ISBN 978-82-471-2746-9 (printed ver.)
ISBN 978-82-471-2747-6 (electronic ver.)
ISSN 1503-8181

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2011:106

Torgeir Stensrud Ustad

A framework for modeling internal mass transfer in atmospheric freeze drying

Doctoral Thesis
Torgeir Stensrud Ustad

Doctoral theses at NTNU, 2011:106

NTNU

NTNU
Norwegian University of
Science and Technology
Thesis for the degree of
philosophiae doctor
Faculty of Engineering Science and Technology
Department of Energy and Process Engineering

NTNU
Norwegian University of
Science and Technology

Torgeir Stensrud Ustad

A framework for modeling internal mass transfer in atmospheric freeze drying

Thesis for the degree of philosophiae doctor

Trondheim, May 2011

Norwegian University of
Science and Technology
Faculty of Engineering Science and Technology
Department of Energy and Process Engineering

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of philosophiae doctor

Faculty of Engineering Science and Technology
Department of Energy and Process Engineering

ISBN 978-82-471-2746-9 (printed ver.)
ISBN 978-82-471-2747-6 (electronic ver.)
ISSN 1503-8181

Doctoral Theses at NTNU, 2011:106

Printed by Tapir Uttrykk

Abstract

In modeling of atmospheric freeze drying (AFD), the modeling of the product is often confined to either one-dimensional models or multi-dimensional models of very basic geometries. Such models are not able to handle a product of complex shape, and require simplifying assumptions on the geometry of the product in order to work. In addition, although many models incorporate equations that account for the effects of several physical phenomena, they often assume that the coefficients in the equations are either constant or certain functions of external parameters, such as the ambient temperature. The effects of local variations in the internal structure of the product, such as the effects of inhomogeneity or anisotropy on transport properties, are rarely modeled.

The main goal of the work presented in this thesis was to develop a three-dimensional, geometrically flexible framework for modeling AFD, in order to make handling of complicated geometries not only possible, but also straightforward. To this end, the definition of the geometry was made implicit, i.e. any surface or interface is defined as the set of points where a certain geometry defining function (GDF) is zero. The GDFs can be made time-dependent, to account for deformations (shrinkage) as time goes by. Thus, we should in principle be able to model any product in any surroundings with this framework, as long as all surfaces and interfaces can be described by GDFs. Moreover, we can encase an implicitly defined product in a box-shaped, fixed domain. This makes the numerical grid simple and fixed, even if the product itself shrinks with time.

Another important goal was to make the framework accommodate quasilinear transport coefficients, i.e. that the coefficients could depend on local conditions, including local moisture concentration. This allows the effects of inhomogeneity to be included. By allowing different coefficients in different directions, the framework also allows anisotropy to be modeled.

The emphasis was on modeling the mass transfer inside the product, which was assumed to be caused by vapor diffusion through Darcy's law. External mass transfer was modeled as convective, with the ambient conditions as given boundary conditions (i.e. not modeled). A mass transfer equation was developed to describe the sublimation of ice and subsequent diffusion of vapor through the product. It was based on the retreating ice front (RIF) assumption. A method of investigating the validity of the RIF assumption was proposed and tested on pieces of cod fillet, using a scanning electron microscope (SEM).

It was verified that the equation was mathematically well posed, and a spectral method was chosen to solve it numerically. The method was adapted to the specific equation, and several improvements were made to it, to decrease memory costs and time consumption.

To test the framework, simulations were carried out on pieces of cod fillet at -5 °C and -10 °C. The calculated drying curves showed good agreement with experimental results, and the qualitative properties were found to be satisfactory within certain bounds on the parameters.

Acknowledgements

This thesis is based on work carried out in connection with SINTEF project 16X448: *Atmospheric Freeze Drying - SIP*, funded by the Research Council of Norway, as well as on subsequent work partially funded by the Department of Energy and Process Engineering at NTNU.

I would like to thank my supervisors, Professor Ingvald Strømmen and Professor Trygve M. Eikevik, for the advice they have given me on the physical aspects of AFD and on modeling issues. I am also grateful for their effort in obtaining funding for my work.

I would also like to thank Dr. Ingrid C. Claussen for providing me with partially dried samples and background data from her drying experiments on pieces of cod fillet. The background data enabled me to test the modeling framework through simulations, while the samples enabled me to test a proposed method of investigating the ice distribution within product samples.

The SEM study of the cod samples took place at the Department of Materials Science and Engineering at NTNU, where Professor Jarle Hjelen and Tor Arild Nilsen were very helpful in providing me with the necessary equipment and training.

Finally, I would like to thank my father, Dr. Terje Ustad, for his help with the SEM study, and other family members and colleagues, for their encouragement and enduring interest in my work through the long and laborious time I have spent on it.

Torgeir Stensrud Ustad

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

Latin symbols

Symbol	Description	Unit
	Scalars	
a	Semi-axis in x direction	m
A	Generic constant	Varying
b	Semi-axis in y direction	m
c	Semi-axis in z direction	m
c_p	Molar heat capacity at constant pressure	J/(mol·K)
c_V	Molar heat capacity at constant volume	J/(mol·K)
f	Generic function	Varying
g	Generic function	Varying
h_m	Convective mass transfer coefficient	m/s
I	Indicator function	1
k	Thermal conductivity	W/(m·K)
K	Permeability	m^2
N	Number of basis functions	1
Nu	Nusselt number	1
P	Pressure	Pa
Pr	Prandtl number	1
r	Radius	m

Continued on next page

Continued from previous page

R	Universal gas constant	J/(mol·K)
R_{H_2O}	Specific gas constant for water vapor	J/(kg·K)
Re	Reynolds number	1
Sc	Schmidt number	1
Sh	Sherwood number	1
t	Time	s
T	Temperature	K
U	Exact solution	kg/m^3
v	Spatial basis function in numerical method	1
w	Basis coefficient in numerical method	kg/m^3
W	Numerical solution	kg/m^3
x	Spatial coordinate	m
y	Spatial coordinate	m
z	Spatial coordinate	m

	Matrices and vectors (unit given is for each element)	
$\boldsymbol{a}$	Semi-axes vector, $\boldsymbol{a} = (a, b, c)$	m
$\boldsymbol{A}$	System matrix in complex system	1
$\boldsymbol{B}$	Coefficient matrix	Varying
$\boldsymbol{C}$	Diffusivity matrix	m^2/s
$\boldsymbol{D}$	Derivative factor matrix	1/m
$\boldsymbol{e}$	System vector in complex system	kg/m^3
$\boldsymbol{E}$	Generic matrix	Varying
$\boldsymbol{f}$	Mass flux vector	$kg/(m^2 \cdot s)$
$\boldsymbol{F}$	Source term vector	Varying
$\boldsymbol{G}$	Basis generation matrix	1
$\widehat{\boldsymbol{G}}$	Basis generation matrix	1

Continued on next page

Continued from previous page

$\boldsymbol{I}$	Identity matrix	1
$\boldsymbol{j}$	Index vector, $\boldsymbol{j} = (j_1, j_2, j_3)$	1
$\boldsymbol{J}$	Exchange matrix	1
$\boldsymbol{k}$	Index vector, $\boldsymbol{k} = (k_1, k_2, k_3)$	1
$\boldsymbol{K}$	Permeability matrix	m^2
$\boldsymbol{m}$	Index vector, $\boldsymbol{m} = (m_1, m_2, m_3)$	1
$\boldsymbol{M}$	Integral matrix	m^5/s
$\boldsymbol{p}$	Index vector, $\boldsymbol{p} = (p, q, r)$	1
$\boldsymbol{R}$	System matrix in real system	1
$\boldsymbol{s}$	Index vector, $\boldsymbol{s} = (s, t, u)$	1
$\boldsymbol{S}$	Sigma factor matrix	1
$\boldsymbol{U}$	Exact solution vector	Varying
$\boldsymbol{v}$	Velocity	m/s
$\boldsymbol{x}$	Position vector, $\boldsymbol{x} = (x, y, z)$	m
$\boldsymbol{y}$	Generic vector	Varying
$\boldsymbol{z}$	Generic vector	Varying

	Sets (unit not applicable)	
$\mathrm{H}^1(\Omega)$	Subspace of L_2, with L_2 first order spatial derivatives	-
$\mathrm{H}_0^1(\Omega)$	Subspace of $\mathrm{H}^1(\Omega)$, with zero boundary function values	-
L_2	Lebesgue space of square integrable functions	-
L_2^0	Subspace of L_2, with zero boundary function values	-
Q_τ	System domain, $\mathrm{Q}_\tau = \Omega \times (0, \tau)$	-
$\mathrm{V}_2(\mathrm{Q}_\tau)$	Extended solution space	-
$\mathrm{V}_2^{1,0}(\mathrm{Q}_\tau)$	General solution space	-

Greek symbols

Symbol	Description	Unit
	Scalars	
δ	Kronecker delta function	1
Δ	Difference operator	-
θ	Angle in xy/xz-plane	rad
Θ	Average operator	-
μ	Dynamic viscosity	kg/(s·m)
ν	Kinematic viscosity	m^2/s
ρ	Mass density	kg/m^3
$\widehat{\rho}$	Mass concentration	kg/m^3
σ	Sigma factor	1
τ	Total drying time	s
ϕ	Porosity	1
χ	Characteristic function	1
ψ	MATLAB indexing function	1
ω	Concentration ratio	1
Ω	Computational domain, $\Omega = (-a, a) \times (-b, b) \times (-c, c)$	m^3
Ω_k	Subset of Ω	m^3
	Dimensionless matrices and vectors	
α	Index vector, $\alpha = (\alpha, \beta, \zeta)$	1
η	Test function vector	1
κ	Index vector, $\kappa = (\kappa, \lambda, \xi)$	1

Subscripts and superscripts used by some symbols

Subscript	Superscript	Description
		Latin scripts
0		Initial condition or constant
air		Air
conv		Convection
diff		Diffusion
	H	Conjugate transpose
i	i	Element index
ice		Ice
j	j	Generic index
k	k	Generic index
l	l	Direction index
m	m	Generic index
n		Time index
p		Counting index
q		Counting index
r		Counting index
s		Counting index
t		Counting index
u		Counting index
sat		Saturation
vap		Vapor
wb		Wet bulb
		Greek scripts
α	α	Counting index

Continued on next page

Continued from previous page

β	β	Counting index
ζ	ζ	Counting index
κ	κ	Counting index
λ	λ	Counting index
ξ	ξ	Counting index
$\partial\Omega$		Boundary condition

Abbreviations

Short form	**Full form**
AFD	Atmospheric freeze drying
BSE	Backscattered electrons
CH	Centrohermitian
CL	Cathodoluminescence
DFT	Discrete Fourier transform
ESEM	Environmental scanning electron microscope
FFT	Fast Fourier transform
GDF	Geometry defining function
HPD	Hermitian positive definite
LV-SEM	Low vacuum scanning electron microscope
RIF	Retreating ice front
SE	Secondary electrons
SEM	Scanning electron microscope
VFD	Vacuum freeze drying

1. INTRODUCTION

In this chapter we give a brief description of atmospheric freeze drying, present the main ideas involved in the modeling work, and look at the structure and notation of this thesis.

1.1 About atmospheric freeze drying

Atmospheric freeze drying (AFD) is a hybrid drying technology which combines the sublimation mechanism of vacuum freeze drying (VFD) with an air/gas atmosphere at about atmospheric pressure (cf. [13, Chapter 21] or [6]). As in VFD, the product is frozen, so (ideally) both its water and dry structure are fixed in place, and the dominant drying mechanism is that of sublimation and subsequent vapor transport. This fixation not only preserves the structure and porosity of the product, but the absence of liquid water prevents desirable substances (e.g. nutrients in foods) from being dissolved, and subsequently removed from the product. This preserves the quality of the product better than drying in warm air. On the other hand, the rate of drying increases dramatically with temperature, and freeze drying is quite slow in comparison to many other types of drying. And AFD is even slower than VFD, because the driving vapor pressure difference between the sublimation site and the product surroundings is smaller in the case of AFD.

However, VFD is very expensive, because it requires equipment capable of establishing and maintaining a vacuum, as well as cooling equipment to keep the temperature sufficiently low. The idea behind AFD is therefore to attain the product quality of freeze dried products at lower costs than VFD. The lack of need to install and run expensive vacuum equipment is the most obvious advantage AFD has over VFD, but there are also some other advantages:

- The presence of an atmosphere enables convective heat and mass transfer to and from the product. This simplifies and increases the heat supply, which in VFD relies on conduction and/or radiation.

- AFD typically takes place at just a few degrees Celsius below zero, while VFD takes place at much lower temperatures, so the cooling costs per unit time and mass are lower for AFD.
- AFD can be used in conjunction with atmosphere-dependent technologies like fluidized beds, and product units can continuously be added to/removed from the drying chamber, without significantly disturbing the process. This is not the case for VFD.
- In closed circuits, AFD can be used in conjunction with heat pumps, enabling the heat surrendered by the vapor as it condenses to be used to heat the air stream before it enters the drying chamber. This saves energy, compared to drying without heat pumps (cf. [18]). Therefore, the energy efficiency is far higher in AFD than in VFD.

It is also possible to combine AFD with warm air drying through a *step-up* program, where the ambient temperature is raised towards the end of the drying. This speeds up the final part of the drying considerably, without significantly reducing product quality. This is an important point, because the main disadvantage with AFD is that it is slow. The limiting factor in AFD is considered to be the rate of internal mass transfer, so AFD is therefore in practice confined to small products.

1.2 About AFD modeling

There has not been done much modeling work on AFD (see [6] for a review of some of it). Although the number of publications on AFD modeling has risen over the last few years, the models are still typically restricted to specific geometries. But the diversity of products, conditions and physical phenomena modeled has increased.

The predominant assumption in modeling the interior of products is the notion that there is a sharp interface (called the *ice front*) where the sublimation takes place. Outside the ice front, in the so-called *dry region*, the ice has been removed, and vapor from the front travels through the open pores. The product still contains ice in the pores in the *moist region* inside the ice front. Initially, the ice front starts out near the product surface, but it gradually recedes towards the center of the product. Models using this assumption are referred to as *retreating ice front* (RIF) models, or *uniformly retreating ice front* (URIF) models (if the ice front is assumed to retreat uniformly).

A characteristic of most existing AFD models is that they assume that the product has a regular shape (e.g. spherical, cubic or slab-shaped), and that it keeps its size and shape during the drying process (shrinkage is usually not a major concern in AFD, and it is barely present i VFD). This is an assumption we will not rely on.

1.3 The ideas behind the modeling framework

The main idea behind the framework is generality. Instead of trying to develop a specific model for a specific product, we wish to create a framework that can be used to model just about any product under different circumstances. To this end, the work presented in this thesis is based on the following ideas:

- The applicability of the framework should be limited by the (lack of) physical data and knowledge, not by mathematical or numerical considerations, i.e. we should be able to model complicated geometries and transport properties, without having to make unrealistic assumptions on these.
- The geometry of the problem should be represented implicitly, to enable the use of a fixed computational grid, and make it easier to model complicated and time-dependent geometries. This way of representing the geometry might be new to drying, but it is well known in mathematics (where it is called the *level set method*, cf. e.g. [19]).
- The framework should be confined to pure freeze drying (i.e. no transport of liquid water), and the emphasis should be on the moisture concentration and mass transfer inside the product.
- Instead of starting with a very simplified problem and expanding the theory into a more advanced one, the development of the framework should start at the other end. It should start with as few assumptions as possible, and add additional assumptions if and when they become necessary.

Of course, the framework might not be optimal for constructing a model for a particular product, but this is not the goal of this work. **The goal is to construct a tool which can be used to study and analyze different models for product geometries and transport parameters.**

1.4 Thesis structure and notation

The structure of the thesis is as follows:

1. In Chapter 2 we present some theory on implicit representation of geometry, construct a method of approximating complicated geometries, present some examples of how to use it in practice, and list advantages and disadvantages.
2. In Chapter 3 we consider the physical aspects of the drying process. We make some assumptions on the fundamental mechanisms involved, as well as on the distribution of ice inside the product during drying. Then we propose a method of investigating this distribution, and present some electron microscope results from a test of this method.
3. In Chapter 4 we make some more assumptions, construct an equation for the mass transfer during drying, and verify that this equation is mathematically well posed.
4. In Chapter 5 we derive a numerical scheme to solve the equation, and prove some important properties of the matrices and vectors in this scheme.
5. In Chapter 6 we first show how to improve the efficiency of the numerical implementation, then present simulation results for pieces of cod fillet.
6. In Chapter 7 we discuss and conclude on the results of the simulations, the framework's limitations and weaknesses, and possible improvements.
7. In Appendix A we give the source code for the simulations.

Since we will rely heavily on the mathematical and numerical theory in [2]-[4], we will closely follow the notation therein (it is the same in all three papers).

We will write vectors and matrices in bold letters, except when referring to single elements. We will also often skip the arguments of functions, when these are obvious or unimportant.

We implement the framework in Matlab 2010a, so there will be some Matlab notation as well. In particular, given a matrix $\boldsymbol{E}$, we will write $E(p, q)$ for the element in row p and column q. We will also write

$$\mathbf{ones}(\boldsymbol{m})$$

for the array of size $\boldsymbol{m} = m_1 \times \cdots \times m_k$ consisting of ones. The MATLAB source code in Appendix A contains a mix of commands which correspond directly to the theory, and commands that have to do with the allocation of memory or information. Many of the latter type of commands have obvious counterparts in other programming languages, but there are also some MATLAB specific commands. The emphasis of many of the MATLAB specific commands has been on saving memory and/or time. Therefore, the source code is not as easy to read as it could have been with a less efficient code.

To make the notation more compact and easy to read, we will often write certain triples as vectors. We write

$\boldsymbol{x}$ for the point (x, y, z) in space,
$\boldsymbol{a}$ for the length triple (a, b, c),
$\boldsymbol{p}$ for the index triple (p, q, r),
$\boldsymbol{s}$ for the index triple (s, t, u),
α for the index triple (α, β, ζ),
κ for the index triple (κ, λ, ξ).

Finally, to avoid too much mathematical technicality, we will often use the somewhat imprecise term *smooth* about mathematical functions. We will use it in the sense *smooth enough*. In practice, this is rarely an issue, as one will typically use infinitely differentiable functions.

2. IMPLICIT REPRESENTATIONS OF GEOMETRY

In this chapter we first see how to create implicit representations of geometry through geometry defining functions (GDFs). Then we give three examples of such representations, and finally, we look at the advantages and disadvantages to using GDFs.

2.1 Approximation by hyperbolic tangent functions

A GDF is constructed to be zero on a surface or an interface. It has nonzero values at other points, but these values do not necessarily indicate how close we are to the surface/interface. They do however tell us on which side of it we are. In this section we show how to include this information in equations.

Denote the entire domain by Ω. Suppose that we divide Ω into disjoint, open subsets Ω_k (i.e. $\overline{\Omega} = \cup_k \overline{\Omega_k}$). If we want some function f to be smooth and behave like the smooth function f_k in Ω_k for each k, we could write f as

$$f = \sum_k \chi_{\Omega_k} f_k,$$

where the characteristic function χ_{Ω_k} is defined by

$$\chi_{\Omega_k}(x,y,z) = \begin{cases} 1, & (x,y,z) \in \Omega_k \\ 0, & (x,y,z) \notin \Omega_k \end{cases}.$$

Unfortunately, this makes f zero on the boundaries of the subsets, since each χ_{Ω_k} is zero on $\partial\Omega_k$. Then f is continuous on $\partial\Omega_k$ if and only if f_k is zero there. This puts a restriction on the choice of each f_k. To avoid this restriction, we can use hyperbolic

tangent approximations to χ_{Ω_k}, as shown below.

Since a GDF is an indicator, we denote the GDF of Ω_k by I_{Ω_k}, so that $I_{\Omega_k} > 0$ defines Ω_k. We also make the following assumption:

Assumption 1

We assume that every I_{Ω_k} is smooth in its arguments in all of Ω.

The smoothness of I_{Ω_k} implies the smoothness of

$$I_{\Omega_k,m}(x,y,z,t) = \tanh(m \cdot I_{\Omega_k}(x,y,z,t)), \text{ where } m > 0.$$

We next set

$$\chi_{\Omega_k,m}(x,y,z,t) = \frac{1}{2}\Big(1 + I_{\Omega_k,m}(x,y,z,t)\Big),$$
$$\overline{\chi}_{\Omega_k,m}(x,y,z,t) = \frac{1}{2}\Big(1 - I_{\Omega_k,m}(x,y,z,t)\Big) = 1 - \chi_{\Omega_k,m}(x,y,z,t).$$

Note that since $I_{\Omega_k,m}$ has values in [-1,1], $\chi_{\Omega_k,m}$ and $\overline{\chi}_{\Omega_k,m}$ have values in [0,1].

We now have the following result:

Theorem 2.1.1 (*Approximation by hyperbolic tangents*)

As $m \to \infty$, $I_{\Omega_k,m}$, $\chi_{\Omega_k,m}$ and $\overline{\chi}_{\Omega_k,m}$ converge pointwise:

$$\lim_{m\to\infty} I_{\Omega_k,m} = sgn(I_{\Omega_k}) = \begin{cases} 1, & I_{\Omega_k} > 0 \\ 0, & I_{\Omega_k} = 0 \\ -1, & I_{\Omega_k} < 0 \end{cases},$$
$$\lim_{m\to\infty} \chi_{\Omega_k,m} = \chi_{\Omega_k,\infty},$$
$$\lim_{m\to\infty} \overline{\chi}_{\Omega_k,m} = \overline{\chi}_{\Omega_k,\infty} = 1 - \chi_{\Omega_k,\infty},$$

where $\chi_{\Omega_k,\infty}$ is χ_{Ω_k} with value 1/2 on $\partial\Omega_k$.

Proof:

1. If $I_{\Omega_k} = 0$, then $I_{\Omega_k,m} = 0 = \text{sgn}(I_{\Omega_k})$ for all m. If $I_{\Omega_k} \neq 0$, then

$$\begin{aligned} |I_{\Omega_k,m} - \text{sgn}(I_{\Omega_k})| &= |\tanh(mI_{\Omega_k}) - \text{sgn}(I_{\Omega_k})| \\ &= \frac{|\sinh(mI_{\Omega_k}) - \text{sgn}(I_{\Omega_k})\cosh(mI_{\Omega_k})|}{|\cosh(mI_{\Omega_k})|} = \frac{\exp(-m|I_{\Omega_k}|)}{\cosh(mI_{\Omega_k})} \to 0, \end{aligned}$$

   when $m \to \infty$. So $I_{\Omega_k,m}$ converges to $\text{sgn}(I_{\Omega_k})$, but since the rate of convergence depends on $I_{\Omega_k} = I_{\Omega_k}(x, y, z, t)$, the convergence is pointwise, not uniform.

