.9 proposes a visual interpretation of implicit integration schemes.

Figure 1.9 • Animation – Integration over space and time with Crank-Nicholson’s semi-implicit scheme.

Let us have a look at how this translates in terms of code, using `scipy.optimize.fsolve`, a very practical function to solve equations and systems of equations.

First of all, we need to define a function returning “zero”, as `fsolve` looks for zeros, which we achieve by putting the right hand side of equation (1.42) to the left hand side:

```
# function for Crank-Nicolson's scheme
# (where Tp is the unknown T^+)
def fc_CN(Tp, T, K, beta, Fo):
    return Tp - T - beta*Fo*np.dot(K, T) - beta*Fo*np.dot(K, Tp)
```

We will then use a similar loop as for explicit integration, with a call to `fsolve` at each time step:

```
# time loop
while t < sim_time:
    #call to fsolve with T_CN as initial guess
    T_plus_CN = fsolve(fc_CN, T_CN, args=(T_CN, K, beta, Fo))
    T_CN = T_plus_CN # replace
    t+=dt
```

A qualitative comparison of both explicit and semi-implicit integration methods is given for pure conduction in Figure 1.10, showing the temperature profile in a one-dimensional bar. The Fourier number is chosen intentionally above the stability limit of Euler's scheme (numerically $Fo = 0.54 > 0.5$). One can observe that the explicit scheme oscillates, whereas the semi-implicit one is stable.

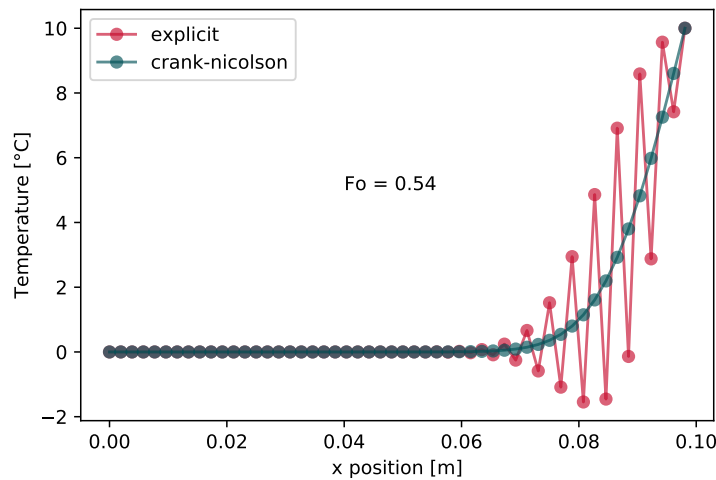


Figure 1.10 • Euler explicit *versus* Crank-Nicolson semi-implicit for conduction ($Fo > 0.5$ and $t = 600$ [s]).

Have a try for yourself with a web browser: [Crank-Nicolson with two layers and superficial heat transfer \(notebook\)](#).

Copy/paste the code to your favorite development environment: [Crank-Nicolson with two layers and superficial heat transfer \(matrix version\)](#).

Note – Regarding the unconditional stability of Crank-Nicolson's scheme, an important additional information must be exposed, however it is often missing in lecture notes about numerical methods. Let us cite [12], themselves quoting Patankar [22], hence the word for word reproduction:

An inexperienced user often interprets this [the unconditionally stable property] to imply that a physically realistic solution will result no matter how large is the time step, and such user is, therefore surprised to encounter oscillatory solutions. The 'stability' in a mathematical sense simply ensures that these oscillations will eventually die out, but it does not guarantee physically plausible solutions.

In other words, Crank-Nicolson's scheme does not guarantee the obtention of physical, non-oscillatory solutions: Take note!

4 A word about computer programming

4.1 Debugging

Part of the fun is to run through errors. The following simple principles below may help debugging your programs. In case of divergent results:

- Reduce the time step and while the computation runs, verify if you respect the analytical stability condition.
- Computing the evolution over a small number of time steps and looking at where in the domain the values start diverging might help finding the culprit*.
- Change the numerical scheme for an unconditionally stable one.

If the results are in the correct order of magnitude but do not match the expected values or profiles:

- Put the same boundary conditions on each side of the domain and see if the behaviour is symmetrical.
- Reduce the number of phenomena: *e.g.* if the model has sources, turn them off and see if it solves the problem.
- Reduce complexity: if you have several layers, try with one only, etc.

A stepwise increase in the complexity of the model is recommended.

4.2 Before starting

The model runs without errors, excellent! Now what about doing a few elementary checks before going further? Following points may be a decent check-list to start with:

- Is there an analytical solution that could help verify* the code?
- Do I get physical results? That is: if I increase the conductivity or heat transfer coefficient, does it match with what would happen in real life?
- Change the time step Δt : are the results somewhat similar?
- Same question with the space discretisation Δx .
- If the model is transient, can I reach a steady state?
- Optionally: use the `time` package to monitor the execution time.

If you intend to make a prediction, it is a healthy habit to run several combinations of possibly high and possibly low values of the model parameters in order to establish the lower/upper bounds of the results.

Last but not least, remember what you are playing with [7]:

"Essentially all models are wrong, but some are useful."

*The latter often sits between the chair and the keyboard.

*We distinguish here *verification* – are the equations properly solved? – and *validation*, for instance against experimental data – are the proper equations used to model the phenomenon?

Chapter 2

Transient problems

Many a problem in building physics exhibits a time dependency. In this chapter, a few of them are examined: an application about phase change materials is first detailed, after what time-dependent phenomena in HVAC control are shown. The third section addresses air quality and filtration using an experimental data set.

1 Phase Change Materials

Increasing thermal mass in buildings is a tangible means of reducing cooling or heating loads. The sensible heat storage capacity of generic construction materials is limited, usually around $\sim 10^3$ [J/kg/K], as is the reasonable thickness of walls. A sound idea is to take advantage of the latent heat released by materials that change phase at ambient temperatures, *e.g.* mixtures of paraffine, releasing $\sim 10^5$ [J/kg] at about 27 [°C].

Over the past decades, numerous applications of phase change materials have been developed for thermal storage in buildings and thermal systems. In this section, we will expose a numerical implementation for conduction in materials including phase change material (PCM), using the *apparent thermal capacity* method, introducing a slight increment in complexity compared to the examples of Chapter 1.

1.1 Modelling phase change in a wall

Compared to sheer conduction, the specificity of phase change is the following: the latent heat of fusion must be taken into account as it ranges roughly between $20 \leq L_f \leq 200$ [kJ/kg] depending on the materials commonly used in construction.

Phase change occurs at a given temperature T_f and absorbs the latent heat of fusion L_f or releases heat during solidification. For pure matter, phase change occurs at a constant temperature, whereas for mixtures this phenomenon happens over a few Kelvins (see for example the experimental curve in [18]).

A simple and efficient way for the implementation of PCM in conduction heat transfer problems is the variable C_p method, which we will explain here. The idea is to find a mathematical artefact enabling the repartition of the latent heat L_f on a small temperature interval ΔT_f , over which the material gradually turns liquid. The procedure is as follows:

- Find a function whose integral is equal to unity over a given, small temperature range and multiply it by the latent heat.
- Determine a function driving the liquid fraction within the material depending on temperature.
- Compute the apparent specific heat capacity C_p , a weighted average of the sensible and latent heat of the material L_f .

The specific heat capacity varies depending on temperature and incorporates the solid heat capacity, the liquid heat capacity and the latent heat gained or lost in the process: let us have a look at the numerical

means of putting this together.

Gaussian distributions have interesting properties regarding integration. The following Gaussian distribution $d(T)$ centred in T_f has an integral equal to unity: it is a numerical equivalent of a Dirac.

$$d(T) = \frac{e^{-\frac{(T-T_f)^2}{\Delta T_f^2}}}{\sqrt{\pi\Delta T_f^2}} \quad (2.1)$$

where T_f is the fusion temperature and ΔT_f the half fusion temperature range [6], meaning the material melts over $2\Delta T_f$. It can be chosen arbitrarily small around $0.5 \sim 1$ [K], as the range does not influence significantly the results. What is more, commercial PCM are generally mixtures, for which melting occurs over a range of temperature, unlike pure matter. Figure 2.1 shows an example of a distribution $d(T)$ (the red solid line).

Let us define the liquid fraction f as a piecewise function, also illustrated on Figure 2.1 (green line):

$$\text{Below fusion } T < T_f - \Delta T_f \rightarrow f = 0 \quad (2.2)$$

$$\text{Above fusion } T > T_f + \Delta T_f \rightarrow f = 1 \quad (2.3)$$

$$\text{During phase change } T_f - \Delta T_f \leq T \leq T_f + \Delta T_f \rightarrow f = \frac{T - T_f + 2\Delta T_f}{4\Delta T_f} \quad (2.4)$$

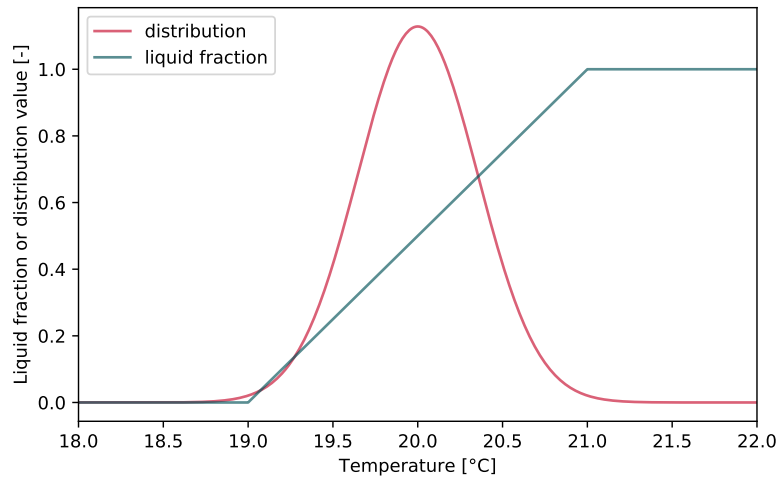


Figure 2.1 • Liquid fraction and distribution function for the apparent capacity method using $T_f = 20$ [°C] and $\Delta T_f = 0.5$ [K].

Eventually, the apparent heat capacity writes as the sum of the latent heat L_f by the distribution over $2\Delta T_f$, the sensible heat of the solid fraction C_p^s and the sensible heat of the liquid fraction C_p^l :

$$C_p(T) = d(T)L_f + C_p^s(1 - f) + C_p^l f \quad (2.5)$$

An arbitrary example of application of Equations (1.1), (2.2) and (2.5) is plotted on Figure 2.2 for a fusion temperature of 20 [°C] and a melting range $\Delta T_f = 0.5$ [K]. Note the drastic increase in apparent thermal capacity around $T_f \pm \Delta T_f$, from ~ 1 [kJ/kg/K] to 120 [kJ/kg/K].

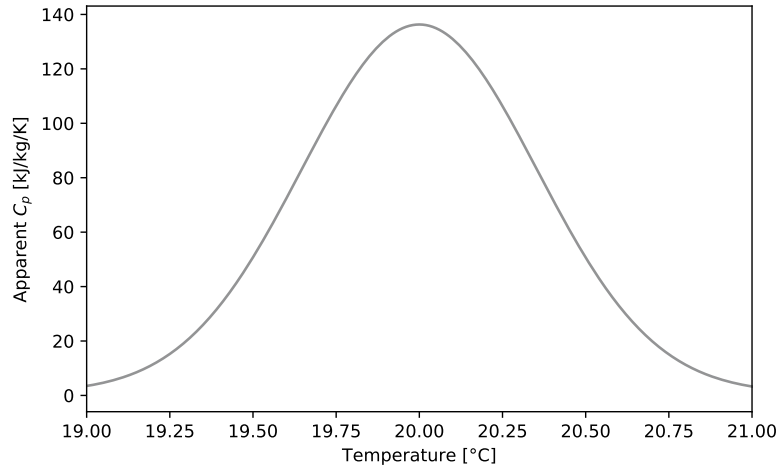


Figure 2.2 • Apparent C_p in the vicinity of the fusion temperature ($T_f = 20$ [°C], $\Delta T_f = 0.5$ [K] and $L_f = 120$ [kJ/kg]).

1.2 Numerical model

What about the stability of the explicit scheme with a variable C_p ? The present case is similar to the conduction case shown in Equation (1.34), where the stability depends on the Fourier number:

$$Fo \leq \frac{1}{2} \Leftrightarrow \frac{\lambda \Delta}{\rho C_p(T) \Delta x^2} \leq \frac{1}{2} \quad (2.6)$$

Given the expression of F_o with C_p at the denominator in the previous equation, when C_p increases, F_o decreases: for the scheme to stay stable, we must verify that it satisfies the criterion for the lowest value of C_p , that is $C_p = \min(C_p^l, C_p^s)$ (the apparent C_p including the latent heat contribution is not mentioned, being obviously much higher).

Technically, we will create a local Fourier number $Fo[i]$, which will be updated at each time step. For the sake of readability, we present below the "vector" formulation rather than the matrix form:

```
for i in range(1,n-1):
    T_plus[i]=T[i]*(1-2*Fo[i])+Fo[i]*(T[i+1]+T[i-1])
    # update the local Fourier numbers
    for i in range(1,n-1):
        # first local apparent Cp (with a function for readability)
        Cp_t[i] = fc_Cp_apparent(T_plus[i],dTf,Tf,Lf,Cp_s,Cp_l)
        # then update Fourier
        Fo[i]=k*dt/(rho*Cp_t[i]*dx**2)
```

The code for Euler's explicit method exposed in Section 2 is directly reusable by adding a local Fourier number: let us make a test with a 10 [cm] broad sample of material containing PCM inclusions such that its apparent properties are as follows:

- liquid heat capacity $C_p^l = 2400$ [J/kg/K]
- solid heat capacity $C_p^s = 1800$ [J/kg/K]
- latent heat of fusion $L_f = 180$ [kJ/kg]
- fusion temperature $T_f = 27$ [°C] and imposed temperatures of 20 and 30 [°C] as boundary conditions

We neglect the variation of conductivity between the solid and liquid state ($\lambda_l = \lambda_s$.)

The result of the simulation is given below on Figure 2.3, where the temperature profile after 6 minutes is presented. Note the typical line break in the vicinity of $x \sim 7$ [cm], corresponding to the phase change temperature of $T_f \simeq 27$ [°C] (represented with a solid line on the graph).

Figure 2.3 • Temperature profile in the board after 6 min.

A web browser-based application can be found here: [10 cm PCM board with \$T_f = 27^\circ\text{C}\$ \(notebook\)](#)

The code provided with following link should allow you to dabble with phase change simulation: [10 cm PCM board with \$T_f = 27^\circ\text{C}\$](#)

Note – Using the code “as is” is the equivalent of imposing the temperature of the solid nodes at the extremities of the model.

1.3 Application – Comparison of temperature profiles

Imagine a laboratory test to compare the previous material with and without PCM inclusions. Two plates are maintained at a constant temperature on each side of the sample. The thermophysical properties of the material without PCM are supposed to be the ones of the solid state (that is $C_p^s = 1800$ [J/kg/K]).

We would like to simulate the temperature evolution with an imposed temperature gradient that is, with 20 [°C] and 30 [°C] respectively on each side. The initial condition is of 20 [°C] and the fusion temperature is $T_f = 27$ [°C].

The result of the simulated temperature evolution after 3 hours is plotted on Figure 2.4, where one can observe an inflection around the fusion temperature for the material with PCM inclusions. The temperature profile without PCM is slightly “ahead”, as its thermal mass is lower.

Figure 2.4 • Temperature with and without PCM after 6 hours.

Continue exploring...

Question 1

So far we have neglected the variation of conductivity (supposing $\lambda_l \sim \lambda_s$): change the model so that $\lambda_l \neq \lambda_s$ and $\lambda = f\lambda_l + (1 - f)\lambda_s$.

Question 2

Build a fusion front tracking routine and plot the evolution of the fusion interface over time. You may want to compare the results with Stefan's analytical formula – explained for instance [in this paper](#) .

Question 3

Change the matrix for a sparse formulation (use `scipy.sparse`). What is the speedup?

2 Indoor Air Quality

In HVAC, maintenance may be overlooked, leading to a significant impact on air systems. In this section, we propose to investigate air filter clogging, estimating its effect on filtration efficiency, fan power consumption and air quality, with measured outdoor air concentrations for PM_{2.5}.

The studied problem is a room with filtered air supply, represented on Figure 2.5. We assume the room volume to be $V = 2000 \text{ m}^3$. Outside air is supplied at flow rate $Q_v = 2000 \text{ [m}^3/\text{h]}$ at the outdoor concentration C_e , while C_s is the concentration of the filtered supply air. The filter characteristics depending on clogging (pressure drop Δp and efficiency η) are taken from the manufacturer's technical sheet, and plotted below on Figure 2.6.

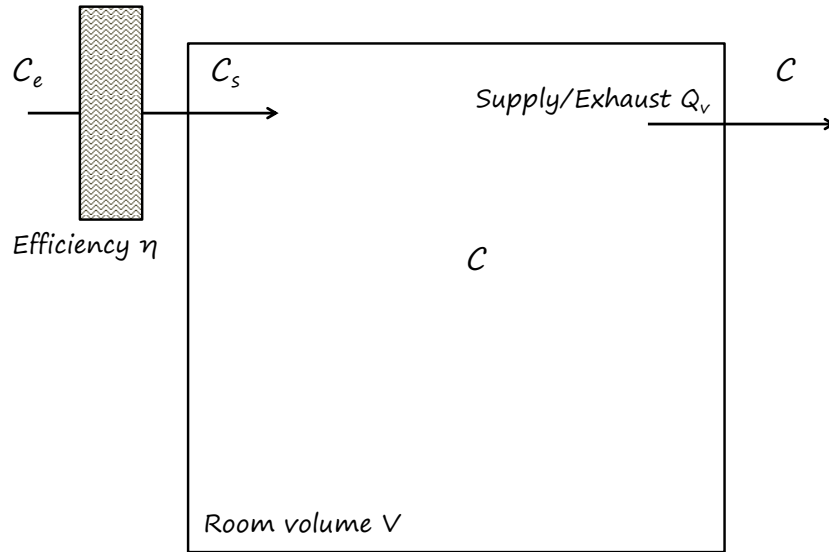


Figure 2.5 • Scheme of the enclosure with air filtration.

The questions we will address in the next pages are the following:

- What is the energy cost of the clogging-related pressure drop increase?
- How does clogging affect air quality?
- What is the best maintenance frequency for this system's filter?

In order to determine an appropriate answer, the construction of a model for the problem is presented in the following section.

2.1 Modelling filter clogging

We will use a so-called "nodal" model, meaning we suppose that the air within the room is fully mixed, with an homogeneous concentration C . Let us write the mass balance of the pollutant concentration in the room, with C_s the supply air concentration*:

$$V \times \frac{dC}{dt} = C_s Q_v - C Q_v \quad (2.7)$$

If we introduce the air change rate $\tau = Q_v/V \text{ [s}^{-1}\text{]}$ in Equation (2.7), we obtain:

$$\frac{dC}{dt} = \tau(C_s - C) \quad (2.8)$$

*Yes, $V \times C$ the room volume multiplied by its concentration is homogeneous to a mass $\text{m}^3 \times \mu\text{g}/\text{m}^3 = \mu\text{g}$

Given the filter efficiency $\eta = \frac{C_e - C_s}{C_e}$, the concentration of the supply air is simply:

$$C_s = (1 - \eta)C_e \quad (2.9)$$

Calculating the mass accumulated in the filter at each time step is straightforward with :

$$m = Q_v \Delta t (C_e - C_s) [\mu\text{g}] \quad (2.10)$$

When the particle mass accumulated in the filter increases, the pressure drop in the clogging porous medium rises, as represented on Figure 2.6 (left – polynomial curve fit from the manufacturer, provided in the code repository). The filtration efficiency is also affected and decreases with the accumulation of mass in the filter, as plotted on Figure 2.6 (right). The pressure drop and efficiency will be updated depending on the mass accumulated in the filter.

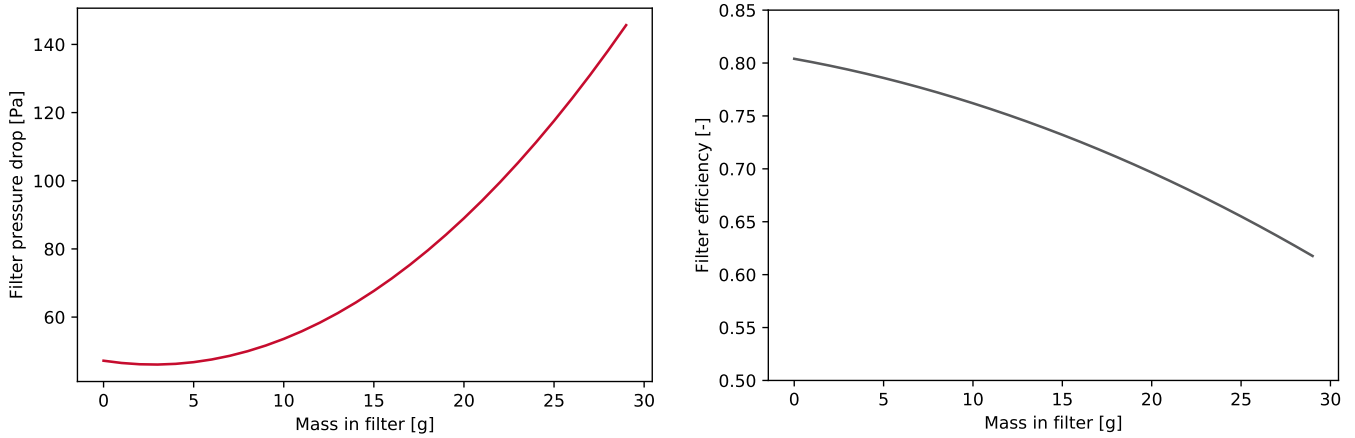


Figure 2.6 • Pressure drop (left) and filtration efficiency (right) depending on the accumulated mass in the filter.

Note – About the effect of pressure drop on ventilation: physically speaking, the fan pressure and system pressure drop adjust to reach the operating point. In the present application, we will suppose that the fan curve is very "steep" and that the flowrate is not impacted by the increase in pressure drop (hence Q_v remains constant throughout the simulation).

2.2 Numerical model

The discretised Equation (2.8) writes numerically when isolating C^+ :

$$C^+ = C + \tau \Delta t (C_s - C) \quad (2.11)$$

Respecting the "cookbook" stability condition presented in Chapter 1 Section 3.1 implies:

$$(1 - \tau \Delta t) > 0 \quad (2.12)$$

$$\Delta t < \frac{1}{\tau} \quad (2.13)$$

At each time step, we will compute the indoor air concentration in PM_{2.5} using a varying outdoor air concentration and updating the filter efficiency η . The mass in the filter is also updated, as well as the increase in pressure drop compared to a clean filter. A proposal of implementation is shown below.

```
Cext=np.loadtxt("PM25.txt") # load data
nb=len(Cext) # how many data points
```

```

# prepare a few vectors for the computation
Csupply=np.zeros(nb) # supply concentration
C=np.zeros(nb) # room concentration
m_filter=np.zeros(nb) # initially nothing in the filter
eta_filter=np.ones(nb)*eta # eta_filter[0] actually (updated in time loop)
pdc_filter=np.zeros(nb) # filter pressure drop

Csupply[0]=Cext[0] # aesthetical fill (for plotting)
m_filter[0]=1e-4 # against div by 0
pdc_ref=fc_pressure_drop(0) #pressure drop of clean filter

# explicit euler: loop over time
for i in range(1,nb):
    # compute efficiency
    eta_filter[i]=fc_eta(m_filter[i-1])
    # compute the additionnal pressure drop compared to clean filter
    pdc_filter[i]=fc_pressure_drop(m_filter[i-1])-pdc_ref
    # supply PM2.5 concentration
    Csupply[i]=(1-eta_filter[i])*Cext[i]
    # mass in the filter, converted to grams as C is in micrograms
    m_filter[i]=(Cext[i]-Csupply[i])*qv/1e6 + m_filter[i-1]
    # check if we change the filter (criterion on m_limit, limit mass in the filter)
    if m_filter[i]>m_limit:
        m_filter[i]=0 # reset the mass
        nb_filt+=1 # count one more filter
    #explicit euler scheme for C
    C[i]=C[i-1] + dt*tau*(Csupply[i]-C[i-1])

```

The results obtained with this model are presented in the following section.

2.3 Simulation results for a given critical mass

The freely available hourly PM_{2.5} measured concentration in Paris [from Airparif's website](#), is the outdoor air concentration data retained. The hourly values over the year are plotted on Figure 2.7:

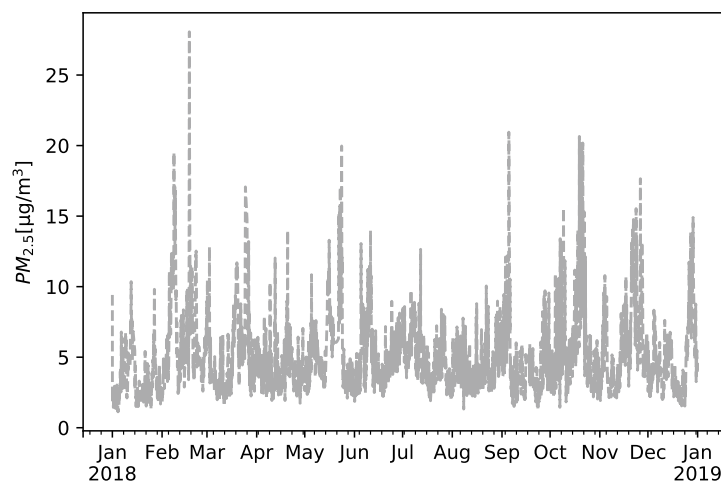


Figure 2.7 • Yearly outdoor PM_{2.5} concentration (Airparif 2018).

The computations presented in the previous section were performed using this PM_{2.5} dataset. We defined a *critical mass* of pollutant in the filter of $m_c = 30$ [g]: when reached, the filter is changed in the model and the accumulated mass reset to zero.

In the sequel, for the sake of readability of the data sets, the average hourly concentration over the week is computed from the yearly results and serves as a means of analysis or comparison for air quality. Indeed, observing the concentration differences on Figures such as 2.7 is difficult.

Figure 2.8 (left) shows the weekly concentration profile outdoors and within the room for $m_c = 30$ [g]. The rise in pressure drop is presented on Figure 2.8 (right). One can notice that the increase reaches 100 [Pa], which is more than half the average pressure drop of the filter (see Figure 2.6).