2. $\chi_{\Omega_k,m}$ and $\overline{\chi}_{\Omega_k,m}$ are linear in $I_{\Omega_k,m}$, and therefore converge pointwise in the same manner:

$$\begin{aligned} \chi_{\Omega_k,m} &= \frac{1}{2}\Big(1 + I_{\Omega_k,m}\Big) \to \frac{1}{2}\Big(1 + \text{sgn}(I_{\Omega_k})\Big) = \chi_{\Omega_k,\infty}, \\ \overline{\chi}_{\Omega_k,m} &= \frac{1}{2}\Big(1 - I_{\Omega_k,m}\Big) \to \frac{1}{2}\Big(1 - \text{sgn}(I_{\Omega_k})\Big) = \overline{\chi}_{\Omega_k,\infty}. \end{aligned}$$

■

The convergence of $I_{\Omega_k,m}$ to $\text{sgn}(I_{\Omega_k})$ is illustrated in Figure 2.1.

If we write f as

$$f = \sum_k \chi_{\Omega_k,m} f_k + \sum_j \overline{\chi}_{\Omega_j,m} f_j, \tag{2.1}$$

we see that for finite m, the value of f extends smoothly from, say, Ω_p to Ω_q, with the average value of f_p and f_q at their common boundary $\partial\Omega_p \cap \partial\Omega_q$. We therefore use coefficients of the form given in Eq. (2.1) in the equation we derive in Section 4.2.4.

2.2 Examples

In this section we derive implicit representations of the surfaces of three different products. In some cases, the derived expression for a GDF may not be smooth in the

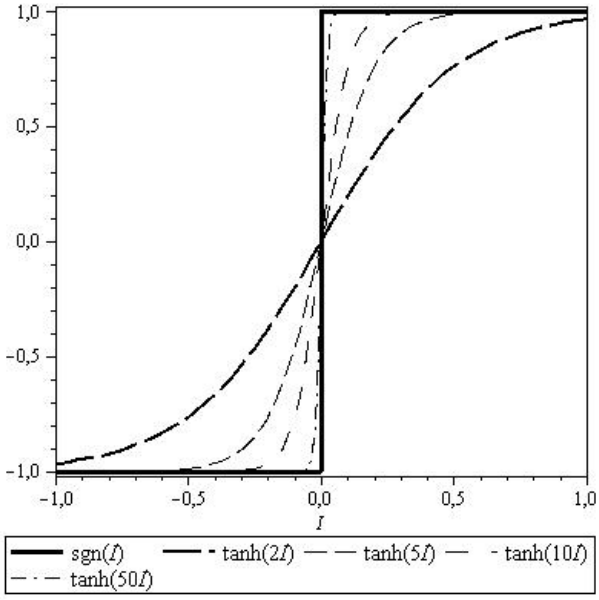


Fig. 2.1: $\mathrm{sgn}(I_{\Omega_k})$ and some approximations $I_{\Omega_k,m}$, for different values of m.

whole of Ω (e.g. there might be a singularity at the origin). If so, we can remove any singularities by adding terms that are small enough to have an insignificant impact on value of the GDF (e.g. terms of size 10^{-10}). This is done in the last two examples in this section.

The plots in this section were made with the Maple 11 functions *implicitplot* and *implicitplot3d*.

2.2.1 Cod slab

A very simple example is a cod slab 20 mm long, 20 mm wide and 5 mm high. It can be represented by a *superellipsoid*, whose surface is given implicitly by

$$\left(\frac{x}{a}\right)^{2p} + \left(\frac{y}{b}\right)^{2p} + \left(\frac{z}{c}\right)^{2p} = 1, \text{ where } p \text{ is a positive integer.} \tag{2.2}$$

For $p = 1$, this is a normal ellipsoid with semi-axes a, b and c. As p increases, the ellipsoid inflates and approximates its bounding box $[-a, a] \times [-b, b] \times [-c, c]$. Consequently, the cod slab can be well represented by setting

$$p = 5,\ a = b = 10^{-2} \text{ and } c = 2.5 \cdot 10^{-3} = \frac{10^{-2}}{4}$$

in Eq. (2.2). After rearranging, we get the following equation for the surface of the slab:

$$I_{\Omega_1}(x, y, z) = 10^{-20} - x^{10} - y^{10} - (4 \cdot z)^{10} = 0. \tag{2.3}$$

The slab is shown in Figure 2.2.

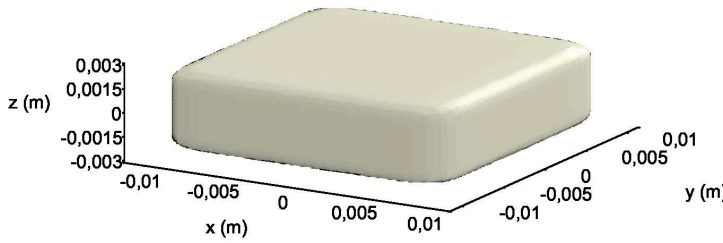


Fig. 2.2: The cod slab given implicitly by Eq. (2.3).

The simulations of a cod slab in Section 6.2 use a time-dependent version of Eq. (2.3) (with $p = 10$), to account for shrinkage.

2.2.2 Strawberry

A product with a shape having cylindrical symmetry can be represented by a *surface of revolution*. To construct such a surface, we begin by constructing a two-dimensional representation of the boundary of its cross-section. A rotation of this two-dimensional expression then turns it into a three-dimensional representation of the whole product.

To illustrate this procedure, we consider an implicit representation of a strawberry. The boundary of a typical strawberry cross-section is given, in polar coordinates, by the equation

$$r = 0.02 - 0.008 \cdot \sin(\theta)^5 + 0.006 \cdot \sin(\theta), \tag{2.4}$$
$$\text{where } |r| = \sqrt{x^2 + z^2}, \text{ for } x = |r|\cos(\theta) \text{ and } z = |r|\sin(\theta).$$

Since r in Eq. (2.4) is positive, $r = |r| = \sqrt{x^2 + z^2}$, but Figure 2.3 shows that the straightforward substitution from θ to x and z,

$$\theta \longmapsto \arctan(\tan(\theta)) = \arctan\left(\frac{\sin(\theta)}{\cos(\theta)}\right) = \arctan\left(\frac{|r|\sin(\theta)}{|r|\cos(\theta)}\right) = \arctan\left(\frac{z}{x}\right)$$

does not work, since

$$f_1(\theta) = \arctan\left(\frac{\sin(\theta)}{\cos(\theta)}\right) \tag{2.5}$$

and its sine are discontinuous. This is because $\tan(\theta)$ jumps from $+\infty$ to $-\infty$ at $\theta = \pi/2$ and $\theta = 3\pi/2$. These values of θ correspond to $x = 0$, i.e. to a change of sign in x, and thus a change of sign in z/x.

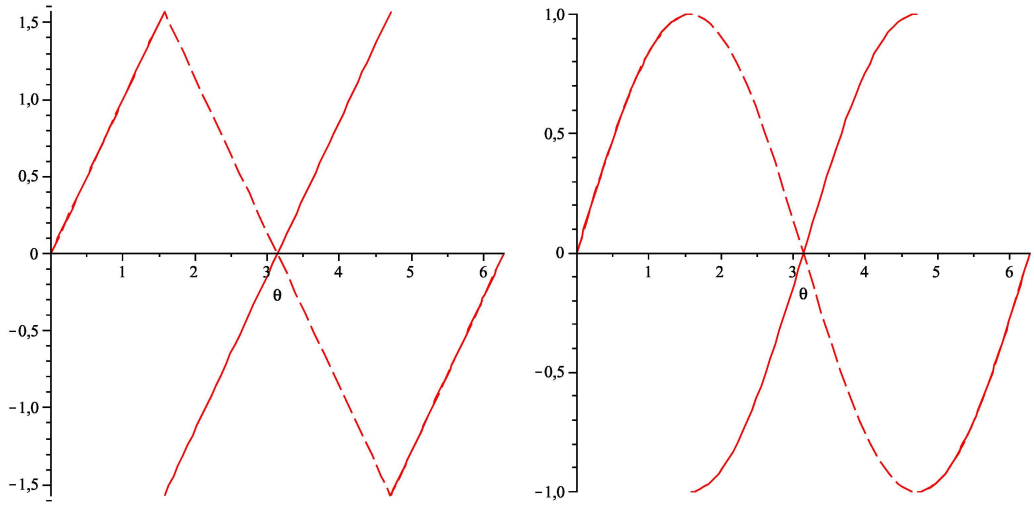
We therefore use the modified substitution

$$\theta \longmapsto f_2(\theta) = \arctan\left(\frac{\sin(\theta)}{|\cos(\theta)|}\right) = \arctan\left(\frac{|r|\sin(\theta)}{||r|\cos(\theta)|}\right) = \arctan\left(\frac{z}{|x|}\right), \tag{2.6}$$

which keeps both the arctangent and its sine continuous. Note from the dashed curve in Figure 2.3(b) that this modified substitution produces a sine curve

$$\sin(f_2(\theta)) = \sin\left(\arctan\left(\frac{\sin(\theta)}{|\cos(\theta)|}\right)\right),$$

which coincides with $\sin(\theta)$, i.e. the modified substitution works well.



(a) f_1 (solid) and f_2 (dashed) (b) $\sin(f_1)$ (solid) and $\sin(f_2)$ (dashed)

Fig. 2.3: The arctangents in Eqs. (2.5) and (2.6) (left), and their corresponding sines (right).

In Cartesian coordinates, Eq. (2.4) thus gives us the following equation for the boundary of the cross-section:

$$I_{\Omega_1}(x,z) = 0.02 - \sqrt{x^2+z^2} - 0.008 \cdot \sin\left(\arctan\left(\frac{z}{|x|}\right)\right)^5 + 0.006 \cdot \sin\left(\arctan\left(\frac{z}{|x|}\right)\right) = 0. \quad (2.7)$$

The cross-section is shown in Figure 2.4.

The whole strawberry can now be constructed by rotating the cross-section about the z-axis. This is done by simply replacing the one-dimensional length $|x|$ by the two-dimensional length $\sqrt{x^2+y^2}$ and, correspondingly, x^2 by x^2+y^2 in Eq. (2.7):

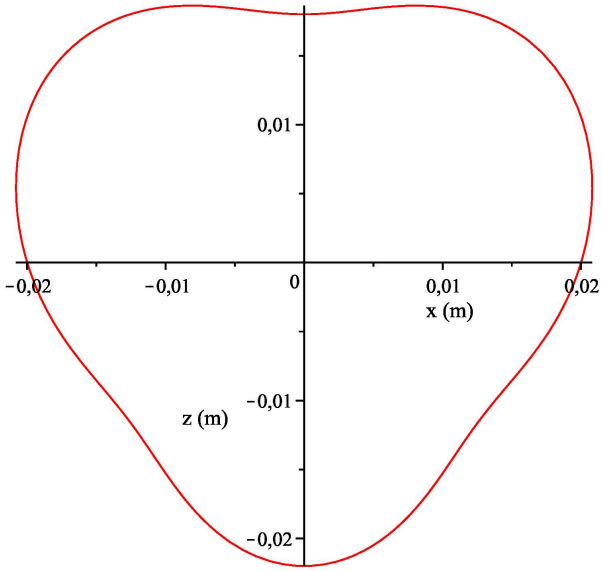


Fig. 2.4: The shape of a strawberry cross-section, given implicitly by Eq. (2.7).

$$I_{\Omega_1}(x,y,z) = 0.02 - \sqrt{x^2+y^2+z^2+10^{-10}} - 0.008 \cdot \sin\left(\arctan\left(\frac{z}{\sqrt{x^2+y^2+10^{-10}}}\right)\right)^5 + 0.006 \cdot \sin\left(\arctan\left(\frac{z}{\sqrt{x^2+y^2+10^{-10}}}\right)\right) = 0. \quad (2.8)$$

Note the addition of terms of size 10^{-10} to avoid a singularity at the origin, as explained above. The small term in the first square root is to prevent numerical problems at the origin for the derivatives of I_{Ω_1}. The strawberry is shown in Figure 2.5, where leaves given by

$$I_{\Omega_2}(x,y,z) = 0.015 - \sqrt{x^2+y^2+10^{-10}} - 4000 \cdot (z-0.02)^2 + 0.005 \cdot \sin\left(8 \cdot \left(\arctan\left(\frac{y}{\sqrt{x^2+10^{-10}}} + \frac{\pi}{16}\right)\right)\right)^3 = 0 \quad (2.9)$$

have been added to enhance the visual impression.

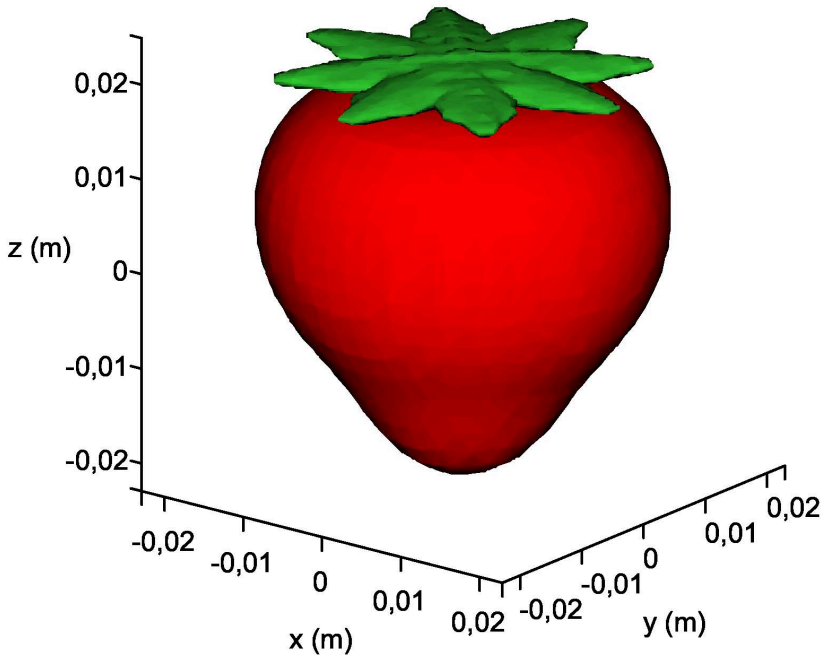


Fig. 2.5: The strawberry given implicitly by Eq. (2.8), with leaves given implicitly by Eq. (2.9).

2.2.3 Maize kernel

Our third and final example is a maize kernel, which we model before and after drying, using a procedure similar to that of the strawberry example above. In the case of the strawberry, we started with a polar coordinate representation of the boundary of a *vertical* cross-section, and turned it into an equation in x and z. Here, we consider a *horizontal* cross-section with boundary given by

$$r = 0.004 + 0.0023 \cdot \sin(\theta)^9 - 0.003 \cdot \sin(\theta)^7$$

in polar coordinates. The resulting Cartesian coordinate equation in x and y becomes

$$I_{\Omega_1}(x,y) = 0.004 - \sqrt{x^2+y^2} + 0.0023 \cdot \sin\left(\arctan\left(\frac{y}{|x|}\right)\right)^9 - 0.003 \cdot \sin\left(\arctan\left(\frac{y}{|x|}\right)\right)^7 = 0. \quad (2.10)$$

The cross-section is shown in Figure 2.6.

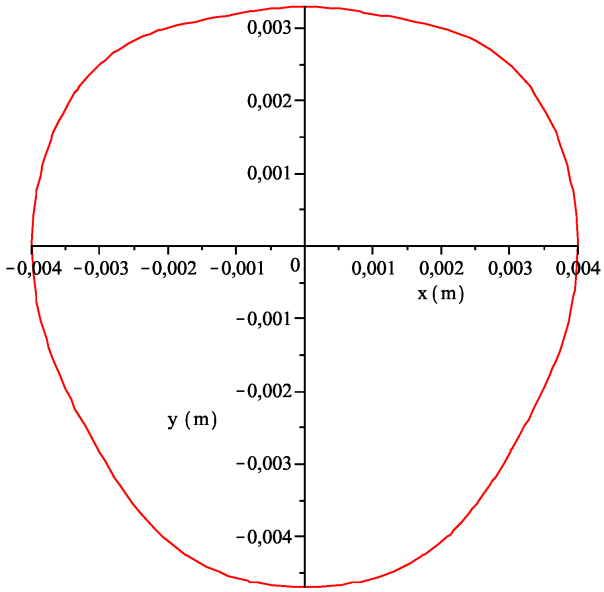


Fig. 2.6: The shape of a maize kernel cross-section, given implicitly by Eq. (2.10).

This time we rotate the cross-section around the y-axis. Unlike in the strawberry example, where the rotation was circular (x^2 was replaced by x^2+y^2), we use an elliptical rotation here, i.e. x^2 is not replaced by x^2+z^2, but the substitution

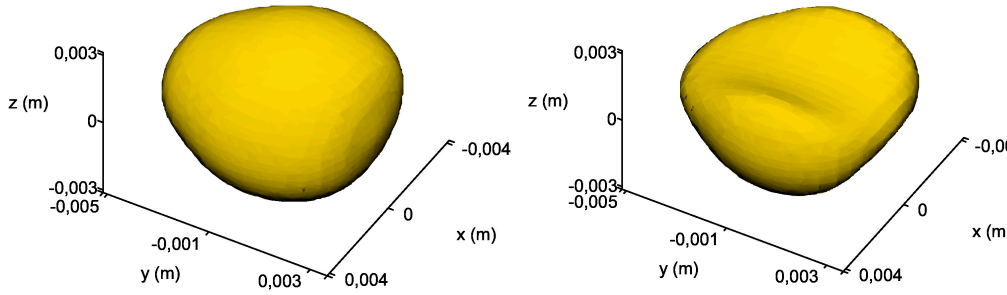
$$x^2 \longmapsto x^2 + \left(2 + f(t) \cdot \cos\left(\frac{\pi}{2} \cdot \frac{x}{0.004}\right)^4\right) \cdot z^2$$

is made instead. Here, f is an arbitrary (smooth) time-dependent function that enables us to model changes in shape as time goes by.

With the addition of terms of size 10^{-10} to prevent a singularity at the origin, the four-dimensional equation for the maize kernel becomes

$$I_{\Omega_1}(x,y,z,t) = 0.004 - \sqrt{x^2 + \left(2 + f(t)\cdot\cos\left(\frac{\pi}{2}\cdot\frac{x}{0.004}\right)^4\right)\cdot z^2 + y^2 + 10^{-10}}$$
$$+ 0.0023\cdot\sin\left(\arctan\left(\frac{y}{\sqrt{x^2 + \left(2 + f(t)\cdot\cos\left(\frac{\pi}{2}\cdot\frac{x}{0.004}\right)^4\right)\cdot z^2 + 10^{-10}}}\right)\right)^9$$
$$- 0.003\cdot\sin\left(\arctan\left(\frac{y}{\sqrt{x^2 + \left(2 + f(t)\cdot\cos\left(\frac{\pi}{2}\cdot\frac{x}{0.004}\right)^4\right)\cdot z^2 + 10^{-10}}}\right)\right)^7 = 0. \tag{2.11}$$

If we assume that f is zero initially, and 10 after drying, we get the deflation effect shown in Figure 2.7.



(a) Initially ($f(0) = 0$)

(b) After drying ($f(t) = 10$)

Fig. 2.7: The maize kernel given implicitly by Eq. (2.11), initially and after drying.

2.3 Advantages and disadvantages

We now look at the advantages and disadvantages to using an implicit representation of the geometry.

2.3.1 Advantages

The most obvious advantage to representing a product's geometry implicitly is that we can use a simple computational domain, regardless of the product's shape. We need not worry about enforcing boundary conditions on the surface of a product of a complicated and/or time-dependent shape. We can simply encase the product and a small part of its surroundings in a box-shaped volume, which then becomes our computational domain. The surface of the box forms a simple boundary on which we can enforce the boundary conditions, instead of having to enforce them on the surface of the product itself.

In addition to the geometry of the product surface, there might be other geometries involved in the problem. For instance, in many types of drying, the product rests on a base. This means that air travels past most of its surface, but its underside faces, completely or partially, a solid surface. In this case, the boundary conditions are obviously different at the top and bottom of the domain. Using GDFs, we can define external interfaces that divide the surroundings into several smaller parts, and thus handle all these parts simultaneously, cf. Eq. (2.1) and Section 6.2.

In some cases, the product might also have an internal structure where different parts have significantly different properties (e.g. granular materials). Then it might be practical to divide the product interior into smaller parts, each with its own transport coefficients. Such a division requires internal interfaces, which can be represented implicitly by GDFs.

Finally, if we use GDFs, we do *not* need to redesign or reimplement a numerical scheme when we change the product's dimensions or proportions, or even when we replace one product with a different one. We only need to modify the GDFs, and possibly remove some of them or add some more.

2.3.2 Disadvantages

Although it offers significant advantages, there are also a couple of disadvantages to representing the geometry implicitly.

One disadvantage is that transport coefficients become more complicated. Although this increases the complexity of both the mathematical and numerical theory, this does not cause a problem in practice, as long as the GDFs are smooth. The real disadvantage lies in an increase in the number of numerical calculations, hence an increase in the time it takes to run simulations.

A far more important disadvantage is the more difficult handling of convective boundary conditions on the product surface, since this surface now becomes part of the interior of the computational domain. Convective boundary conditions are usually handled as specified flux conditions (Neumann conditions) on the domain boundary. However, specifying the flux on an internal surface creates problems, both in terms of mathematical theory and in terms of numerical implementation (e.g. how to include the conditions when the grid points do not coincide with the surface). The solution is to impose the flux conditions throughout the surroundings, but this is not without its issues either (cf. Section 7.2.1).

3. THE PHYSICS OF THE DRYING PROCESS

In this chapter we look at the physics involved in AFD, and make some important assumptions. We also present and test a method of investigating the distribution of ice inside the product during drying.

3.1 Physical assumptions

In this section we make assumptions on the mechanisms of moisture removal during AFD. We start with two very common assumptions:

Assumption 2 (*No free liquid water*)

Any liquid water in the product is assumed to be bound and unavailable for removal by AFD. It is therefore ignored, and the removable water in the product is considered to exist only as ice (in the moist regions) and vapor (in the dry region).

Disregarding bound water is clearly a source of error, but it also greatly simplifies the modeling, because we do not have to worry about flow or evaporation of liquid water. We can concentrate on the sublimation mechanism:

Assumption 3 (*Retreating ice front*)

At any given time during drying, each remaining moist region within the product is separated from the dry region by a sharp interface, the ice front. *As drying continues, ice at the interface turns into vapor through sublimation. This removal of ice causes the interface to recede deeper into the product, making the moist region smaller and the dry region larger. The vapor is transported away from the interface through the dry region. There is no moisture transport within the moist regions.*

Asmp. 3 is a RIF assumption, so we get a moving boundary problem. The velocity of this motion depends on the vapor flux through the dry region, which is porous,

so we can assume that flow through it is governed by Darcy's law:

Assumption 4 (*Internal mass transfer*)

We assume that the transport of vapor through the dry region follows Darcy's law. We also assume that it is adiabatic, i.e. that the vapor is not heated by the air or dry matter on its way from the ice front to the surface of the product.

Remark: The assumption of adiabatic vapor transport may seem somewhat dubious, since there will be a temperature difference between the ice front and the ambient air (cf. Asmp. 7 below). This means that cold vapor from the ice front will pass through warmer regions, which might heat it up. However, we can argue that since the heat capacities of the air and vapor are several orders of magnitude smaller than the latent heat of sublimation, practically all the supplied heat is used for sublimation, instead of local heating of the air and vapor in the pores. ■

3.2 Investigation of the sharp interface assumption

In this section we present and test a method of investigating the validity of the sharp interface assumption (Asmp. 3). The idea is to use salt crystals as markers for the dry region of a sample, and study how sharply the concentration of these crystals decreases as we move from the dry region to a moist one.

When studying the sample, it is best to use a scanning electron microscope (SEM). Although the profile might be visible even to the naked eye, there are several advantages to using a SEM instead:

- The salt profile is protected by the vacuum in the vacuum chamber of the SEM (cf. the first remark following Algorithm 1 below).
- It is easy to study the profile at different magnifications, making it possible to look at both small details and the big picture.
- It is easy to create images of the profile. The images also contain a length scale, making comparisons easy.
- A typical SEM has several different types of detectors able to detect the salt, making the identification of the profile more reliable.

When studying a moist, organic sample, it is preferable to use an environmental SEM (ESEM, cf. [8]) or a low vacuum SEM (LV-SEM). An LV-SEM is a SEM with a relatively high chamber pressure (typically up to a few hundred Pa). A high vacuum (typically a fraction of a Pa) would take a very long time to establish, and it prevents the discharge of built-up negative charges in non-conducting (e.g. organic) specimens, due to the electron bombardment. These charges eventually cause the electron beam to be deflected and the signal to be lost.

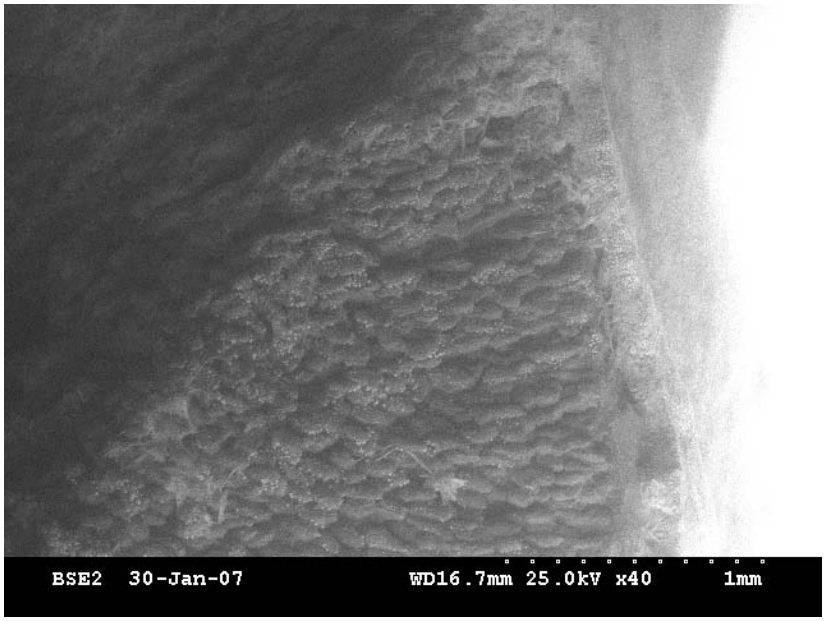
3.2.1 Electron microscope detection of NaCl

There are at least four types of relevant SEM detectors for NaCl crystals:

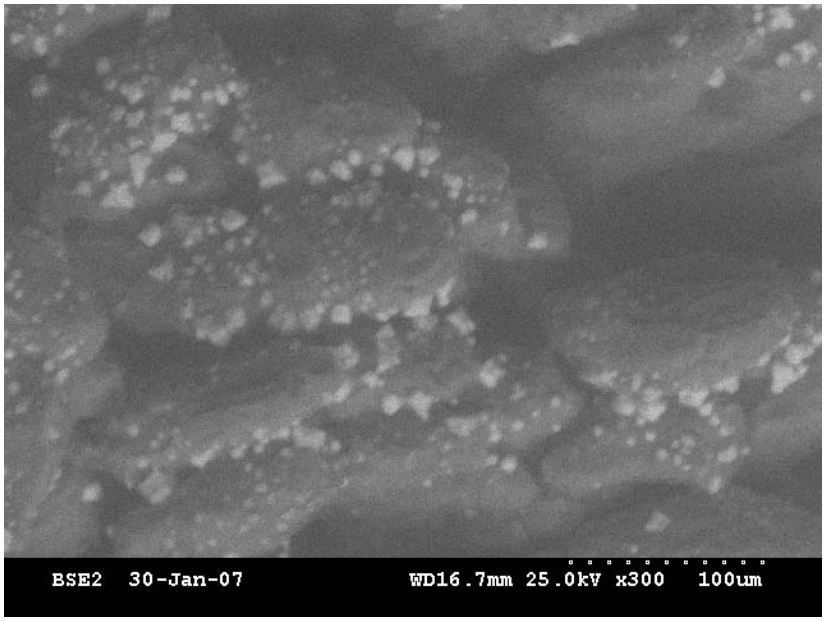
- **Secondary electron (SE) detector.** Detects low energy electrons emitted from the specimen when irradiated with electrons.
- **Backscattered electron (BSE) detector.** Detects high energy electrons emitted from the specimen when irradiated with electrons. Heavier elements produce stronger BSE signals than lighter.
- **Cathodoluminescence (CL) detector.** Detects light emitted from the specimen when irradiated with electrons. The intensity of the CL signal depends on the type of material.
- **X-ray spectrometer.** Detects X-rays emitted from the specimen when irradiated with electrons. Different elements emit X-rays at different energies, so an X-ray detector can map the chemical composition of the specimen.

More information about these detectors can be found in [16, Chapter 3].

What makes NaCl a good indicator is that in organic samples, most of the atoms are of light elements (like hydrogen, carbon, nitrogen and oxygen). In most types of products, sodium and chlorine are only present in small quantities, and are easily distinguishable with the detectors mentioned above (cf. Figure 3.1 and Figure 3.2). The X-ray spectrometer can obviously identify sodium and chlorine, and these atoms also produce a stronger BSE signal than the organic matter. In addition, the CL signal appears to be much stronger from the crystalline NaCl than from the non-crystalline organic matter, at least at some angles (in general, in the case of both X-rays and CL, the strength of the signal can depend significantly on the angle between the specimen surface and the detector).

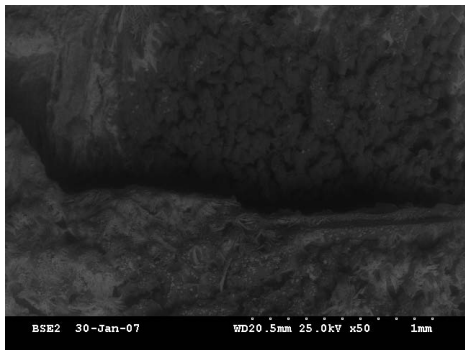


(a) Magnified 40 times

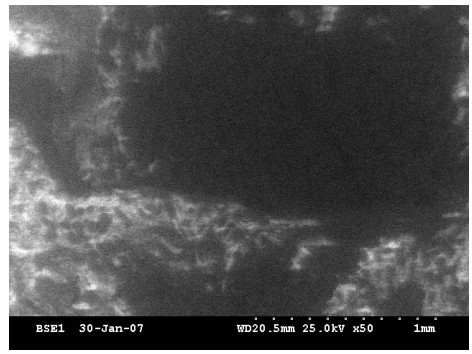


(b) Magnified 300 times

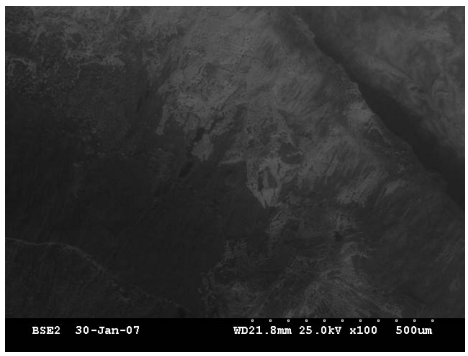
Fig. 3.1: BSE images of small NaCl crystals inside a frozen pea.



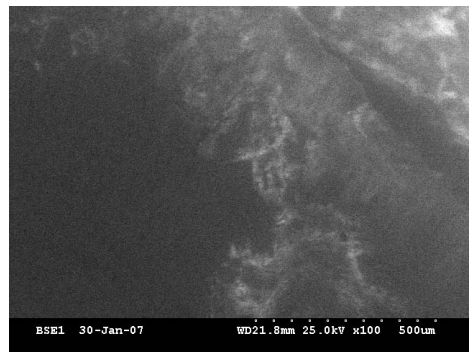
(a) BSE image, center area



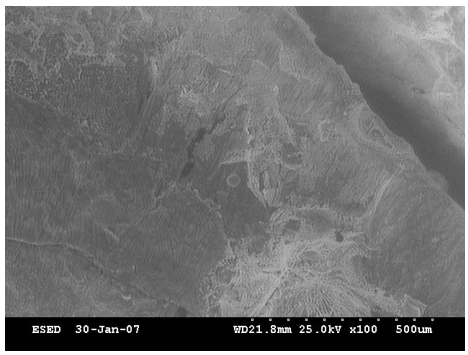
(b) CL image, center area



(c) BSE image, edge



(d) CL image, edge



(e) SE image, edge

Fig. 3.2: Visibility of NaCl in a frozen pea, for different detectors.

3.2.2 The method

A procedure for preparing and investigating samples is given in Algorithm 1:

Algorithm 1 (*Dry region identification, using NaCl*)

1. ***Store samples from the drying process at a temperature below the eutectic temperature of NaCl brine (-21.2 °C).***
2. ***Immerse each sample in saturated NaCl brine that is just above its eutectic temperature.*** *Assuming that the pores in the moist regions are blocked by ice, and those in the dry region are not, the brine should penetrate and rehydrate the dry region only. As the brine encounters ice, the ice will begin to melt, causing the local temperature to decrease to the brine's eutectic temperature. Since the brine is already close to its eutectic temperature, only a tiny amount of ice should melt before eutectic equilibrium between the brine and the ice is reached, preventing further melting (and thus preventing the brine from penetrating into the moist regions).*
3. ***Immerse the rehydrated sample in liquid nitrogen.*** *This rapid freezing immediately turns the brine into ice, precipitating out the dissolved salt as tiny crystals.*
4. ***Remove the sample from the liquid nitrogen and break it in half.*** *At liquid nitrogen temperature, the sample should be brittle, and breaking it in half exposes a cross-sectional surface where salt crystals indicate where the dried (and later rehydrated) region is. If possible, one should avoid cutting the sample with a knife, as the blade might smear the salt profile.*
5. ***Transfer the sample to an ESEM or a LV-SEM and study the cross-sectional salt profile.*** *A sharp jump in the concentration of salt crystals indicates a sharp interface, while a gradual change in concentration indicates an unsharp interface.*

Remark: In the last two steps in Algorithm 1, it is very important to prevent the sample from thawing. This is because the presence of liquid water would dissolve the salt crystals, destroying or blurring any salt profile. If possible, these steps should take place in refrigerated rooms using a SEM with a cryo-stage (this was unfortunately not possible in our case). If the ambient temperature cannot be kept low, the transfer of the sample from the liquid nitrogen to the microscope must be very quick. Once the sample is in the vacuum chamber of the SEM, the vacuum protects the salt profile by quickly drying out the surface of the cross-section. ■

Remark: Algorithm 1 is based on NaCl. This works fine as long as the product is not already saturated with NaCl. If it is, another salt (or other practical substance) must be used as the indicator, and the temperature in the algorithm must be changed accordingly. ■

3.2.3 A test of the method on cod slabs

The method in Algorithm 1 was tested on samples taken from drying experiments conducted by Dr. Ingrid C. Claussen (cf. [5]). The drying was carried out at -5 °C, and partially dried samples were taken out at random positions at different times and stored at -24 °C. For each drying time, a few samples were immersed in NaCl brine, each for a unique length of time, before being frozen in liquid nitrogen.

Unfortunately, the somewhat intricate process of breaking the frozen samples in half and fastening one of the halves to a sample holder, took place at room temperature. This resulted in about half of the samples being lost (i.e. rendered useless due to thawing of the cross-sectional surface). After preparation, each usable sample was transferred to a Hitachi S-3500N LV-SEM at NTNU's Department of Materials Science and Engineering, and studied at 270 Pa low vacuum. Figure 3.3 shows two sample arrangements in the microscope's vacuum chamber.

Some of the images from the investigations are shown in Figures 3.4-3.8. We see that the increased salt concentration in the rehydrated region makes it clearly visible, and that there generally is a sharp jump in salt concentration at the interface between the part of the sample that is rehydrated and the part that is not. In fact, to the naked eye, each sample looked like a little piece of Brie cheese, with a beige middle region surrounded by a white outer one. Comparing the images, we also see from the length indicators that the thickness of the rehydrated region increases with drying time (as expected).

We expect an increase in either immersion time or drying time (or both) to cause the brine to penetrate the sample to a greater depth. Thus, if two or more samples with the same drying time, but different immersion times, have rehydrated regions of the same thickness, this *could* imply that this common thickness is the typical dry region thickness after this drying time.

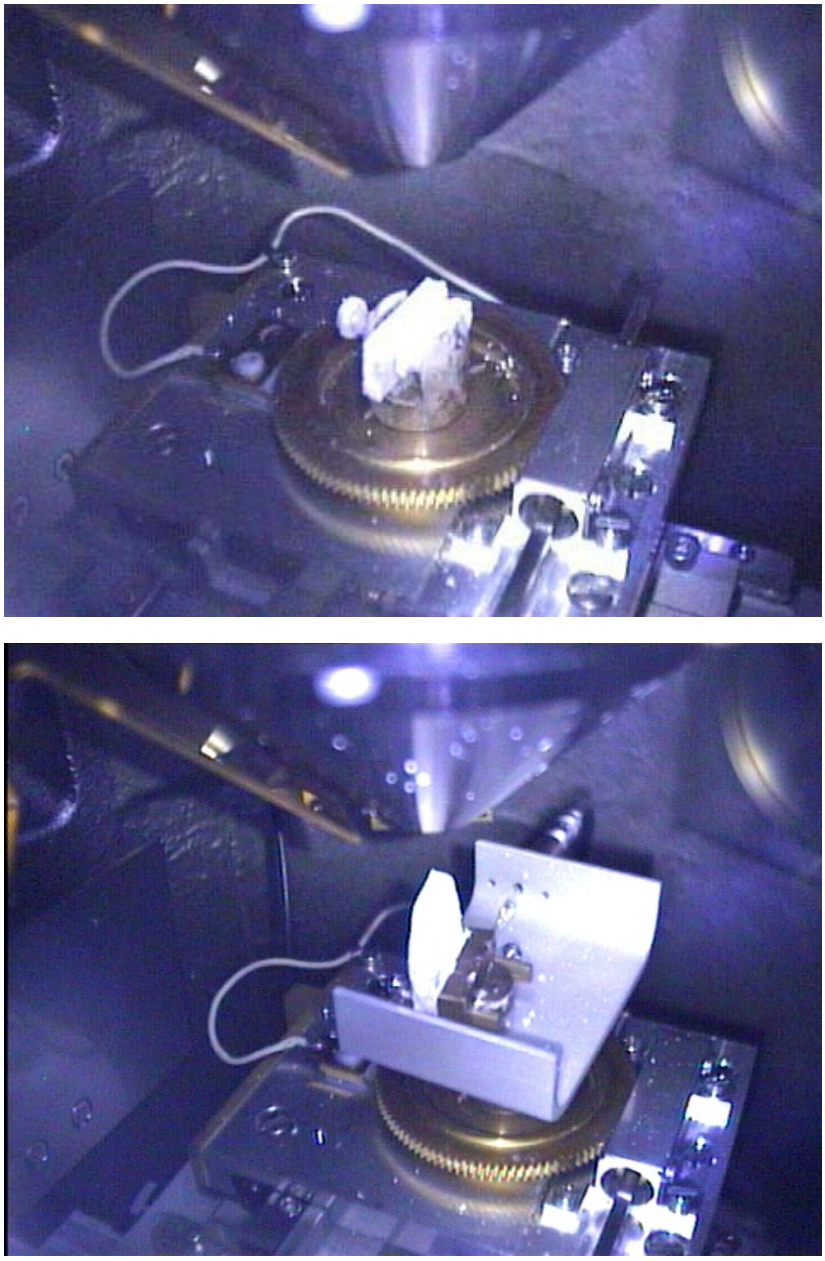
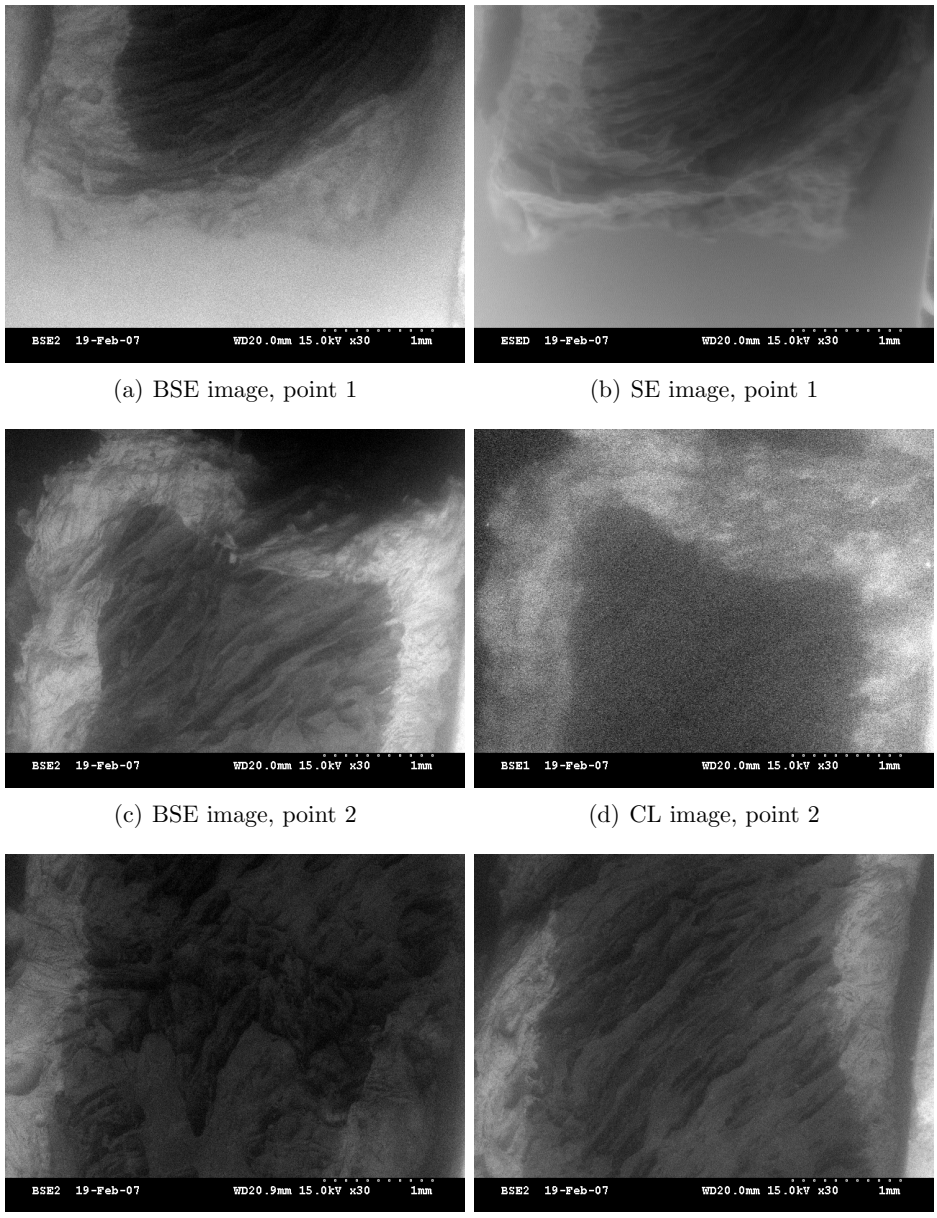


Fig. 3.3: Frozen samples of cod in the vacuum chamber of the LV-SEM.



(a) BSE image, point 1

(b) SE image, point 1

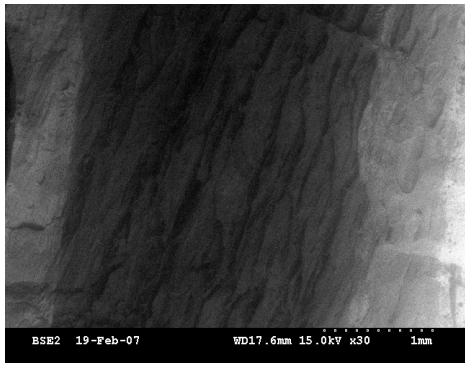
(c) BSE image, point 2

(d) CL image, point 2

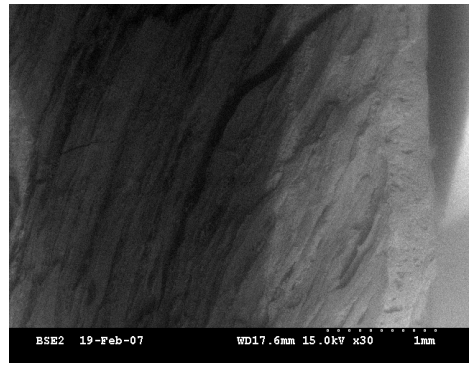
(e) BSE image, point 3

(f) BSE image, point 4

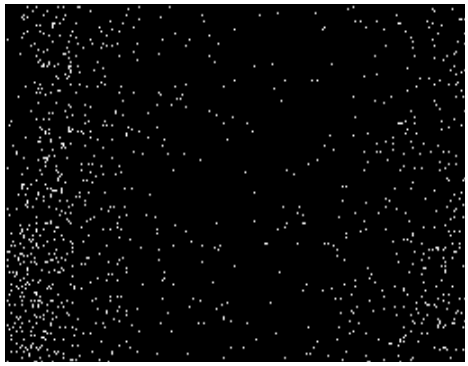
Fig. 3.4: BSE, SE and CL images of a frozen cod sample dried for 5 h and immersed in saturated NaCl brine for 30 s.