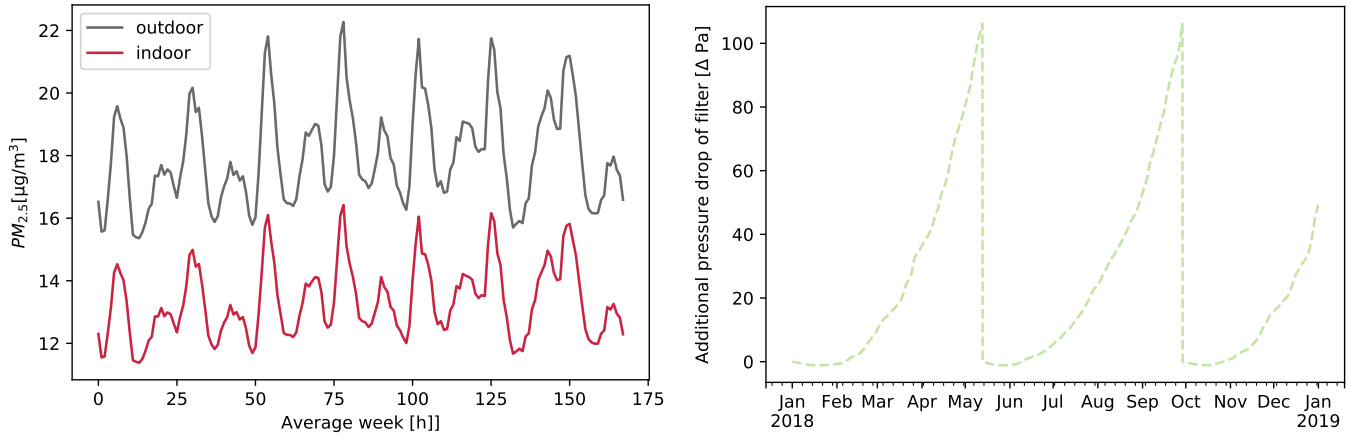


Figure 2.8 • Average weekly concentration, outdoor versus indoor concentration (left) and increase of pressure drop related to clogging compared to a clean filter (right).

On Figure 2.9 (left) the filtration efficiency is plotted over the year. The regular increases correspond to the moment when the filter is replaced. On the right plot, the mass accumulated in the filter is shown, limited here to $m_c = 30$ [g].

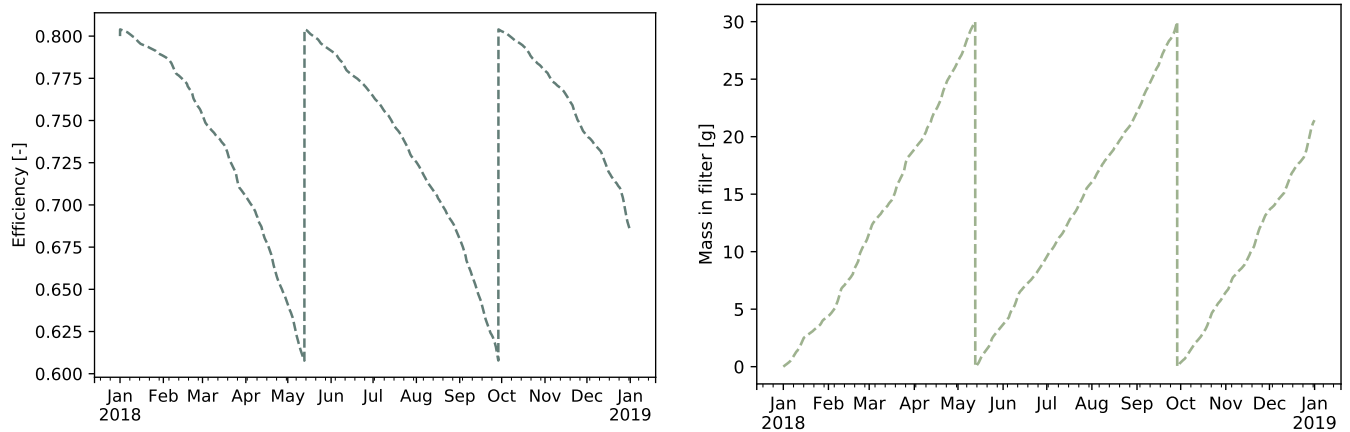


Figure 2.9 • Filtration efficiency (left) and accumulated mass in the filter over time (right).

2.4 Cost or air quality?

In this section, we will investigate the impact of the "moment" at which the filter is changed, represented as a particle mass threshold, both on cost and air quality.

From the model, it is possible to compute an estimate of the operation and maintenance costs c by multiplying the hourly power consumption by the energy cost c_{energy} and adding a unitary service cost

c_{filter} each time the filter is replaced (n_{filter}):

$$c = c_{\text{energy}} \Sigma P(t) \Delta t + c_{\text{filter}} n_{\text{filter}} \quad (2.14)$$

where $c_{\text{energy}} = 15$ [€ct/kWh] and $c_{\text{filter}} = 30$ [€/filter]. The unitary cost of the filter is low, however we suppose other filters would be changed at the same time.

As the relation between mass increase and pressure drop is not linear, unlike the maintenance cost, the result is given for a wide range of accumulated mass in the filter on Figure (2.10). The analysis of the values obtained can be summarised with the following observations:

- Although the energy cost is pretty low, the increase in pressure drop can still be an argument to change filters if the unitary servicing costs are around 30€/intervention (play around with the code to make your own mind about it: you may find interesting to observe the non-linearity of the problem). In this case, a trade-off in terms of costs would probably be to select a value around $m_c \sim 30$ [g], the lower being the better in order to preserve air quality.
- Noticeably after $m_c \sim 50$ [g], owing to the increased pressure drop Δp , continuing using the filter without replacement costs more than replacing it (see the corresponding arrow on Figure 2.10).
- After ~ 70 [g], the curve is flat: the filter is not replaced and as the filtration efficiency drops significantly with mass intake, the mass increases very slowly and so does the pressure drop.

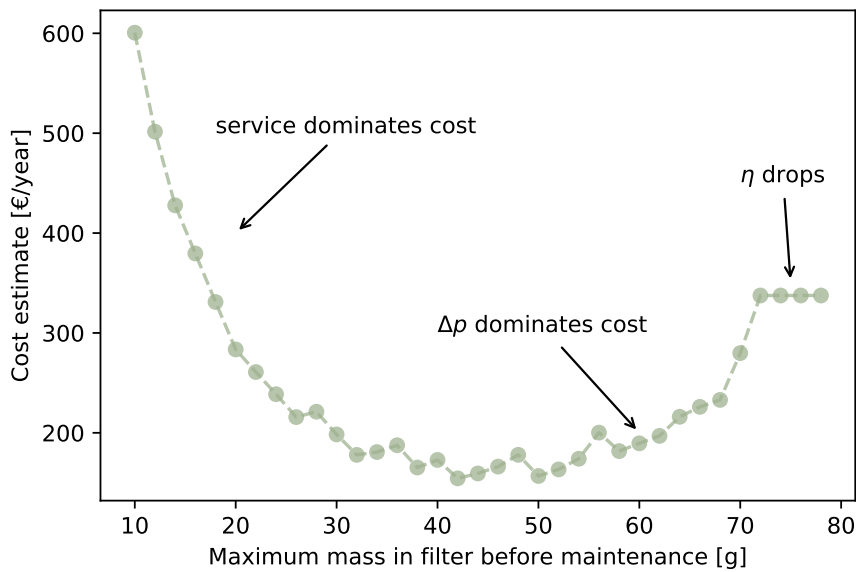


Figure 2.10 • Maintenance and operation costs estimate depending on the accumulated mass in filter before maintenance.

Not only the cost but also the air quality is affected by the filter maintenance frequency. On Figure 2.11 one can observe the average indoor air quality for $m_c = 20$ [g] and $m_c = 70$ [g]: the indoor air quality is affected by a low servicing rate (twice a year versus a dozen times) and exhibits concentrations that are about threefold higher than for $m_c = 20$ [g].

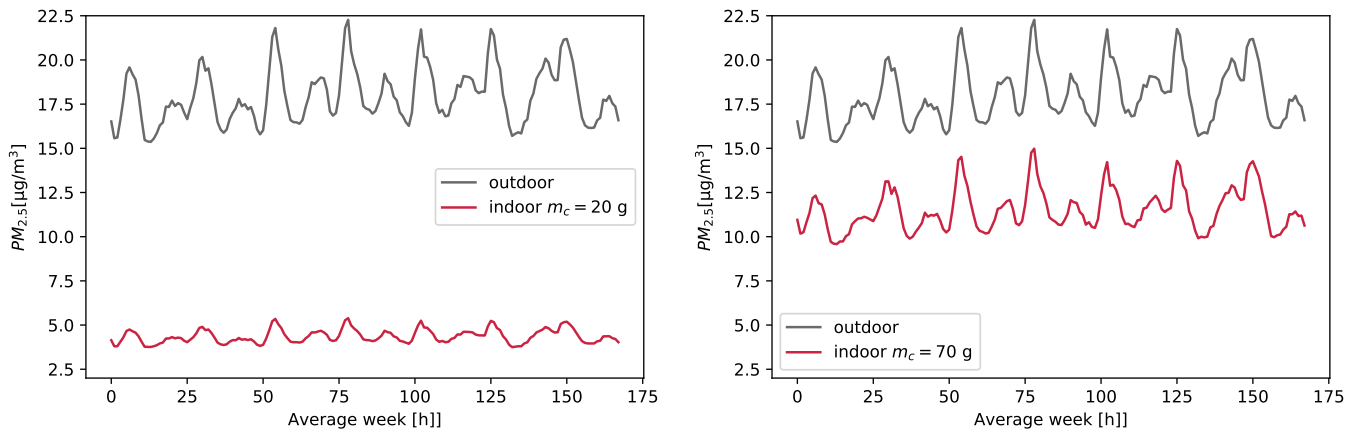


Figure 2.11 • Comparison of the $PM_{2.5}$ weekly average concentration for two critical masses in the filter $m_c = 20$ [g] (left) and $m_c = 70$ [g] (right).

The objectives we are trying to reach are actually contradictory: the better the air quality, the higher the costs. Let us introduce the notion of *Pareto front*: when comparing solutions having two contradictory objectives, the Pareto front is the ensemble of solutions for which it is not possible to improve one objective without decreasing the performance of the other (see Figure 2.12, left).

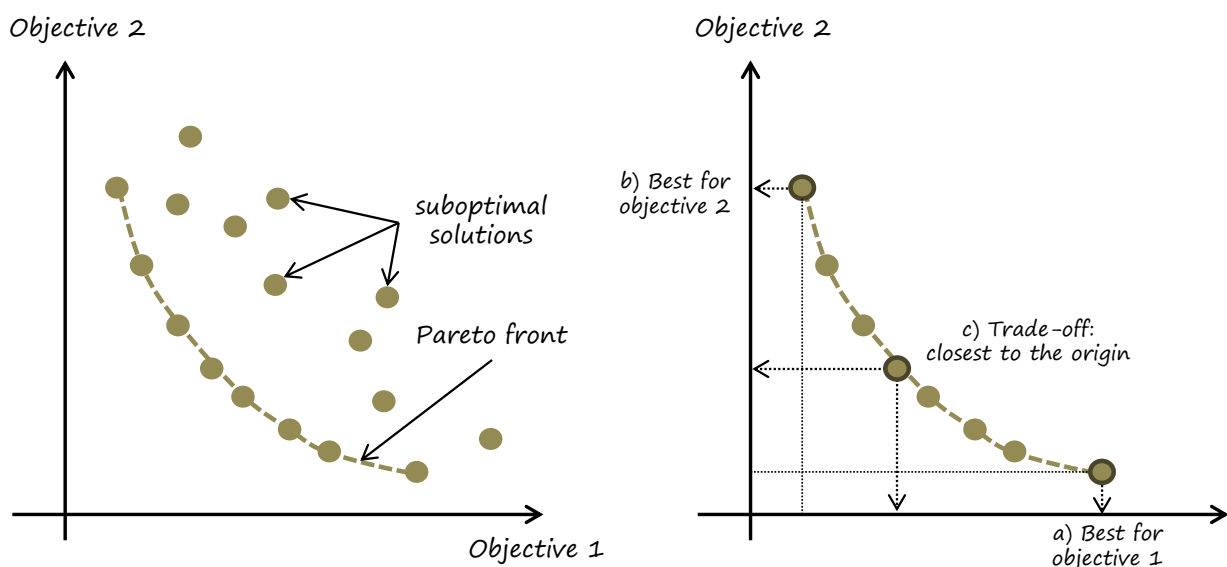


Figure 2.12 • Illustration of the Pareto front for the comparison of solutions with contradictory objective functions (left) and possible choices amongst the Pareto optima (right).

In short: the Pareto front is the ensemble of optimal solutions with regard to both objectives and all other solutions can be discarded. Now, how can we pick one out of this short list? Three rational options are possible*, also illustrated on Figure 2.12 (right) above:

- Choose the best solution for objective 1.
- Choose the best solution for objective 2.
- Choose the "average" solution, defined as the closest to the origin.

*Often the most affordable solution, or the one that technically exists is chosen, regardless of all these computational efforts... Bother!

Let us determine the Pareto front for critical masses ranging between $10 \leq m_c \leq 80$ [g], opposing cost and mean yearly air quality: Figure 2.13 shows the results obtained. One can observe a satisfactory minimum cost of ~ 150 [€/year] yielding 6 [$\mu\text{g}/\text{m}^3$] average concentration in $\text{PM}_{2.5}$. Note that the “flat” part of the curve of Figure 2.10 is present under the form of superimposed points at the higher end of the abscissa.

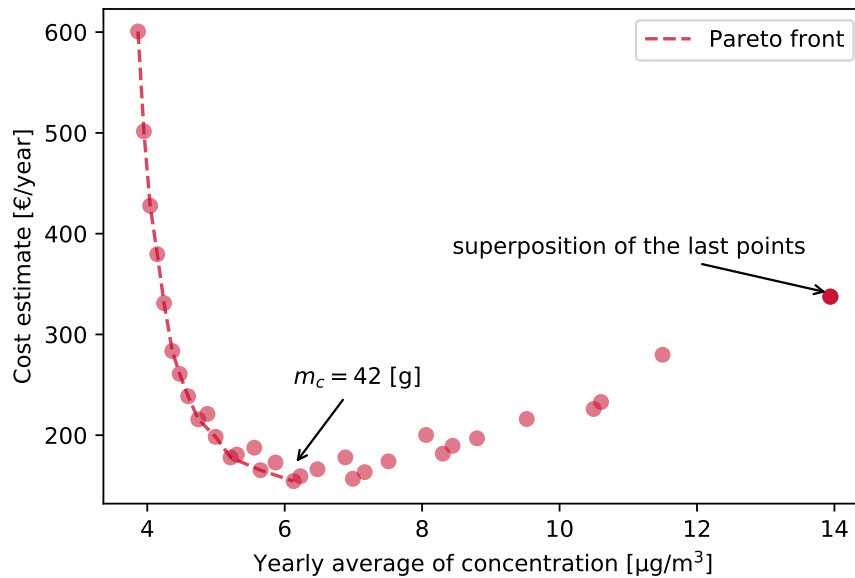


Figure 2.13 • Cost estimate *versus* average indoor air quality: example of Pareto front.

Two additional results put to light by the Pareto front are that the cost cannot decrease below ~ 150 [€/year] for $m_c = 42$ [g] and that the air quality ranges between $4 \sim 6$ [$\mu\text{g}/\text{m}^3$]. As the mean air quality does not significantly vary and is way below the yearly recommended average of 25 [$\mu\text{g}/\text{m}^3$], the choice of solution in the present application could be driven by the cost objective.

A web-based application with sliders for the parameters of the problem can be found there: [Filter clogging over time \(notebook\)](#).

The python code and $\text{PM}_{2.5}$ data for this application are provided on Github, feel free to make your own tests: [Filter clogging over time](#).

Continue exploring...

Question 4

In real life, the maintenance signals of filters usually rely on pressure drop sensors. Change the algorithm in order to replace m_c by a critical pressure drop Δp_c and plot the equivalent of Figure 2.10. As the shape of the slopes of accumulated mass and pressure drop are different, the results may differ.

Question 5

Vary the unitary cost for filter servicing and observe the outcome on the results presented Figure 2.10.

Question 6

Process the PM_{2.5} data set to increase/decrease the pollution level (or alternately use the data of your local air pollution monitoring agency): are the optimal results the same as in the present study?

Question 7

Add a fan curve to ameliorate the model and take into account the impact of pressure drop increase on power consumption and flowrate. What is the difference with the previous model? You may want to add an indoor pollution source term to Equation (2.7) in order to observe the reduction of air supply on the indoor air quality.

3 PID Controllers in HVAC

Numerous applications in HVAC make use of controllers in order to maintain quantities at a set value (temperature, concentration, flow rate amongst others). *Proportional-Integral-Derivative* controllers ("PID") are widespread in the built environment but may have also become the acronym of a nightmare for many students. We propose here to take a closer look at control theory in two steps: first understanding the PID operation with a simple physical problem (water draining from a tank), second a more complex example, namely controlling a room temperature with a three-way-valve.

3.1 Mathematical model of a PID controller

Control in general is based on maintaining a chosen set value (*e.g.* a temperature, a flowrate, a water level) with the help of an actuator (a motor, pump or valve).

Let us define the error $e(t)$ between the measured output and its set value, for example considering the temperature T and its set value T_0 :

$$e(t) = T(t) - T_0 \quad (2.15)$$

Three different actions may be used in order to reach the desired value of the measured quantity, generally combined:

- **Proportional** action: it represents *the present* value of the error and the correction action is proportional to the difference between actual and set values. Noticeably, if the error is nil, the correction will be zero as well, meaning the set value is never reached. Mathematically, it is proportional to coefficient K_p :

$$\text{proportional correction} \sim K_p \times e(t) \quad (2.16)$$

- **Integral** action: it represents *the past* values of the error and is practically the algebraic sum of the past errors. It allows for the exact correction of the error and is tuned by the integration time T_i :

$$\text{integral correction} \sim \frac{1}{T_i} \int_0^t e(t) dt \quad (2.17)$$

- **Derivative** action: it is *the future* value of the error. The correction action represents a prediction of the value of error at the next time step using the error slope de/dt and the derivation time T_d :

$$\text{derivative correction} \sim T_d \times \frac{de(t)}{dt} \quad (2.18)$$

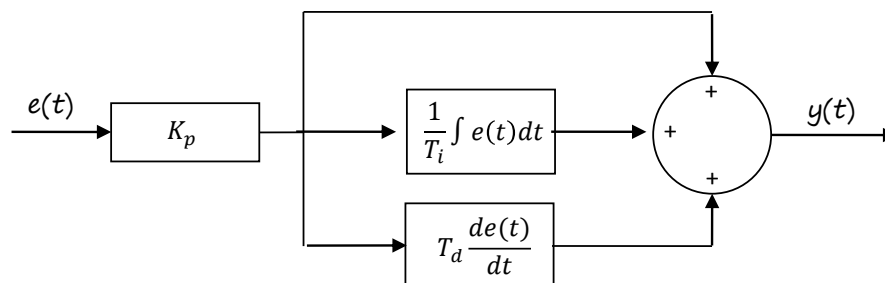


Figure 2.14 • Example of architecture of a PID controller.

These actions are often combined as "PI" or "PID", though more exotic versions may exist in the combinations or the form used. A common way of symbolising such PID controllers is depicted on Figure 2.14, which translates into the equation below:

$$y(t) = K_p \times \left(e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \times \frac{de}{dt} \right) \quad (2.19)$$

The following sections will provide a deeper understanding of PID controls through two physical problems.

3.2 Application - Gravity drainage of a tank

In order to reduce the overall complexity of the problem, let us start with a self draining tank, as presented on Figure 2.15, emptying through an open outlet at the bottom with the flow rate Q . The aim is to maintain the set level H_{set} by controlling the percentage of opening of a throttling valve*. The error is defined as $e(t) = H_{set} - h(t)$. Note that the outlet flow rate Q depends on the water height h .

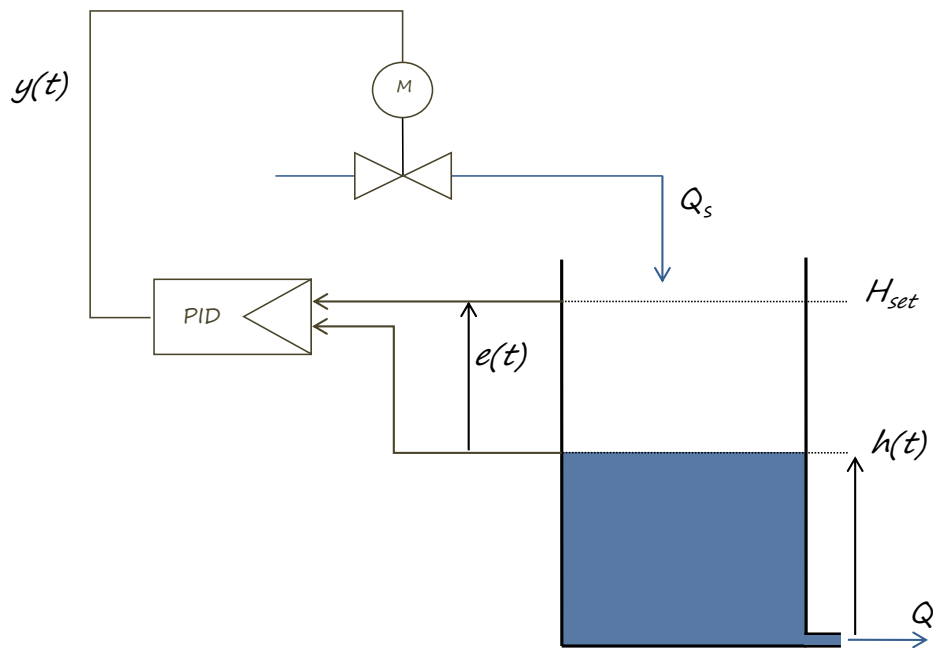


Figure 2.15 • Illustration of the tank with gravity drainage.

Let us establish the mathematical model of the problem, Q_s and Q being respectively the supply and outlet water flow. The variation of volume dV over a small time increment dt is:

$$dV = (Q_s - Q)dt \quad (2.20)$$

Bernoulli's theorem provides us the outlet flow rate Q at height $h = 0$ depending on the water height h and the outlet section s :

$$Q = s\sqrt{2gh} \quad (2.21)$$

As $dV = Sdh$, S being the water surface in the tank, we can rewrite Equation (2.20) using (2.21):

$$Sdh = (Q_s - s\sqrt{2gh})dt \quad (2.22)$$

The differential equation to be solved is then the water height h such that:

*Throttling is pure energy loss: it would be more common nowadays to use speed variation, however this example allows us to introduce the characteristic curves of valves.

$$\frac{dh}{dt} = \frac{1}{S}(Q_s - s\sqrt{2gh}) \quad (2.23)$$

Equation (2.23) has an analytical solution involving Lambert's W function*, however we will solve it numerically. Following expression is the translation of Equation (2.23) into an Euler explicit scheme:

$$h^+ = h + \frac{\Delta t}{S}(Q_s - s\sqrt{2gh}) \quad (2.24)$$

We will now introduce another element: the valve's *characteristic curve*. It represents the flow rate going through the valve depending on the control signal $y(t)$ and adds non-linearity to the problem. Three types of characteristic equations are provided below:

$$Q_s = Q_{\max} \times y \text{ for linear valves} \quad (2.25)$$

$$Q_s = Q_{\max} \times y^2 \text{ for quadratic valves} \quad (2.26)$$

$$Q_s = Q_{\max} \times e^{3.5 \times (y-1)} \text{ for equal percentage valves} \quad (2.27)$$

The corresponding flow curves are plotted on Figure 2.16. Note the non-negligible leakage flow rate for the equal percentage type (labelled *equal_pct*) when the valve is closed (valve position=0):

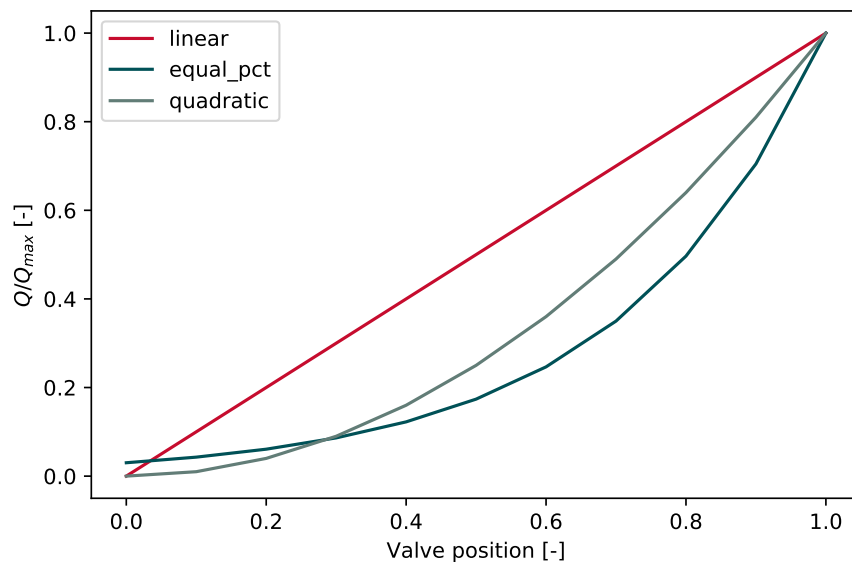


Figure 2.16 • Characteristic flow curves for three types of valves.

Having defined all the constitutive elements of the problem, a slightly simplified numerical implementation, is as follows:

```
# Ss/Sr = ratio of tank/outlet surfaces
B = Ss / Sr * np.sqrt(2 * 9.81)
sum_error=0 # initialise integral term
while t <= sim_time:
    # define A for readability
    A = Qsupply / Sr
    # Euler explicit for h
    h = dt * (A - B * np.sqrt(h)) + h
    # integral action
    sum_error = dt / Tn * (H_set - h) + sum_error
```

*Have you ever heard about this function before? Me neither.

```

# derivative action using current and previous error
d_error_dt = ((H_set - h) - delta_previous)/dt
# computation of the valve position with PID
valve_position = Kp * ((H_set - h) + sum_error + Td * d_error_dt)
# compute the flow rate depending on the valve_position
Qsupply=fc_valve_curve(valve_type, valve_position, Qmax)
# update the error for the next "D" action calculation
delta_previous = H_set - h

```

The code above may produce negative values of the valve position. In order to avoid inconsistent behaviour or negative flow rates, a condition is added in the actual code (see the online version).