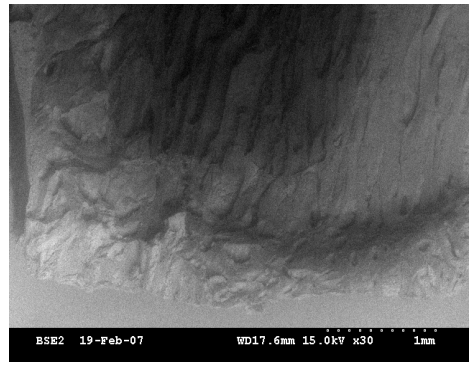
(a) BSE image, point 1



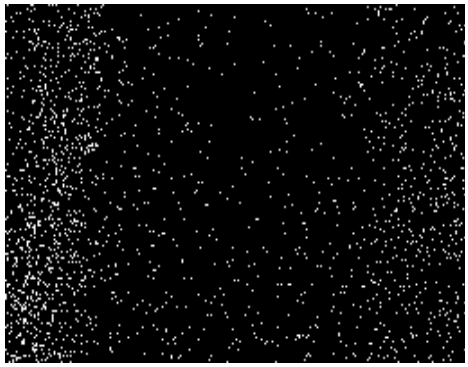
(b) BSE image, point 2



(c) Na counts, point 1

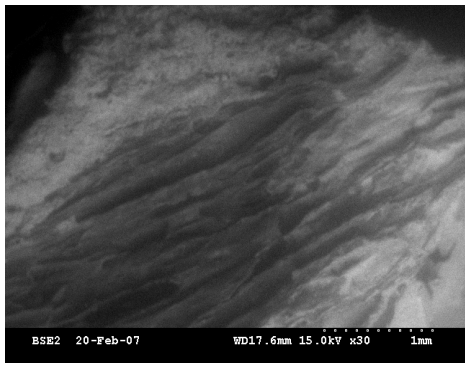


(d) BSE image, point 3

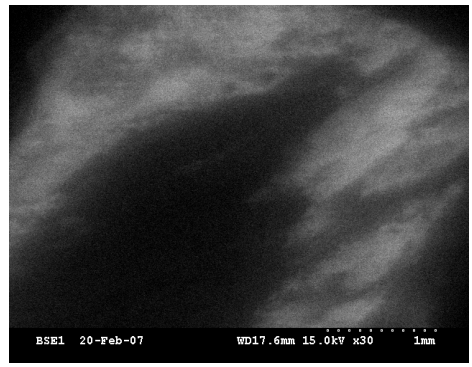


(e) Cl counts, point 1

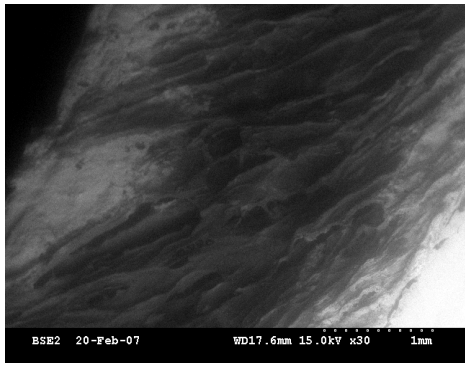
Fig. 3.5: BSE images of a frozen cod sample dried for 5 h and immersed in saturated NaCl brine for 300 s, with corresponding sodium and chlorine count distributions.



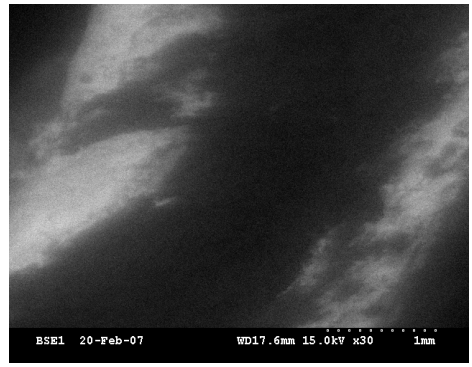
(a) BSE image, point 1



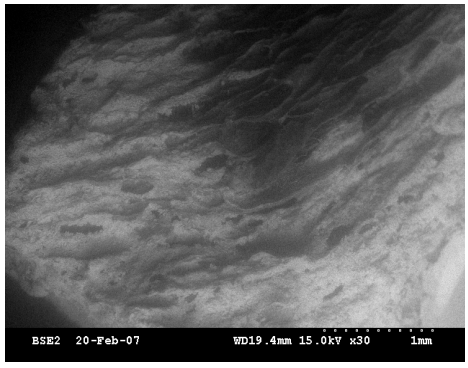
(b) CL image, point 1



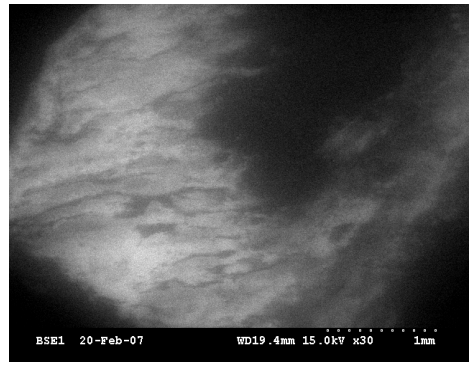
(c) BSE image, point 2



(d) CL image, point 2

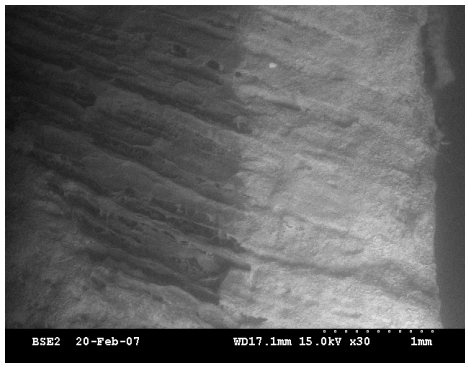


(e) BSE image, point 3

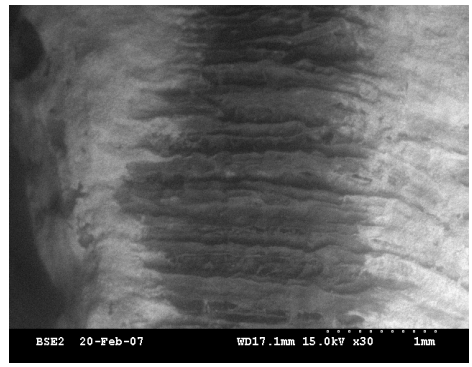


(f) CL image, point 3

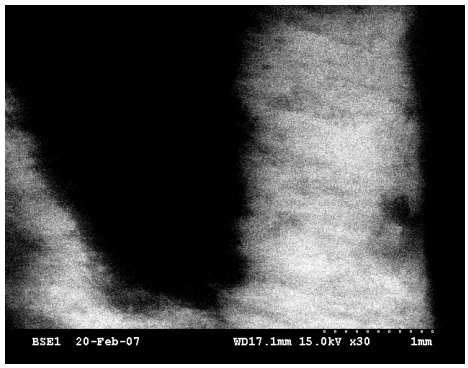
Fig. 3.6: BSE and CL images of a frozen cod sample dried for 11 h 30 min and immersed in saturated NaCl brine for 30 s.



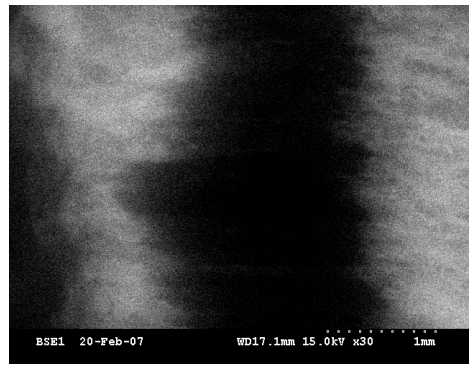
(a) BSE image, point 1



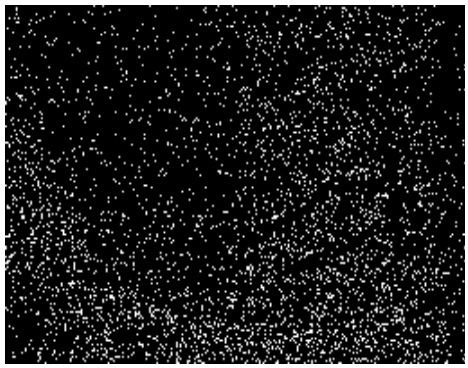
(b) BSE image, point 2



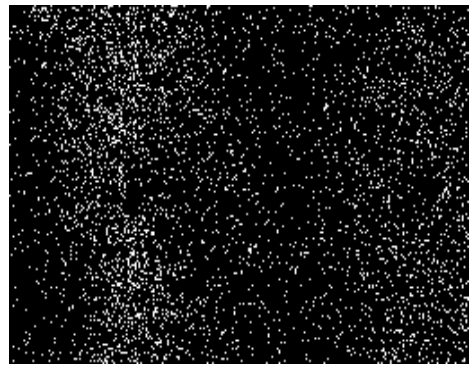
(c) CL image, point 1



(d) CL image, point 2

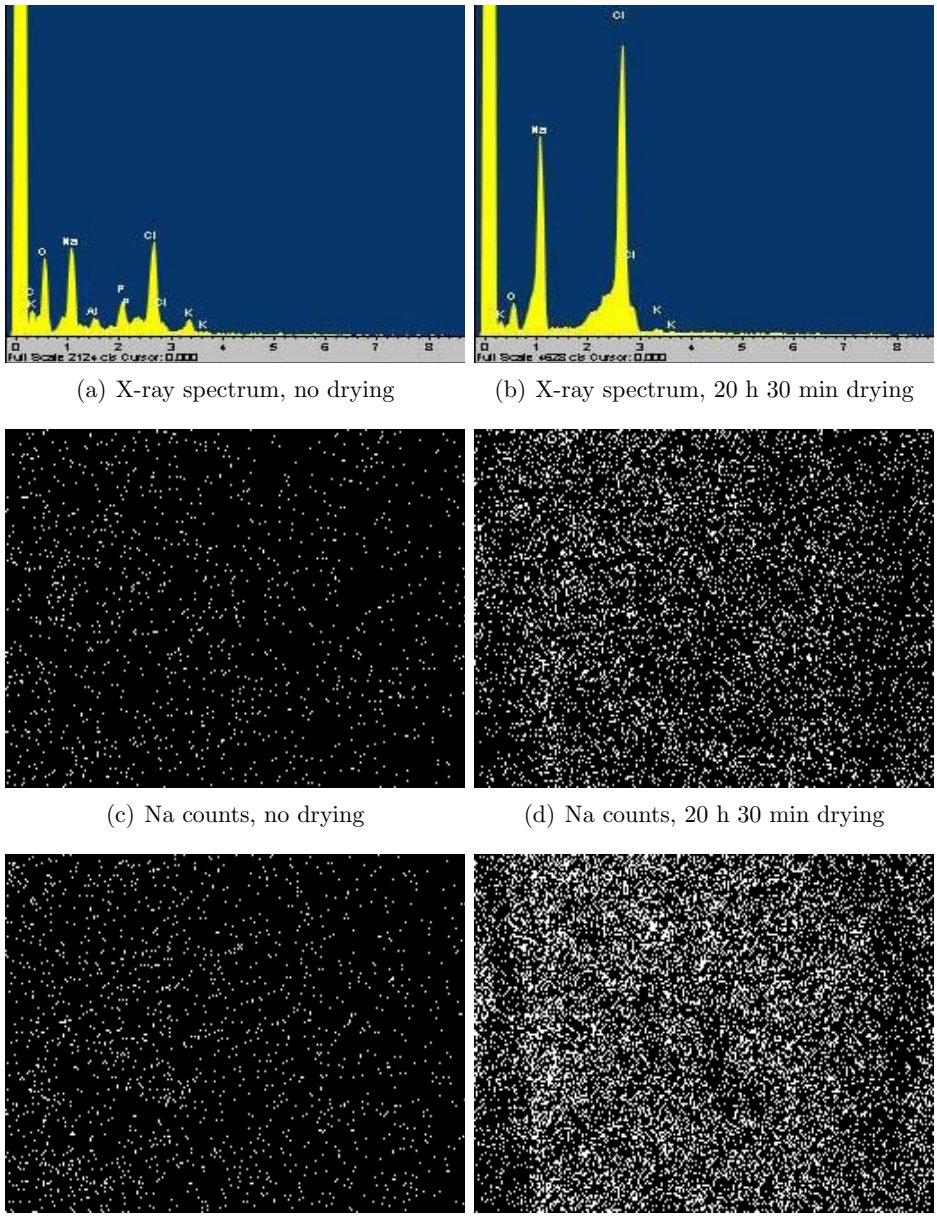


(e) Cl counts, point 1



(f) Cl counts, point 2

Fig. 3.7: BSE and CL images of a frozen cod sample dried for 14 h 15 min and immersed in saturated NaCl brine for 240 s, with corresponding chlorine count distributions (the sodium distributions were similar).



(a) X-ray spectrum, no drying

(b) X-ray spectrum, 20 h 30 min drying

(c) Na counts, no drying

(d) Na counts, 20 h 30 min drying

(e) Cl counts, no drying

(f) Cl counts, 20 h 30 min drying

Fig. 3.8: X-ray spectra and distribution of sodium and chlorine counts in frozen cod, after immersion in saturated NaCl brine (for 300 s for the undried sample and 360 s for the dried sample).

The sample in Figure 3.4 seems to have a thinner dry region than the sample in Figure 3.5, with equal drying times. Some conceivable reasons for this might be that:

- The two samples have dried differently.
- The 30 s sample was not immersed in brine for a long enough time for the dry region to be fully rehydrated. This is unlikely, since the sample in Figure 3.6 shows that a thicker rehydration region is possible for a 30 s immersion.
- The brine penetrated into the icy core of the 300 s sample. This should not be the case if the theory in Algorithm 1 is sound.

Since the samples were collected at random, it seems plausible that there might be a significant difference in the dryness of samples collected at the same time.

The undried and fully dried samples were uniform, without any distinct regions. This is because the fully dried sample had no moist region left, so it was fully rehydrated by the brine (creating a high salt concentration everywhere), while the undried sample (which had no dry region) was not rehydrated by the brine at all (keeping its original, low salt concentration). We can see this in the X-ray spectra of the undried and fully dried samples, shown in Figure 3.8. By comparing the *relative heights* of the sodium and chlorine peaks with the oxygen peak in each spectrum, we see that the salt concentration is clearly higher in the fully dried sample. The *absolute heights* of the peaks can be ignored, because they represent the number of counts of each element, so they depend on e.g. the scale of the vertical axis, the length of the detection period and the distance and angle between the specimen and the detector.

Remark: In X-ray analysis, elements having atomic number smaller than a certain lower bound (depending on the type of X-ray detector, cf. [16, Table 3-9]) cannot be detected. This is the reason why some of the main elements in organic matter do not show up in the spectra in Figure 3.8. Hydrogen does not have an X-ray spectrum, and carbon and nitrogen can be hard to detect (in our case, the carbon peak is hidden inside the noise peak on the left edge in both spectra). ■

The overall impression from these images (and others not shown here) is that there is in general an easily observable and quite sharp jump in NaCl concentration. This jump is consistent with the common sharp interface hypothesis.

Remark: This experiment was only meant to test the method itself, so only a few samples were studied. We therefore emphasize that although the results in this section are consistent with the sharp interface hypothesis, the number of samples was simply too low to prove anything, especially regarding the quantitative evolution of the moisture profile. Moreover, the information from this test is of course only valid for cod slabs. ■

4. THE MATHEMATICS OF THE FRAMEWORK

Any mathematical model must contain a system of one or more equations to describe the physics of the problem at hand. Not only must this system give a satisfactory description of the physics, it must also be *mathematically well posed*, i.e. there must exist a unique solution that must depend continuously on the system parameters. If a system cannot be solved, it has little value as a model, and if a solution is not unique and/or does not depend continuously on the data, the results of the modeling will be unreliable.

In this chapter we derive our framework, and show that it is well posed.

4.1 The general equation system

We base our mathematical theory on a slight modification of the equation system in [4]. The theory therein uses the weak (i.e. integrated) formulation of the system to prove its properties.

4.1.1 The general system

Let $\Omega \in \mathbb{R}^n$ be the spatial domain and τ the total drying time. We write

$$\mathrm{Q}_\tau = \Omega \times (0, \tau)$$

for the set of relevant points in space and time. The general equation system in [4] consists of k equations for the solution vector $\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{x}, t)$. Equation i is given by

$$\frac{\partial U^i(\boldsymbol{x},t)}{\partial t} = \sum_{l=1}^{n} \frac{\partial}{\partial x_l} \left\{ \sum_{j=1}^{k} \sum_{m=1}^{n} C_{ij}^{lm}(\boldsymbol{x},t,\boldsymbol{U}(\boldsymbol{x},t)) \, \frac{\partial U^j(\boldsymbol{x},t)}{\partial x_m} + B_i^l(\boldsymbol{x},t,\boldsymbol{U}(\boldsymbol{x},t)) \right\} + F_i(\boldsymbol{x},t,\boldsymbol{U}(\boldsymbol{x},t)), \text{ for } (\boldsymbol{x},t) \in \mathrm{Q}_\tau, \quad (4.1)$$

and the corresponding initial condition by

$$U^i(\boldsymbol{x},0) = U_0^i(\boldsymbol{x}), \text{ for } \boldsymbol{x} \in \Omega. \quad (4.2)$$

In our drying problem, we know the ambient conditions, so we want to specify values for the solution on the boundary $\partial\Omega$ (i.e. use *Dirichlet boundary conditions*). The boundary conditions in [4] are of *Neumann type* (i.e. specified boundary flux), but we can adapt the theory to the *homogeneous Dirichlet conditions*

$$\boldsymbol{U}(\boldsymbol{x},t) = \boldsymbol{0}, \text{ for } (\boldsymbol{x},t) \in \partial\Omega \times (0,\tau). \quad (4.3)$$

This adaptation is quite trivial, because, in the weak formulation of the equations,

- Neumann conditions are enforced by boundary integrals,
- homogeneous Dirichlet conditions are enforced by the basis functions.

Hence, to use homogeneous Dirichlet conditions, we only need to choose an appropriate basis (this makes the boundary integrals vanish by themselves). We can therefore formulate our problem with specified boundary values, and translate it from a general Dirichlet problem to a homogeneous one, satisfying Eqs. (4.1)-(4.3). This is done below, for the case of a single equation.

4.2 The particular system

Although the drying process in reality involves coupled heat and mass transfer, we restrict the framework to modeling just mass transfer. This reduces our system to just a single equation (i.e. $k = 1$, and the indices i and j become redundant). The reasons why we do this are discussed in Section 7.2.1.

For simplicity, we also restrict the form of the diffusivity matrix, by assuming that

$$C^{lm} = 0, \text{ for } l \neq m,$$

which makes m redundant as well. Note that this still allows for anisotropic diffusivities, through the remaining index l.

4.2.1 Vapor diffusivity

Before we derive the transport equation for water vapor in the dry region, we make a very common assumption:

Assumption 5 (*Vapor as an ideal gas*)

We assume that water vapor can be considered an ideal gas.

We have already assumed that the vapor travels through the dry region in accordance with Darcy's law (Asmp. 4). To allow for anisotropy, we write the permeability of the product as a matrix,

$$\boldsymbol{K} = \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_3 \end{bmatrix},$$

where we allow $K_l = K_l(\boldsymbol{x}, t, \boldsymbol{U}(\boldsymbol{x}, t)) \geq 0$, to include possible inhomogeneity. Darcy's law then becomes

$$v^l = -\frac{1}{\mu_{\text{vap}}}(\boldsymbol{K} \cdot \nabla P_{\text{vap}})_l = -\frac{1}{\mu_{\text{vap}}} K_l \, \frac{\partial P_{\text{vap}}}{\partial x_l},$$

where μ_{vap} is the dynamic viscosity of the vapor, and P_{vap} its local vapor pressure. Since we also assume that the vapor transport is adiabatic (Asmp. 4), we can use the adiabatic relation

$$P_{\text{vap}} V^\gamma = P_0 V_0^\gamma = \text{ constant}$$

to relate the local vapor pressure to the local vapor density:

$$P_{\text{vap}} = P_0 \left(\frac{V_0}{V}\right)^\gamma = P_0 \left(\frac{\rho_{\text{vap}}}{\rho_{\text{vap},0}}\right)^\gamma = \frac{P_0}{\rho_{\text{vap},0}^\gamma} \, \rho_{\text{vap}}^\gamma.$$

Here, γ is the adiabatic index of the vapor, given by

$$\gamma = \frac{c_{\mathrm{p,vap}}}{c_{\mathrm{V,vap}}} = \frac{c_{\mathrm{p,vap}}}{c_{\mathrm{p,vap}} - R} = \frac{c_{\mathrm{p,vap}}/R}{c_{\mathrm{p,vap}}/R - 1} \in (1, 2), \tag{4.4}$$

where $c_{\mathrm{p,vap}}$ and $c_{\mathrm{V,vap}}$ are the vapor's molar heat capacities at constant pressure and volume, respectively, and $R = c_{\mathrm{p,vap}} - c_{\mathrm{V,vap}}$ the universal gas constant.

The diffusive mass flux through the dry region is thus given by

$$\begin{aligned} f^l_{\mathrm{diff}} = \rho_{\mathrm{vap}} v^l = -\frac{\rho_{\mathrm{vap}}}{\mu_{\mathrm{vap}}} K_l \, \frac{\partial P_{\mathrm{vap}}}{\partial x_l} &= -\frac{P_0}{\mu_{\mathrm{vap}} \rho^\gamma_{\mathrm{vap},0}} \, \rho_{\mathrm{vap}} K_l \, \frac{\partial \rho^\gamma_{\mathrm{vap}}}{\partial x_l} \\ &= -\frac{\gamma P_0}{\mu_{\mathrm{vap}} \rho^\gamma_{\mathrm{vap},0}} \, \rho^\gamma_{\mathrm{vap}} K_l \, \frac{\partial \rho_{\mathrm{vap}}}{\partial x_l}. \end{aligned} \tag{4.5}$$

Next, we make a distinction between the density and the concentration of vapor:

$$\begin{aligned} \text{Density: } \rho_{\mathrm{vap}} &= \frac{\text{mass of vapor}}{\text{volume of vapor}}. \\ \text{Concentration: } \widehat{\rho}_{\mathrm{vap}} &= \frac{\text{mass of vapor}}{\text{porous volume}}. \\ \text{Mutual relation: } \widehat{\rho}_{\mathrm{vap}} &= \phi \rho_{\mathrm{vap}}. \\ \text{Porosity: } \phi &= \frac{\text{volume of pores}}{\text{porous volume}} \left(= \frac{\text{volume of vapor}}{\text{porous volume}} \right). \end{aligned} \tag{4.6}$$

The porosity ϕ of the dry region is important for the mass transfer properties of the product. We assume that it is constant:

Assumption 6

We assume that the porosity of the dry region is constant, and equal to the final porosity of the dry product.