A simple PI control using following parameters was set up:

- the maximum supply flow rate is 10 [L/s],
- the initial water height is $h = H_{\text{set}} = 1$ [m],
- the valve used is linear and initially closed ($y = 0$),
- the proportional band is $PB = 0.5$ [m], i.e. $K_p = 1/0.5 = 2$ [m⁻¹],
- the integration constant is set to $T_i = 30$ [s],
- the derivative action is set to zero with $T_d = 0$ [s].

Figure 2.17 shows the results obtained. On the left, the water height in the tank is plotted: after an overshoot in the vicinity of ~ 150 [s] followed by a few oscillations, it stabilises at the set value of 1 [m]. On the right, the contribution of the "P" and "I" actions are shown, as well as the valve position. In the present case study, the "P" action is the major contributor to control during the first 100 [s], after what the "I" action takes over and stabilises the valve's position, from ~ 300 [s].

Figure 2.17 • Animation – Water height over time for the tank draining exercise with a PI controller (left) and contribution of each action over time (right).

As you are about to try the model, two common sense observations about the PID parameter values may be of some interest:

- The proportional band is the quantity required for the controller to react with 100% of its capacity: in the case of the tank, if the proportional band is 0.5 [m], the valve will be fully open only if the error

reaches 0.5 [m], which may be a bit too much. On the other hand, if we set it to 0.01 [m], the controller will for sure overreact and exhibit an On-Off behaviour.

- The integral action allows to reach exactly the set point. However, the "inertia" of the integral time T_i may lead to large instabilities and overshoot, that add up to other actions ("P" or "D"). Take it easy with low values of T_i .

Note – A particular care must be given to the choice of the time step: indeed, such models add up numerical stability and choice of the sampling rate, that is, at what time interval we are going to check what the difference is with the set value. Large time steps may lead to increased (numerical or "physical") instabilities: for instance, letting the tank drain for a long period of time (*i.e.* Δt is important) before letting the system react may provoke important "P", "I" and "D" actions, followed by overshoot and/or oscillations.

Moreover, you will not be able to perform a stringent " Δt test" proposed in the check-list to verify your model (see Chapter 1 – Section 4.2): the controller actions modify the behaviour of the system depending on the measured error, and the latter varies with different time steps.

You now know how the model works. Before moving on to the next application that combines more physical phenomena leading to non-linear interactions, you may want to play with the tank model in order to get a grasp of the influence of each of the PID parameters applied to a rather linear system: [PID model of the tank drain](#).

You can also access directly the corresponding Jupyter Notebook with a web browser: [PID model of the tank drain \(notebook\)](#).

Continue exploring...

Question 8

Use Ziegler and Nichols' empirical method to tune the PID for the drainage tank:

https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method

Spoiler: the principle of this method is to set the "I" and "D" actions to zero and, starting from a broad proportional band, diminish its value until you find value of K_p^o provoking self-oscillations with a period T_o . The values of T_i, T_d are then calculated from T_o and K_p^o is a multiple of K_p^o .

Question 9

Change the valve's characteristic curve or the maximum flow rate: do you observe the same behaviour?

Question 10

Build your own Crank-Nicolson's formulation of the model to leverage the stability behaviour of this scheme.

3.3 Application - Three-way-valve controller for space heating

A first idea to control the power P delivered in a room would be to use the flow rate, as the heat delivery equation writes $P = \dot{m}C_p\Delta T$. Actuating on the flow in this case actually leads the valve to undergo frequent oscillations as the power dependence to flow rate is asymptotic. Although it would be convenient, thermal power emission does not respond to linear command laws.

Using an appropriate combination of radiator heat transfer coefficient, valve characteristic curve and valve pressure drop, it is possible to create a linear relation between valve position and emitted power. In this section, we will introduce the non-linearity in the physical phenomenon described above, going succinctly through:

- Basic heat exchanger theory to qualify the heat transfer from the radiator to the ambience.
- Valve curves, this time considering the K_v value, a proxy for hydraulic conductance at a given flow rate.
- Valve pressure drop compared to the system pressure drop, also called authority a .

An illustration of the problem considered is given on Figure 2.18, where a room with envelope losses US (the product of wall conductance U and surface S in $[W/K]$) contains a heater delivering the power P . The flow rate \dot{m} in the heater is constant but its inlet temperature is controlled with a motorised three-way valve, whose control signal depends on a PID controller, measuring the difference between the set value T_{set} and actual value T_a .

For the sake of the example, a constant temperature at the inlet of the three-way valve is assumed (in real life it would vary with the outdoor temperature and the room's heater would be equipped with a thermostatic head).

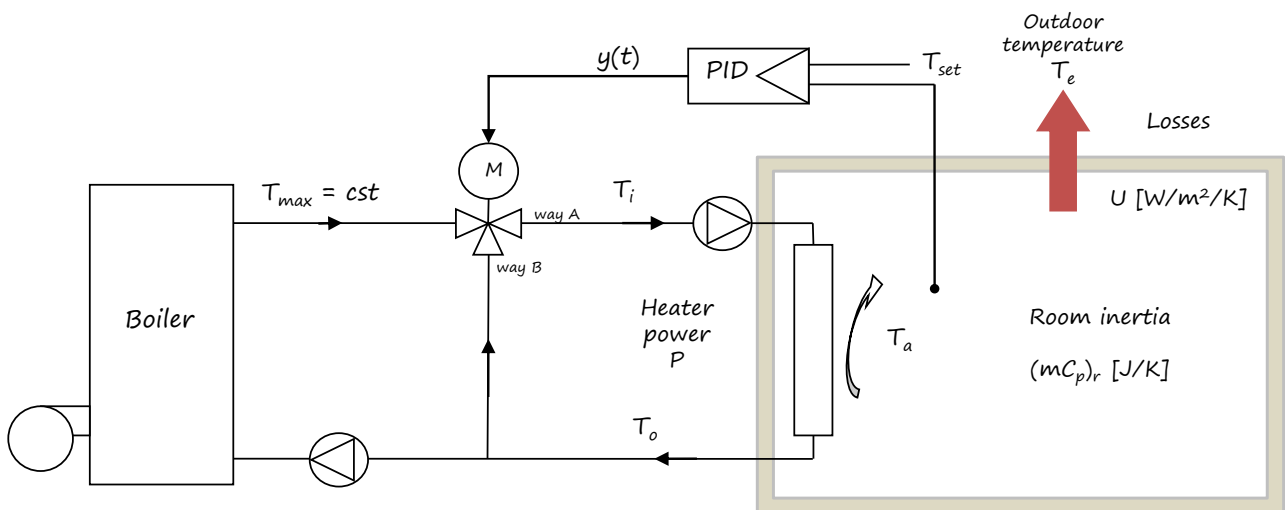


Figure 2.18 • Schematic of the heating control problem.

The following hypotheses are made in order to reduce the complexity of the problem:

- The room has a bulk inertia $(mC_p)_r$ $[J/K]$.
- The heater has a negligible thermal mass.
- The walls of the envelope have a negligible thermal mass (it could be a lightweight construction).
- Infiltrations, air change heat losses and thermal bridges are neglected.

The hypotheses about thermal mass may seem significant but actually they just delay the temperature rise and enables us to reduce the number of equations. Noticeably, the step response of a 2nd order system, alike the room with wall and heater thermal capacities, is very similar to the one of a 1st order one with delay.

Mathematical modelling

A simple way to control temperature in HVAC is to use mixing valves. In this section, we will start with the theory in order to be able to calculate the mixing temperature.

Let us first calculate the flow rate going through the valve depending on the valve position. A commonly used quantity is K_v : it is usually defined as the flow rate going through a valve for 1 [bar] pressure difference across inlet and outlet such that $Q_v = K_v \sqrt{\Delta p}$. It is homogeneous however to $[\text{m}^3 \cdot \text{s}^{-1} \cdot \text{bar}^{-0.5}]$, a proxy of hydraulic conductance (see section 1.2 of the Appendix for more details about the K_v).

The evolution of K_v in the valve is defined as a function of y , the percentage of valve opening and K_{vs} , its value when the valve is fully open:

$$\frac{K_v}{K_{vs}} = f(y) \quad (2.28)$$

where the function f is one of the equations given above, *i.e.* *quadratic, linear, or equal percentage*.

A valve's authority is defined as the ratio of the valve pressure drop and the total pressure drop of the circuit it is controlling, such that:

$$a = \Delta p_v / (\Delta p_v + \Delta p_r) \quad (2.29)$$

where Δp_v and Δp_r are respectively the valve pressure drop and the controlled element's pressure drop (the heater in our case).

It is possible to demonstrate that the flow rate depends on the valve's authority a and the characteristic curve of the valve K_v (see Section 1.3 in the Appendix). We express it below as the ratio between the maximum flow rate Q_{vs} and the actual flow rate Q_v :

$$\frac{Q_v}{Q_{vs}} = \frac{1}{\sqrt{a \left(\frac{K_{vs}}{K_v} \right)^2 + 1 - a}} \quad (2.30)$$

The temperature at the outlet of the valve is then a simple weighted average of the boiler outlet temperature and heater return temperature, neglecting the temperature-dependent variations of density for water:

$$T_i = Q_v T_{\max} + (1 - Q_v) T_o \quad (2.31)$$

The temperature drop in a heat exchanger is generally not linear but follows an exponential trend. The mean logarithmic temperature difference at the surface of the radiator is defined by:

$$\Delta T_{LM} = \frac{\Delta T_A - \Delta T_B}{\ln \frac{\Delta T_A}{\Delta T_B}} \quad (2.32)$$

where ΔT_A and ΔT_B are respectively the temperature differences between the inlet of the heater T_i and the ambient air T_a , and the outlet of the heater T_o and the ambient air.

$$\Delta T_A = T_i - T_a \quad (2.33)$$

$$\Delta T_B = T_o - T_a \quad (2.34)$$

For domestic heaters, the power released by a heater is defined by an empirical law, derived from the heat exchanger theory:

$$P = K (\Delta T_{LM})^n \quad (2.35)$$

with $n \sim 1.3$ for radiators and $n \sim 1$ for heated floors and K the heat transfer coefficient in $[\text{W}/\text{K}]$.

In order to obtain the fluid temperature T_o at the outlet of the heater we need to solve for the following equation stemming from (2.35):

$$P = \dot{m}C_p(T_i - T_o) = K \left(\frac{T_i - T_o}{\ln \left(\frac{T_i - T_a}{T_o - T_a} \right)} \right)^n \quad (2.36)$$

a reminder of the temperature notations can be found on Figure 2.18.

Let us suppose that the air in the room is fully mixed. With the assumptions made above, the variation of the room temperature is then a function of the power P delivered by the heater and the losses of the room envelope:

$$(mC_p)_r \frac{dT}{dt} = P - US(T_a - T_e) \quad (2.37)$$

Discretising Equation (2.37) provides the temperature in the room T_a^+ at the following time step:

$$T_a^+ = T_a + \frac{\Delta t}{(mC_p)_r} \left(K(\Delta T_{LM})^n - US(T_a - T_e) \right) \quad (2.38)$$

Note – In the present example, the determination of the "cookbook" stability condition from Equation (2.38) is difficult, as T_a cannot be factored out of the ΔT_{LM} :

$$0 \leq T_a - T_a \frac{US\Delta t}{(mC_p)_r} - \frac{K\Delta t}{(mC_p)_r} \left(\frac{T_i - T_o}{\ln \left(\frac{T_i - T_a}{T_o - T_a} \right)} \right)^n \quad (2.39)$$

In this case, it is preferable to either use a Crank-Nicolson scheme or carry on with Euler's one, at the cost of a conservative *-i.e.* small- Δt .

Case study

The case parameters used in the sequel are the following:

- The room losses are $US = 50$ [W/K], for instance 50 m² surface and 1 [W/m²/K] conductance.
- The thermal mass is taken as $(mC_p)_r \sim 63$ [kJ/K], which is the equivalent of about 62 [m³] of air. This value is low but allows us to observe the evolution on a rather short simulated period.
- The nominal power of the radiator is $P_{\max} = 3000$ [W].
- The water flow rate is computed from the power with 20 [K] temperature drop in the radiator:
 $\dot{m} = P_{\max}/(4182 \times 20) \sim 129$ [kg/h].
- The K coefficient is computed from the nominal power with a reference ΔT_{LM} of 50 [K]:
 $K = P_{\max}/\Delta T_{LM}^{1.3} = 3000/50^{1.3} \sim 19$ [W/K].
- The proportional band is set to 5 [K].
- The integration time is set to 3000 [s].
- The derivation time is set to 5 [s].

The results are given on Figure 2.19. On the left, the temperatures are plotted: heater inlet and outlet, set value and room actual temperature over time. The set value is reached after ~ 3000 [s] and the heater inlet and outlet temperature stabilise.

On the right, the share of the "P", "I" and "D" control actions is shown. One can observe that the "P" action is first at work, progressively replaced by the "I" action. The "D" share accounts for a significant fraction of the total in the first 500 [s], after what the variation of the error is low and it has a negligible influence.

Figure 2.20 shows the evolution of the power delivered by the heater and the room losses during the first 2600 [s]. One can observe that the power delivered by the heater almost equates the room's losses after this time span.

Figure 2.19 • Animation – Temperature within the room with a PI controller (left) and contribution of each action over time (right).

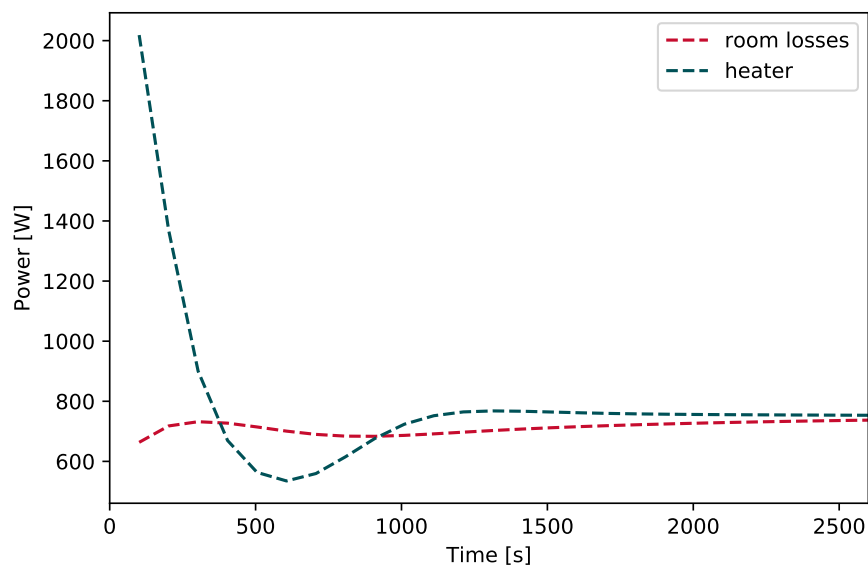


Figure 2.20 • Heat delivered by the heater versus losses.

The code for this application is available, implemented with a Crank-Nicolson scheme. Give it a try: [PID model of the heated room](#).

Note – Doing tests on the code you may experience "math error" if you play a bit too hard with derivation times and/or small proportional band values as oscillations may try to feed the logarithm function with negative values.

Continue exploring...

- Question 11** Tune the controller after Ziegler and Nichols method (*cf. supra* for the instructions).
- Question 12** Change the valve's authority, the characteristic curve or the maximum heating power: what happens?
- Question 13** Make a simulation over a longer period including a time-dependent outdoor temperature (use for instance a `sin` function or actual meteorological data).

4 Geothermal heat pump

The code of this section owes much to Pr. Desodt of ENS Paris-Saclay, may he be thanked here for his benevolence.

Heat pumps are used for room heating and/or cooling using vapour compression cycles. Such systems are similar to domestic refrigerators, extracting heat at one end of the cycle and releasing it to the ambience at the other end. The reverse principle is used in the case of domestic heating: energy is transferred to the heating system, whereas the cooling effect has to be evacuated somewhere. It is released to the atmosphere when the evaporator is an air unit, or to the ground for geothermal heat pumps.

Geothermal heat pumps may be constructed in different ways, using open or closed-loop systems and vertically or horizontally distributed probes. We will consider here that heat is exchanged in a closed loop water circuit running from the evaporator to vertical probes in the soil, where ground water is supposed to flow. A schematic example is drawn on Figure 2.21, with a unique vertical probe for the sake of clarity. On this Figure, the room heat demand, heat taken from the ground and ground water velocity are also represented.

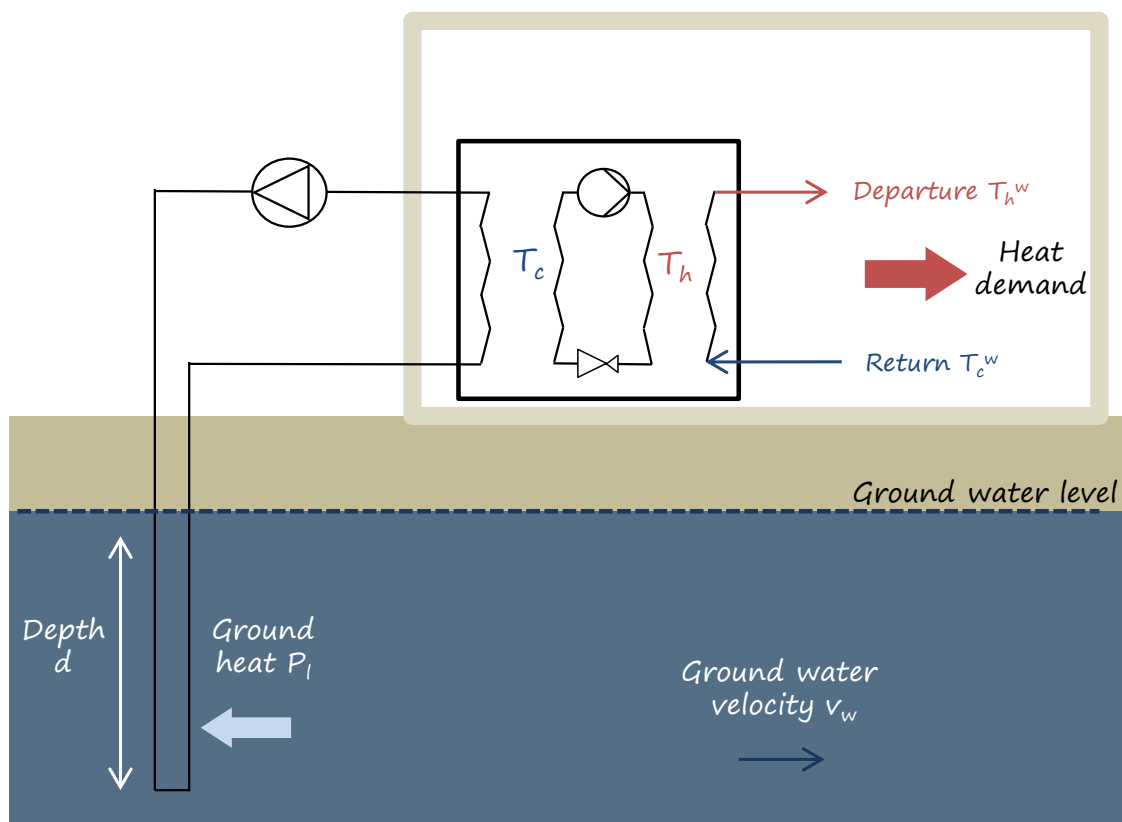


Figure 2.21 • Operating diagram of a domestic geothermal heat pump with one underground pipe.

The aim in this section is to evaluate the performance of such a heat pump. As the efficiency of such systems depends on the temperatures of the thermodynamic cycle, it implies simulating both the heating demand and the ground temperature, *id est* the hot and cold sources.

4.1 Mathematical model of ground heat pump

In this section, we present the simple model retained for the heat pump system and the determination of temperatures in the cycle.

Coefficient of performance of a heat pump

The first principle of thermodynamics, also called *conservation principle* indicates that the energy is conserved in the cycle (suppose we neglect losses). We obtain the following relationship between the energy Q_h on the hot side at the condenser, Q_c on the cold side at the evaporator and the electrical energy used for compression Q_{e-} :

$$Q_h = Q_c + Q_{e-} \text{ [J/kg]} \quad (2.40)$$

Noticeably, this relationship can also be observed graphically on Figure 2.22 representing Mollier's chart (see the equation below the x-axis).

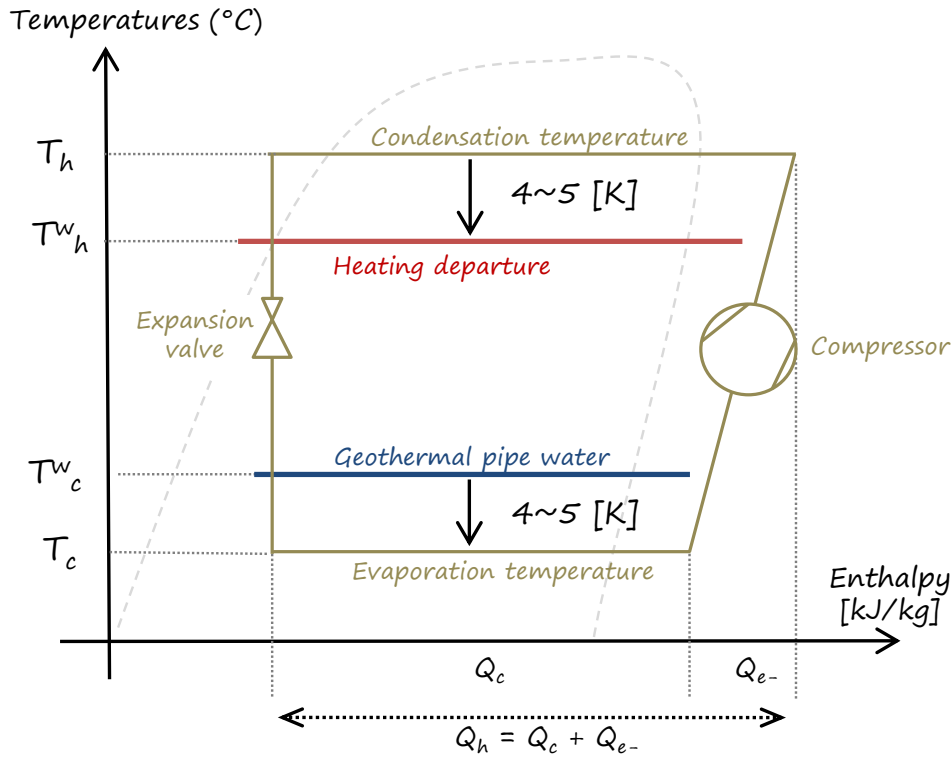


Figure 2.22 • Illustration of temperatures and heat exchange around the thermodynamic cycle.

Multiplying Equation (2.40) by the mass flow rate of refrigerant \dot{m} [kg/s] in the heat pump, one obtains:

$$\dot{m}Q_h = \dot{m}Q_c + \dot{m}Q_{e-} \quad (2.41)$$

$$P_h = P_c + P_{e-} \quad (2.42)$$

The coefficient of performance (COP) is the ratio of the thermal power P_h used for heating to the electrical power put in the compressor P_{e-} (respectively the enthalpies Q_h, Q_{e-} in [J/kg]). It writes as follows in heating mode:

$$\text{COP}_h = \frac{P_h}{P_{e-}} = \frac{Q_h}{Q_{e-}} \quad (2.43)$$

Note – Dividing Equation (2.41) by the compressor electrical power P_{e-} yields:

$$\text{COP}_h = \text{COP}_c + 1 \quad (2.44)$$

COP_c being the COP in cooling mode, also called *Energy Efficiency Ratio* (EER). The latter is thus systematically smaller by one unit than the heating mode COP_h .

Above the freezing temperature at the evaporator^{*}, an approximative performance of vapour compression cycles may be expressed as a percentage of Carnot's cycle theoretical efficiency:

$$\text{COP}_h \sim \eta_C \frac{T_h}{T_h - T_c} \quad (2.45)$$

where T_c, T_h are respectively the cold and hot temperature of the thermodynamic cycle and η_C is the efficiency of the cycle compared to Carnot's theoretical efficiency, usually around $\eta_C \sim 35\%$ for commercial applications. The challenge in the current section is to calculate T_c and T_h in order to obtain an estimate of the COP.

As defined in Equation (2.45), the higher the temperature difference between the hot and cold side, the lower the coefficient of performance, and *vice versa*. Interestingly, heat pumps exhibit the best performance when the temperature difference is low, that is when only a little heating or cooling is required.

In order to determine the cycle temperatures T_c, T_h , we need to consider heat exchange at the condenser and evaporator. Practically, the heat exchanger pinching may reasonably be assumed as $3 \sim 5$ [K]. With this assumption the actual cycle temperatures are respectively higher and lower than the usage temperature:

$$T_h = T_h^w + 5 \quad (2.46)$$

$$T_c = T_c^w - 5 \quad (2.47)$$

A graphical illustration of the position of temperatures T_h, T_c in the thermodynamic cycle and of Equations (2.46) and (2.47) is proposed on Figure 2.22, superimposed on Mollier's chart.

As seen in Section 3, water-based heating systems are driven by temperature levels. The heat release depends on the logarithmic mean temperature difference between the radiator surface and the ambient air (as a reminder, take a look at Equation (2.35)). Provided with Equation (2.45) that heat pumps are efficient with low temperature differences, the heating departure temperature is to be the lowest as possible: heat pumps are particularly adapted to floor heating or radiators with large surfaces, enabling the use of low water temperatures.

For the simplicity of the model, suppose that the heating power demand only depends on the outdoor temperature. The chosen water logic is classical: when the outdoor temperature is the lowest, the departure temperature is at its maximum (see Figure 2.23) and when the outdoor temperature rises, the departure temperature goes down.

The departure temperature of the heating system T_h^w is a function of the outdoor temperature, as presented on Figure 2.23. The outdoor dry bulb temperature can then be used as a proxy for the return temperature, such that:

$$T_h^w = aT_{\text{out}} + b \quad (2.48)$$

where $a = -1.05$ [-] and $b = 33.7$ [°C] for the case study.