Remark: In inhomogeneous products, the dry region might not have a uniform porosity, but we model this effect on the local mass transfer properties through the permeability $\boldsymbol{K}$ instead. ■

Asmp. 6, Eqs. (4.5)-(4.6) and the porous continuity equation

$$\frac{\partial(\phi\rho_{\text{vap}})}{\partial t} + \nabla\cdot(\rho_{\text{vap}}\boldsymbol{v}) = 0,$$

give us a diffusion equation for the transport of vapor through the dry region:

$$\begin{aligned}\frac{\partial\widehat{\rho}_{\text{vap}}}{\partial t} = -\nabla\cdot\boldsymbol{f}_{\text{diff}} &= \sum_{l=1}^{3}\frac{\partial}{\partial x_l}\left(\frac{\gamma P_0}{\mu_{\text{vap}}\rho_{\text{vap},0}^{\gamma}}\,\rho_{\text{vap}}^{\gamma}K_l\,\frac{\partial\rho_{\text{vap}}}{\partial x_l}\right)\\ &= \sum_{l=1}^{3}\frac{\partial}{\partial x_l}\left(\left(\frac{\gamma P_0}{\mu_{\text{vap}}\rho_{\text{vap},0}^{\gamma}}\,K_l\,\frac{\widehat{\rho}_{\text{vap}}^{\gamma}}{\phi^{\gamma+1}}\right)\frac{\partial\widehat{\rho}_{\text{vap}}}{\partial x_l}\right). \quad (4.7)\end{aligned}$$

4.2.2 Convection

The convective mass flux $\boldsymbol{f}_{\text{conv}}$ from the surface of the product is given by

$$\boldsymbol{f}_{\text{conv}} = h_{\text{m}}(\rho_{\text{vap}} - \rho_{\text{vap,air}})\boldsymbol{n}, \quad (4.8)$$

where h_{m} is the convective mass transfer coefficient, ρ_{vap} the local vapor density, $\rho_{\text{vap,air}}$ the ambient vapor density, and $\boldsymbol{n}$ the outer unit normal for the product surface. Usually, this expression is used as a boundary condition, but since our domain boundary does not coincide with the product surface, we must extend Eq. (4.8) from the surface into the surroundings.

In the surrounding air, $\phi = 1$, so $\rho_{\text{vap}} = \widehat{\rho}_{\text{vap}}$, and the continuity equation yields

$$\frac{\partial\widehat{\rho}_{\text{vap}}}{\partial t} = -\nabla\cdot\boldsymbol{f}_{\text{conv}} = \nabla\cdot(-h_{\text{m}}(\widehat{\rho}_{\text{vap}} - \rho_{\text{vap,air}})\boldsymbol{n}). \quad (4.9)$$

Remark: In the ambient air, the convective mass transfer coefficient h_{m} is given by fluid dynamics, through the Sherwood number (cf. Eq. (6.15)). ■

4.2.3 Sublimation

Modeling the phase change from ice to vapor presents a challenge, because the two phases have very different concentrations ($\widehat{\rho}_{\text{ice}}/\widehat{\rho}_{\text{vap}} \sim 10^5$). This makes it hard to model the mass balance at the ice front, because a microscopic decrease in the local ice concentration implies a significant increase in the local vapor concentration. If the two concentrations are modeled separately, this very easily causes the system to become unstable.

We avoid this problem by using a trick. **We model the mass transfer as if the ice itself is being transported, according to the vapor transport equation.** This of course makes no sense physically, but since we use the coefficients from the vapor transport equation, the mathematical result is just a massively scaled up version of the vapor transport. The major advantage is that we get away with using only one equation, which does not involve phase change, and is stable (cf. the discussion in Section 7.2.1).

At the ice front, we can assume that the ice concentration is equal to the initial ice concentration $\widehat{\rho}_{\text{ice},0}$. In fact, **we define the moist region to be where $\widehat{\rho}_{\mathbf{ice}} > K_{\mathbf{ice}}\widehat{\rho}_{\mathbf{ice},0}$, for some constant $K_{\mathbf{ice}}$ close to 1.** The ice front thus becomes the interface where this inequality turns into an equality.

Next, we assume that the vapor at the ice front is saturated:

Assumption 7 (*Conditions at the ice front*)

We assume that the temperature at the ice front is equal to the wet bulb temperature T_{wb} of the ambient air, and that the density of water vapor at the ice front is equal to the saturation density $\rho_{sat}(T_{wb})$ at that wet bulb temperature.

This assumption is sensible when the ice front is close to the product surface, and once the sublimation has started, its energy consumption should balance the energy supply to the ice front, keeping the temperature at the ice front reasonably stable.

Asmp. 7 raises the question of how the ambient wet bulb temperature T_{wb} evolves. It depends implicitly on the ambient conditions:

$$T_{\text{wb}} = T_{\text{wb}}(T_{\text{air}}, \rho_{\text{vap,air}}) = T_{\text{air}} + \frac{\nu_{\text{air}}\Delta H_{\text{sub}}}{k_{\text{air}}Pr}\,[\rho_{\text{vap,air}} - \rho_{\text{sat}}(T_{\text{wb}})]. \tag{4.10}$$

Here, T_{air} is the ambient temperature, $\rho_{\text{vap,air}}$ the ambient vapor density, ΔH_{sub} the latent heat of sublimation, Pr the Prandtl number, and ν_{air} and k_{air} the kinematic viscosity and thermal conductivity of air, respectively. We make an assumption on the first two:

Assumption 8 (*Constant ambient temperature and humidity*)

We assume that the ambient temperature and humidity are constant.

Remark: If the product rests on a solid base, the humidity inside this base is in reality of course not $\rho_{\text{vap,air}}$, but zero. But since $h_{\text{m}} = 0$ inside the base, the true value of ρ_{vap} there does not matter to the modeling anyway. ■

We next define the ratio of the concentrations of ice and vapor:

$$\omega = \frac{\widehat{\rho}_{\text{ice}}}{\widehat{\rho}_{\text{vap}}}, \text{ evaluated at the ice front.} \tag{4.11}$$

Note that Asmp. 7 and Eq. (4.6) imply that

$$\omega = \frac{\widehat{\rho}_{\text{ice},0}}{\phi \rho_{\text{sat}}(T_{\text{wb}}(T_{\text{air}}, \rho_{\text{vap,air}}))}. \tag{4.12}$$

Now to the mass flux at the ice front. It must be continuous, so $\boldsymbol{f}_{\text{ice,diff}} = \boldsymbol{f}_{\text{diff}}$, and Eqs. (4.7), (4.11) and (4.12) imply that

$$\begin{aligned} \frac{\partial \widehat{\rho}_{\text{ice}}}{\partial t} &= -\nabla \cdot \boldsymbol{f}_{\text{ice,diff}} = -\nabla \cdot \boldsymbol{f}_{\text{diff}} = \sum_{l=1}^{3} \frac{\partial}{\partial x_l} \left(\left(\frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} \; K_l \; \frac{\widehat{\rho}_{\text{vap}}^{\gamma}}{\phi^{\gamma+1}} \right) \frac{\partial \widehat{\rho}_{\text{vap}}}{\partial x_l} \right) \\ &= \sum_{l=1}^{3} \frac{\partial}{\partial x_l} \left(\left(\frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} \; K_l \; \frac{\widehat{\rho}_{\text{ice}}^{\gamma}}{(\phi\omega)^{\gamma+1}} \right) \frac{\partial \widehat{\rho}_{\text{ice}}}{\partial x_l} \right) = \sum_{l=1}^{3} \frac{\partial}{\partial x_l} \left(\left(\widehat{K}_l \; \widehat{\rho}_{\text{ice}}^{\gamma} \right) \frac{\partial \widehat{\rho}_{\text{ice}}}{\partial x_l} \right), \end{aligned} \tag{4.13}$$

where

$$\widehat{K}_l = \frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} \left(\frac{\rho_{\text{sat}}(T_{\text{wb}}(T_{\text{air}}, \rho_{\text{vap,air}}))}{\widehat{\rho}_{\text{ice},0}} \right)^{\gamma+1} K_l. \tag{4.14}$$

Eqs. (4.11)-(4.14) are in principle valid only at the ice front, but our aforementioned trick is that **we extend them to the whole of the dry region**. We can do

this because we do not need $\widehat{\rho}_{\text{ice}}$ to be physically meaningful outside the ice front (because we know that there is no ice there anyway). Therefore, **we define $\widehat{\rho}_{\text{ice}}$ to be the ice concentration when we are inside the ice front, and just an indicator function when we are outside it**.

We can extend $\widehat{\rho}_{\text{ice}}$ to the surroundings as well, using the same procedure:

$$\frac{\partial \widehat{\rho}_{\text{ice}}}{\partial t} = -\nabla \cdot \boldsymbol{f}_{\text{ice,conv}} = -\nabla \cdot \boldsymbol{f}_{\text{conv}} = \nabla \cdot (-h_{\text{m}}(\widehat{\rho}_{\text{vap}} - \rho_{\text{vap,air}})\boldsymbol{n}) \\ = \nabla \cdot \left(-\frac{h_{\text{m}}}{\omega}(\widehat{\rho}_{\text{ice}} - \omega\ \rho_{\text{vap,air}})\boldsymbol{n} \right), \quad (4.15)$$

by Eq. (4.9). The obvious choice of boundary condition then becomes

$$\widehat{\rho}_{\text{ice}}(\boldsymbol{x}, t) = \omega\ \rho_{\text{vap,air}} = \text{ constant, for } (\boldsymbol{x}, t) \in \partial\Omega \times (0, \tau). \quad (4.16)$$

4.2.4 The mass transfer equation

We now rewrite Eqs. (4.1)-(4.3) as a single equation in $(\boldsymbol{x}, t)$. After removing F, this equation becomes

$$\frac{\partial U(\boldsymbol{x}, t)}{\partial t} = \sum_{l=1}^{3} \frac{\partial}{\partial x_l} \left\{ C^l(\boldsymbol{x}, t, U(\boldsymbol{x}, t))\ \frac{\partial U(\boldsymbol{x}, t)}{\partial x_l} + B^l(\boldsymbol{x}, t, U(\boldsymbol{x}, t)) \right\}, \text{ for } (\boldsymbol{x}, t) \in \mathrm{Q}_\tau, \quad (4.17)$$

with the initial condition

$$U(\boldsymbol{x}, 0) = U_0(\boldsymbol{x}), \text{ for } \boldsymbol{x} \in \Omega,$$

and the boundary condition

$$U(\boldsymbol{x}, t) = 0, \text{ for } (\boldsymbol{x}, t) \in \partial\Omega \times (0, \tau). \quad (4.18)$$

We set

$$U(\boldsymbol{x}, t) = \widehat{\rho}_{\text{ice}}(\boldsymbol{x}, t) - \omega\ \rho_{\text{vap,air}}.$$

It is clear from Eq. (4.16) that U satisfies Eq. (4.18), and, assuming that

$$\begin{aligned} &K_l \text{ is nonzero only in the dry region,} \\ &h_{\mathrm{m}} \text{ is nonzero only in the air,} \end{aligned}$$

we can merge the contributions from Eqs. (4.13) and (4.15) into Eq. (4.17), for

$$B^l(\boldsymbol{x},t,U(\boldsymbol{x},t)) = -\frac{h_{\mathrm{m}}(\boldsymbol{x},t)}{\omega}(\widehat{\rho}_{\mathrm{ice}} - \omega\ \rho_{\mathrm{vap,air}})n_l(\boldsymbol{x},t) = -\frac{h_{\mathrm{m}}(\boldsymbol{x},t)}{\omega}\ U(\boldsymbol{x},t)n_l(\boldsymbol{x},t),$$

and

$$\begin{aligned} C^l(\boldsymbol{x},t,U(\boldsymbol{x},t)) &= \widehat{K}_l(\boldsymbol{x},t,U(\boldsymbol{x},t))\ \widehat{\rho}_{\mathrm{ice}}(\boldsymbol{x},t)^\gamma \\ &= \widehat{K}_l(\boldsymbol{x},t,U(\boldsymbol{x},t))\ [U(\boldsymbol{x},t) + \omega\ \rho_{\mathrm{vap,air}}]^\gamma, \end{aligned}$$

for

$$\widehat{K}_l(\boldsymbol{x},t,U(\boldsymbol{x},t)) = \frac{\gamma P_0}{\mu_{\mathrm{vap}}\rho_{\mathrm{vap},0}^\gamma}\ \left(\frac{\rho_{\mathrm{sat}}(T_{\mathrm{wb}}(T_{\mathrm{air}},\rho_{\mathrm{vap,air}}))}{\widehat{\rho}_{\mathrm{ice},0}}\right)^{\gamma+1} K_l(\boldsymbol{x},t,U(\boldsymbol{x},t)).$$

However, to reflect the fact that the transport coefficients K_l and h_{m} only apply in the dry region and the surroundings, respectively, we write them on the form (2.1). Let Ω_1 be the product. We allow K_l and h_{m} to have the following composite forms:

$$h_{\mathrm{m}}^k(\boldsymbol{x},t) = \sum_i \chi_{\Omega_i,k}(\boldsymbol{x},t)\ h_{\mathrm{m},i}(\boldsymbol{x},t) + \sum_j \overline{\chi}_{\Omega_j,k}(\boldsymbol{x},t)\ h_{\mathrm{m},j}(\boldsymbol{x},t),\ \text{for disjoint } \Omega_i,\Omega_j,$$

and

$$\begin{aligned} &K_l^k(\boldsymbol{x},t,U(\boldsymbol{x},t)) = \overline{\chi}_{\Omega_{\mathrm{ice}},k}(U(\boldsymbol{x},t))\sum_i \chi_{\Omega_i,k}(\boldsymbol{x},t)\ K_{l,i}(\boldsymbol{x},t),\ \text{for disjoint } \Omega_i \text{ and} \\ &\Omega_{\mathrm{ice}} = \{(\boldsymbol{x},t)\in \mathrm{Q}_\tau\ \big|\ (U(\boldsymbol{x},t) + \omega\rho_{\mathrm{vap,air}})^2 > (K_{\mathrm{ice}}\widehat{\rho}_{\mathrm{ice},0})^2\},\ \text{for } K_{\mathrm{ice}} \text{ just below 1.} \end{aligned}$$

These new coefficients allow us to model highly composite products. The factor $\overline{\chi}_{\Omega_{\text{ice}},k}$ makes $K_l^k \approx 0$ in the moist regions.

We accordingly redefine B^l and C^l as

$$B_k^l(\boldsymbol{x}, t, U(\boldsymbol{x}, t)) = -\frac{h_{\text{m}}^k(\boldsymbol{x}, t)}{\omega}\, U(\boldsymbol{x}, t) n_l(\boldsymbol{x}, t), \tag{4.19}$$

and

$$C_k^l(\boldsymbol{x}, t, U(\boldsymbol{x}, t)) = \varepsilon + \widehat{K}_l^k(\boldsymbol{x}, t, U(\boldsymbol{x}, t))\, |U(\boldsymbol{x}, t) + \omega\, \rho_{\text{vap,air}}|^\gamma, \tag{4.20}$$

where

$$\widehat{K}_l^k(\boldsymbol{x}, t, U(\boldsymbol{x}, t)) = \frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^\gamma} \left(\frac{\rho_{\text{sat}}(T_{\text{wb}}(T_{\text{air}}, \rho_{\text{vap,air}}))}{\widehat{\rho}_{\text{ice},0}} \right)^{\gamma+1} K_l^k(\boldsymbol{x}, t, U(\boldsymbol{x}, t)).$$

Here, ε is a very small parameter we add to prevent C_k^l from becoming zero. Note that we take the absolute value of $U + \omega\, \rho_{\text{vap,air}}$ in C_k^l. This is because $(U + \omega\, \rho_{\text{vap,air}})^\gamma$ becomes complex if $U < -\omega\, \rho_{\text{vap,air}}$. In practice, U is nonnegative, but we need C_k^l to be written in this way in the proof of Theorem 4.2.1 below.

Now that we have made B and C depend on k, the solution U will also depend on k, according to the equation

$$\frac{\partial U_k(\boldsymbol{x}, t)}{\partial t} = \sum_{l=1}^{3} \frac{\partial}{\partial x_l} \left\{ C_k^l(\boldsymbol{x}, t, U_k(\boldsymbol{x}, t))\, \frac{\partial U_k(\boldsymbol{x}, t)}{\partial x_l} + B_k^l(\boldsymbol{x}, t, U_k(\boldsymbol{x}, t)) \right\}, \quad \text{for } (\boldsymbol{x}, t) \in \text{Q}_\tau. \tag{4.21}$$

The boundary condition (4.18) becomes

$$U_k(\boldsymbol{x}, t) = 0, \text{ for } (\boldsymbol{x}, t) \in \partial\Omega \times (0, \tau), \tag{4.22}$$

and we choose

$$U_k(\boldsymbol{x}, 0) = U_{k,0}(\boldsymbol{x}) = \chi_{\Omega_1,k}(\boldsymbol{x}, 0)\, [\widehat{\rho}_{\text{ice},0} - \omega\, \rho_{\text{vap,air}}], \text{ for } \boldsymbol{x} \in \Omega, \tag{4.23}$$

which approximately satisfies Eq. (4.22) at $t = 0$, for k sufficiently large.

4.2.5 Well-posedness

In [4], the authors prove the well-posedness of (4.1)-(4.2) with Neumann boundary conditions. As mentioned above, by choosing a basis of functions that are zero on the boundary, we can use their theory on our homogeneous Dirichlet problem as well. We therefore make the following assumptions:

Assumption 9 (*Properties of the solution and the transport coefficients*)

We assume that the following relations hold for the system (4.21)*:*

1. *B_k^l and C_k^l are continuous in $\overline{Q_\tau} \times \mathbb{R}$. This translates to each $h_{\mathrm{m},i}$, $h_{\mathrm{m},j}$, $K_{l,i}$ and n_l being continuous (and thus bounded) in Q_τ.*
2. *The first order derivatives of a solution U are continuous in $\overline{Q_\tau}$. Note that we have already indirectly assumed this in the derivations above.*

These assumptions are sufficient to ensure the well-posedness of our system:

Theorem 4.2.1 (*Well-posedness*)

With the assumptions in Asmp. 9, the system (4.21)-(4.23)*, with coefficients given by Eqs.* (4.19) *and* (4.20)*, is well posed.*

Proof:

1. Since $(h_{\mathrm{m}}^k n_l)/\omega$ is bounded in Q_τ and B_k^l is linear in U, B_k^l is Lipschitz continuous in U:

$$|B_k^l(\cdot,\cdot,V) - B_k^l(\cdot,\cdot,W)| \leq \max_{(\boldsymbol{x},t)\in \mathrm{Q}_\tau} \left| \frac{h_{\mathrm{m}}^k n_l}{\omega} \right| \, |V - W|,$$

   for all V, W in the space $\mathrm{V}_2(\mathrm{Q}_\tau)$. Note that C_k^l can be written on the form

$$C_k^l(\boldsymbol{x},t,U) = \varepsilon + f_l(\boldsymbol{x},t)\ \overline{\chi}_{\Omega_{\mathrm{ice}},k}(U)\ |U + \omega\rho_{\mathrm{vap,air}}|^\gamma,$$

   where f_l is bounded and nonnegative. We see that for $A = (K_{\mathrm{ice}}\widehat{\rho}_{\mathrm{ice},0})^2$ and $g = g(U) = |U + \omega\rho_{\mathrm{vap,air}}|$,

$$\begin{aligned}\overline{\chi}_{\Omega_{\text{ice}},k}(U)\ |U+\omega\rho_{\text{vap,air}}|^\gamma &= \frac{1}{2}\left(1-\tanh(k(g^2-A))\right)g^\gamma \\ &= \frac{1}{2}\left(\frac{\cosh(k(g^2-A))-\sinh(k(g^2-A))}{\cosh(k(g^2-A))}\right)g^\gamma \\ &= \exp\bigl(-k(g^2-A)\bigr)\ \frac{g^\gamma}{\exp(k(g^2-A))+\exp(-k(g^2-A))} \\ &= \frac{g^\gamma}{1+\exp(2k(g^2-A))}, \quad (4.24)\end{aligned}$$

which is bounded for $k>0$. Hence,

$$\varepsilon \le C_k^l \le \varepsilon + \max_{\mathrm{Q}_\tau} f_l \cdot \max_{g\ge 0}\left\{\frac{g^\gamma}{1+\exp(2k(g^2-A))}\right\} \le C_{\max} = \text{ constant.}$$

This implies that, for any $\boldsymbol{y}\in\mathbb{R}^3$,

$$\varepsilon\sum_{l=1}^3 y_l^2 \le \sum_{l=1}^3 y_l C_k^l y_l \le C_{\max}\sum_{l=1}^3 y_l^2, \text{ and } \sum_{l=1}^3 (C_k^l y_l)^2 \le C_{\max}^2 \sum_{l=1}^3 y_l^2.$$

Furthermore, let $g_1=|V+\omega\rho_{\text{vap,air}}|$ and $g_2=|W+\omega\rho_{\text{vap,air}}|$. Since $\gamma>1$, the function

$$g \longmapsto \frac{1}{2}\left(1-\tanh(k(g^2-A))\right)g^\gamma$$

has a bounded first order derivative in $[0,\infty)$, and is thus Lipschitz continuous for nonnegative g. Eq. (4.24) then implies that

$$\begin{aligned}&|C_k^l(\cdot,\cdot,V)-C_k^l(\cdot,\cdot,W)| \\ &\quad\le \max_{\mathrm{Q}_\tau} f_l \left|\frac{1}{2}\left(1-\tanh(k(g_1^2-A))\right)g_1^\gamma - \frac{1}{2}\left(1-\tanh(k(g_2^2-A))\right)g_2^\gamma\right| \\ &\qquad\le K_C\ |g_1-g_2| = K_C\ |\ |V+\omega\rho_{\text{vap,air}}|-|W+\omega\rho_{\text{vap,air}}|\ | \\ &\qquad\quad\le K_C\ |\ (V+\omega\rho_{\text{vap,air}})-(W+\omega\rho_{\text{vap,air}})\ | \le K_C\ |V-W|,\end{aligned}$$

for some constant K_C, i.e. C_k^l is Lipschitz continuous. Since [4, Eq. (2.16)] implies [4, Eq. (2.5)], Asmps. 2.1 and 2.2 in [4] are thus satisfied (except for Eq. (2.3), which is redundant for homogeneous Dirichlet conditions).