The heating power demand is supposed to be proportional to the difference between indoor and outdoor air temperatures, taking into account conduction losses US and air change $\rho_a Q_v C_{pa}$, providing the bulk equation:

$$P_h = (US + \rho_a Q_v C_{pa})(T_{\text{in}} - T_{\text{out}}) \quad (2.49)$$

The maximal power P_{max} of the heat pump is defined as the heat demand at the outdoor sizing temperature, e.g. $T_{\text{out}} = -6$ [°C]. Depending on the outdoor air temperature, the fraction of the maximum heating power can be determined by:

$$\frac{P_h}{P_{\text{max}}} = \frac{T_{\text{in}} - T_{\text{out}}}{\Delta T_{\text{max}}} \quad (2.50)$$

where $T_{\text{in}} = 19$ [°C] is the constant indoor temperature and ΔT_{max} is the difference of temperatures used for the heating system sizing, i.e. $19 - (-6) = 25$ [K], where -6 [°C] is the minimum temperature of the chosen location.

^{*}Evaporator defrosting (electrically or by cycle inversion) requires more complex models than the simple one described here.

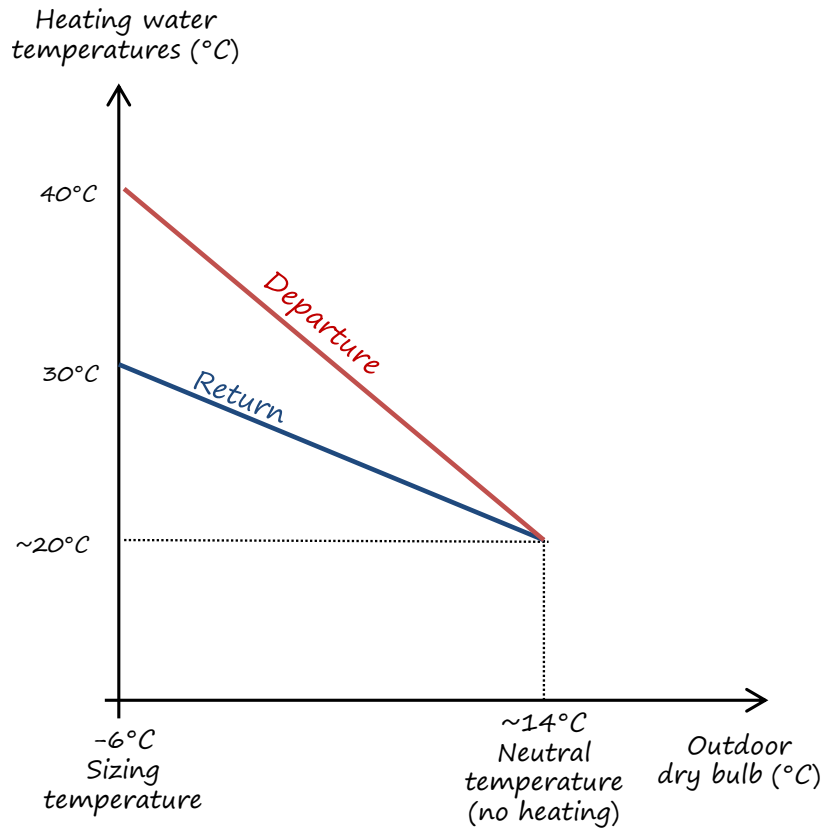


Figure 2.23 • Water temperatures of the heating system depending on the outdoor temperature.

The heating power required is a fraction of P_{\max} and varies linearly as per:

$$P_h = P_{\max} \frac{T_{\text{in}} - T_{\text{out}}}{\Delta T_{\max}} \text{ [W]} \quad (2.51)$$

Using Equations (2.43) and (2.44), the heat flux P_c to be taken from the ground at the evaporator is derived from the following three successive equations:

$$\text{COP}_c = \text{COP}_h - 1 \quad (2.52)$$

$$P_{e-} = \frac{P_h}{\text{COP}_h} \quad (2.53)$$

$$P_c = \text{COP}_c P_{e-} \quad (2.54)$$

The power extracted per unit length of pipe in the ground is then obtained by allocating P_c in each of the underground pipes, such that:

$$P_l = \frac{P_c}{d \times n_{\text{pipes}}} \text{ [W/m]} \quad (2.55)$$

where d is the immersed depth of the pipes and n_{pipes} the number of pipes.

Using the value of P_l obtained in a 2D model of the ground, the temperature on the cold side T_c^w can be determined as the average return temperature of the geothermal pipes. The calculation is detailed in Section 4.2 of this chapter.

In real life, the geothermal probe looks much alike the drawing presented on Figure 2.24 where the probe containing two U-shaped pipes is filled with bentonite* (a type of clay enhancing conduction heat transfer). However, as it would be convenient for our model to remain two-dimensional, we want to avoid a detailed modeling of:

*at least in the French context, in order to comply with DTU 65.16

- the temperature evolution of the heat carrier in each pipe with depth,
- the water-to-wall convective resistance of the liquid in the U-pipes,
- what happens in the cross-section of bentonite with four different heat sources,
- the probe wall resistance.

Using P_l as a simple source term on the central point of the cell containing the pipe hence appears as a very practical alternative (what is more, we do not exactly know what are the real thermophysical properties of ground around the probes, hence we may probably afford an extra approximation).

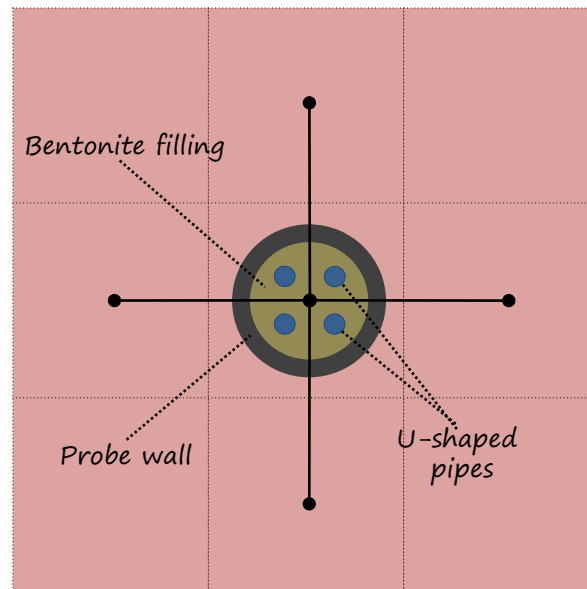


Figure 2.24 • Scheme of an actual pipe within the 2D mesh.

Summary of the modelling assumptions

For the sake of simplicity, the following assumptions are made on the geothermal side:

- The ground water flow is parallel to the vertical axis of the grid.
- Heat transfer is bi-dimensional: vertical heat transfer is neglected, which allows for a two-dimensional simulation*.
- Heat is transferred through the geothermal pipes using a heat flux term P_l [W/m].

Regarding the building heating system, the assumptions are listed below:

- Thermal mass of the building is neglected.
- The transients of the heat pump are not considered.
- Heating demand only depends on the outdoor temperature (no solar or internal gains) and no heating schedule is considered.
- The departure temperature is reached instantly.
- The heat exchanger pinching is constant at 5 [K] and imposes the cycle temperatures to the heat pump.
- The efficiency η_C compared to Carnot's cycle is constant.

In the next section, the governing equations and their numerical implementations are presented.

*This very handy assumption is of course wrong.

4.2 Numerical model in 2D

The equation of heat transfer in the ground resembles much mere conduction, as presented in Section 2.2. In addition to conduction, the advection term $v_w \rho_w C_{pw} \frac{\partial T}{\partial x}$ and the heat source/sink P_l appear in the balance:

$$\rho C_p \frac{dT}{dt} = v_w \rho_w C_{pw} \frac{\partial T}{\partial x} + \lambda \frac{\partial^2 T}{\partial x^2} + \lambda \frac{\partial^2 T}{\partial y^2} + P_l \quad (2.56)$$

where P_l [W/m] is the heat transferred from ground water to the pipe and v_w [m/s] is the underground water velocity, usually in the order of magnitude of ~ 1 [m/day].

Numerical formulation

Let us suppose that the ground water flows along the \vec{y} axis at velocity v_w . The thickness of the discretised layer is $L = 1$ [m]. Using a centered discretisation scheme, the advection term in Equation (2.56) writes numerically as:

$$\rho_w Q_w C_{pw} (T_{i+1,j} + T_{i-1,j} - 2T_{i,j}) \quad (2.57)$$

$$\rho_w v_w \Delta x L C_{pw} (T_{i+1,j} + T_{i-1,j} - 2T_{i,j}) \text{ [W]} \quad (2.58)$$

Establishing the heat balance at the probe node, Equation (2.56) hence translates to:

$$\begin{aligned} \rho C_p \Delta x^2 L \frac{T_{i,j}^+ - T_{i,j}}{\Delta t} = & v_w \rho_w C_{pw} \Delta x L \frac{T_{i,j-1} + T_{i,j+1} - 2T_{i,j}}{\Delta x} \\ & + \lambda \Delta x L \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x} \\ & + \lambda \Delta x L \frac{T_{i,j+1} + T_{i,j-1} - 2T_{i,j}}{\Delta x} + P_l L \end{aligned} \quad (2.59)$$

Simplifying the equation (2.59) yields:

$$\begin{aligned} T_{i,j}^+ - T_{i,j} = & \frac{v_w \Delta t}{\Delta x} (T_{i,j-1} - T_{i,j+1} - 2T_{i,j}) \\ & + \frac{\lambda \Delta t}{\rho C_p \Delta x^2} ((T_{i+1,j} + T_{i-1,j} - 2T_{i,j}) + (T_{i,j+1} + T_{i,j-1} - 2T_{i,j})) \\ & + \frac{P_l \Delta t}{\rho C_p \Delta x^2} \end{aligned} \quad (2.60)$$

Let us introduce Courant's adimensional number $C_o = \frac{v_w}{\Delta x / \Delta t}$, *id est* the ratio of the fluid velocity to the velocity $\Delta x / \Delta t$ of a particle that would pass from a cell to the adjacent one, distant of Δx , during time Δt . Practically, this is the maximum velocity that can be "captured" by the time and space discretisation used, otherwise fluid particles travel more than a cell at each time step and cause numerical diffusion.

For clarity, let also $r = \frac{\rho_w C_{pw}}{\rho C_p}$ be the water to ground ratio of heat storage per Kelvin. Using the non-dimensional numbers and factorising the temperatures, we can now rewrite Equation (2.60) as:

$$\begin{aligned} T_{i,j}^+ = & T_{i,j} (1 - 2rC_o - 4F_o) \rightarrow \text{balance at the node} \\ & + (T_{i-1,j} + T_{i+1,j})(rC_o + F_o) \rightarrow \text{advection+conduction along } \vec{y} \\ & + (T_{i,j+1} + T_{i,j-1})F_o \rightarrow \text{conduction along } \vec{y} \\ & + \frac{P_l \Delta t}{\rho C_p \Delta x^2} \rightarrow \text{source term} \end{aligned} \quad (2.61)$$

Note – The stability condition for an explicit scheme in this case is $1 - 2rC_o - 4F_o > 0$. Compared to mere conduction, the presence of underground water flow hence lowers the stability limit for a given space discretisation. This translates to a time step Δt smaller than:

$$\Delta t \leq \frac{1}{\frac{2rv_w}{\Delta x^2} + \frac{4\alpha}{\Delta x^2}} \quad (2.62)$$

Additionally, the centered differentiation advection-diffusion scheme is stable for $F_o > \frac{C_o^2}{2}$ [10], which translates to a relationship between time step, diffusivity and ground water velocity:

$$\frac{2\alpha\Delta t}{\Delta x^2} \geq \frac{v_w^2\Delta t^2}{\Delta x^2} \quad (2.63)$$

$$\Delta t \leq \frac{2\alpha}{v_w^2} \quad (2.64)$$

The boundary conditions used for the simulation are presented on Figure 2.25: the inlet temperature is imposed whereas the outlet is a "zero-gradient" condition. The sides of the domain are considered as adiabatic (do not mind the low number of cells).

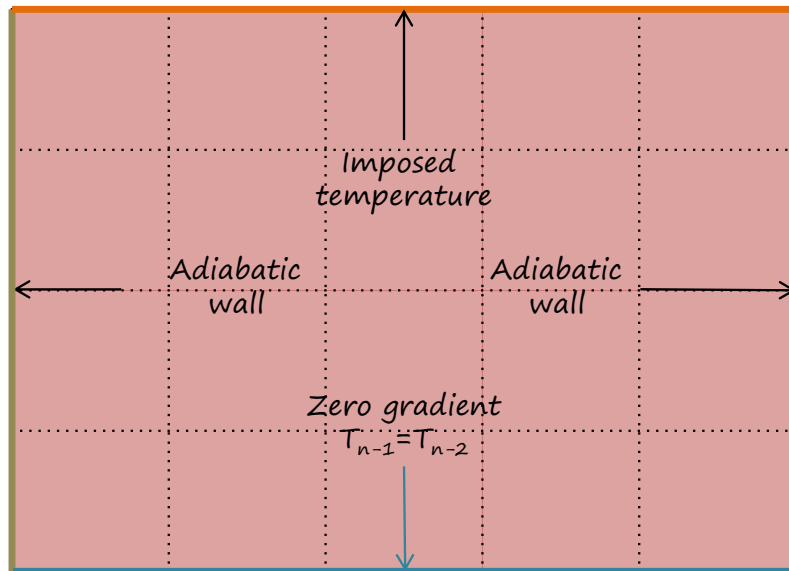


Figure 2.25 • Boundary conditions of the 2D simulation.

Similarly to the applications seen before, the numerical implementation of the 2D Euler scheme is detailed in the code excerpt that follows:

```
# beginning of the time loop
while t < sim_time:

    # compute the heating departure temperature as per the water logic
    T_hot=-1.05*Tout[ii]+33.7
    # heating power required
    P_h = (19-Tout[ii])/dTmax*Pmax
    # compute the heating mode COP
    COP_h = eta_carnot*(T_hot+273.15+5) / (T_hot+5-T_avg-5)
    # compute the cooling mode COP
```

```

COP_c = COP_h - 1
# compute the electrical power
Pcomp = P_h/COP_h
# compute the cooling power in the ground
P_c = Pcomp*COP_c
# update the heat pump power (in W/m)
P_l=-P_c/n_pipes/depth
# get the numerical source term
source_HP=P_l*dt/(rho*Cp*dx**2)

# apply the in/out boundary conditions
for j in range(0,m):

    # inlet upper boundary
    T[0,j]=T_inlet
    T_temp[0,j]=T[0,j]

    # outlet lower boundary
    T[n-1,j]=T[n-2,j]
    T_temp[n-1,j]=T[n-1,j]

# apply the adiabatic boundary conditions
for i in range(1,n-1):

    # j=0 left boundary
    T[i,0]= T[i,0]*(1-r*Co-3*Fo)\
        + Fo * (T[i+1,0] + T[i-1,0]) \
        + Fo * (T[i,1])\
        + r*Co*(T[i-1,0])
    T_temp[i,0]=T[i,0]

    # j=m-1 right boundary
    T[i,m-1]= T[i,m-1]*(1-r*Co-3*Fo)\
        + Fo * (T[i-1,m-1] + T[i+1,m-1] ) \
        + Fo * (T[i,m-2]) \
        + r*Co*(T[i-1,m-2])
    T_temp[i,m-1]=T[i,m-1]

# compute the inside of the domain
for i in range(1,n-1):
    for j in range(1,m-1):

        # if the cell contains a pipe, add the source term
        if [i,j] in ij_pipes:
            source=source_HP
        else:
            source=0
        # generic node temperature equation
        T_temp[i,j]=T[i,j]*(1-r*Co-4*Fo)\
            + Fo * (T[i+1,j]+T[i-1,j]) \
            + Fo * (T[i,j+1]+T[i,j-1]) \
            + r*Co*(T[i-1,j]) \
            + source

```

4.3 Application - Geothermal heat pump

Using the methodology detailed in the previous sections, the calculation of the pipe temperatures and the seasonal COP are presented in the next paragraphs.

The simulation set up is the following:

- The maximum heating power is $P_{\max} = 10$ [kW] for -6 [°C] outdoor temperature.
- Paris' outdoor air temperature from a meteorological file serves as input.
- The temperatures and COP are calculated after the model presented in Section 4.1 of this chapter.
- The saturated ground thermophysical properties are $\rho = 2150$ [kg/m³] and $C_p = 2000$ [J/kg/K].
- The geothermal probes reach the depth $d = 20$ [m].
- The domain is 40×40 [m²] wide, with 100 nodes in each direction.
- The stable time step being rather high, a maximum of $\Delta t = 3600$ [s] is set in order to reduce the numerical integration error.

The temporal evolution of the average pipe temperature and the power taken from the ground are plotted on Figure 2.26 and follow the same trend. The power extracted from the ground remains in the vicinity of ~ 50 [W/m], which is a common order of magnitude for vertical geothermal probes (~ 20 to 50 [W/m] is the acceptable range for P_l in the French building construction code, depending on ground conductivity and heat pump service time over the year).

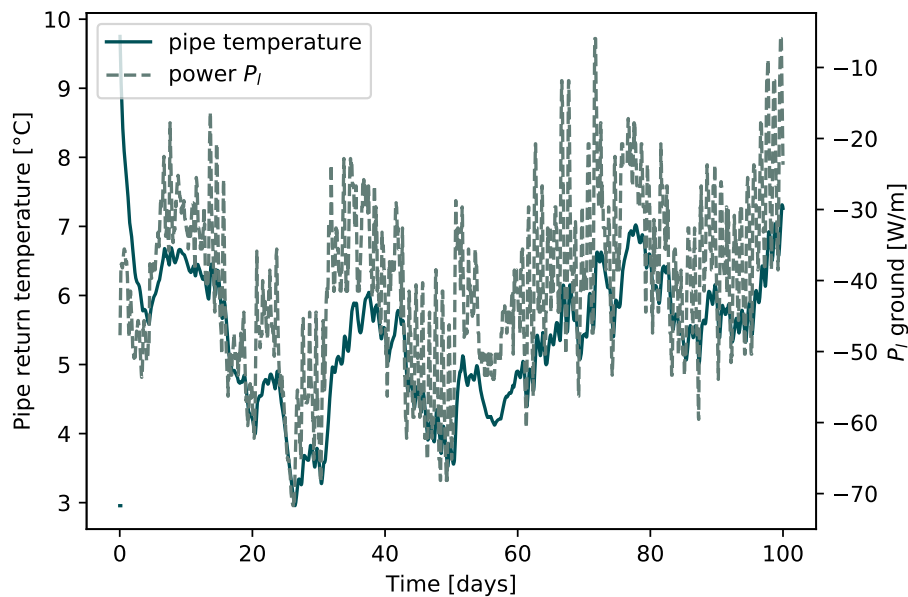


Figure 2.26 • Example of evolution of average pipe temperature and linear power P_l taken from the ground.

The ground temperature field can be observed on Figure 2.27 with a staggered configuration of ground pipes, from day 50 to day 75 of the simulation (or simply day 75 if your PDF reader does not support animations). The last row of pipes is in the “plume” of the upstream ones and hence cooler* than the other ground pipes.

Figure 2.27 • Temperature field in the ground for staggered probes.

Regular configurations of pipes may also be used, as presented on Figure 2.28, also showing days 50 to 75. One can observe qualitatively that the reached temperature is lower, as downstream pipes are immersed in the water cooled by upstream pipes.

Figure 2.28 • Temperature field in the ground for aligned probes.

*Please note that the colour scale is reversed: although it is counter-intuitive, it looks much nicer.

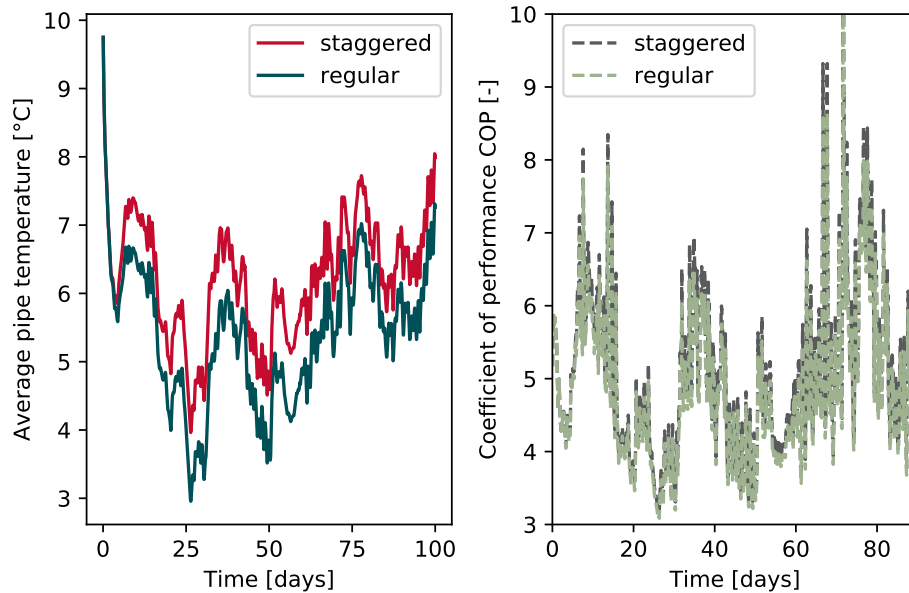


Figure 2.29 • Comparison of the staggered and aligned probes.

A comparison of staggered and regular configuration in terms of temperatures (left) and COP (right) is provided on Figure 2.29. Although the profiles exhibit the same trends, one can observe that the average pipe temperature of the staggered set up is slightly higher, as well as the COP. Indeed, the average coefficients of performance differ by a few percent over the year: the regular set up of pipes yields an average $COP_{re} = 4.9$, whereas the staggered configuration performs better with $COP_{st} = 5.2$.

The temperature of the different pipes in the ground is also different depending on the configuration: Figure 2.30 shows the staggered configuration (right) *versus* the regular one (left), the latter being more heterogeneous and with temperatures colder by a few degrees.

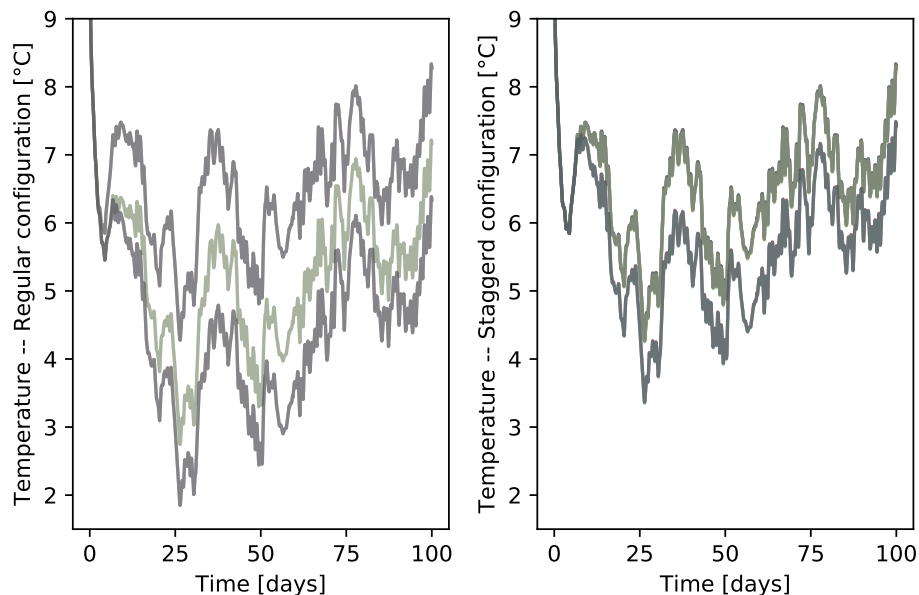


Figure 2.30 • Comparison of the probes temperatures for staggered and aligned pipes.

The code for this application can be downloaded from: [2D water heat pump model](#).

Continue exploring...

- Question 14** Set the water velocity v_w to zero and change the soil properties to unsaturated ρ and C_p . How does the coefficient of performance behave?
- Question 15** Perform a parametric study of the influence of water velocity v_w on the COP.
- Question 16** Change the formulation so that the velocity may be unaligned with the grid axis, *i. e.* $\vec{v} = \vec{v}_x + \vec{v}_y$. Numerically, it consists in splitting the underground velocity into terms $\frac{v_x \Delta t}{\Delta x} (T_{i+1,j} + T_{i-1,j} - 2T_{i,j}) + \frac{v_y \Delta t}{\Delta x} (T_{i,j+1} + T_{i,j-1} - 2T_{i,j})$.
- Question 17** Imagine the radiators are sized for a 70-50 [°C] departure/return system. Change the slope of the heating system as per Figure 2.18. What is the difference of COP compared to the floor heating temperature levels?
- Question 18** Change the maximum heating power P_{\max} [W] and evaluate the water temperature difference. Observe the duration of the soil temperature perturbation over a longer period of time.
- Question 19** Following a similar methodology with the outdoor temperature, add a cooling function for summer and observe the results over a complete year with/without summer cooling.
- Question 20** Add a cylindrical heat resistance for the initially neglected probe wall, as sketched in Figure 2.24, and evaluate the influence of the hypothesis.

Chapter 3

Coupled problems & minimisation

The field of building physics proposes a number of coupled phenomena. In this chapter, we will examine the numerical methods that can be applied to solve systems of coupled partial differential equations with problems of gradually increasing complexity. Assumptions are made in order to slightly simplify the problem: the point here is not to discuss whether the governing equations are well chosen to represent the underlying physics, but to broach their solving.

1 Indoor Air Quality: *two-compartments* models

We have seen in Section 2 an example of air concentration modelling with a *single-compartment* approach, that is, considering only the *air compartment*: deposition onto surfaces and resuspension from them were both neglected. These two phenomena are coupled and form an interesting case study for this chapter. Much could be said about aerosols. Since this book aims at staying concise, we will only go through some of the details in the sections below. References will be provided for more precisions.

1.1 What is an aerosol?

Aerosols are clouds of particles in suspension in the air. Those particles have different sizes and shapes, without distinction of species. Particles may be formed of solid matter, gas or liquid droplets. The following paragraphs briefly introduce the quantities generally used when considering aerosols.

Size

Different mechanisms drive particles into the air such as erosion or combustion, leading to discrepancies in particle sizes. Imagine a herd with animals as big as whales, elephants, rabbits and ants: this is, proportions kept, particles of the orders of magnitude between $\sim 10 \text{ } [\mu\text{m}]$ and $\sim 0.001 \text{ } [\mu\text{m}]$, illustrated on Figure 3.1.