2. The existence of a solution to the system now follows from [4, Theorem 4.1].

3. With the addition of the second assumption in Asmp. 9, the uniqueness of the solution follows from [4, Theorem 5.1].

4. Finally, the continuous dependence of the solution on the problem data follows from [4, Theorem 6.1].

■

Remark: According to [4, Theorem 4.2], the solution in Theorem 4.2.1 belongs to the Banach space $\mathrm{V}_2^{1,0}(\mathrm{Q}_\tau)$ of functions U satisfying

$$|U|^2_{\mathrm{Q}_\tau} = \sum_{l=1}^{3} \int_0^\tau \int_\Omega \left(\frac{\partial U(\boldsymbol{x},t)}{\partial x_l} \right)^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}t + \max_{t\in[0,\tau]} \int_\Omega U(\boldsymbol{x},t)^2 \, \mathrm{d}\boldsymbol{x} < \infty.$$

In practice, the solution will have more smoothness than the minimum necessary to satisfy this requirement. ■

5. NUMERICAL THEORY

In this chapter we develop a numerical scheme to solve (4.21)-(4.23). This scheme is based on a spectral extrapolation method by Cannon and Ewing [3]. The extrapolations require the solution at the two previous time steps to be known, so we must use another method to calculate the solution at the first time step. This other method is a spectral predictor-corrector method by the same authors [2].

Both these methods are made for solving a system of partial differential equations coupled to a system of ordinary differential equations. However, our system does not include ordinary differential equations, so we drop the dependence on their solutions (denoted $\boldsymbol{Y}$ in [2] and [3]) in our theory.

In addition, the methods are made for solving Neumann problems, while we have a homogeneous Dirichlet problem. But, as mentioned in the theory in the previous chapter, we can fix this problem by using a basis of functions that vanish on $\partial\Omega$.

5.1 The choice of basis

Our chosen numerical methods require a basis for a subspace of $\mathrm{H}^1(\Omega)$. As mentioned above, the basis functions must all be zero on $\partial\Omega$. We denote the subspace of functions in $\mathrm{H}^1(\Omega)$ that are zero on $\partial\Omega$ by $\mathrm{H}_0^1(\Omega)$. The choice of basis determines the subspace of $\mathrm{H}_0^1(\Omega)$ in which we seek a solution, as well as the properties of the resulting discrete, linear system. We give two possible basis alternatives below.

5.1.1 Alternative 1: Associated Legendre functions

Perhaps the most intuitive alternative is to use associated Legendre functions. A k-th degree, second order associated Legendre function P_k^2 is given by

$$P_k^2(x) = (1 - x^2)\,\frac{\mathrm{d}^2 P_k(x)}{\mathrm{d}x^2},$$

where P_k is the k-th degree ordinary Legendre polynomial. The set $\{P_k^2\}_{k=2}^{\infty}$ forms a basis for the space $\mathrm{L}_2^0(-1,1)$, which includes $\mathrm{H}_0^1(-1,1)$. The basis is orthogonal in [-1,1], and each function is zero on the boundary, i.e.

$$P_k^2(\pm 1) = 0, \text{ for all } k. \tag{5.1}$$

Since $\Omega = (-a,a) \times (-b,b) \times (-c,c)$ is just $(-1,1) \times (-1,1) \times (-1,1)$ stretched by the three factors a, b and c, a possible basis for $\mathrm{H}_0^1(\Omega)$ is the set $\{v_{pqr}\}_{p,q,r=2}^{\infty}$, where

$$v_{pqr}(x,y,z) = P_p^2\Big(\frac{x}{a}\Big)\, P_q^2\Big(\frac{y}{b}\Big)\, P_r^2\Big(\frac{z}{c}\Big)\,. \tag{5.2}$$

The basis of functions given by Eq. (5.2) is orthogonal in Ω, and

$$v_{pqr}(\pm a,y,z) = v_{pqr}(x,\pm b,z) = v_{pqr}(x,y,\pm c) = 0, \tag{5.3}$$

by Eq. (5.1). This is thus our first basis alternative.

5.1.2 Alternative 2: Modified complex Fourier basis

Since the solution to (4.21)-(4.23) must be zero whenever $x = \pm a$, $y = \pm b$ or $z = \pm c$, we can consider it to be periodic, with periods $2a$ in x, $2b$ in y and $2c$ in z. Periodic functions in $\mathrm{H}_0^1(\Omega)$ can be expanded in a real or complex Fourier basis. We will consider the complex alternative here.

Remark: The numerical methods in [2] and [3] are real, and must be adapted to handle a complex basis. This is straightforward. We only need to replace all the real inner products and norms by their complex counterparts, e.g.

$$\int_\Omega f \cdot g \, \mathrm{d}\boldsymbol{x} \longmapsto \int_\Omega f \cdot \overline{g} \, \mathrm{d}\boldsymbol{x},$$

for the $\mathrm{L}_2(\Omega)$ inner product. ■

Ordinary complex Fourier basis functions are of the form

$$\exp(\pi i(px/a))\,,$$

where p is an integer and i the imaginary unit. Unfortunately, we cannot use

$$v_{pqr}(x,y,z) = \exp(\pi i\,(px/a + qy/b + rz/c))$$

as a basis function, because $|v_{pqr}|$ is everywhere equal to one, so the zero boundary value condition is violated. The solution to this problem is to use a modified complex Fourier basis instead. This new basis consists of linear combinations of ordinary Fourier basis functions:

$$v_{pqr}(x,y,z) = \sum_{\alpha,\beta,\zeta} v_{pqr}^{\alpha\beta\zeta}\ \exp(\pi i(\alpha x/a + \beta y/b + \zeta z/c))\,. \tag{5.4}$$

The coefficients $v_{pqr}^{\alpha\beta\zeta}$ must be chosen so that $v_{pqr} = 0$ on $\partial\Omega$.

These modified basis functions are unfortunately not orthogonal with repect to the complex L_2 inner product:

$$\begin{aligned}
&\int_\Omega v_{pqr}(x,y,z)\ \overline{v_{stu}}(x,y,z)\ \mathrm{d}\boldsymbol{x} \\
&\qquad = \sum_{\alpha,\beta,\zeta}\sum_{\kappa,\lambda,\xi}\left(v_{pqr}^{\alpha\beta\zeta}\ \overline{v_{stu}^{\kappa\lambda\xi}}\ \prod_{l=1}^{3}\int_{-a_l}^{a_l}\exp(\pi i(\alpha_l-\kappa_l)x_l/a_l)\ \mathrm{d}x_l\right) \\
&\qquad = \sum_{\alpha,\beta,\zeta}\sum_{\kappa,\lambda,\xi}\left(v_{pqr}^{\alpha\beta\zeta}\ \overline{v_{stu}^{\kappa\lambda\xi}}\ \prod_{l=1}^{3}(2a_l\delta_{\alpha_l\kappa_l})\right) = 8abc\sum_{\kappa,\lambda,\xi}\left(v_{pqr}^{\kappa\lambda\xi}\ \overline{v_{stu}^{\kappa\lambda\xi}}\right).
\end{aligned} \tag{5.5}$$

This is, however, not a major issue (it just increases the complexity of the scheme). The basis of functions given by Eq. (5.4) is our second basis alternative.

5.1.3 Selecting a basis

We now compare the two basis alternatives:

1. **Arithmetic: Advantage Legendre**
   The associated Legendre basis requires real arithmetic, while the modified complex Fourier basis requires complex arithmetic, which is much slower.

2. **Orthogonality: Advantage Legendre**
   The associated Legendre basis is orthogonal, so the resulting linear system requires fewer computations to construct than that of the modified complex Fourier basis, which is not orthogonal.

3. **Symmetries: Advantage Fourier**
   The modified complex Fourier basis results in a linear system with several types of symmetry, while the associated Legendre basis results in a linear system which is only symmetric. In general, the more symmetries a linear system has, the faster it can be solved.

4. **Numerical integration: Decisive advantage Fourier**
   With any basis, we must calculate a huge number of integrals. With the modified complex Fourier basis, we can do this *very fast and simultaneously*, using a three-dimensional fast Fourier transform (FFT) (cf. Section 6.1.1). The associated Legendre basis results in integrals which must be calculated one by one, which is extremely slow.

The last point in this list simply rules out the associated Legendre basis in practice. **Our choice is therefore to use the modified complex Fourier basis.**

The coefficients v_{pqr} in Eq. (5.4) must satisfy Eq. (5.3) for all (x, y, z). We therefore rewrite v_{pqr} as

$$v_{pqr}(x, y, z) = \prod_{l=1}^{3} \left(\exp(\pi i(p_l x_l / a_l)) - (-1)^{p_l}\right), \tag{5.6}$$

for $p, q, r = -N, \ldots, -1, 1, \ldots, N$. Note that $p, q, r \neq 0$, since, if any of them are zero, v_{pqr} is identically zero. It is easy to see that v_{pqr} in Eq. (5.6) does satisfy Eq. (5.3).

Using v_{pqr} from Eq. (5.6) and v_{stu} from Eq. (5.4), we get,

$$\begin{aligned}
&\int_\Omega v_{pqr}(x,y,z)\ \overline{v_{stu}}(x,y,z)\ \mathrm{d}\boldsymbol{x} \\
&\quad = \sum_{\kappa,\lambda,\xi} \overline{v_{stu}^{\kappa\lambda\xi}} \prod_{l=1}^{3} \int_{-a_l}^{a_l} \exp(\pi i(p_l-\kappa_l)x_l/a_l) - (-1)^{p_l}\exp(-\pi i(\kappa_l x_l/a_l))\ \mathrm{d}x_l \\
&\quad = 8abc \sum_{\kappa,\lambda,\xi} \overline{v_{stu}^{\kappa\lambda\xi}} \prod_{l=1}^{3} (\delta_{p_l\kappa_l} - (-1)^{p_l}\delta_{\kappa_l 0}), \quad (5.7)
\end{aligned}$$

for $p,q,r \neq 0$. Comparing Eqs. (5.5) and (5.7), we get a possible choice for $v_{pqr}^{\kappa\lambda\xi}$:

$$v_{pqr}^{\kappa\lambda\xi} = \prod_{l=1}^{3} (\delta_{p_l\kappa_l} - (-1)^{p_l}\delta_{\kappa_l 0}), \text{ for } p,q,r \neq 0. \quad (5.8)$$

5.2 Some important matrices

In this section we present some matrices that are essential ingredients in our discrete system.

5.2.1 The basis generation matrix

We define the rearrangement function ψ by

$$\psi(p,q,r,N) = \lceil (2N+1)^3/2 \rceil + p + q\cdot(2N+1) + r\cdot(2N+1)^2,$$

i.e. ψ rearranges the triples $\{(p,q,r)\}_{p,q,r=-N}^{N}$ into the same order as the Matlab function *ndgrid*. Note that the smallest number, $\psi(-N,-N,-N,N)$ is equal to one, not zero. This is in accordance with the indexing of Matlab arrays. Note also that

$$\psi(-p,-q,-r,N) = (2N+1)^3 + 1 - \psi(p,q,r,N), \quad (5.9)$$

which is illustrated in Table 5.1 for $N=1$.

(p,q,r)	$\psi(p,q,r,1)$	$\psi(-p,-q,-r,1)$	$(2\cdot 1+1)^3+1-\psi(p,q,r,1)$
(-1,-1,-1)	1	27	27
(0,-1,-1)	2	26	26
(1,-1,-1)	3	25	25
(-1,0,-1)	4	24	24
(0,0,-1)	5	23	23
(1,0,-1)	6	22	22
(-1,1,-1)	7	21	21
(0,1,-1)	8	20	20
(1,1,-1)	9	19	19
(-1,-1,0)	10	18	18
(0,-1,0)	11	17	17
(1,-1,0)	12	16	16
(-1,0,0)	13	15	15
(0,0,0)	14	14	14
(1,0,0)	15	13	13
(-1,1,0)	16	12	12
(0,1,0)	17	11	11
(1,1,0)	18	10	10
(-1,-1,1)	19	9	9
(0,-1,1)	20	8	8
(1,-1,1)	21	7	7
(-1,0,1)	22	6	6
(0,0,1)	23	5	5
(1,0,1)	24	4	4
(-1,1,1)	25	3	3
(0,1,1)	26	2	2
(1,1,1)	27	1	1

Tab. 5.1: The values of $\psi(p,q,r,1)$, $\psi(-p,-q,-r,1)$ and $(2\cdot 1+1)^3+1-\psi(p,q,r,1)$ for different values of (p,q,r).

Using ψ and Eq. (5.8), we can construct a matrix $\boldsymbol{G}$ by first setting

$$G(\psi(\kappa,\lambda,\xi,N),\psi(p,q,r,N)) = \begin{cases} v_{pqr}^{\kappa\lambda\xi}, & p,q,r \neq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{5.10}$$

for $p,q,r,\kappa,\lambda,\xi = -N,\ldots,N$, and then removing the zero columns. We call the resulting $(2N+1)^3 \times 8N^3$ matrix $\boldsymbol{G}$ our **basis generation matrix**.

Remark: Every column $\psi(p,q,r,N)$, $p,q,r \neq 0$ in $\boldsymbol{G}$ has precisely 8 nonzero elements, regardless of the size of N, so $\boldsymbol{G}$ is very sparse. This is because there are only two values for each κ_l that cause the factor $(\delta_{p_l\kappa_l} - (-1)^{p_l}\delta_{\kappa_l 0})$ in Eq. (5.8) to be nonzero, namely $\kappa_l = 0$ and $\kappa_l = p_l \neq 0$. Since there are three such factors, a total of $2^3 = 8$ elements are nonzero (and all these are either +1 or -1). ■

We see from Eqs. (5.5) and (5.10) that element $(\psi(s,t,u,N),\psi(p,q,r,N))$ of the matrix $\boldsymbol{G}^H\boldsymbol{G}$ is equal to

$$\sum_{\kappa,\lambda,\xi} \left(v_{pqr}^{\kappa\lambda\nu}\ \overline{v_{stu}^{\kappa\lambda\xi}} \right) = \frac{1}{8abc} \int_\Omega v_{pqr}(x,y,z)\ \overline{v_{stu}}(x,y,z)\ \mathrm{d}\boldsymbol{x},$$

which by Eq. (5.6) is equal to

$$\begin{aligned} &\frac{1}{8abc} \prod_{l=1}^{3} \int_{-a_l}^{a_l} \left(\exp(\pi i(p_l x_l/a_l)) - (-1)^{p_l}\right)\left(\exp(-\pi i(s_l x_l/a_l)) - (-1)^{s_l}\right)\ \mathrm{d}x_l \\ &= \frac{1}{8abc} \prod_{l=1}^{3} \left[2a_l(\delta_{p_l s_l} - (-1)^{p_l}\delta_{s_l 0} - (-1)^{s_l}\delta_{p_l 0} + (-1)^{p_l+s_l})\right] = \prod_{l=1}^{3} (\delta_{p_l s_l} + (-1)^{p_l+s_l}), \end{aligned}$$

for $p,q,r,s,t,u = -N,\ldots,-1,1,\ldots,N$. Hence,

$$\boldsymbol{G}^H\boldsymbol{G} = \widehat{\boldsymbol{G}} \otimes \widehat{\boldsymbol{G}} \otimes \widehat{\boldsymbol{G}},$$

where

$$\widehat{\boldsymbol{G}}(N+1+s_l-\delta_{s_l,|s_l|}, N+1+p_l-\delta_{p_l,|p_l|}) = \delta_{p_l s_l} + (-1)^{p_l+s_l}, \quad \text{for } p_l, s_l = -N,\ldots,-1,1,\ldots,N,$$

i.e. $\boldsymbol{G}^H\boldsymbol{G}$ is a Kronecker product of the $2N \times 2N$ matrix $\widehat{\boldsymbol{G}}$. Note that $\widehat{\boldsymbol{G}}$ is equal to the identity matrix plus a matrix where the elements in any given column are all $+(-1)^{s_l}$ or all $-(-1)^{s_l}$, $s_l = -N, \ldots, -1, 1, \ldots, N$. The latter matrix has rank one, so it has $2N-1$ zero eigenvalues and one that is equal to the trace $2N$. Hence, $\widehat{\boldsymbol{G}}$ has $2N-1$ eigenvalues equal to one and one that is equal to $2N+1$. Thus, it is invertible, and it follows that

$$(\boldsymbol{G}^H\boldsymbol{G})^{-1} = \widehat{\boldsymbol{G}}^{-1} \otimes \left(\widehat{\boldsymbol{G}}^{-1} \otimes \widehat{\boldsymbol{G}}^{-1}\right). \tag{5.11}$$

To save memory, we never construct $(\boldsymbol{G}^H\boldsymbol{G})^{-1}$ explicitly in the numerical implementation. Instead, we use Eq. (5.11) and [10, Eq. (4.5.19)] when calculating $\boldsymbol{w}_0$ in Eq. (5.15) (cf. Code A.5).

5.2.2 Other important matrices

Differentiating the complex exponential functions produces imaginary factors, which we store (without the imaginary unit) in the $(2N+1)^3 \times (2N+1)^3$ diagonal **derivative factor matrices** $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$, defined by

$$D_l(\psi(\kappa,\lambda,\xi,N),\psi(\alpha,\beta,\zeta,N)) = \frac{\pi\kappa_l}{a_l}\delta_{\alpha_l\kappa_l}.$$

Since each $\boldsymbol{D}_l$ is a real diagonal matrix, it obviously satisfies

$$\boldsymbol{D}_l^H = \overline{\boldsymbol{D}_l} = \boldsymbol{D}_l.$$

We also introduce the $m \times m$ **exchange matrix** $\boldsymbol{J}_m$:

$$\boldsymbol{J}_m = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 1 \\ \vdots & & \iddots & 1 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & 1 & \iddots & & \vdots \\ 1 & 0 & \cdots & \cdots & 0 \end{bmatrix}, \text{ e.g. } \boldsymbol{J}_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Note that $\boldsymbol{J}_m$ is orthogonal:

$$\boldsymbol{J}_m = \boldsymbol{J}_m^H = \boldsymbol{J}_m^{-1}.$$

The effect of multiplying a matrix by $\boldsymbol{J}_m$ from the left side, from the right side and from both sides is to reverse the order of its rows, to reverse the order of its columns, and to rotate it by 180°, respectively.

5.3 Derivation of the discrete system

We are now in position to derive our discrete, linear system. The theory in this section is based on a complex extension of the equations (3.5) in [3] and (3.6) in [2]. Both sets of equations are based on the weak (integrated) formulation of the problem, so we shall use the test functions η_{stu}, given by

$$\eta_{stu} = v_{stu}.$$

With a time step size of Δt, we write the approximate solution at time step $t_n = n\Delta t$ as W_n, given by

$$W_n(x, y, z) = \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{n,pqr} v_{pqr}(x, y, z), \tag{5.12}$$

where v_{pqr} are the basis functions given by Eq. (5.4), with coefficients given by Eq. (5.8).

A key ingredient in the discrete system is the $(2N + 1)^3 \times 1$ vector of Fourier coefficients of f. We write it as $\mathbf{Fourier}_f$, where

$$\mathbf{Fourier}_f(\psi(\kappa, \lambda, \xi, N)) = \frac{1}{8abc} \int_\Omega f(\boldsymbol{x}) \exp(-\pi i(\kappa x/a + \lambda y/b + \xi z/c)) \, \mathrm{d}\boldsymbol{x}. \tag{5.13}$$

Remark: We derive a more general numerical scheme than necessary, with the source term $F = F(\boldsymbol{x}, t, U(\boldsymbol{x}, t))$ back in place. ■

5.3.1 The discrete initial condition

We start by deriving the discrete initial condition. Given the initial function U_0, the equation for the initial solution W_0 becomes

$$\int_\Omega W_0 \ \overline{\eta_{stu}} \ \mathrm{d}\boldsymbol{x} = \int_\Omega U_0 \ \overline{\eta_{stu}} \ \mathrm{d}\boldsymbol{x}. \tag{5.14}$$

The left side of Eq. (5.14) becomes

$$\begin{aligned}
\int_\Omega W_0 \ \overline{\eta_{stu}} \ \mathrm{d}\boldsymbol{x} &= \int_\Omega \left(\sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{0,pqr} v_{pqr} \right) \overline{v_{stu}} \ \mathrm{d}\boldsymbol{x} \\
&= \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \left(\sum_{\alpha,\beta,\zeta=-N}^{N} \sum_{\kappa,\lambda,\xi=-N}^{N} v_{pqr}^{\alpha\beta\zeta} \ \overline{v_{stu}^{\kappa\lambda\xi}} \prod_{l=1}^{3} \int_{-a_l}^{a_l} \exp(\pi i(\alpha_l - \kappa_l)x_l/a_l)) \ \mathrm{d}x_l \right) w_{0,pqr} \\
&= 8abc \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \left(\sum_{\kappa,\lambda,\xi=-N}^{N} v_{pqr}^{\kappa\lambda\xi} \ \overline{v_{stu}^{\kappa\lambda\xi}} \right) w_{0,pqr},
\end{aligned}$$

which is element $\psi(s,t,u,N)$ in the vector

$$8abc(\boldsymbol{G}^H\boldsymbol{G}) \ \boldsymbol{w}_0.$$

The right side becomes

$$\int_\Omega U_0 \ \overline{\eta_{stu}} \ \mathrm{d}\boldsymbol{x} = \sum_{\kappa,\lambda,\xi=-N}^{N} \overline{v_{stu}^{\kappa\lambda\xi}} \int_\Omega U_0(\boldsymbol{x}) \ \exp(-\pi i(\kappa x/a + \lambda y/b + \xi z/c)) \ \mathrm{d}\boldsymbol{x},$$

which is element $\psi(s,t,u,N)$ in the vector

$$8abc \ \boldsymbol{G}^H\mathbf{Fourier}_{U_0}.$$

Comparing the left and right sides for all (s,t,u), we get the initial basis coefficients:

$$\boldsymbol{w}_0 = (\boldsymbol{G}^H\boldsymbol{G})^{-1}\boldsymbol{G}^H\mathbf{Fourier}_{U_0}. \tag{5.15}$$

5.3.2 The discrete equations

Now we derive the discrete system itself. The first time step requires a different approach than the rest, as mentioned at the beginning of this chapter. The equations in [2] and [3] are the same, except for different function arguments and a slight difference in notation: In [3, Eq. (3.5)], W_n is written as W_{*n}, while in [2, Eq. (3.6)], it is written as W_{*n} or W_{**n}. We simply write W_n and treat both cases together. To further simplify the notation, we define

$$\begin{aligned} &\text{the difference operator } \Delta, \text{ given by } \Delta_n f = f_{n+1} - f_n, \\ &\text{the average operator } \Theta, \text{ given by } \Theta_n f = \frac{f_{n+1} + f_n}{2}. \end{aligned} \tag{5.16}$$

We use a constant time step Δt (without the index n, since it is a constant), and write

$$t_n = n\ \Delta t, \text{ and } t_{n+1/2} = \Theta_n t = \frac{t_{n+1} + t_n}{2}.$$

The main equation then becomes (with the boundary term removed)

$$\begin{aligned} \int_\Omega \frac{\Delta_n W}{\Delta t}\ \overline{\eta_{stu}}\ \mathrm{d}\boldsymbol{x} = &- \int_\Omega \sum_{l=1}^{3} \left(\frac{\partial \overline{\eta_{stu}}}{\partial x_l}\ C^l(\boldsymbol{x}, t_{n+1/2}, \cdot)\ \frac{\partial(\Theta_n W)}{\partial x_l} \right)\ \mathrm{d}\boldsymbol{x} \\ &- \int_\Omega \sum_{l=1}^{3} \left(\frac{\partial \overline{\eta_{stu}}}{\partial x_l}\ B^l(\boldsymbol{x}, t_{n+1/2}, \cdot) \right)\ \mathrm{d}\boldsymbol{x} + \int_\Omega \overline{\eta_{stu}}\ F(\boldsymbol{x}, t_{n+1/2}, \cdot)\ \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{5.17}$$

where $(\cdot)$ is a temporary wildcard that symbolizes the dependence on some approximation of $W_{n+1/2}$.