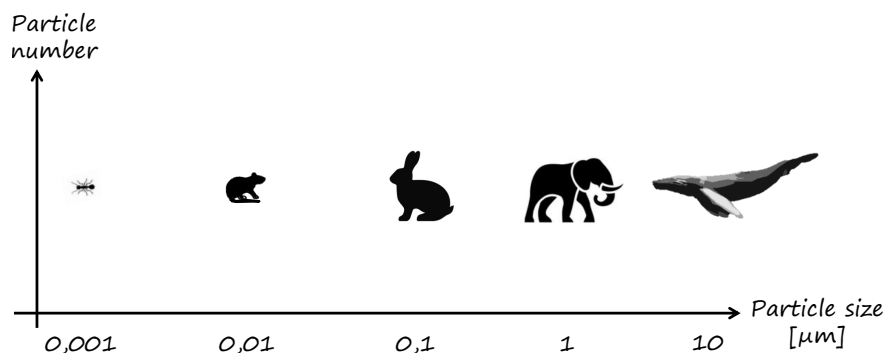


Figure 3.1 • Illustration of aerosol particles with animal size analogy (*the size of animal drawings is not proportional*).

As one can imagine, the underlying physics of deposition or resuspension for particles of the various size classes are governed by different mechanisms. Extending the analogy: elephants are less affected by wind velocity than ants*.

Shape / Diameter

As the shape of particles is most of the time unknown, an equivalent diameter is used for comparison: Stokes' diameter is the diameter of a particle having the same particle density and the same measured deposition velocity. A few other equivalences exist, such as the aerodynamic diameter using the same principle for a density of 1 [g/cm³], or the electrical mobility diameter – the report of the Swedish Transports Agency is very clear about those definitions, see [26].

Density

The density ρ of particles depends on their nature and varies with the context of generation of aerosols (erosion, combustion, industry, brake wear...). For the indoor and outdoor environments, the particle density is around 1 [g/cm³], whereas for specific environments it may be higher, such as in the underground where, owing to a higher iron content in the aerosols, the bulk density can reach up to 4 ~ 5 [g/cm³] [24].

Quantification

Two quantities are commonly used as a means of comparison for aerosols:

- The particle mass of the aerosol for a given range of diameters, *e.g.* the PM_{2.5} or PM₁₀, respectively the aerosols of particles until 2.5 [μm] and 10 [μm]. Noticeably, the mass is generally calculated using the assumption of spherical particles and is proportional to the power of three of the particles' diameter ($m_p \sim \rho_p \times d^3$).
- The number of particles per size-class.

Mass and number concentration can be used either for the whole aerosol or for each of the size classes, meaning respectively a one bulk value for all the size classes considered, one or per size class. The toxicological debate is still open depending on the correlated influence of size and species of particles, for instance in the underground context [13]. All in all, a simple rule applies: the less particles, the better, be it in mass or numbers.

1.2 Physical model

In most applications the bulk approach is used, considering the aerosol in its ensemble without distinction for each individual particle density or size. The simulated aerosol is supposed to be an equivalent of the real one, choosing adequate properties.

Let us consider a homogeneous aerosol and add two phenomena to the previous mass balance model for air quality exposed in Section 2:

- **Deposition** δ [s⁻¹] a constant depending on the particle size, deposition velocity and geometrical features of the enclosure, namely the ratio of wall, ceiling and floor (for an excellent description of the complex underlying physics, see [19]). In regular enclosures, δ is actually a weighted average of the deposition velocities upon vertical surfaces, horizontal surfaces and ceilings (respectively the subscripts v, h, c in following equation):

$$\delta = \frac{v_v S_v + v_h S_h + v_c S_c}{V} \quad (3.1)$$

*... Assuming they are exposed to the same wind velocity, which is unlikely to happen given the existence of a boundary layer near the ground, correct.

→ **Resuspension** ρ [s^{-1}] a term for the quantity of particles deposited on the surfaces of the enclosure (or *surface compartment*) that can be sent back to the air compartment. This phenomenon is less known and up to now no unified modelling approach has been proposed. However, one can note that the resuspension flux in [$\text{kg}/\text{m}^2/\text{s}$] is roughly proportional to the local air velocity v^α [16, 21, 25].

It is to be noted that other physical interactions do occur in aerosols, such as coagulation, that appears to be important for higher numbers of particles. We consider here they are negligible and do not integrate them in the model.

For a monodispersed aerosol, that is with one size class, writing down the "closure" equation of the particle concentration model with L [$\mu\text{g}/\text{m}^2$] the quantity of particles deposited on surfaces, we obtain following system of partial differential equations, also explained in [23]:

$$\left\{ \begin{array}{l} V \times \frac{\partial C}{\partial t} = Q_v(C_e - C) - \delta VC + \rho SL \quad [\mu\text{g}/\text{s}] \\ S \times \frac{dL}{dt} = \delta VC - \rho SL \quad [\mu\text{g}/\text{s}] \end{array} \right\} \quad (3.2)$$

where S is the surface available for deposition and resuspension, the same for both phenomena in the present case.

The governing phenomena of deposition and resuspension in an enclosure described by the set of Equations (3.2) are represented on Figure 3.2.

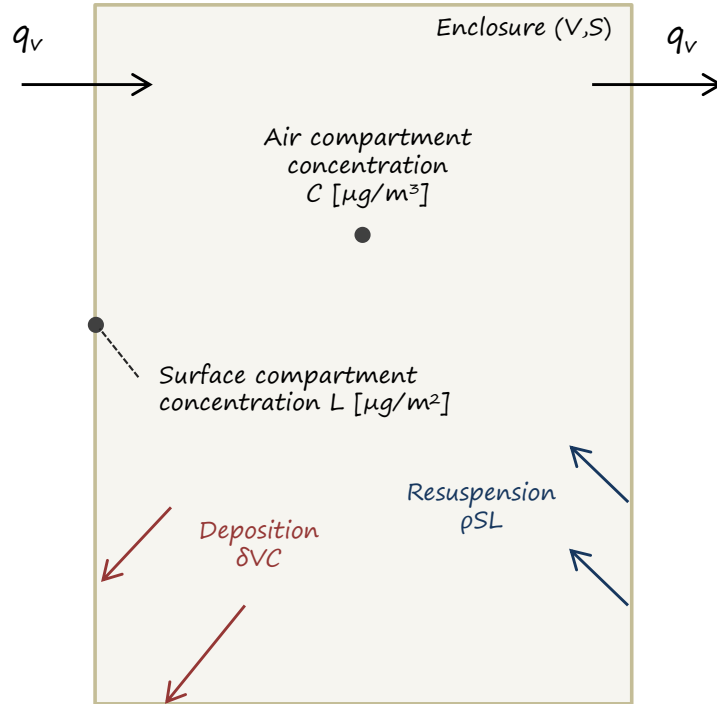


Figure 3.2 • Schematic of the deposition and resuspension phenomena in an enclosure of volume V and total surface S .

We can identify the mass of particles going from the air to the surface δVC and the mass of particles resuspended from the surfaces ρSL , forming a coupled system: C depends on L and *vice versa*.

Dividing the equations in (3.2) respectively by the air volume V and the total deposition surface S , one obtains:

$$\left\{ \begin{array}{l} \frac{\partial C}{\partial t} = \tau(C_e - C) - \delta C + \frac{\rho SL}{V} \quad [\mu\text{g}/\text{m}^3/\text{s}] \\ \frac{dL}{dt} = \frac{\delta VC}{S} - \frac{\rho SL}{S} \quad [\mu\text{g}/\text{m}^2/\text{s}] \end{array} \right\} \quad (3.3)$$

The distribution of aerosols is often partially known, however the deposition rate of particles varies by several orders of magnitude depending on their size [19], as shown on Figure 3.3. In simple terms, large particles tend to “fall” as gravity is the governing phenomenon (above 1 μm), whereas small particles (below 0.01 μm) diffuse towards surfaces by Brownian diffusion. In between, particles having a diameter in the vicinity of ~ 0.1 μm are too large to diffuse but too small to “fall”: they are maintained in the air by the slightest movements of natural convection.

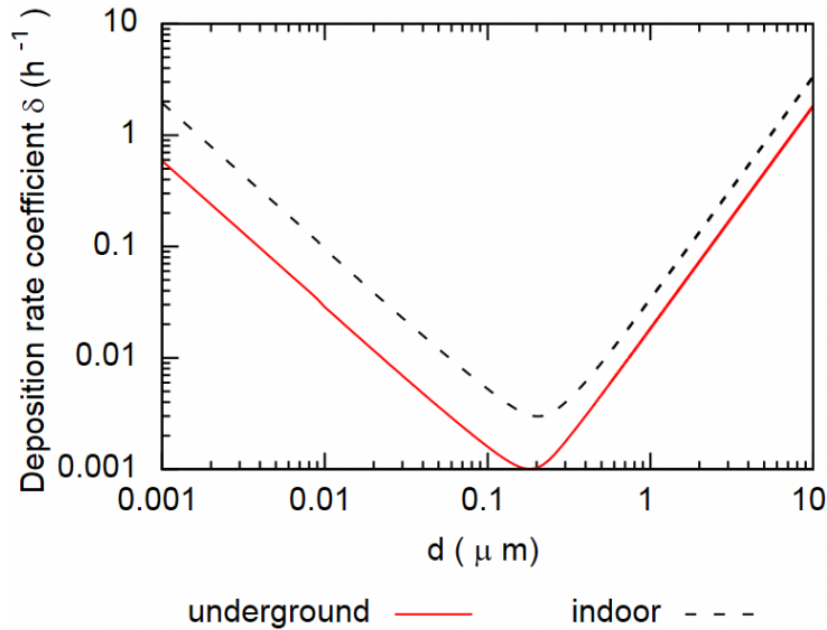


Figure 3.3 • Deposition rate δ as a function of particle size, shape of the enclosure and friction velocity (from [30]).

The “bulk” approach presented in the system of equations (3.2) however does not impede the separation of particles, grouping them into size classes, for a better evaluation of the behaviour of the aerosol (an example will be shown in the coming pages of Section 1.4). Considering there is no interaction between the size classes, *e. g.* neglecting coagulation that would transfer mass between size classes, the system can be solved independently from each particle class. A qualitative illustration of the effects of coagulation and deposition on a polydisperse aerosol is proposed on Figure 3.4.

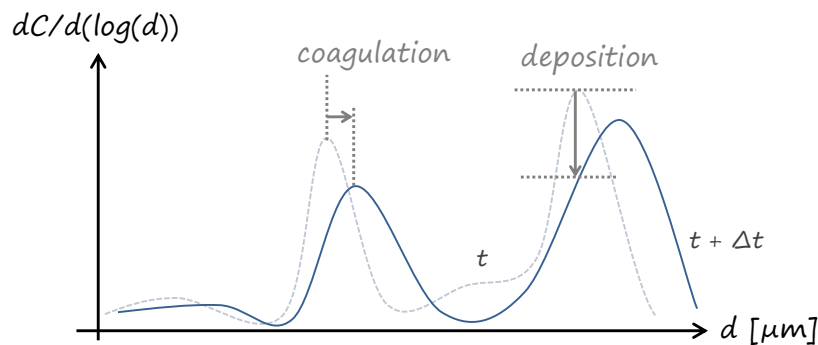


Figure 3.4 • Qualitative representation of the coagulation and deposition effects on a polydispersed aerosol.

In the two following sections, we will examine how to put this together numerically, first with a monodispersed aerosol and then with a polydisperse one.

1.3 Numerical implementation for one size-class

Similarly as introduced in Section 3.2 of Chapter 1, let us define a function to solve for with the very handy `fsolve` method. Interestingly, the latter can process either scalars (size one arrays), or arrays. The construction of the numerical equivalent to system (3.2) is described in the snippet below. For readability purposes, we split and reconstruct the vector $[C, L]$ as follows:

```
def fc_IAQ_coupled(vec_CLp, vec_CL, tau,delta,dt,rho,S,V,Ce):
    #let us split the input array in two
    C,L = vec_CL[0],vec_CL[1] # C [microg/m3], L [microg/m2]
    Cp,Lp = vec_CLp[0],vec_CLp[1]
    # crank-nicolson system of equations for coupled IAQ
    C_term = -Cp + C + 0.5*dt*( tau*(Ce-C) -delta*C+ rho*S*L/V)
                + 0.5*dt*( tau*(Ce-Cp) -delta*Cp+ rho*S*Lp/V)
    L_term = -Lp + L + 0.5*dt*( delta*V*C/S - rho*S*L/S )
                + 0.5*dt*( delta*V*Cp/S - rho*S*Lp/S )
    return [C_term,L_term] # return as an array for fsolve not to crash
```

The setup of the case study and the model hypotheses are listed below:

- The enclosure dimensions are $5 \times 5 \times 3$ [m³].
- C_e varies with time around 20 [µg/m³] such that $C_e + 5 \times \cos(2\pi t/24)$, as a simplistic mimic of outdoor air quality variations.
- The air change rate in the enclosure $\tau = 0.1$ [h⁻¹] is constant.
- The deposition rate $\delta = 0.15$ [h⁻¹] is constant.
- The resuspension rate ρ arbitrarily varies between 0 and 0.5 [h⁻¹] such that $\rho + |\sin(2\pi t/24)|$, so that we may observe something else than pure asymptotic behaviour.
- We consider the aerosol is represented by one equivalent particle size. Looking at the deposition rate curve in Figure 3.3, with the chosen value of δ it could be $d \sim 4$ [µm].

Computing the time evolution of the aerosol is then simply made with a loop as follows, solving for C^+ and L^+ with Crank-Nicolson's semi-implicit scheme defined in the previous code excerpt:

```
while t < sim_time:
    # add a time varying outdoor concentration
    Ce=Ce_base + 5*abs(np.cos(t*2*np.pi/period))
    # resuspension rate depends on time
    rho=rho_base*abs(np.sin(t*2*np.pi/period))
    # compute C+,L+ with Crank-Nicolson
    C_plus,L_plus=fsolve(fc_IAQ_coupled,[C,L],
                        args=([C,L],tau,delta,dt,rho,S,V,Ce))
    C,L=C_plus,L_plus
    t+=dt
```

The results obtained with this model are presented on Figure 3.5. As one can see, the initial mass deposited on surfaces L_0 is ~ 10 times superior to the steady state (right). Indeed, the initial conditions for C and L have an impact on each other*. Looking for initial conditions promptly leading to steady state may well take a few trials and errors. The air and surfaces behaviour in Figure 3.5 can be explained looking at Equations (3.2): owing to the high quantity of deposited matter, the transferred to air (L to C) is important

*This is the essence of coupled problems.

and the peak of $\rho LS/V$ in the first hours causes the increase in C , simultaneously with the drop of L . Noticeably, obtaining a correct idea of the quantity of particles deposited on surfaces is also complex in practice.

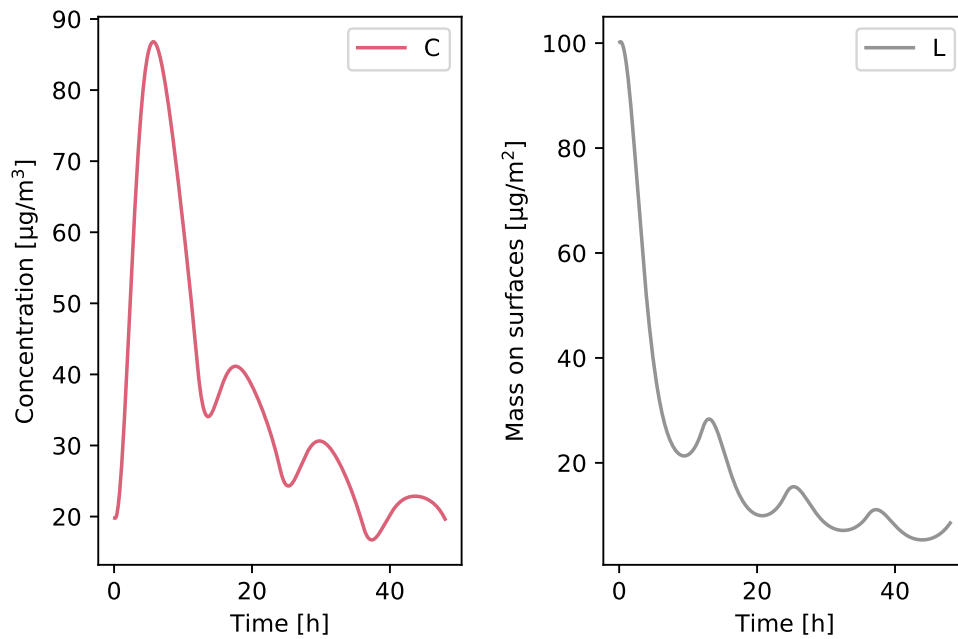


Figure 3.5 • Concentration evolution in the air compartment (left) and on the surfaces compartment (right) after 48 hours.

Figure 3.6 (left) is a plot of the value of the resuspension coefficient ρ over time, following a periodical sine function. The right-hand side of Figure 3.6 shows the mass transfer from surface to air and *vice versa*. The initial peak of matter transferred from surfaces to the air is related to the available quantity of deposited particles and proportional to $\sim \rho LS/V$.

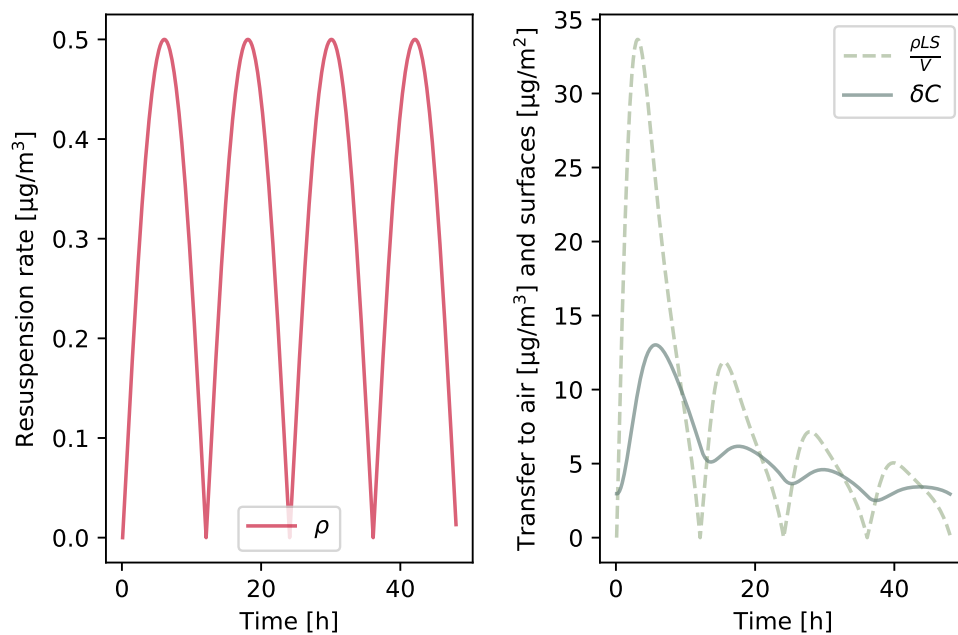


Figure 3.6 • Arbitrary periodic resuspension rate ρ (left) and mass transfer to/from the surface compartment (right).

The code using a Crank-Nicolson scheme is provided on Github: [Single-class air quality model](#).

1.4 Extension to an n size-class aerosol

As interactions between size classes are not taken into account, we may spare the writing of a matrix formulation of the problem and be content with arrays. Indeed, extending the method to n classes of particles is relatively easy once the implementation with one class is done: declaring two vectors $[C]$, $[L]$ each having the size of the number of size-classes is basically sufficient.

There might certainly be a more elegant or efficient way of realising the coupling*. However, following "array splitting" technique happens to be straightforward: the vector containing C and L is cut in two halves, which allows the conservation of most of the one-size-class formulation. With this technique, the air and surfaces concentrations for the size class i are called with the same index, $C[i]$, $L[i]$ as the positions in the split array remains the same. The python function for solving writes as:

```
def fc_IAQ_coupled_classes(vec_CLp, vec_CL, tau, delta, dt, rho, S, V, Ce):
    # number of size classes
    n=int(len(vec_CL)/2)
    C,L = vec_CL[0:n], vec_CL[n:] # split into C and L
    Cp,Lp = vec_CLp[0:n], vec_CLp[n:] # split into Cplus and Lplus
    # solve with delta being a vector this time
    C_term = -Cp + C + 0.5*dt*( tau*(Ce-C) -delta*C+ rho*S*L/V)
    + 0.5*dt*( tau*(Ce-Cp) -delta*Cp+ rho*S*Lp/V)
    L_term = -Lp + L + 0.5*dt*( delta*V*C/S - rho*S*L/S )
    + 0.5*dt*( delta*V*Cp/S - rho*S*Lp/S )
    # sending back as an array so that fsolve works
    # (the use of np.hstack 'flattens' in one single array)
    return np.hstack([C_term,L_term])
```

Application to mere deposition

In order to get a grasp of the effects of pure deposition on a polydisperse aerosol following case study is proposed: the enclosure is identical as in the previous paragraph, however with $\tau = 0$ and $\rho = 0$.

For this example, let us take a virtual, 6-classes, polydisperse aerosol with mass the distribution* shown on Figure 3.7 (left), each of the size-classes being centred around $d = [0.001, 0.01, 0.1, 1, 10]$ [μm], and use a deposition per size class close to the values described in [19], plotted on Figure 3.7 (right): interestingly they exhibit two orders of magnitude difference.

The evolution of the aerosol over one hour is simulated. One can observe on Figure 3.8 the concentration in the air (left) and deposited on surfaces (right). Without surprise, pure deposition leads to a decrease of concentration in the air compartment, mass being transferred to the surface compartment.

*Computational expense is not a challenge in this case.

*Note that mass being proportional to the cube of diameter $m \sim d^3$, this means the numbers of particles in the smaller size classes are very high.

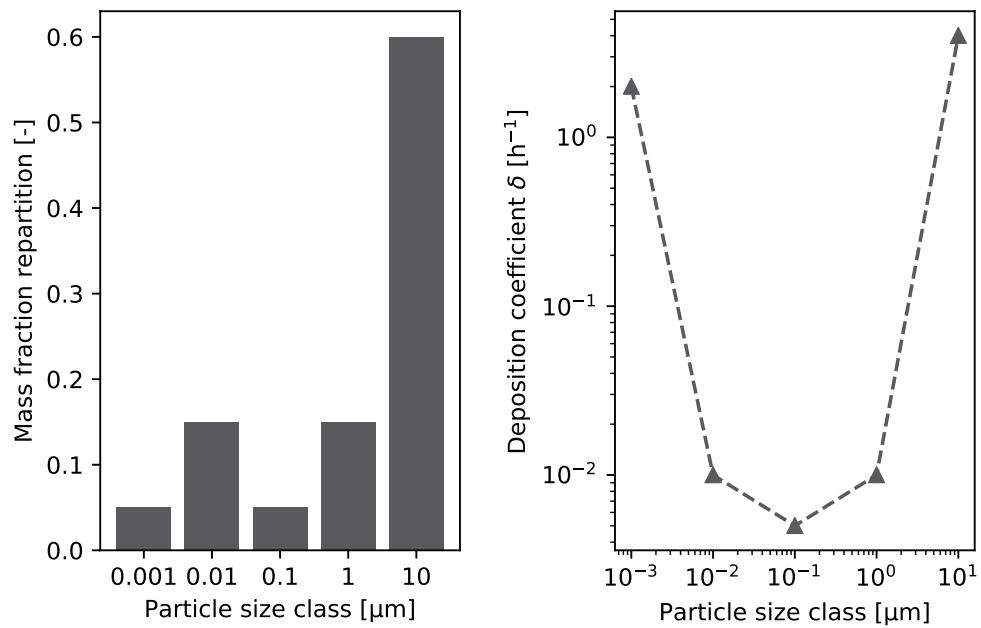


Figure 3.7 • Chosen mass repartition in the current example (left) and chosen deposition coefficients (read from Figure 3.3).

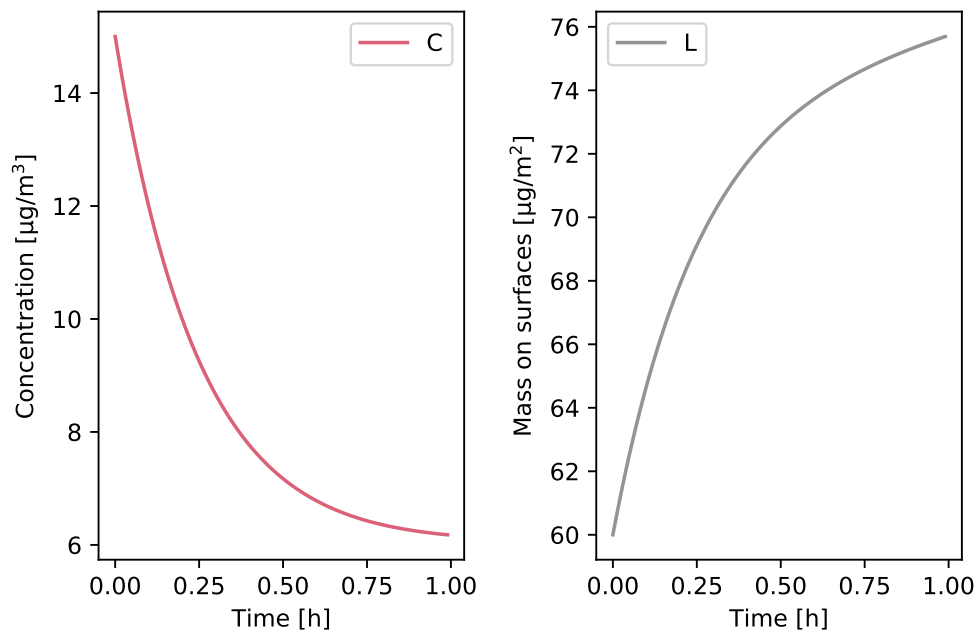


Figure 3.8 • Particle concentration in the air (left) and on surfaces (right) for the chosen polydisperse aerosol in the case of pure deposition.

Let us now examine the evolution of mass distribution presented on Figure 3.9 as fractions of the total mass: we can observe that particles in the size classes at both extremities of the diameter range (0.001 and 10 μm) tend to vanish over time. The 10 μm particle size-class is particularly affected in this example case. Interestingly, at the beginning of the simulation, the 10 μm size-class contains 60% of mass, whereas after an hour the 0.01 μm and 0.1 μm contain $\sim 50\%$ of the aerosol's mass, leading to a clear shift of mass distributions over time (see Figure 3.4 as a reminder of the qualitative effect of deposition).