Remark: In both [2] and [3], the integral involving F in Eq. (5.17) has a negative sign in front of it. This is a misprint. ■

The left side of Eq. (5.17) becomes

$$\int_\Omega \frac{\Delta_n W}{\Delta t}\ \overline{\eta_{stu}}\ \mathrm{d}\boldsymbol{x} = \int_\Omega \left(\sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \frac{(\Delta_n \boldsymbol{w})_{pqr}}{\Delta t}\ v_{pqr} \right) \overline{v_{stu}}\ \mathrm{d}\boldsymbol{x}$$

$$= \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \left(\sum_{\alpha,\beta,\zeta=-N}^{N} \sum_{\kappa,\lambda,\xi=-N}^{N} v_{pqr}^{\alpha\beta\zeta}\ \overline{v_{stu}^{\kappa\lambda\xi}} \prod_{l=1}^{3} \int_{-a_l}^{a_l} \exp(\pi i(\alpha_l - \kappa_l)x_l/a_l)\ \mathrm{d}x_l \right) \frac{(\Delta_n \boldsymbol{w})_{pqr}}{\Delta t}$$

$$= 8abc \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \left(\sum_{\kappa,\lambda,\xi=-N}^{N} v_{pqr}^{\kappa\lambda\xi}\ \overline{v_{stu}^{\kappa\lambda\xi}} \right) \frac{(\Delta_n \boldsymbol{w})_{pqr}}{\Delta t},$$

which is element $\psi(s,t,u,N)$ in the vector

$$\frac{8abc}{\Delta t}(\boldsymbol{G}^H \boldsymbol{G})\ (\Delta_n \boldsymbol{w}). \tag{5.18}$$

The first term on the right side becomes

$$-\int_\Omega \sum_{l=1}^{3} \left(\frac{\partial \overline{\eta_{stu}}}{\partial x_l}\ C^l(\boldsymbol{x}, t_{n+1/2}, \cdot)\ \frac{\partial(\Theta_n W)}{\partial x_l} \right)\ \mathrm{d}\boldsymbol{x}$$

$$= -\sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} \left(\sum_{\alpha,\beta,\zeta=-N}^{N} \sum_{\kappa,\lambda,\xi=-N}^{N} v_{pqr}^{\alpha\beta\zeta}\ \overline{v_{stu}^{\kappa\lambda\xi}} \sum_{l=1}^{3} \left[\left(\frac{\pi i \alpha_l}{a_l} \right) \left(-\frac{\pi i \kappa_l}{a_l} \right) \right. \right.$$

$$\left. \left. \cdot \int_\Omega C^l(\boldsymbol{x}, t_{n+1/2}, \cdot) \exp(\pi i((\alpha-\kappa)x/a + (\beta-\lambda)y/b + (\zeta-\xi)z/c))\ \mathrm{d}\boldsymbol{x} \right] \right) (\Theta_n \boldsymbol{w})_{pqr}.$$

It can be shown that this equals element $\psi(s,t,u,N)$ in the vector

$$-\sum_{l=1}^{3} \boldsymbol{G}^H \boldsymbol{D}_l \boldsymbol{M}_n^l(\cdot) \boldsymbol{D}_l \boldsymbol{G}\ (\Theta_n \boldsymbol{w}), \tag{5.19}$$

where each $\boldsymbol{M}_n^l(\cdot)$ is a $(2N+1)^3 \times (2N+1)^3$ **integral matrix**, defined by

$$M_n^l(\cdot)(\psi(\kappa,\lambda,\xi,N),\psi(\alpha,\beta,\zeta,N)) = \int_\Omega C^l(\boldsymbol{x},t_{n+1/2},\cdot)\,\exp(\pi i((\alpha-\kappa)x/a+(\beta-\lambda)y/b+(\zeta-\xi)z/c)))\;\mathrm{d}\boldsymbol{x}. \quad (5.20)$$

The second term becomes

$$-\int_\Omega \sum_{l=1}^{3}\left(\frac{\partial\overline{\eta_{stu}}}{\partial x_l}\,B^l(\boldsymbol{x},t_{n+1/2},\cdot)\right)\mathrm{d}\boldsymbol{x}$$
$$=-\sum_{l=1}^{3}\sum_{\kappa,\lambda,\xi=-N}^{N}\overline{v_{stu}^{\kappa\lambda\xi}}\left(-\frac{\pi i\kappa_l}{a_l}\right)\int_\Omega B^l(\boldsymbol{x},t_{n+1/2},\cdot)\exp(-i\pi(\kappa x/a+\lambda y/b+\xi z/c))\;\mathrm{d}\boldsymbol{x},$$

which is element $\psi(s,t,u,N)$ in the vector

$$i\;8abc\sum_{l=1}^{3}\boldsymbol{G}^H\boldsymbol{D}_l\mathbf{Fourier}_{B^l(\boldsymbol{x},t_{n+1/2},\cdot)}. \quad (5.21)$$

The third term becomes

$$\int_\Omega \overline{\eta_{stu}}\;F(\boldsymbol{x},t_{n+1/2},\cdot)\;\mathrm{d}\boldsymbol{x} = \sum_{\kappa,\lambda,\xi=-N}^{N}\overline{v_{stu}^{\kappa\lambda\xi}}\int_\Omega F(\boldsymbol{x},t_{n+1/2},\cdot)\;\exp(-i\pi(\kappa x/a+\lambda y/b+\xi z/c))\;\mathrm{d}\boldsymbol{x},$$

which is element $\psi(s,t,u,N)$ in the vector

$$8abc\;\boldsymbol{G}^H\mathbf{Fourier}_{F(\boldsymbol{x},t_{n+1/2},\cdot)}. \quad (5.22)$$

Collecting Eqs. (5.19), (5.21) and (5.22) and equating them with Eq. (5.18) for all (s,t,u), we get

$$\frac{8abc}{\Delta t}(\boldsymbol{G}^H\boldsymbol{G})\;(\Delta_n\boldsymbol{w}) = -\sum_{l=1}^{3}\boldsymbol{G}^H\boldsymbol{D}_l\boldsymbol{M}_n^l(\cdot)\boldsymbol{D}_l\boldsymbol{G}\;(\Theta_n\boldsymbol{w}) + i\;8abc\sum_{l=1}^{3}\boldsymbol{G}^H\boldsymbol{D}_l\mathbf{Fourier}_{B^l(\boldsymbol{x},t_{n+1/2},\cdot)} + 8abc\;\boldsymbol{G}^H\mathbf{Fourier}_{F(\boldsymbol{x},t_{n+1/2},\cdot)},$$

i.e.

$$\begin{aligned}\boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3}+\frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l\boldsymbol{M}_n^l(\cdot)\boldsymbol{D}_l\right)\boldsymbol{G}\boldsymbol{w}_{n+1}\\ =\boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3}-\frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l\boldsymbol{M}_n^l(\cdot)\boldsymbol{D}_l\right)\boldsymbol{G}\boldsymbol{w}_n\\ +\Delta t\,\boldsymbol{G}^H\left(\sum_{l=1}^{3}(i\boldsymbol{D}_l)\mathbf{Fourier}_{B^l(\boldsymbol{x},t_{n+1/2},\cdot)}+\mathbf{Fourier}_{F(\boldsymbol{x},t_{n+1/2},\cdot)}\right).\end{aligned}$$

Here, $\boldsymbol{I}_{(2N+1)^3}$ is the $(2N+1)^3\times(2N+1)^3$ identity matrix. Finally, we set

$$\begin{aligned}\boldsymbol{A}_n(\cdot)&=\boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3}+\frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l\boldsymbol{M}_n^l(\cdot)\boldsymbol{D}_l\right)\boldsymbol{G},\\ \boldsymbol{e}_n(\cdot)&=\Delta t\,\boldsymbol{G}^H\left(\sum_{l=1}^{3}(i\boldsymbol{D}_l)\mathbf{Fourier}_{B^l(\boldsymbol{x},t_{n+1/2},\cdot)}+\mathbf{Fourier}_{F(\boldsymbol{x},t_{n+1/2},\cdot)}\right),\end{aligned}$$

and write our discrete system as

$$\boldsymbol{A}_n(\cdot)(\Theta_n\boldsymbol{w})=\boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n+\frac{\boldsymbol{e}_n(\cdot)}{2}. \tag{5.23}$$

We can now present the algorithm we use to solve (4.21)-(4.23) numerically. It is given in Algorithm 2, where the wildcards $(\cdot)$ have been replaced by the proper arguments.

Algorithm 2 (*Modified complex Cannon & Ewing spectral method*)

1. ***Initial condition:*** *Calculate* $\boldsymbol{w}_0$ *from Eq.* (5.15) *and* W_0 *from Eq.* (5.12).
2. ***Time step 1, prediction:*** *Solve*

$$\boldsymbol{A}_0(W_0)(\Theta_0\boldsymbol{w}) = \boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_0 + \frac{\boldsymbol{e}_0(W_0)}{2}$$

   for $\Theta_0\boldsymbol{w}$. *Then calculate* $\boldsymbol{w}_1$ *from Eq.* (5.16) *and* W_1 *from Eq.* (5.12).
3. ***Time step 1, correction:*** *Solve*

$$\boldsymbol{A}_0((W_1+W_0)/2)(\Theta_0\boldsymbol{w}) = \boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_0 + \frac{\boldsymbol{e}_0((W_1+W_0)/2)}{2}$$

   for $\Theta_0\boldsymbol{w}$. *Then calculate* $\boldsymbol{w}_1$ *from Eq.* (5.16) *and* W_1 *from Eq.* (5.12).
4. ***All other time steps:*** *For all remaining time steps, solve*

$$\boldsymbol{A}_n(EW_n)(\Theta_n\boldsymbol{w}) = \boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n + \frac{\boldsymbol{e}_n(EW_n)}{2}$$

   for $\Theta_n\boldsymbol{w}$. *Then calculate* $\boldsymbol{w}_{n+1}$ *from Eq.* (5.16) *and* W_{n+1} *from Eq.* (5.12). *The arguments* EW_n *are given by [3, Eq. (3.6a)] as*

$$EW_n = \frac{3}{2}\,W_n - \frac{1}{2}\,W_{n-1}.$$

5.4 Properties of the discrete system

In this section we show that each of the systems in Algorithm 2 have certain properties which enable us to solve them faster. First we recall that an $m \times k$ matrix $\boldsymbol{E}$ is centrohermitian (CH) if

$$\boldsymbol{E} = \overline{\boldsymbol{J}_m\boldsymbol{E}\boldsymbol{J}_k},$$

or, equivalently,

$$E(p,q) = \overline{E(m+1-p, k+1-q)}, \tag{5.24}$$

i.e. if it equals a 180° rotation of its complex conjugate. Note that the following matrices are CH:

- Linear combinations of CH matrices, if the coefficients are real.
- Products of CH matrices.
- Hermitian transposes of CH matrices.
- Inverses of CH matrices.

The quantities in our system (5.23) have this property, and some others:

Theorem 5.4.1 (*Properties of the linear system*)

In Eq. (5.23)*, the matrix* $\boldsymbol{A}_n$ *is Hermitian positive definite (HPD) and centrohermitian, and the vectors* $\boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n + \boldsymbol{e}_n/2$ *and* $\Theta_n\boldsymbol{w}$ *are centrohermitian as well.*

Proof: We frequently rely upon the identities (5.9) and (5.24) in the theory below.

1. Let $p, q, r, \kappa, \lambda, \xi \in -N, \ldots, -1, 1, \ldots, N$. Then we see from Eqs. (5.8) and (5.10) that row $\psi(\kappa, \lambda, \xi, N)$ in $\boldsymbol{G}$ consists of the elements $\sum_{l=1}^{3} \delta_{p_l \kappa_l}$. This evaluates to one when $(p, q, r) = (\kappa, \lambda, \xi)$, and zero otherwise. Hence, the column corresponding to (p, q, r) is the only column to have a nonzero element in this row, i.e. it is linearly independent of the other columns. Repeating the argument for the other triples (κ, λ, ξ) shows that all columns are linearly independent, i.e. $\boldsymbol{G}$ has full column rank.

   Furthermore, we see from Eq. (5.8) that

$$v_{-p-q-r}^{-\kappa-\lambda-\xi} = v_{pqr}^{\kappa\lambda\xi}.$$

   Consider now the construction of $\boldsymbol{G}$ in Eq. (5.10):

$$\begin{aligned} G(\psi(\kappa,\lambda,\xi,N),\psi(p,q,r,N)) &= \left\{ \begin{array}{rl} v_{pqr}^{\kappa\lambda\xi}, & p,q,r \neq 0 \\ 0, & \text{otherwise} \end{array} \right\} \\ = \left\{ \begin{array}{rl} v_{-p-q-r}^{-\kappa-\lambda-\xi}, & p,q,r \neq 0 \\ 0, & \text{otherwise} \end{array} \right\} &= G(\psi(-\kappa,-\lambda,-\xi,N),\psi(-p,-q,-r,N)) \\ &= G((2N+1)^3+1-\psi(\kappa,\lambda,\xi,N),(2N+1)^3+1-\psi(p,q,r,N)), \end{aligned}$$

   i.e. $\boldsymbol{G}$ was CH before its zero columns were removed. But this removal did not not alter the order of the nonzero columns, so $\boldsymbol{G}$ is still CH.

2. Eq. (5.20) implies that

$$\begin{aligned} M_n^l(\psi(\kappa,\lambda,\xi,N),\psi(\alpha,\beta,\zeta,N)) &= \overline{M_n^l(\psi(-\kappa,-\lambda,-\xi,N),\psi(-\alpha,-\beta,-\zeta,N))} \\ &= \overline{M_n^l((2N+1)^3+1-\psi(\kappa,\lambda,\xi,N),(2N+1)^3+1-\psi(\alpha,\beta,\zeta,N))}, \end{aligned}$$

so $\boldsymbol{M}_n^l$ is CH.

3. Because the identity matrix is (trivially) CH and

$$\boldsymbol{J}_{(2N+1)^3}\overline{(i\boldsymbol{D}_l)}\boldsymbol{J}_{(2N+1)^3} = i\boldsymbol{D}_l,$$

the matrix

$$\widehat{\boldsymbol{A}}_n = \boldsymbol{I}_{(2N+1)^3} - \frac{\Delta t}{16abc}\sum_{l=1}^{3}(i\boldsymbol{D}_l)\boldsymbol{M}_n^l(i\boldsymbol{D}_l)$$

is CH, since it consists of sums and products of CH matrices. Then

$$\boldsymbol{A}_n = \boldsymbol{G}^H\widehat{\boldsymbol{A}}_n\boldsymbol{G}$$

is a product of CH matrices, and is therefore itself CH.

4. Next, we see from Eq. (5.20) that

$$M_n^l(\psi(\kappa,\lambda,\xi,N),\psi(\alpha,\beta,\zeta,N)) = \overline{M_n^l(\psi(\alpha,\beta,\zeta,N),\psi(\kappa,\lambda,\xi,N))},$$

i.e. $\boldsymbol{M}_n^l$ is Hermitian. It follows that $\boldsymbol{A}_n$ is Hermitian:

$$\begin{aligned} \boldsymbol{A}_n^H &= \left[\boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3} + \frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l\boldsymbol{M}_n^l\boldsymbol{D}_l\right)\boldsymbol{G}\right]^H \\ &= \boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3}^H + \frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l^H(\boldsymbol{M}_n^l)^H\boldsymbol{D}_l^H\right)\boldsymbol{G} \\ &= \boldsymbol{G}^H\left(\boldsymbol{I}_{(2N+1)^3} + \frac{\Delta t}{16abc}\sum_{l=1}^{3}\boldsymbol{D}_l\boldsymbol{M}_n^l\boldsymbol{D}_l\right)\boldsymbol{G} = \boldsymbol{A}_n. \end{aligned}$$

5. Let $(\tilde{x}, \tilde{y}, \tilde{x}) = ((\pi x)/a, (\pi y)/b, (\pi z)/c)$ and $\tilde{\Omega} = (-\pi, \pi) \times (-\pi, \pi) \times (-\pi, \pi)$. We have

$$\int_{\Omega} C^l(\boldsymbol{x}) \exp(\pi i((\alpha - \kappa)x/a + (\beta - \lambda)y/b + (\zeta - \xi)z/c)) \, \mathrm{d}\boldsymbol{x} = \frac{8abc}{(2\pi)^3} \int_{\tilde{\Omega}} C^l(a\tilde{x}/\pi, b\tilde{y}/\pi, c\tilde{z}/\pi) \exp(-i((\kappa - \alpha)\tilde{x} + (\lambda - \beta)\tilde{y} + (\xi - \zeta)\tilde{z})) \, \mathrm{d}\tilde{\boldsymbol{x}},$$

so for $C^l = C^l_k \geq \varepsilon$, a three-dimensional generalization of [17, Theorem 2.1] shows that the smallest eigenvalue of $\boldsymbol{M}^l_n$ is positive, i.e. $\boldsymbol{M}^l_n$ is positive definite.

Let $\boldsymbol{y} \neq 0$ be an arbitrary nonzero vector and $\boldsymbol{z} = \boldsymbol{G}\boldsymbol{y}$. Because $\boldsymbol{G}$ has full column rank, $\boldsymbol{z} \neq 0$, and we have

$$\begin{aligned}\boldsymbol{y}^H \boldsymbol{A}_n \boldsymbol{y} &= \boldsymbol{y}^H \boldsymbol{G}^H \boldsymbol{G}\boldsymbol{y} + \frac{\Delta t}{16abc} \sum_{l=1}^{3} \left(\boldsymbol{y}^H \boldsymbol{G}^H \boldsymbol{D}_l \boldsymbol{M}^l_n \boldsymbol{D}_l \boldsymbol{G}\boldsymbol{y}\right) \\ &= \boldsymbol{z}^H \boldsymbol{z} + \frac{\Delta t}{16abc} \sum_{l=1}^{3} (\boldsymbol{D}_l^H \boldsymbol{z})^H \boldsymbol{M}^l_n (\boldsymbol{D}_l^H \boldsymbol{z}).\end{aligned}$$

This quantity is positive, because the first term is positive and the three terms in the sum are all nonnegative (because $\boldsymbol{M}^l_n$ is positive definite). Hence, $\boldsymbol{A}_n$ is positive definite.

6. Next, we see from Eq. (5.13) that $\mathbf{Fourier}_f$ is CH for all real f:

$$\begin{aligned}\mathbf{Fourier}_f(\psi(\kappa, \lambda, \xi, N)) &= \overline{\mathbf{Fourier}_f(\psi(-\kappa, -\lambda, -\xi, N))} \\ &= \overline{\mathbf{Fourier}_f((2N+1)^3 + 1 - \psi(\kappa, \lambda, \xi, N))}.\end{aligned}$$

7. We have already established that $\boldsymbol{G}$, $\boldsymbol{A}_n$ and $\mathbf{Fourier}_f$ are CH. The implication chart in Figure 5.1 shows that they imply that several other matrices and vectors, among them $\boldsymbol{G}^H \boldsymbol{G}\boldsymbol{w}_n + \boldsymbol{e}_n/2$ and $\Theta_n \boldsymbol{w}$, are also CH.

■

In Section 6.1.2, we use the properties mentioned in Theorem 5.4.1 to speed up the solution of the systems in Algorithm 2. The numerical implementation of the algorithm is given in Code A.5.

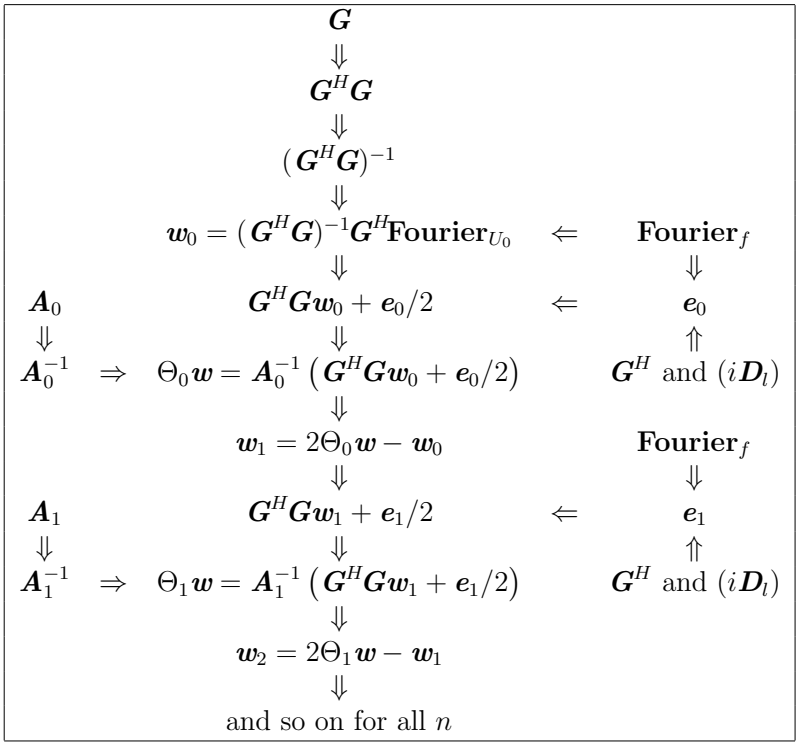


Fig. 5.1: Chart showing how some centrohermitian matrices and vectors cause other matrices and vectors to become centrohermitian as well.