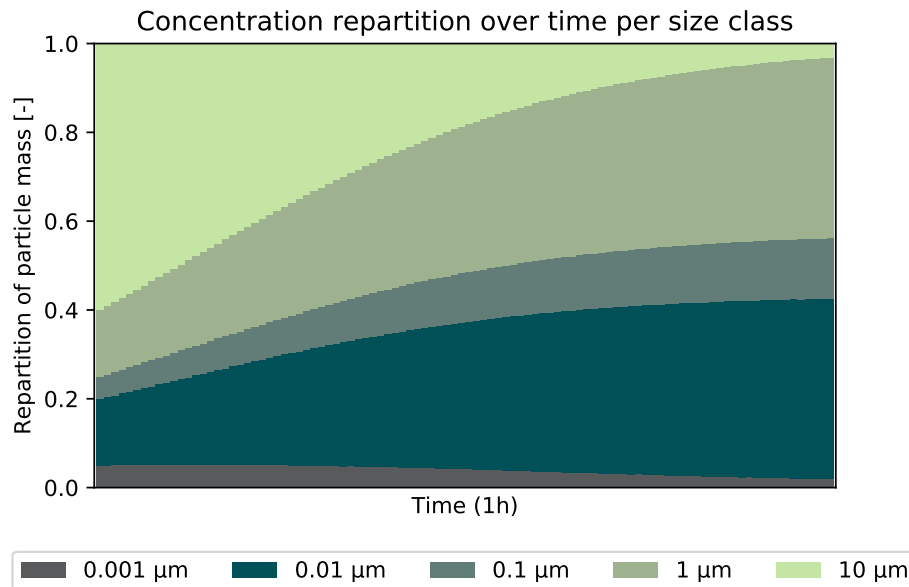


Figure 3.9 • Evolution of the fraction of each particle size class over time.

Application to an enclosure

Taking again the case study exposed in 1.3 and using the mass distribution chosen in the previous example with pure deposition, we calculate the evolution of the quantity of particles on surfaces and in the air. Figure 3.10 shows the results obtained, alike those obtained with one size-class: on the left the air concentration is plotted and on the right the surface compartment concentration is shown. Similarly, a marked increase of air concentration C is coupled to a decrease of L and the steady state is not reached after 48 hours.

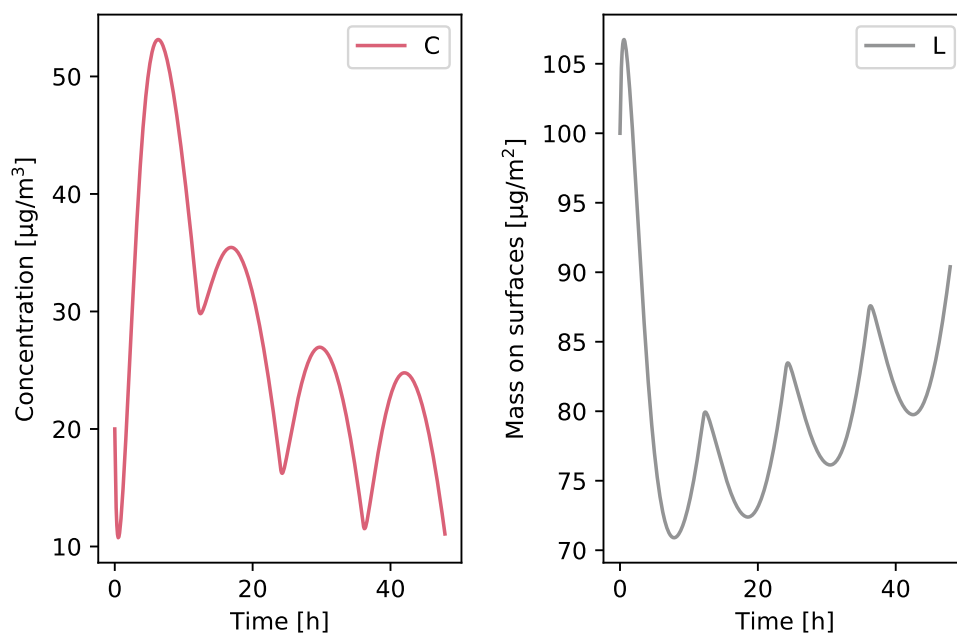


Figure 3.10 • Polydisperse aerosol: Concentration evolution in the air (left) and surfaces (right) compartments after 48 hours.

The evolution of the fractions of mass distribution is presented on Figure 3.11: the periodic increase of the fraction of particles of the 10 μm size class are related to the resuspension spectrum being homoge-

neous (ρ is the same for every size-class). On this Figure too we can observe that the steady state is not reached after 48 hours.

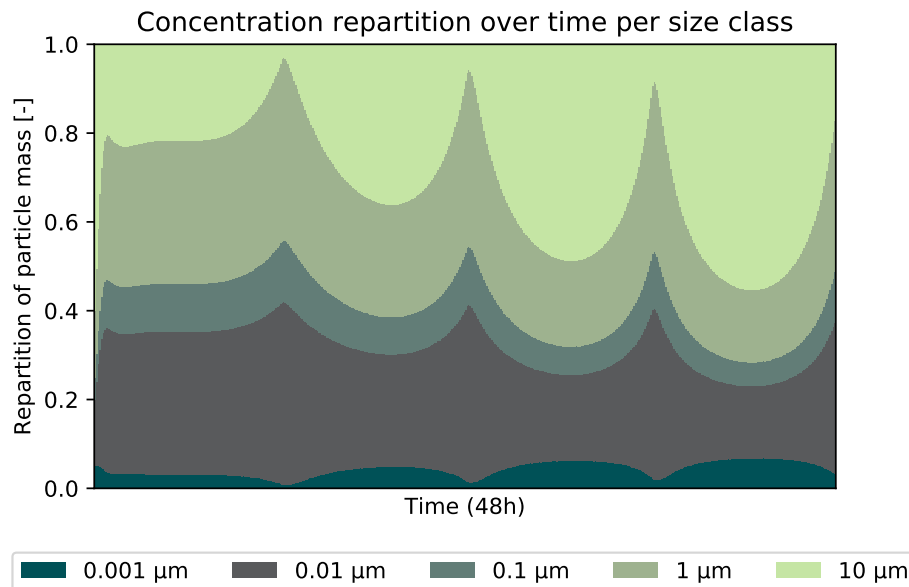


Figure 3.11 • Evolution of the fraction of each particle size class over time.

Note – The simplified resuspension rate ρ used for the example assumes there is a sufficient quantity of particles of every size class deposited on surfaces and available for resuspension, which may well be wrong. It is also noteworthy that obtaining a representative value of ρ depending on the size classes remains a challenge in practical applications.

The code for n size classes using a Crank-Nicolson scheme is provided on Github: [n size-classes air quality model](#).

Continue exploring...

Question 21 Use a log-normal distribution for a proper n -size class distribution. How does the distribution evolve over time?
Details about the properties of log-normal functions can be found in [15].

Question 22 Using a filter efficiency per size-class as per the simple and useful definition in [17], perform the filter clogging computation done in Section 2 for a polydisperse aerosol (you will have to choose a MERV filter class).
What do you observe as far as the indoor distribution is concerned?

2 Heat and mass transfer in walls

The evolution of temperature and moisture content in building walls is a coupled heat and mass transfer problem. Its interesting feature lies in the times scales of both phenomena that may vary by one to two orders of magnitude. The non-linearity arising from the dependance of transfer coefficients to moisture makes it even more challenging. In the following section, a numerical means of solving such problems is proposed.

2.1 Phenomenon & Governing equations

Most of the materials used in construction are permeable to air and water vapour through their porous microstructure. Within the pore network, complex heat transfer occur as radiation adds up to convection and conduction. The phenomena at the pore scale however are not contradictory with the measured equivalent properties of materials made on macroscopic samples.

The common macroscopic assumption is that walls are constituted of media with open pores where water vapour may circulate and is likely to be adsorbed at the solid surface of pores. Humidity is hence adsorbed on the surface of pores as superimposed layers of H₂O molecules. When the water obstructs the pore, capillary condensation occurs and liquid water appears. However, up to ~ 90 [%] relative humidity, water transport is essentially driven by vapour transfer and this range is called the *hygroscopic region*. Above this threshold, liquid water fills most of the pores volume, starting with the smallest ones.

A fundamental relationship in coupled heat and mass transfer modeling is the water sorption isotherm, illustrated on Figure 3.12. It provides a link between the mass water content w [kg_w/kg] and the relative humidity φ , for example using a mathematical equation:

$$w = \frac{a}{\left(1 - \frac{\ln(\frac{p}{p_s})}{b}\right)^{\frac{1}{c}}} \quad (3.4)$$

where p and p_s are the vapour pressure and saturated vapour pressure and a, b, c are coefficients fitted on experimental data.

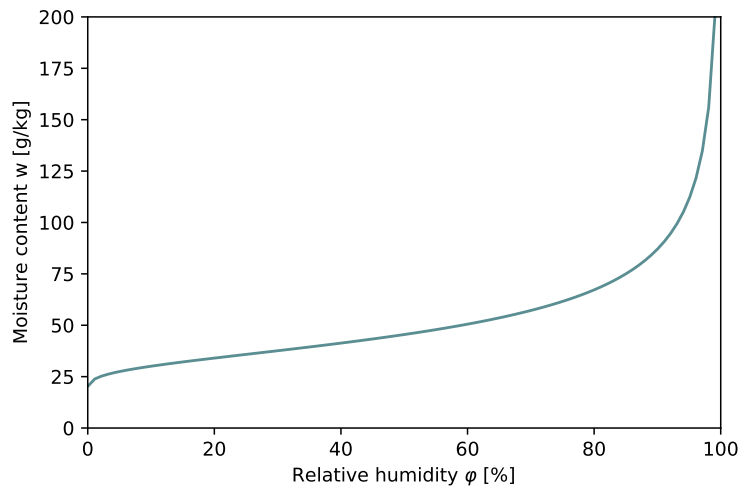


Figure 3.12 • Sorption curve for brick
($a = 1000, b = 146 \times 10^{-6}, c = 1.59$ in Equation (3.4)).

Note – The macroscopic behaviour of the material, for instance its water content depending on relative humidity, is mainly affected by the size distribution of its pores, that is: its microstructure.

Conversely, the desorption process, when the material releases vapour, follows a similar trend as presented on Figure 3.12, excepted for a hysteresis which we will neglect here. Capillary pressure owing to the presence of water in pores is also supposed to be negligible within the moisture content range considered. In the sequel, we will assume the driving potential of moisture transfer is the vapour pressure gradient and the material remains within the hygroscopic region. Sound idea: Indeed, vapour pressure is continuous even at the interfaces between materials, unlike water content!

In porous media, the coupled system between temperature T and water vapour pressure p writes as [4]:

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) + L_v \frac{\partial}{\partial x} \left(\delta_v \frac{\partial p}{\partial x} \right) \quad (3.5)$$

$$C_m \frac{\partial p}{\partial t} = \frac{\partial}{\partial x} \left(\delta_v \frac{\partial p}{\partial x} \right) \quad (3.6)$$

The coupling term appears as $L_v \frac{\partial}{\partial x} \left(\delta_v \frac{\partial p}{\partial x} \right)$. It is the product of the latent heat of water and the change in water content driven by pressure gradient.

Interestingly, the water content w appears in the moisture storage capacity C_m of Equation (3.6):

$$C_m = \left| \frac{\partial w}{\partial \varphi} \right| \frac{1}{p_s(T)} \quad (3.7)$$

Note – The moisture storage capacity C_m in [kg_w/kg/Pa] is defined as the slope of the sorption curve and acts similarly as a variable heat capacity C_p [J/kg/K] would for heat transfer (replacing p by T and δ_v by λ in Equation (3.6) makes it look even more familiar). At each time step of the simulation, the slope of Equation (3.4) will hence be evaluated with respect to the actual relative humidity φ in the material's elementary volume.

In this coupled problem, the heat diffusion and mass diffusion processes exhibit different time scales. Indeed, the characteristic time for heat transfer $t_h^* = L^2/\alpha$ and the one for mass transfer $t_m^* = L^2/\delta_v$ may differ by one or two orders of magnitude, owing to their respective heat and mass diffusivities.

The coefficients of system (3.5), (3.6) depend on the water content w , as already explicited for C_m . The other water content dependencies are listed below:

- The specific heat capacity depends on the water content $C_p(w) = C_p + \frac{w}{\rho} C_{pw}$, although the water contribution may seem low compared to the dry value*.
- The conductivity depends on the water content such that $\lambda = \lambda_0 + kw$. For brick, the coefficients are equal to $\lambda_0 = 0.996$ [W/m/K] and $k = 0.006$ [W.m³/m/K/kg_w].
- The vapour permeability δ_v [m²/s] varies for instance as $\delta_v(w) = \delta_0 + kw$. Values for spruce can be $\delta_0 = 1.1 \times 10^{-7}$ and $k = -1.57 \times 10^{-10}$.

Other mathematical formulations fitting different behaviours of materials may be found in literature. For instance references [5, 14] provide a few typical values for the water and temperature-dependent thermophysical properties of materials. Figure 3.13 illustrates the evolution of the vapour diffusion coefficient and of conductivity with water content.

*The order of magnitude of the water content $w \sim 10^{-3}$ [kg_w/kg] yields $w \times C_{pw} \sim 10^1$ [J/kg/K], which is small *versus* the dry material heat capacity $C_p \sim 10^3$ [J/kg/K]. Hence the variation can arguably be neglected, but we are trying to make it non-linear, don't we?

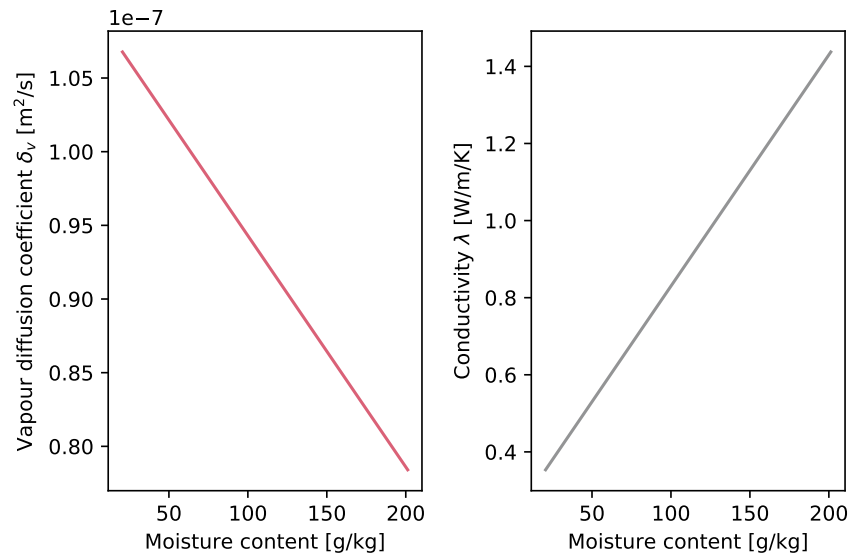


Figure 3.13 • Examples of evolution of the diffusion coefficient δ_v (left) and the conductivity λ (right) in function of water content.

2.2 Modelling heat & mass transfer through walls

Having broached the fundamentals of heat and mass transfer within construction materials, we will now move on to the numerical modelling. Figure 3.14 presents the notations of material properties and boundary conditions around a wall.

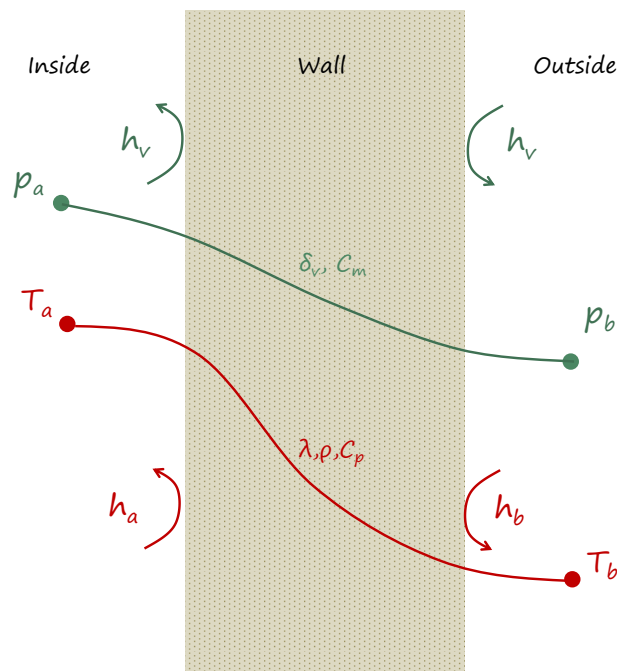


Figure 3.14 • Schematic of the case study: wall with temperature and pressure boundary conditions.

Typical values for the superficial heat transfer coefficients are $h_a = 8$ [W/m²/K] and $h_b = 25$ [W/m²/K] on the indoor and outdoor sides. The surface mass transfer coefficient on both sides can be supposed to be equal as in [5] $h_v = 3 \times 10^{-8}$ [m/s].

Numerical implementation

In the solid nodes, the system of Equation 3.5 writes numerically as follows, with all transfer coefficients depending on water content:

$$\Delta x \rho C_p \frac{T_i^+ - T_i}{\Delta t} = \lambda \frac{T_{i+1} + T_{i-1} - 2T_i}{\Delta x} + L_v \delta_v \frac{p_{i+1} + p_{i-1} - 2p_i}{\Delta x} \quad (3.8)$$

$$\Delta x C_m \frac{p_i^+ - p_i}{\Delta t} = \delta_v \frac{p_{i+1} + p_{i-1} - 2p_i}{\Delta x} \quad (3.9)$$

Rearranging the previous equations, using Fourier's number and introducing $F_{ow} = \frac{\delta_v \Delta t}{\Delta x^2}$ as an equivalent of the Fourier number for mass transfer, we obtain:

$$T_i^+ = T_i(1 - 2F_o) + F_o(T_{i+1} + T_{i-1}) + F_{ow} \frac{L_v}{\rho C_p} (p_{i+1} + p_{i-1} - 2p_i) \quad (3.10)$$

$$p_i^+ = p_i(1 - 2\frac{F_{ow}}{C_m}) + \frac{F_{ow}}{C_m} (p_{i+1} + p_{i-1}) \quad (3.11)$$

At the air interface, the superficial vapour transfer coefficient and the vapour diffusion within the porous media occur. Figure 3.15 shows the configuration with h_v [m/s] the vapour transfer coefficient and δ_v [m²/s] the vapour diffusion coefficient.

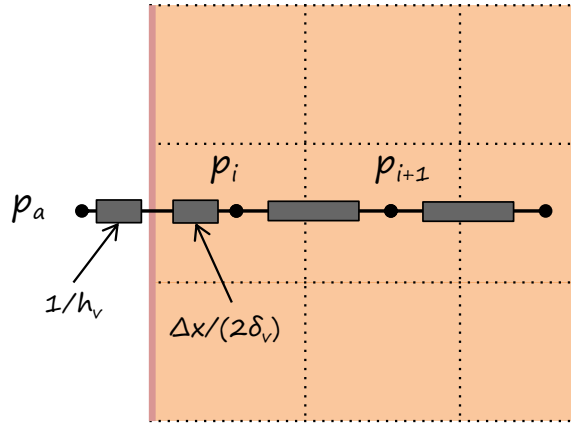


Figure 3.15 • Finite volume method for vapour transfer at the air interface.

Similarly to Section 2.3 of Chapter 1, for the first solid node *i.e.* at the interface between air and solid matter, the system writes:

$$\rho C_p \Delta x \frac{T_i^+ - T_i}{\Delta t} = \frac{\lambda}{\Delta x} (T_{i+1} - T_i) + \frac{T_a - T_i}{\frac{1}{h_a} + \frac{\Delta x}{2\lambda}} + L_v \left(\delta_v \frac{p_{i+1} - p_i}{\Delta x} + \frac{p_a - p_i}{\frac{1}{h_v} + \frac{\Delta x}{2\delta_v}} \right) \quad (3.12)$$

$$\Delta x C_m \frac{p_i^+ - p_i}{\Delta t} = \frac{\delta_v}{\Delta x} (p_{i+1} - p_i) + \frac{p_a - p_i}{\frac{1}{h_v} + \frac{\Delta x}{2\delta_v}} \quad (3.13)$$

Much alike the example of heat transfer at the air interface for mere conduction presented in Chapter 1, Section 2.3, we will use an "equivalent" Fourier number that includes convection. Rearranging and simplifying, we obtain:

$$T_i^+ = T_i(1 - F_o - F_o^{eq}) + F_o T_{i+1} + F_o^{eq} T_a + \frac{L_v}{\rho C_p} \left(-p_i(F_{ow} + F_{ow}^{eq}) + F_{ow} p_{i+1} + F_{ow}^{eq} p_a \right) \quad (3.14)$$

$$p_i^+ = p_i \left(1 - \frac{F_{ow}}{C_m} - \frac{F_{ow}^{eq}}{C_m} \right) + \frac{F_{ow}}{C_m} p_{i+1} + \frac{F_{ow}^{eq}}{C_m} p_a \quad (3.15)$$

where $F_{ow}^{eq} = \frac{\Delta t}{\Delta x (\frac{1}{h_v} + \frac{\Delta x}{2\delta_v})}$ is the equivalent Fourier number for mass transfer including the convective term (its heat transfer equivalent F_o^{eq} has been defined previously in Section 2.3).

Solving this system of equations is accomplished with the array-splitting technique introduced in Section 1.4 of Chapter 3. In the present case, one array containing the temperature and vapour pressure is concatenated in a single row $[T, p_v]$. With the array split in two halves, the nodes T_i and p_i correspond to the same position in space and hence the access by index is straightforward. The implementation for the function to be solved is proposed below:

```
def fc_coupled_HAM(vec_Tpv, vec_Tpv, K, Fo, Fow, dt, rho, Cp, Cpm, Lv) :
    # split the array
    n=int(len(vec_Tpv)/2) # n is the index of a half array
    T,pv = vec_Tpv[0:n] ,vec_Tpv[n:]
    Tp,pvp = vec_Tpv[0:n],vec_Tpv[n:]
    # we need phi in order to actualise w
    phi=pv/fc_pvsat(T)*100
    # now compute w from the sorption curve
    w=fc_w_phi(phi)
    # update the properties in the media
    Cpm,dw,dphi=update_Cpm(w,phi,T)
    Fo=update_Fo(w,k,rho,Cp,dt,dx)
    Fow=update_Fow(w,dt,dx)
    # Crank-Nicolson explicit and implicit parts for T
    exp_T=0.5 * (Fo * np.dot(K,T) + Lv * Fow/(rho * Cp) * np.dot(K,pv))
    imp_T=0.5 * (Fo * np.dot(K,Tp) + Lv * Fow/(rho * Cp) * np.dot(K,pvp))
    T_term = -Tp + T + exp_T + imp_T
    # Crank-Nicolson explicit and implicit parts for pv
    exp_pv=0.5 * (Fow/Cpm * np.dot(K,pv))
    imp_pv=0.5 * (Fow/Cpm * np.dot(K,pvp))
    pv_term = -pvp + pv + exp_pv + imp_pv
    # send back to 'fsolve' as one array
    return np.hstack([T_term,pv_term])
```

The definition of initial conditions, of the transfer matrix and the time loop setup are very similar to the previous applications for heat transfer. Take a look at the details in the comments of [the code available online](#) and its comments.

Application: Imposed pressure and temperature boundary conditions

Before moving on to convective boundary conditions, we may want to get familiar with a stripped down version of the problem: Our case study is a brick wall sample with imposed temperature and pressure on both sides. The boundary conditions T_a, T_b, p_a, p_b remaining constant during the simulation.

The presence of variable coefficients in the equations implies to use a relatively small time step when solving, irrespectively of the unconditional stability of the Crank-Nicolson scheme. In the sequel, the pressure, temperature, relative humidity and water content are hence plotted at larger time intervals in order to clarify the presentation of results.

Figure 3.16 presents the evolution of relative humidity and water content in the wall every 6 minutes over ~ 144 [min]. The simulation is characterised by boundary conditions (T_a, T_b) and initial conditions (T_{init}, p_{init}) such that:

$$T_a = T_b = 20 \text{ [}^\circ\text{C]} \quad (3.16)$$

$$p_a = p_b = 1200 \text{ [Pa]} \quad (3.17)$$

$$T_{init} = 20 \text{ [}^\circ\text{C]} \quad (3.18)$$

$$p_{init} = 1000 \text{ [Pa]} \quad (3.19)$$

One observes that the relative humidity increases over this period from ~ 42 to ~ 52 [%] in the central

part of the sample. Reflecting the shape of the sorption curve, the water content is higher where relative humidity is high.

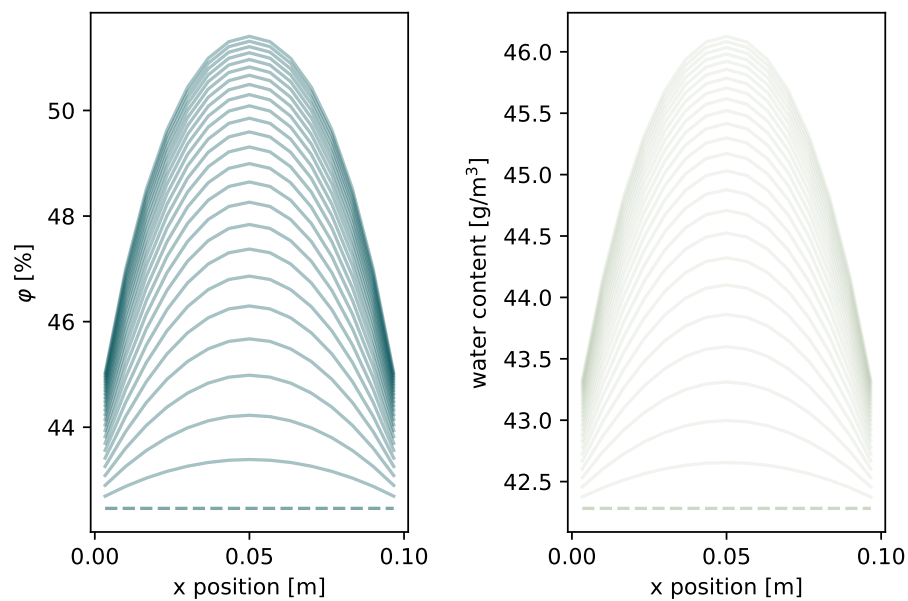


Figure 3.16 • Relative humidity and water content profiles within the sample (dashed lines representing initial conditions).

The temperature and vapour pressure profiles for the same period are plotted on Figure 3.17: the temperature drop in the center of the sample owes to the sorption process.

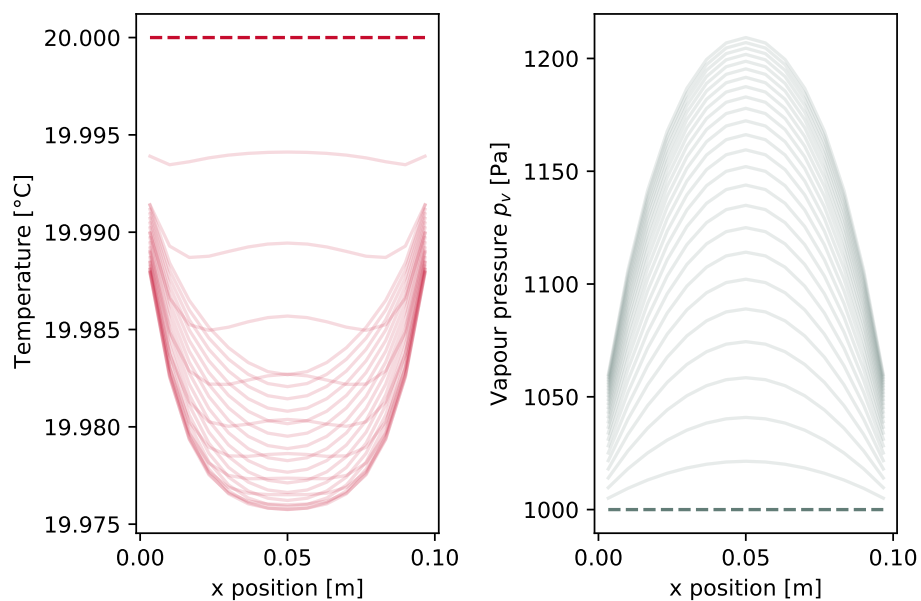


Figure 3.17 • Vapour pressure and temperature profiles within the sample (dashed lines representing initial conditions).

Another simulation is performed with following boundary conditions:

$$T_a = 20 \text{ [}^\circ\text{C]} \quad (3.20)$$

$$T_b = 40 \text{ [}^\circ\text{C]} \quad (3.21)$$

$$p_a = 1200 \text{ [Pa]} \quad (3.22)$$

$$p_b = 1500 \text{ [Pa]} \quad (3.23)$$

$$T_{\text{init}} = 15 \text{ [}^\circ\text{C]} \quad (3.24)$$

$$p_{\text{init}} = 1300 \text{ [Pa]} \quad (3.25)$$

The results obtained are presented on Figure 3.18, where the evolution of relative humidity, water content, temperature and vapour pressure is plotted over a ~ 144 [min] period. The evolution of vapour pressure is especially affected by the change in C_m depending on the material's humidity and exhibits an S-shape.

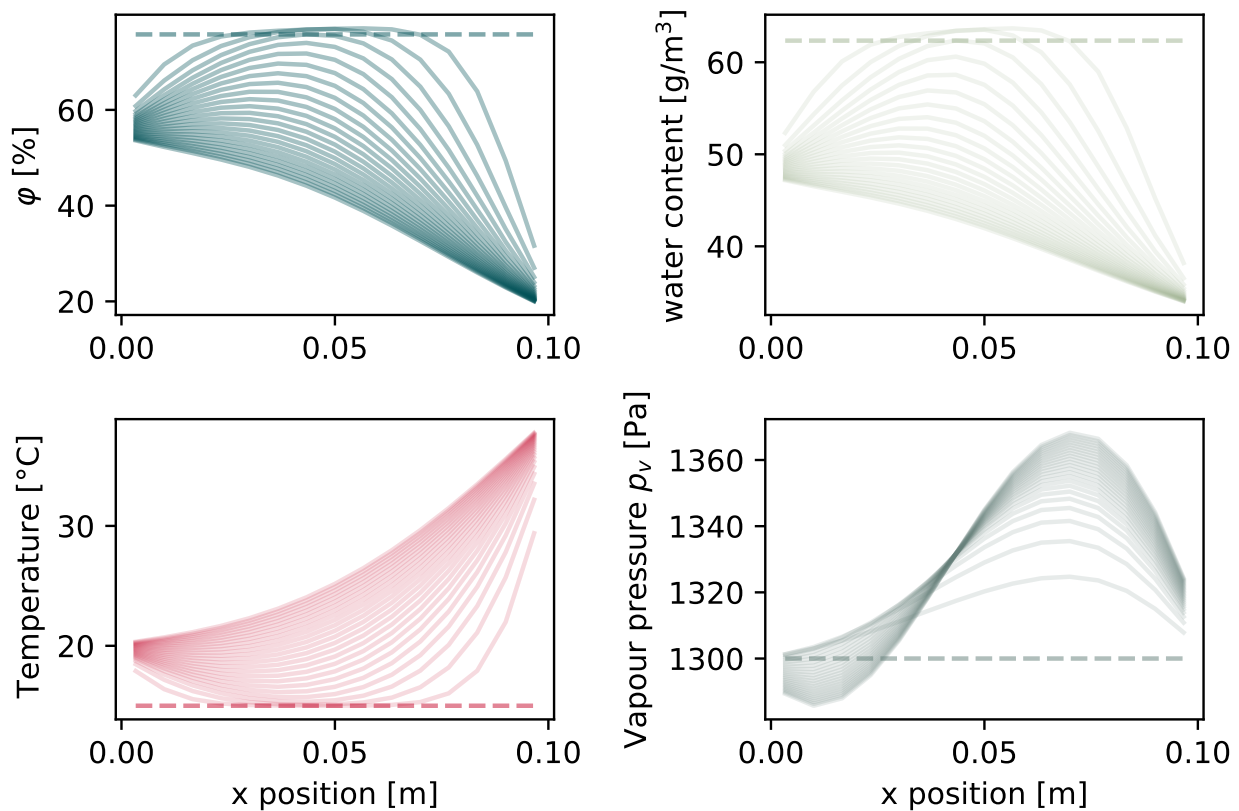


Figure 3.18 • Relative humidity and water content profiles within the sample (dashed lines representing initial conditions).

The 1D heat and mass transfer code for this setup can be downloaded here: [Heat & mass transfer in 1D with imposed pressure and temperature](#).

2.3 Observations on the methods

A strong non-linearity originating from the water content dependency to relative humidity makes this problem quite particular.

Initial conditions

The work by [5] underlines that the choice of initial condition is not straightforward. Quoting previous work by [20], an accepted initial humidity level would be to start with 40 % relative water content for existing structures.

At early time steps, the values obtained may exhibit an oscillatory behaviour (remember Chapter 1, Section 3.2: *unconditionnally stable* only means the oscillations eventually vanish). The release of latent heat, especially when the chosen initial conditions cause large moisture gradients, may provoke oscillations and/or non-physical results.

Integration scheme

Given the non-linearity of transfer coefficients, the implicit method presented above has the drawback of requiring small time steps plus sub-iterations to ensure a proper solving of equations. In recent years, the explicit Du Fort-Frankel scheme [9], became increasingly popular in our field of interest [3, 4, 12] and is recognised as a state-of-the-art method to solve heat and mass transfer problems.

Briefly, the numerical scheme writes as follows for mere conduction:

$$T_i^+ = \frac{1 - F_o}{1 + F_o} T_i^- + \frac{F_o}{1 + F_o} (T_{i+1} + T_{i-1}) \quad (3.26)$$

where, similarly as T^+ is the value of the temperature field at time $t + \Delta t$, T^- is the temperature field at $t - \Delta t$ and T remains the value of the current time step.

Figure 3.19, provides a graphical interpretation of the Du Fort-Frankel scheme, compared to the explicit and implicit ones (we reproduce here the scheme presented in Chapter 1 in order to facilitate the graphical understanding of differences).

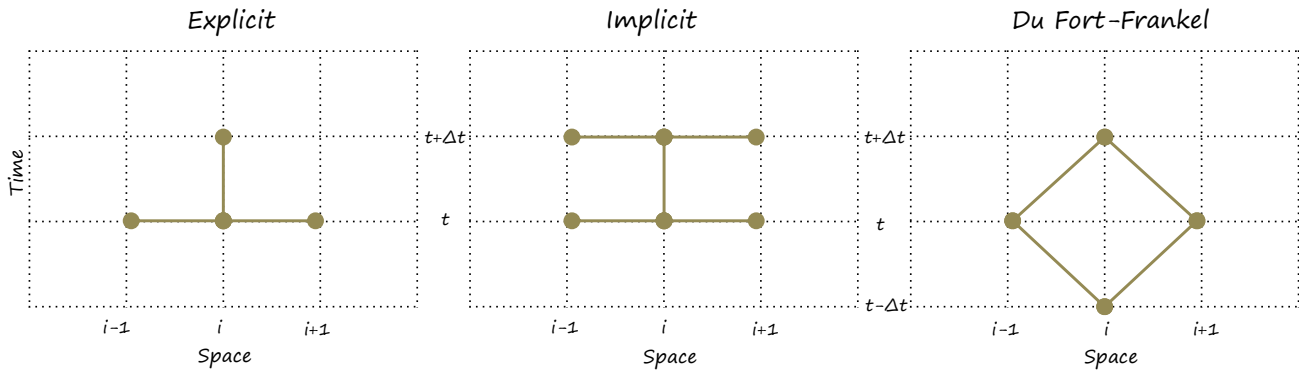


Figure 3.19 • Time and space steps involved in the explicit, implicit and Du Fort-Frankel schemes.

The initial conditions $T_i(t = 0)$ are known. However, $T_i(t = \Delta t)$ has to be determined with another scheme or to be duplicated from the initial conditions. A specific treatment also applies to boundary conditions, as they have to be defined "twice".

Another mathematical approach called "super time-stepping" was recently applied to building simulation and allows for an efficient computation of heat and mass transfer over longer periods. An example of application and a description of the method can be found in [1, 2].

Continue exploring...

- Question 23** Does the specific heat capacity has to be amended with water content? Perform the simulation with/without and conclude.
- Question 24** Modify the implicit scheme for treatment of non-linearity, using the cross-Crank-Nicolson scheme "*cCN*", presented in the remarkably clear reference [11].
- Question 25** Evaluate the impact of variable transfer coefficients: perform two simulations with and without variable properties. Check the values obtained and the simulation time.
Hint: The easiest implementation is to change the functions for the properties calculation so they return constant values.
- Question 26** Adapt the model in order to include convective boundary conditions as per the sketch on Figure 3.14.

3 Parameter fitting on differential equations

This section provides an insight at problems that combine minimisation with the topics exposed in the previous chapters: temporal evolution, implicit integration schemes and partial differential equations. Its purpose is to provide a kick-start for other applications through two examples.

3.1 Minimisation in practice

Most of us have already used the "add trendline" function in a famous spreadsheet commercial software. It is a useful means of putting a mathematical relationship (*e.g.* a polynom) passing near data points (*e.g.* measurements), enabling the use of a continuous function between known points. Such procedures were used for the particle size dependency of filter efficiency in [17] or to fit the pressure drop and efficiency functions used in Section 2.1 of Chapter 2.

Apart from abstract polynomial functions, identifying the parameters of an equation may also be of practical interest. A common application in buildings is the gas tracer method: monitoring the concentration decay allows for an *a posteriori* determination of the air change rate [8], minimising the distance between model and measurements. For simple problems, manual or spreadsheet determination may be sufficient, however in the case of more complex cases a minimisation procedure can prove to be useful.

Three items are required for proper minimisation:

- A "complex" problem: it is always a sound alternative to use a nifty workaround rather than a conjugate gradient algorithm.
- An *objective function*: this is the function to be minimised *e.g.* the error between the simulated and measured quantity. The objective function is the compass of any algorithm used and determines the results obtained.
- An appropriate minimisation procedure: for linear problems, gradient-based methods are in general well adapted (an introduction can be found [on wikipedia](#)). In the case of non-linear or non-differentiable problems, genetic algorithms are often useful (if need be, take a look at the outstanding *deap* library).

The following points may be regarded as good practice:

- Give your parameters physical bounds. Unlike their users, minimisation procedures do not mind the likeliness of the results obtained (*e.g.* negative pressure drop coefficients, temperatures below 0 [K], *etc.*).
- As you may discover reading [the documentation](#), it is embarrassingly simple to try different minimisation procedures with python. Depending on the studied problem you may experience generous speed-ups when changing the solving procedure.
- Give a try to various initial guesses: depending on the starting point of the algorithm *local minima* may be found and influence the displayed results or the execution velocity.

Tried all of the above and it still does not fit? If you are fitting measured data, maybe the equations used are not representative of the phenomenon measured: reconsider the physical model. If you are trying to optimise a system, re-examining the objective function might be a good start.

Let us now have a look at examples of minimisation in practice, depicted in the two next sections.

3.2 Automatic tuning of a PID

Using the tank drainage problem presented in Chapter 2 Section 3.2, the automatic tuning of a PID controller will be exposed hereinafter.

Presentation of the problem

Consider the gravity draining tank problem presented in the previous chapter. From a given initial filling level, we now aim at reaching the set value as fast as possible. As PID controllers are prone to oscillations and overshoot, a stable combination of the PB, T_u, T_d parameters must be found. We perform an *a posteriori* optimisation in the sense that combinations of parameters are evaluated on the total simulation time.

As a summary, the requirements are the following:

- Find the combination of proportional band, integration time and derivation time allowing to swiftly reach the set value.
- Avoid overshoot in order to prevent water overflow.
- Avoid oscillations to preserve the valves from wearing.

The primary goal of the controller being rapidity, the objective function can be defined as the average of the absolute differences between the actual water height and the set water height, taking into account both the water height increase and the potential overshoot:

$$f = \frac{\Delta t}{t_{\max}} \sum_{t=0}^{t_{\max}} |h(t) - H_{\text{set}}| \quad (3.27)$$

Numerically, adding the objective function at the end of the computation of the water height over time consists in writing the following simple lines:

```
# [... perform the water height evolution with a given PID set of values]
# create a vector of set values
height_set= H_set*np.ones(len(height))
# difference between actual and set values
diff=np.asarray(height)-height_set
# average of the absolute difference
diff=abs(diff)
# we want to return a scalar
objective=np.mean(diff)
return objective
```

In order to avoid oscillating systems, a (very) rudimentary penalisation method can be implemented: for instance check if the valve closes during simulation. The proposal below accounts for the number of times when the valve position is equal to zero during the simulation, and provides a penalisation after the time evolution has been computed:

```
# [...] PID calculations etc.
#compute the objective function
objective=np.sum(abs(height)- H_set))
# penalise the result if
penalty=0
# let us find out if the valve is closed (position=0)
# np.where gives the indexes of the vector
# v_pos where its values are nil:
idx=np.where(v_pos==0)

# if more than 3 times..
if len(idx[0])>3:
    penalty=1000 # ... sanction
return objective+penalty
```

Minimisation setup

The values used for the studied case are summarized below:

- The initial water height is $h_{(t=0)} = 0$ [m], for 1 [m] set water height in the tank.
- The maximum supply flow rate is $Q_{\max} = 50$ [L/s].
- The proportional band PB , the integration time T_i and the derivation time T_d are bound as follows:

$$0.1 \leq PB \leq 1 \text{ [m]} \quad (3.28)$$

$$5 \leq T_i \leq 200 \text{ [s]} \quad (3.29)$$

$$0.5 \leq T_d \leq 100 \text{ [s]} \quad (3.30)$$

- The algorithms supporting bounds in `scipy.optimize.minimize` are "TNC", "L-BFGS-B" and "SLSQP", detailed in the documentation. In the studied case, all three provide the same optimal solution and exhibit a similar behaviour in terms of computational time.

Note – Equation (3.30) provides a relatively limited interval for the variation of T_d . As the derivation time in combination with low values of integration time tends to provoke oscillation, the aim here is to limit such behaviour.

The resulting evolution of water height over time obtained using the simulation set up described is given on Figure 3.20. It corresponds to the optimal PID parameters such that:

- The proportional band is $PB = 0.97$ [m]
- The integration time is $T_i = 143$ [s]
- The derivation time is $T_d = 0.5$ [s]

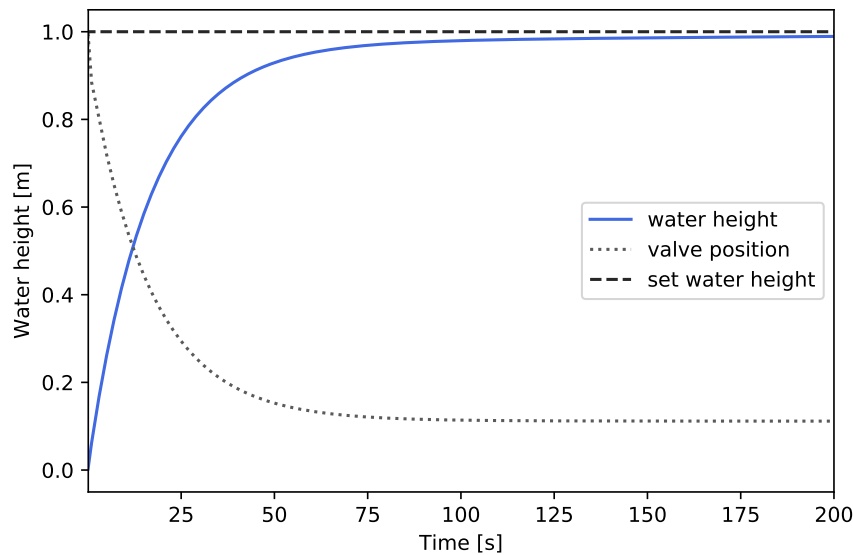


Figure 3.20 • Evolution of the water height and valve position over time for the set of parameters obtained ($PB = 0.97$ [m], $T_i = 143$ [s], $T_d = 0.5$ [s]).

Noticeably, the optimum value for T_d is the lower bound, which could mean that a better set of solutions may exist for a lower value of this parameter: this result is surprising, as the derivative action is supposed to make controllers act faster. In such cases, you may want to reconsider the bounds: *i.e.* extending them and observe the resulting behaviour (*e.g.* forcing a smaller proportional band, for a faster 0 – 100% valve opening coupled with a larger integration time T_i).

Comparison with Ziegler & Nichols' method

A comparative of the results obtained by the optimisation procedure and Ziegler and Nichols' method is proposed on Figure 3.21. The control is faster and the water height does not overreach the set value in the case of optimisation whereas it does for Ziegler-Nichols' method.

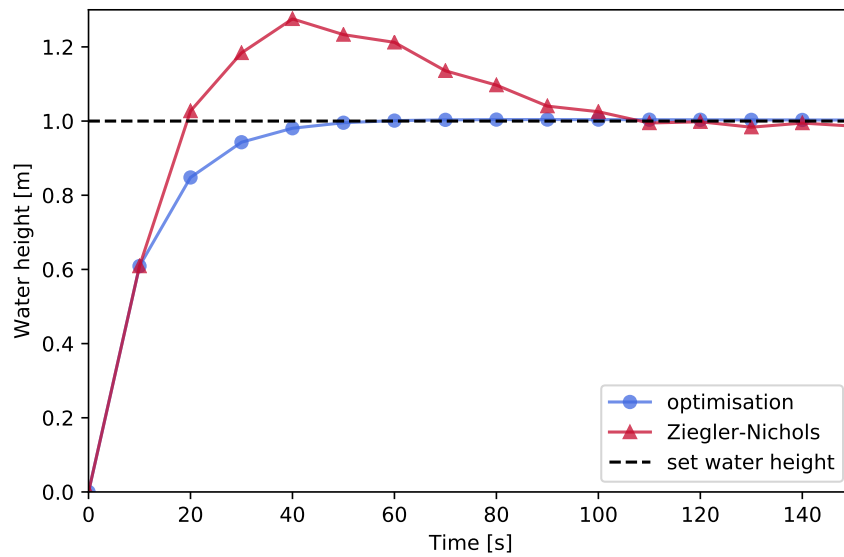


Figure 3.21 • Comparison of the results obtained with Ziegler & Nichols method *versus* the optimisation.

Note – Using this tuning method yields an optimum set up for this particular combination of initial conditions and flow rate: the set of parameters obtained may well be unstable under other operating conditions.

The code used to obtain these results is provided at following address: [Automatic PID tuning](#).

Alternately, get access with your preferred web browser: [Automatic PID tuning \(notebook\)](#).

Continue exploring...

Question 27 Get rid off the penalisation function and see what happens.

Question 28 Despite the penalisation function used, you may have noticed that some sets of parameters provoking oscillations of the valve may be considered as fitting to the optimality criterion: indeed, if the valve does not close completely, no penalisation applies! Build your own function for the detection of oscillations and implement a more generic penalisation in order to improve your objective function.
Hint: computing the average absolute gradient between two consecutive time steps may be a good option. A proposal can be found below:

```
# let's consider only the 2nd half of the valve positions
# (a steep gradient when filling the tank is expected,
# however it is not desirable at the end of the simulation,
# when the set value is reached)
i=int(len(v_pos)/2) #
# valve position at t
v=v_pos[i:-2]
# valve position at t+1
vp=v_pos[i+1:-1]
# proxy for the mean gradient
md=np.mean(abs(vp-v))
# if the gradient is too steep: penalisation
if md>0.02:
    penalty+=100
```

Question 29 Build an objective function that allows for the determination of a stable and fast system
a) when Q_{\max} is low and $h_{(t=0)} \ll H_{\text{set}}$
b) when Q_{\max} is important and $h_{(t=0)} \sim H_{\text{set}}$.

3.3 Air Quality in Underground Stations

In this section, the air quality model presented in [28] serves as a basis for the case study.

Presentation of the problem

In underground stations, a correlation exists between train circulation and particulate matter pollution, as can be observed from Figure 3.22 where the average train traffic and the underground PM₁₀ concentration are plotted. The PM₁₀ data set originates from Paris' Gare du Nord station [28] normalised data*. For the sake of the example, the present data set was arbitrarily multiplied by 100 [µg/m³], which will be the peak concentration value. In the underground stations around the world, the PM₁₀ measured air pollution ranges from 50 ~ 1000 [µg/m³]. Make your own opinion and [check what were the underground pollution levels lately in Paris](#).

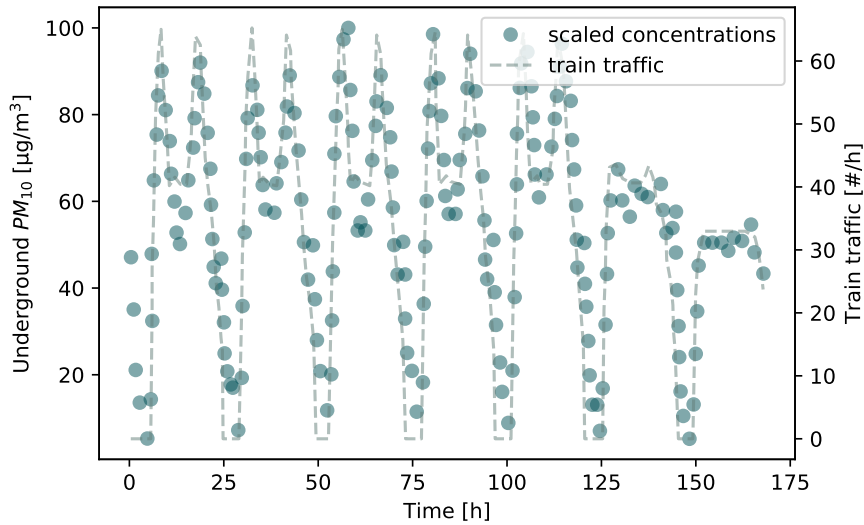


Figure 3.22 • PM₁₀ concentration and average hourly train traffic (the PM₁₀ values are scaled normalised data from [28])

Starting from the observation that train traffic and underground concentration are linked, the physical model proposed in [28] uses a bulk aerosol approach to simulate PM₁₀. The evolution of concentration over time is supposed to be driven by three phenomena. An apparent source term is producing PM₁₀ depending on train traffic N , ventilation brings outside air and particles vanish from the air compartment by deposition onto surfaces:

$$\frac{\partial C}{\partial t} = \alpha N^2 + (\beta N + \tau)(C_e - C) - \delta C \quad (3.31)$$

- The pollutant sources originate both from brake wear and resuspension of particles from surfaces. As measuring the share of each term is not straightforward, an apparent emission term is used. The review of resuspension [21] shows that a resuspension rate varying with the square of velocity is likely. The apparent emission term hence writes αN^2 (this point will be discussed in the following paragraph).
- The air change rate is supposed to be the result of natural ventilation denoted τ , driven by pressure and/or temperature differences, and of the piston effect ventilation $\beta \times N$. The piston effect term corresponds here to the outdoor air brought into the station by the arrival and departure of trains. We define β as the fraction of the station's volume replaced by outside air per train.

*This means that, in the article, the highest concentration has been chosen to divide all others.

- The bulk deposition δC is the one of a monodisperse aerosol, for which the diameter of particles is not known.
- The surfaces compartment is not considered therefore no coupling equation is written.

In the model, train traffic is assumed to be a proxy for velocity far from the surfaces where resuspension may occur. A field measurement campaign in Paris Saint-Michel's underground station showed that this hypothesis satisfactorily represents reality: on Figure 3.23, train traffic and measured hourly average of air velocity on the platform are plotted over time and exhibit correlated patterns. Thus the air velocity can be considered as proportional to the number of trains. The apparent source term including resuspension is correlated to train traffic numbers via air velocity.

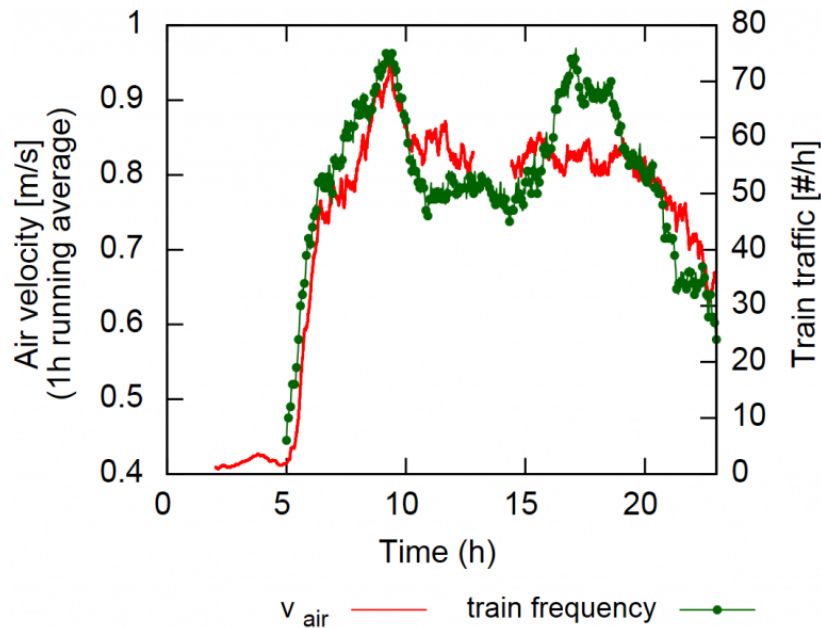


Figure 3.23 • Measured running average air velocity and train traffic over time in Saint-Michel's underground station from [30].

The purpose of minimisation in this case is to find the most appropriate combination of parameters $(\alpha, \beta, \delta, \tau)$ that minimises the distance between the model and the measurements.

Numerical model

Following the method and notations proposed along the previous chapters, Equation (3.31) translates to the function below, combining the explicit part `term_C` and implicit part `term_Cp`:

```
# Crank-Nicolson scheme for air pollution equation in the underground
def fc_crank_nicolson(Cp, C, dt, alpha, beta, tau, delta, N, Ce):
    term_C = dt * (alpha * N ** 2 + (beta * N + tau) * (Ce - C) - delta * C)
    term_Cp = dt * (alpha * N ** 2 + (beta * N + tau) * (Ce - Cp) - delta * Cp)
    return -Cp + C + 0.5 * term_C + 0.5 * term_Cp
```

Note – In the present case, the “cookbook” stability condition for the explicit scheme would write:

$$0 \leq 1 - \Delta t(\beta N + \tau + \delta) \quad (3.32)$$

In the frame of minimisation, the parameters β, δ, τ are varying, hence the time step has to be subsequently adapted. The measured data should be resampled at each evaluation: indeed the difference

between model and measurements must be done at the same moment in time. Instead of implementing such a (tedious) routine, the Crank-Nicolson formulation is preferred.

The temporal evolution can be concisely written as a `while` loop:

```
Ce=15 # constant outdoor concentration
C=PM10[0] # the initial concentration is the 1st measurement
while t < sim_time:
    N=traffic[i] # read the train traffic array
    # solve for C+
    Cp=fsolve(fc_crank_nicolson, C, args=(C,dt,alpha,beta,tau,delta,N,Ce))
    C=Cp # replace C by C+
    t+=dt # increment time
    i+=1 # increment traffic index
```

The model for this evolution depending on the parameters $\alpha, \beta, \delta, \tau$ can be found on Github in the `min_func` function: [Underground IAQ model](#).

Minimisation setup

The values used for the studied case are summarised below:

- The initial concentration is the measured PM_{10} concentration at $t = 0$.
- The time step is set such that $\Delta t = 1$ [h].
- The apparent source term, the piston effect volume, the bulk deposition rate and the base natural ventilation air change rate are bound as follows:

$$0.2 \leq \alpha \leq 0.5 [\mu\text{g}/\text{m}^3/\text{train}^2] \quad (3.33)$$

$$0.05 \leq \beta \leq 0.2 [-] \quad (3.34)$$

$$0.1 \leq \delta \leq 10 [\text{h}^{-1}] \quad (3.35)$$

$$0.1 \leq \tau \leq 1 [\text{h}^{-1}] \quad (3.36)$$

- The objective function is chosen as the mean distance between the simulated values and the measured ones, similarly as Equation (3.27):

$$f = \frac{\Delta t}{t_{\max}} \sum_{t=0}^{t_{\max}} |C_{\text{model}}(t) - C_{\text{measured}}(t)| \quad (3.37)$$

The results obtained with the model are presented on Figure 3.24, yielding a reasonable $4.96 [\mu\text{g}/\text{m}^3]$ average difference between the model and measurements. One can observe the dynamics and amplitude are coherent. The values of the optimal parameters are:

$$\alpha = 0.25 [\mu\text{g}/\text{m}^3/\text{train}^2] \quad (3.38)$$

$$\beta = 0.19 [-] \quad (3.39)$$

$$\delta = 0.17 [\text{h}^{-1}] \quad (3.40)$$

$$\tau = 0.48 [\text{h}^{-1}] \quad (3.41)$$

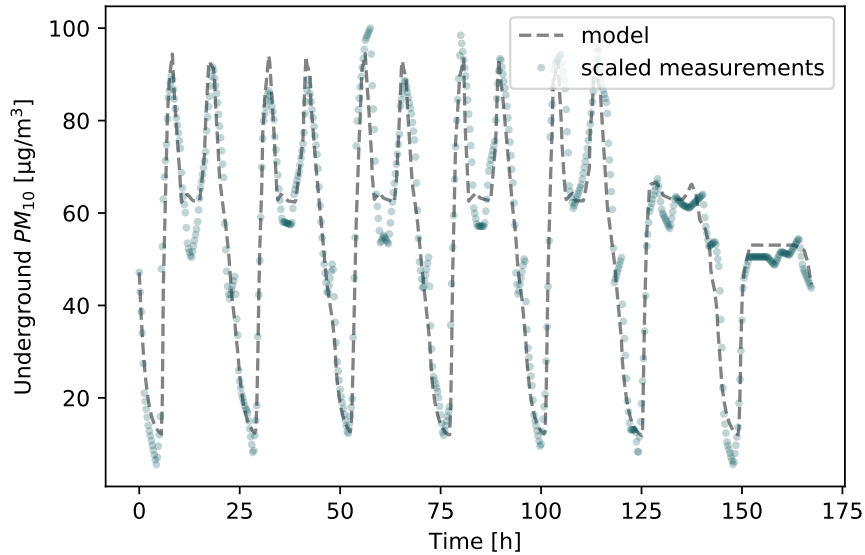


Figure 3.24 • Model *versus* scaled measurements

Note – The value of δ obtained is of particular interest: indeed it allows to find the equivalent diameter that would represent the bulk aerosol, for instance reading Chapter 3, Figure 3.3 “backwards”, starting from the value of δ . Provided it is compared to values in the literature, *e.g.* the diameters found in [26], the equivalent diameter constitutes additional information about the likeliness of the estimation of parameter δ .

The code can be found on Github: [Minimisation of the underground IAQ model](#).

You can also access it via a browser-based application: [Minimisation of the underground IAQ model \(notebook\)](#).

Continue exploring...

- Question 30** Change the model of the apparent source term in order to account for direct emission and resuspension *e.g.* $(\alpha_d \times N + \alpha_r \times N^2)$.
Hint : the paper by [27] provides an order of magnitude of the contribution of both terms.
- Question 31** Add a variable outdoor concentration using the openly, online available data sets of [Air-parif air quality monitoring agency](#).
Hint: use the functions of Chapter 2, Section 2.1 in order to build the average weekly PM₁₀ concentration.
- Question 32** Change the objective function: build your own function focussing on the error during day-time peak concentration, concentration between the rush hours and night time.
- Question 33** Extend the model to two size-classes as in [29].

Chapter 4

Appendix

The appendix contain reminders or details about the theory exposed in the book. Useful code snippets are also to be found here.

1 Demonstrations & complements

1.1 Matrix formulation in 2D

The 2D formulation of the equation (1.15) is proposed here. In order to use the solving procedure, in the form of $[T^+] = [T] + [K][T]$ the matrix of temperatures is to be set to a one-dimensional array. The numbering used to access every element of the matrix must be converted from the 2D array indices to a 1D array, which is less intuitive and will be explained now.

Figure 4.1 illustrates a grid of dimensions $n = 3$ and $m = 4$. The conversion consists in adding the length of the columns, *i.e.* m columns, each time a row is "complete", following the relation $T_{i,j} = T_{i \times m + j}$. In order to access the element $T_{1,2}$, the index is $T_{1,2} = T_{1 \times 4 + 2} = T_6$.

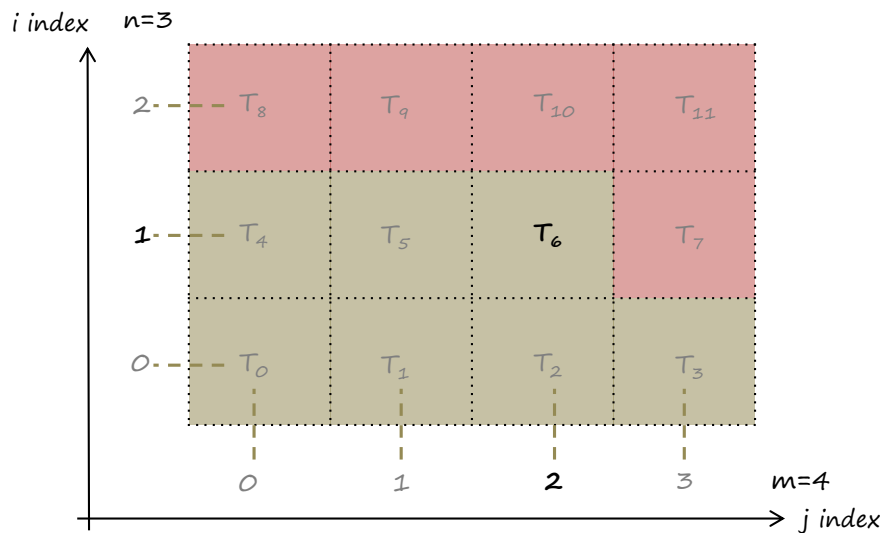


Figure 4.1 • Illustration of the 2D to 1D array index conversion.

Note – Remember that in python numbering is from 0 to $n-1$ or $m-1$, hence the size m of the columns instead of $m-1$, the last index.

The 1D array of temperatures in the 2D space is written as follows:

$$\begin{bmatrix} T_{0,0} \\ \dots \\ T_{i,j} \\ \dots \\ T_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} T_0 \\ \dots \\ T_{i \times m + j} \\ \dots \\ T_{(n-1)m + (m-1)} \end{bmatrix} \quad (4.1)$$

In order to construct the conductivity matrix, we can observe the adjacent terms in the 1D array notation are the "northern" and "southern" terms, respectively $j + 1$ and $j - 1$:

$$i, j - 1 \rightarrow im + j - 1 \quad (4.2)$$

$$i, j \rightarrow im + j \quad (4.3)$$

$$i, j + 1 \rightarrow im + j + 1 \quad (4.4)$$

The "east" and "west" terms $(i + 1, j - 1)$ stand m points away from the index of the central node i, j in the 1D array:

$$i - 1, j \rightarrow (i - 1)m + j \quad (4.5)$$

$$i, j \rightarrow im + j \quad (4.6)$$

$$i + 1, j \rightarrow (i + 1)m + j \quad (4.7)$$

Knowing "where" all the contributors of the heat balance are, a pentadiagonal matrix can be constructed. Given the size of it, only the upper part is presented, the lower being symmetrical:

$$\begin{bmatrix} 1 - 4F_o & F_o & \dots(m) \dots & F_o & \dots & & & & \\ F_o & 1 - 4F_o & F_o & \dots(m) \dots & F_o & & & & \\ \dots & \dots & \dots & & & & & & \\ \dots & F_o & \dots(m) \dots & F_o & 1 - 4F_o & F_o & \dots(m) \dots & F_o & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (4.8)$$

This formulation is less intuitive for the imposition of boundary condition and has a higher memory consumption than the array method, especially when storing the evolution over time of the studied field. In the application presented in Chapter 2, Section 4.3, the array version will be used.

1.2 The K_v value

An equivalence for the K_v value is proposed in this section. The relationship between flow rate and pressure drop may seem somewhat obscure:

$$Q_v = K_v \sqrt{\Delta p} \quad (4.9)$$

Let us start from Bernoulli's pressure drop equation across a valve, combining linear pressure drop $\lambda L/D$ and singular loss ξ :

$$\Delta p = \frac{\rho v^2}{2} \times \left(\frac{\lambda L}{D} + \xi \right) \quad (4.10)$$

Isolating v^2 and multiplying by the surface S in order to make the flow rate Q_v appear, it yields:

$$Q_v^2 = \frac{2S^2}{\rho \left(\frac{\lambda L}{D} + \xi \right)} \Delta p \quad (4.11)$$

We obtain the formulation of Q_v :

$$Q_v = \sqrt{\frac{2S^2}{\rho \left(\frac{\lambda L}{D} + \xi \right)}} \sqrt{\Delta p} \quad (4.12)$$

From which is derived the equation containing the hydraulic impedance Z :

$$\Delta p = Z \times Q_v^2 \quad (4.13)$$

$$Q_v = \frac{1}{\sqrt{Z}} \sqrt{\Delta p} \quad (4.14)$$

The initial relation is completed and K_v is the inverse of the square root of a hydraulic impedance:

$$K_v = \frac{1}{\sqrt{Z}} \quad (4.15)$$

1.3 Demonstration of the relation between Q_v , K_v and a

Using the hydraulic impedance method, the total pressure drop in the circuit equals the sum of the valve pressure drop Z_v and the heater pressure drop Z_r :

$$\Delta p = (Z_v + Z_r) \times Q_v^2 \quad (4.16)$$

As $Q_v = K_v \sqrt{\Delta p}$, it is possible to rewrite the previous equation as:

$$\left(\frac{Q_v}{Q_{v100}} \right)^2 = \frac{Z_{r100} + Z_{v100}}{Z_r + Z_v} = \frac{1 + \frac{Z_{r100}}{Z_{v100}}}{\frac{Z_v}{Z_{v100}} + \frac{Z_r}{Z_{v100}}} \quad (4.17)$$

Noticeably the relationship between the fully opened Z_{v100} and the partially opened Z_v can be written as:

$$\frac{Z_v}{Z_{v100}} = \left(\frac{K_{v100}}{K_v} \right)^2 \quad (4.18)$$

Let us introduce the valve authority a as the ratio of the valve's pressure drop on the total pressure drop:

$$a = \frac{\Delta p_{v100}}{\Delta p_{r100} + \Delta p_{v100}} = \frac{Z_{v100}}{Z_{r100} + Z_{v100}} \quad (4.19)$$

Hence $Z_{r100} = Z_{v100} \frac{1-a}{a}$, also written as:

$$\frac{Z_{r100}}{Z_{v100}} = \frac{1-a}{a} \quad (4.20)$$

Let us hypothesise that

$$\frac{Z_{r100}}{Z_{v100}} \sim \frac{Z_r}{Z_{v100}} \quad (4.21)$$

This assumption means that the hydraulic impedance of the heater when the valve is fully opened is similar to the hydraulic impedance when the valve is partially open.

It is then possible to rewrite Equation (4.17) as:

$$\left(\frac{Q_v}{Q_{v100}} \right)^2 = \frac{1 + \frac{1-a}{a}}{\left(\frac{K_{v100}}{K_v} \right)^2 + \frac{1-a}{a}} \quad (4.22)$$

Finally, we obtain a relationship providing the evolution of the flow rate depending on the K_v value and the authority of the valve:

$$\frac{Q_v}{Q_{v100}} = \frac{1}{\sqrt{a \left(\frac{K_{v100}}{K_v} \right)^2 + 1 - a}} \quad (4.23)$$

The link between the K_v value and the valve position y is provided explicitly by equation $\frac{K_v}{K_{v100}} = f(y)$, representing the valve's characteristic curve.

2 Code

In case you do not have an access to `stackoverflow`, you may find it convenient to have these code snippets at hand.

2.1 Solving (systems of) Equations

A reminder of the syntax for solving a non linear equation.

```
import numpy as np
from scipy.optimize import fsolve

# find x in this non linear
def fc_inverse(x,a,b,n,q):
    return x- q/a*np.exp(b*np.power(x,n))

q=4440 # parameter
tau0=65 # initial guess

# solve and get the solution!
tau=fsolve(fc_inverse, tau0, args=(a,b,n,q))
```

It also works with systems of equations, as exposed all along this book:

```
def fc_CN(Tp,T,K,beta,Fo):
    return Tp -T - 0.5*Fo*np.dot(K,T) - 0.5*Fo*np.dot(K,Tp)

T_plus_CN = fsolve(fc_CN, T_CN, args=(T_CN,K,beta,Fo))
```

2.2 Data resampling

A handy routine to resample data using Fourier's theory:

```
import numpy as np
from scipy import signal
# load data
pm10=np.loadtxt("PM10.txt")
t=pm10[:,0] # column 0 is the time
PM=pm10[:,1] # column 1 is the data
# prepare resampling
total_time=168 # 24 h * 7 days = 168 points
nb_pts=168 # how many points
PM_resample = signal.resample(PM, nb_pts)
tnew = np.arange(0, total_time, total_time/nb_pts)

# get rid off potential negative values
idx=np.where(PM_resample<0)[0]
for elt in idx:PM_resample[elt]=0
# plot to compare original/resampled
plt.plot(tnew, PM_resample, '--')
plt.plot(t,PM, 'r--',alpha=0.65)
```


2.3 Data interpolation

Quintessential code lines allowing to interpolate between data points (e.g. digitised data) and compute the values at regular intervals:

```
import scipy.interpolate
# the measured data set
x = t
y = PM
# create an interpolation method from the data
y_interp = scipy.interpolate.interpld(x, y)
# define your interpolation interval
dt=0.25
t_resample= np.arange(min(t), max(t), dt)
# calculate the interpolated values
PM_resample=y_interp(t_resample)
```

2.4 Data fitting

Fitting data may be done using your favourite spreadsheet editor, however if you wish to automate the process or use more complex equations than the usual "polynomial fit", the following may prove to be useful. In the example provided we need a fit of the analytical pressure drop of singularities $\Delta p = \xi \frac{\rho v^2}{2}$, such that $\Delta p = Av + Bv^2$, where A and B are unknown.

```
from scipy.optimize import minimize
import numpy as np
import matplotlib.pyplot as plt

# analytical formula used as a reference for fitting
xi=25.97
rho=1.2
v=np.arange(0,10,0.5)
dp=xi*rho*v**2/2

plt.xlabel("v (m/s)")
plt.ylabel("pressure drop (Pa)")
plt.plot(v,dp,label=r'analytical $\xi \rho v^2/2$')

# minimise discrepancy between model and analytical formula:
# dp = A*v + B*v**2
def fc_to_minimise(x):
    A,B=x[0],x[1]
    dp_num=A*v+B*v**2
    return np.sum(abs(dp-dp_num))

x0=[0,0] # initial guess
sol = minimize(fc_to_minimise, x0, method='SLSQP', tol=5e-3)
A,B=sol.x[0],sol.x[1]
print("solutions : ",round(A,3), round(B,3))

# plot the result
dp_num=A*v+B*v**2
```

```
plt.plot(v, dp_num, marker='o', linestyle='', label=r'fit $Av +Bv^2$')
plt.legend()
```

2.5 Pareto front computation

I found this handy routine somewhere on the internet a while ago*. You may have to play with `maxY=False` and change it to `True` to get the proper Pareto front.

```
def pareto_front(Xs, Ys, maxY=False):
    '''Pareto frontier selection process'''
    sorted_list=sorted([[Xs[i], Ys[i]] for i in range(len(Xs))], reverse=maxY)
    pareto_front=[sorted_list[0]]
    for pair in sorted_list[1:]:
        if maxY:
            if pair[1] >= pareto_front[-1][1]:
                pareto_front.append(pair)
        else:
            if pair[1] <= pareto_front[-1][1]:
                pareto_front.append(pair)
    pf_X = [pair[0] for pair in pareto_front]
    pf_Y = [pair[1] for pair in pareto_front]
    return pf_X, pf_Y
```

*I humbly ask its author to accept my apologies for not mentioning the source, owing to my forgetfulness.

Bibliography

- [1] Madina Abdykarim, Julien Berger, Denys Dutykh, and Amen Agbossou. An efficient numerical method for a long-term simulation of heat and mass transfer: the case of an insulated rammed earth wall. *arXiv preprint arXiv:1909.08416*, 2019.
- [2] Madina Abdykarim, Julien Berger, Denys Dutykh, Lucile Soudani, and Amen Agbossou. Critical assessment of efficient numerical methods for a long-term simulation of heat and moisture transfer in porous materials. *International journal of thermal sciences*, 145:105982, 2019.
- [3] Julien Berger, Thomas Busser, Thibaut Colinart, and Denys Dutykh. Critical assessment of a new mathematical model for hysteresis effects on heat and mass transfer in porous building material. *International Journal of Thermal Sciences*, 151:106275, 2020.
- [4] Julien Berger, Thomas Busser, Sohail Reddy, and George S Dulikravich. Evaluation of the reliability of a heat and mass transfer model in hygroscopic material. *International Journal of Heat and Mass Transfer*, 142:118258, 2019.
- [5] Julien Berger, Sihem Tascas-Guernouti, Monika Woloszyn, and Catherine Buhe. Mould growth damages due to moisture: comparing 1D and 2D heat and moisture models. In *13th International Conference on Building Performance Simulation 2013, Chambéry, France, August 25*, volume 28, pages 2876–2884, 2013.
- [6] Pascal Henry Biwolé, Pierre Eclache, and Frédéric Kuznik. Phase-change materials to improve solar panel’s performance. *Energy and Buildings*, 62:59–67, 2013.
- [7] George E. P. Box, Norman Richard Draper, et al. *Empirical model-building and response surfaces*, volume 424. Wiley New York, 1987.
- [8] Shuqing Cui, Michaël Cohen, Pascal Stabat, and Dominique Marchio. CO₂ tracer gas concentration decay method for measuring air change rate. *Building and Environment*, 84:162–169, 2015.
- [9] E. C. Du Fort and S. P. Frankel. Stability conditions in the numerical treatment of parabolic differential equations. *Mathematical Tables and other aids to computation*, 7(43):135–152, 1953.
- [10] C. P. Dullemond. Numerische Strömungsmechanik Lecture Notes – Chapter 3: Numerical Advection, Summer 2009 Universität Heidelberg.
- [11] Denys Dutykh. How to overcome the Courant-Friedrichs-Lewy condition of explicit discretizations? *arXiv preprint arXiv:1611.09646*, 2016.
- [12] Suelen Gasparin, Julien Berger, Denys Dutykh, and Nathan Mendes. Stable explicit schemes for simulation of nonlinear moisture transfer in porous materials. *Journal of Building Performance Simulation*, 11(2):129–144, 2018.
- [13] Mats Gustafsson, Göran Blomqvist, Erik Swietlicki, Andreas Dahl, and Anders Gudmundsson. Inhalable railroad particles at ground level and subterranean stations—Physical and chemical properties and relation to train traffic. *Transportation Research Part D: Transport and Environment*, 17(3):277–285, 2012.
- [14] C. E. Hagentoft and T. Blomberg. 1D-HAM coupled heat, air and moisture transport in multi-layered wall structures. *Manual with brief theory and an example, Version, 2*, 2000.

- [15] Jost Heintzenberg. Properties of the log-normal particle size distribution. *Aerosol Science and Technology*, 21(1):46–48, 1994.
- [16] Cheng-Hsiung Huang, Chin-I Lee, and Chuen-Jinn Tsai. Reduction of particle reentrainment using porous fence in front of dust samples. *Journal of Environmental Engineering*, 131(12):1644–1648, 2005.
- [17] Wladyslaw J Kowalski and William Parry Bahnfleth. MERV filter models for aerobiological applications. *Air Media, Summer*, 1, 2002.
- [18] Frédéric Kuznik, Joseph Virgone, and Jean Noel. Optimization of a phase change material wallboard for building use. *Applied Thermal Engineering*, 28(11-12):1291–1298, 2008.
- [19] Alvin C. K. Lai and William W. Nazaroff. Modeling indoor particle deposition from turbulent flow onto smooth surfaces. *Journal of aerosol science*, 31(4):463–476, 2000.
- [20] Hyeun Jun Moon. *Assessing mold risks in buildings under uncertainty*. PhD thesis, Georgia Institute of Technology, 2005.
- [21] K. W. Nicholson. A review of particle resuspension. *Atmospheric Environment (1967)*, 22(12):2639–2651, 1988.
- [22] S. V. Patankar. Numerical heat transfer and fluid flow, Series in Computational Methods in Mechanics and Thermal Sciences. *Mechanics*, 1980.
- [23] Jing Qian, Andrea R Ferro, and Kathleen R Fowler. Estimating the resuspension rate and residence time of indoor particles. *Journal of the Air & Waste Management Association*, 58(4):502–516, 2008.
- [24] Imre Salma, Tamás Weidinger, and Willy Maenhaut. Time-resolved mass concentration, composition and sources of aerosol particles in a metropolitan underground railway station. *Atmospheric Environment*, 41(37):8391–8405, 2007.
- [25] G. A. Sehmel and F. D. Lloyd. Particle resuspension rates. Technical report, Battelle Pacific Northwest Labs., Richland, Wash.(USA), 1974.
- [26] Peter Torstensson, Tore Verneresson, Sara Janhäll, Anders Andersson, Fredrik Blennow, and Kristoffer Mossheden. Use of numerical simulation to map and mitigate railway particle emissions. 2019.
- [27] Minghui Tu, Yingying Cha, Jens Wahlström, and Ulf Olofsson. Towards a two-part train traffic emissions factor model for airborne wear particles. *Transportation Research Part D: Transport and Environment*, 67:67–76, 2019.
- [28] E. Walther and M. Bogdan. A novel approach for the modelling of air quality dynamics in underground railway stations. *Transportation Research Part D: Transport and Environment*, 56:33–42, 2017.
- [29] E. Walther, M. Bogdan, and R. Cohen. Modelling of airborne particulate matter concentration in underground stations using a two size-class conservation model. *Science of The Total Environment*, 607:1313–1319, 2017.
- [30] Walther, E. L’hypercube’s webpage on Air Quality. <https://lhypercube.arep.fr/en/qualite-dair/qualite-dair-en-gares-souterraines/>. Accessed: 2020-04-19.

You flipped to the very last page, congratulations! Unfortunately, no correction is –yet– proposed for the questions at the end of each chapter.

How to cite

This book is freely available online. You are most welcome to share it!

If you found it of any use, you may cite it with this reference:

```
@book{walther2021,  
  title      = {{Building Physics Applications in Python}},  
  author     = {Walther, E.},  
  year       = {2021},  
  publisher  = {DIY Spring},  
  address    = {Paris}  
}
```

Errors may well remain in this manuscript: I would be grateful to the reader to send me notice [per inmail](#).