6. NUMERICAL SIMULATIONS

In this chapter we test the framework by simulating AFD of pieces of cod fillet at -5 °C and -10 °C. Before we present the results of these simulations, we present some ways to make the numerical implementation more efficient, and show how to plot the results.

6.1 Improving performance and plotting results

In this section we show how to create cross-sectional plots of the product, and how to dramatically improve the performance of the numerical implementation. Without these improvements, the simulations would run very slowly.

6.1.1 Evaluating the integrals

We need to evaluate a large number of Fourier integrals, i.e. integrals of the form

$$\int_\Omega f(\boldsymbol{x}) \exp(\pi i(\kappa x/a + \lambda y/b + \xi z/c)) \ \mathrm{d}\boldsymbol{x}.$$

We can evaluate all such integrals very fast, by using the three-dimensional FFT method in [11].

First, we must define a uniform spatial grid in Ω. Let $\boldsymbol{m} = (m_1, m_2, m_3)$ be the number of grid points in the three directions, and let $\boldsymbol{j} = (j_1, j_2, j_3)$ and $\boldsymbol{k} = (k_1, k_2, k_3)$. Then, for

$$\tilde{a}_l = -a_l + \frac{a_l}{m_l} = \left(\frac{1}{m_l} - 1\right) a_l,$$

the points

$$x_l^{j_l} = \tilde{a}_l + \frac{2a_l}{m_l}\, j_l = \left(\frac{2j_l+1}{m_l} - 1\right) a_l, \text{ where } 0 \le j_l \le m_l - 1, \tag{6.1}$$

define a uniform partition of $(-a_l, a_l)$.

The size $\boldsymbol{m}$ three-dimensional discrete Fourier transform (DFT) and its inverse are defined by

$$\begin{aligned}
\mathrm{DFT}_{\boldsymbol{k}}(\boldsymbol{g_j}; \boldsymbol{m}) &= \sum_{j_1=0}^{m_1-1} \sum_{j_2=0}^{m_2-1} \sum_{j_3=0}^{m_3-1} \boldsymbol{g_j} \exp\left(-2\pi i \sum_{l=1}^{3} j_l k_l / m_l\right), \\
\mathrm{DFT}^{-1}_{\boldsymbol{k}}(\boldsymbol{g_j}; \boldsymbol{m}) &= \frac{1}{m_1 m_2 m_3} \sum_{j_1=0}^{m_1-1} \sum_{j_2=0}^{m_2-1} \sum_{j_3=0}^{m_3-1} \boldsymbol{g_j} \exp\left(2\pi i \sum_{l=1}^{3} j_l k_l / m_l\right),
\end{aligned}$$

for a three-dimensional array $\boldsymbol{g}$ of size $\boldsymbol{m}$, and $0 \le j_l, k_l \le m_l - 1$, $l = 1, 2, 3$.

In our case, we let $\boldsymbol{g}$ be of size $2\boldsymbol{m}$, and

$$\boldsymbol{g_j} = f(x^{j_1}, y^{j_2}, z^{j_3}) \exp(2\pi i(\kappa j_1/m_1 + \lambda j_2/m_2 + \xi j_3/m_3)), \tag{6.2}$$

for $0 \le j_l \le m_l - 1$, $l = 1, 2, 3$, and zero otherwise.

Note that the one-dimensional DFT of a vector of only ones is

$$\mathrm{DFT}_{k_l}(\mathbf{ones}(2m_l); 2m_l) = \sum_{j_l=0}^{2m_l-1} \exp(-2\pi i(j_l k_l / m_l)) = 2m_l \delta_{k_l 0},$$

for $0 \le k_l \le 2m_l - 1$. Hence, the product

$$\begin{aligned}
\mathrm{DFT}_{\boldsymbol{k}}(\boldsymbol{g_j}; 2\boldsymbol{m}) \prod_{l=1}^{3} \mathrm{DFT}_{k_l}(\mathbf{ones}(2m_l); 2m_l) \\
= 8m_1 m_2 m_3\ \mathrm{DFT}_{000}(\boldsymbol{g_j}; 2\boldsymbol{m})\ \delta_{k_1 0} \delta_{k_2 0} \delta_{k_3 0}
\end{aligned}$$

becomes nonzero only for $\boldsymbol{k} = (0, 0, 0)$. This greatly simplifies the theory, because

$$\mathrm{DFT}_{\boldsymbol{k}}^{-1}(8m_1m_2m_3\ \mathrm{DFT}_{000}(\boldsymbol{g_j};2\boldsymbol{m})\ \delta_{k_10}\delta_{k_20}\delta_{k_30}\cdot\mathbf{ones}(2\boldsymbol{m});2\boldsymbol{m})=\mathrm{DFT}_{000}(\boldsymbol{g_j};2\boldsymbol{m}),$$

i.e. a constant, independent of $\boldsymbol{k}$.

In our notation, the integral in [11, Eq. (4.1)] reads

$$\int_{-c}^{c}\int_{-b}^{b}\int_{-a}^{a} f(\boldsymbol{x})\ \exp(i\ [\pi\kappa/a,\pi\lambda/b,\pi\xi/c]\cdot\boldsymbol{x})\ \mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z,$$

for $r=s=1$.

It then follows from the above simplifications and [11, Eq. (4.8)] that

$$\begin{aligned}
&\int_\Omega f(\boldsymbol{x})\ \exp(\pi i(\kappa x/a+\lambda y/b+\xi z/c))\ \mathrm{d}\boldsymbol{x}\\
&\qquad=\int_{-c}^{c}\int_{-b}^{b}\int_{-a}^{a} f(\boldsymbol{x})\ \exp(i\ [\pi\kappa/a,\pi\lambda/b,\pi\xi/c]\cdot\boldsymbol{x})\ \mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z\\
&\qquad\simeq\frac{8abc}{m_1m_2m_3}\ \exp(i(\tilde{a}_1\pi\kappa/a_1+\tilde{a}_2\pi\lambda/a_2+\tilde{a}_3\pi\xi/a_3))\,\mathrm{DFT}_{000}(\boldsymbol{g_j};2\boldsymbol{m})\\
&=\frac{8abc}{m_1m_2m_3}\ \exp(\pi i((1/m_1-1)\kappa+(1/m_2-1)\lambda+(1/m_3-1)\xi))\,\mathrm{DFT}_{000}(\boldsymbol{g_j};2\boldsymbol{m}).
\end{aligned}$$

By Eq. (6.2),

$$\begin{aligned}
\mathrm{DFT}_{000}(\boldsymbol{g_j};2\boldsymbol{m})&=\sum_{j_1=0}^{2m_1-1}\sum_{j_2=0}^{2m_2-1}\sum_{j_3=0}^{2m_3-1}\boldsymbol{g_j}\\
&=\sum_{j_1=0}^{m_1-1}\sum_{j_2=0}^{m_2-1}\sum_{j_3=0}^{m_3-1} f(x^{j_1},y^{j_2},z^{j_3})\ \exp(2\pi i(\kappa j_1/m_1+\lambda j_2/m_2+\xi j_3/m_3))\\
&=\sum_{j_1=0}^{m_1-1}\sum_{j_2=0}^{m_2-1}\sum_{j_3=0}^{m_3-1} f(x^{j_1},y^{j_2},z^{j_3})\ \exp(-2\pi i(j_1k_1/m_1+j_2k_2/m_2+j_3k_3/m_3))\\
&=\mathrm{DFT}_{\boldsymbol{k}}(f;\boldsymbol{m}),
\end{aligned}$$

where $\boldsymbol{k}$ is given by

$$k_l = \begin{cases} -\kappa_l, & \kappa_l \leq 0, \\ m_l - \kappa_l, & \kappa_l > 0 \end{cases}, \tag{6.3}$$

for $l = 1, 2, 3$. Combining the last three equations, we finally get

$$\begin{aligned} &\int_\Omega f(\boldsymbol{x}) \exp(\pi i(\kappa x/a + \lambda y/b + \xi z/c)) \, \mathrm{d}\boldsymbol{x} \\ &= \frac{8abc}{m_1 m_2 m_3} \exp(\pi i((1/m_1 - 1)\kappa + (1/m_2 - 1)\lambda + (1/m_3 - 1)\xi)) \, \mathrm{DFT}_{\boldsymbol{k}}(f; \boldsymbol{m}), \end{aligned} \tag{6.4}$$

i.e. we can use the three-dimensional FFT to evaluate the integrals, as promised in Section. 5.1.3. The FFT simultaneously calculates the DFT for all values of $\boldsymbol{k}$, so *all* the integrals in $\boldsymbol{A}_n$ and $\boldsymbol{e}_n$ can be calculated very fast.

Remark: Note that to ensure that all k_l in Eq. (6.3) are nonnegative, we need each m_l to be greater than the maximum value of κ_l (i.e. N), or, in the case of the integrals in $\boldsymbol{M}_n$ (given in Eq. (5.20)), greater than the maximum value of $\alpha_l - \kappa_l$ (i.e. $2N$). In addition, we should choose each m_l equal to a power of 2, since this is the optimal choice for the FFT. ■

The formula (6.3)-(6.4) is implemented in Code A.2.

6.1.2 Solving the linear system

We now turn our attention to solving Eq. (5.23), using the properties mentioned in Theorem 5.4.1.

Since $\boldsymbol{A}_n$ is HPD, we could solve Eq. (5.23) as it stands, using the complex Cholesky factorization. However, we can do better.

Let $\boldsymbol{Q}_k$ be the $k \times k$ matrix in [15, Eq. (2.2)], defined by

$$\boldsymbol{Q}_k = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{I}_m & i\boldsymbol{I}_m \\ \boldsymbol{J}_m & -i\boldsymbol{J}_m \end{bmatrix}, \text{ for } k = 2m,$$
$$\boldsymbol{Q}_k = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{I}_m & 0 & i\boldsymbol{I}_m \\ 0 & \sqrt{2} & 0 \\ \boldsymbol{J}_m & 0 & -i\boldsymbol{J}_m \end{bmatrix}, \text{ for } k = 2m+1.$$

This matrix is unitary, i.e. $\boldsymbol{Q}_k \boldsymbol{Q}_k^H = \boldsymbol{I}_k$, so we can rewrite Eq. (5.23) as

$$\left(\boldsymbol{Q}_{8N^3}^H \boldsymbol{A}_n \boldsymbol{Q}_{8N^3}\right)\left(\boldsymbol{Q}_{8N^3}^H(\Theta_n \boldsymbol{w})\right) = \boldsymbol{Q}_{8N^3}^H\left(\boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n + \boldsymbol{e}_n/2\right).$$

The matrix

$$\boldsymbol{R}_n = \boldsymbol{Q}_{8N^3}^H \boldsymbol{A}_n \boldsymbol{Q}_{8N^3}$$

is real, according to [15]. Since $\boldsymbol{Q}_1 = 1$, the vectors

$$\begin{aligned} \boldsymbol{y}_n &= \boldsymbol{Q}_{8N^3}^H(\Theta_n \boldsymbol{w}) = \boldsymbol{Q}_{8N^3}^H(\Theta_n \boldsymbol{w})\, \boldsymbol{Q}_1 \\ \text{and } \boldsymbol{z}_n &= \boldsymbol{Q}_{8N^3}^H\left(\boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n + \boldsymbol{e}_n/2\right) = \boldsymbol{Q}_{8N^3}^H\left(\boldsymbol{G}^H\boldsymbol{G}\boldsymbol{w}_n + \boldsymbol{e}_n/2\right) \boldsymbol{Q}_1 \end{aligned}$$

are real as well. This implies that we can solve the real system

$$\boldsymbol{R}_n \boldsymbol{y}_n = \boldsymbol{z}_n$$

instead of Eq. (5.23), and then obtain $\boldsymbol{w}_{n+1}$ from

$$\boldsymbol{w}_{n+1} = 2\boldsymbol{Q}_{8N^3}\boldsymbol{y}_n - \boldsymbol{w}_n.$$

Moreover, since $\boldsymbol{Q}_{8N^3}$ is unitary, $\boldsymbol{R}_n$ retains the symmetry and positive definiteness of $\boldsymbol{A}_n$, so we can use the real Cholesky factorization, which is much faster than the complex one.

6.1.3 Accelerating convergence

To achieve an acceptable level of accuracy, we need to use a certain number of basis functions v_{pqr} to represent our solution. The number of basis functions grows as $(2N+1)^3$, so it quickly becomes so large that the memory costs and time usage become unacceptable. Fortunately, we can reduce the number of basis functions necessary to obtain the desired accuracy, by accelerating the convergence of the Fourier expansion of the solution. We do this by using a three-dimensional generalization of the one-dimensional Lanczos sigma filter in [14, pp. 65-75].

The ordinary, infinite Fourier series of a function f is given by

$$f(x,y,z) = \sum_{p,q,r=-\infty}^{\infty} f_{pqr} \exp(\pi i(px/a + qy/b + rz/c)) \,. \tag{6.5}$$

Of course, we must restrict ourselves to a finite approximation of this series. The default alternative is to use the truncated Fourier series:

$$f(x,y,z) \approx f^N(x,y,z) = \sum_{p,q,r=-N}^{N} f_{pqr} \exp(\pi i(px/a + qy/b + rz/c)) \,,$$

but the graph of this series will typically oscillate vigorously, and display significant overshoots and undershoots compared to the graph of f. However, we can reduce this unwanted effect by replacing f^N with the function $f_k^N, k > 0$, averaged over $(x-a/k, x+a/k) \times (y-b/k, y+b/k) \times (z-c/k, z+c/k)$:

$$\begin{aligned} f_k^N(x,y,z) &= \frac{k^3}{8abc} \int_{-c/k}^{c/k} \int_{-b/k}^{b/k} \int_{-a/k}^{a/k} f^N(x+\alpha, y+\beta, z+\zeta) \, \mathrm{d}\alpha \, \mathrm{d}\beta \, \mathrm{d}\zeta \\ &= \sum_{p,q,r=-N}^{N} \frac{k^3}{8abc} \prod_{n=1}^{3} \left(\int_{-a_l/k}^{a_l/k} \exp(\pi i(p_l(x_l+\alpha_l)/a_l)) \, \mathrm{d}\alpha_l \right) f_{pqr} \\ &= \sum_{p,q,r=-N}^{N} \frac{k^3}{8abc} \prod_{n=1}^{3} \left(\exp(\pi i(p_l x_l/a_l)) \int_{-a_l/k}^{a_l/k} \exp(\pi i(p_l \alpha_l/a_l)) \, \mathrm{d}\alpha_l \right) f_{pqr}. \end{aligned}$$

Since

$$\int_{-a_l/k}^{a_l/k} \exp(\pi i(p_l\alpha_l/a_l))\ \mathrm{d}\alpha_l = \frac{2a_l}{k}\ \delta_{p_l0} + \left(\frac{\exp(\pi i(p_l/k)) - \exp(-\pi i(p_l/k))}{\pi i p_l/a_l}\right)\delta_{p_l\neq 0}$$
$$= \frac{2a_l}{k}\ \left(\delta_{p_l0} + \frac{k}{\pi p_l}\left(\frac{\exp(\pi i(p_l/k)) - \exp(-\pi i(p_l/k))}{2i}\right)\delta_{p_l\neq 0}\right)$$
$$= \frac{2a_l}{k}\left(\delta_{p_l0} + \frac{k}{\pi p_l}\ \sin\Big(\frac{\pi p_l}{k}\Big)\,\delta_{p_l\neq 0}\right)$$
$$= \frac{2a_l}{k}\left(\delta_{p_l0} + \mathrm{sinc}\Big(\frac{\pi p_l}{k}\Big)\,\delta_{p_l\neq 0}\right) = \frac{2a_l}{k}\ \mathrm{sinc}\Big(\frac{\pi p_l}{k}\Big)\,,$$

for each l, it follows that

$$f_k^N(x,y,z) = \sum_{p,q,r=-N}^{N} f_{pqr}^k\ \exp(\pi i(px/a + qy/b + rz/c))\,,$$

where the Fourier coefficients of f_k^N are

$$f_{pqr}^k = \mathrm{sinc}\Big(\frac{\pi p}{k}\Big)\,\mathrm{sinc}\Big(\frac{\pi q}{k}\Big)\,\mathrm{sinc}\Big(\frac{\pi r}{k}\Big)\,f_{pqr} = \sigma_{pqr}^k f_{pqr}.$$

The factors

$$\sigma_{pqr}^k = \mathrm{sinc}\Big(\frac{\pi p}{k}\Big)\,\mathrm{sinc}\Big(\frac{\pi q}{k}\Big)\,\mathrm{sinc}\Big(\frac{\pi r}{k}\Big)$$

are known as *Lanczos sigma factors*. As expected, when $k \to \infty$,

$$\sigma_{pqr}^k \to 1,\ \text{and thus } f_{pqr}^k \to f_{pqr} \text{ and } f_k^N(x,y,z) \to f^N(x,y,z).$$

When $|p|, |q|, |r| \leq k-1$, we see that $\sigma_{pqr}^k \in (0,1]$. Hence,

$$\left|f_{pqr}^k\right| \leq |f_{pqr}|\,, \tag{6.6}$$

i.e. the sigma factors reduce the amplitudes of the complex wave functions, making the graph of f^N smoother. We therefore choose $k = N+1$, and use the approximation

$$f(x,y,z) \approx \sum_{p,q,r=-N}^{N} f_{pqr}^{N+1}\ \exp(\pi i(px/a + qy/b + rz/c))\,.$$

Because the original series (6.5) converges, Eq. (6.6) implies that this approximation converges when $N \to \infty$ (and often faster than the original series). Since the sigma-filtered series is just another Fourier series, we can apply the filter again (as many times as we wish). The more times the filter is applied, the smoother the graph of the approximation, but at the expense of a larger and larger deviation from f in areas where f changes rapidly (cf. [14, Figure 10, p. 68] in the one-dimensional case).

6.1.4 Calculating point values at grid points

We now turn to the problem of efficient calculation of point values from Fourier coefficients. We use the grid points $(x^{j_1}, y^{j_2}, z^{j_3})$, given by Eq. (6.1), for $0 \le j_l \le m_l - 1$, $l = 1, 2, 3$.

Given the coefficients $\boldsymbol{w}_n$, we must efficiently calculate $W_n(x^{j_1}, y^{j_2}, z^{j_3})$ for each grid point $(x^{j_1}, y^{j_2}, z^{j_3})$. By Eqs. (5.12) and (5.4),

$$\begin{aligned} W_n(x^{j_1}, y^{j_2}, z^{j_3}) &= \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{n,pqr} v_{pqr}(x^{j_1}, y^{j_2}, z^{j_3}) \\ &= \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{n,pqr} \left(\sum_{\alpha,\beta,\zeta=-N}^{N} v_{pqr}^{\alpha\beta\zeta} \exp\bigl(\pi i(\alpha x^{j_1}/a + \beta y^{j_2}/b + \zeta z^{j_3}/c)\bigr) \right) \\ &= \sum_{\alpha,\beta,\zeta=-N}^{N} \left(\sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{n,pqr} v_{pqr}^{\alpha\beta\zeta} \right) \exp\bigl(\pi i(\alpha x^{j_1}/a + \beta y^{j_2}/b + \zeta z^{j_3}/c)\bigr) . \end{aligned}$$

If we set

$$f_{\alpha\beta\zeta} = \sum_{\substack{p,q,r=-N \\ p,q,r\neq 0}}^{N} w_{n,pqr} v_{pqr}^{\alpha\beta\zeta} = (\boldsymbol{G}\boldsymbol{w}_n)(\psi(\alpha, \beta, \zeta, N)),$$

we see that we get a finite Fourier expansion for W_n:

$$W_n(x^{j_1}, y^{j_2}, z^{j_3}) = \sum_{\alpha,\beta,\zeta=-N}^{N} f_{\alpha\beta\zeta} \exp\bigl(\pi i(\alpha x^{j_1}/a + \beta y^{j_2}/b + \zeta z^{j_3}/c)\bigr) .$$

In practice, we will also apply the sigma filter $\sigma_{\alpha\beta\zeta}^{N+1}$ m times, so we get

$$W_n(x^{j_1}, y^{j_2}, z^{j_3}) = \sum_{\alpha,\beta,\zeta=-N}^{N} f_{\alpha\beta\zeta}^{N+1,m} \exp\bigl(\pi i(\alpha x^{j_1}/a + \beta y^{j_2}/b + \zeta z^{j_3}/c)\bigr), \tag{6.7}$$

where

$$f_{\alpha\beta\zeta}^{N+1,m} = (\sigma_{\alpha\beta\zeta}^{N+1})^m f_{\alpha\beta\zeta} \text{ is element } \psi(\alpha,\beta,\zeta,N) \text{ of the vector } \boldsymbol{S}_N^m \boldsymbol{G}\boldsymbol{w}_n,$$

for the $(2N+1)^3 \times (2N+1)^3$ diagonal matrix $\boldsymbol{S}_N$, defined by

$$S(\psi(\kappa,\lambda,\xi,N), \psi(\alpha,\beta,\zeta,N)) = \sigma_{\alpha\beta\zeta}^{N+1} \delta_{\alpha\kappa}\delta_{\beta\lambda}\delta_{\zeta\xi}.$$

Next, we rewrite Eq. (6.7) to a more compact form,

$$W_n(x^{j_1}, y^{j_2}, z^{j_3}) = \sum_{\alpha,\beta,\zeta=-N}^{N} f_{\alpha\beta\zeta}^{N+1,m} \exp\left(\pi i \sum_{l=1}^{3} \alpha_l x_l^{j_l}/a_l\right), \tag{6.8}$$

and use Eq. (6.1) to split the exponentials:

$$\begin{aligned}
\exp\left(\pi i \sum_{l=1}^{3} \alpha_l x_l^{j_l}/a_l\right) &= \exp\left(-N\pi i \sum_{l=1}^{3} x_l^{j_l}/a_l\right) \exp\left(\pi i \sum_{l=1}^{3} (\alpha_l + N) x_l^{j_l}/a_l\right) \\
&= \exp\left(-N\pi i \sum_{l=1}^{3} x_l^{j_l}/a_l\right) \exp\left(\pi i \sum_{l=1}^{3} [(\alpha_l + N)((2j_l+1)/m_l - 1)]\right) = \\
\exp\left(-N\pi i \sum_{l=1}^{3} \frac{x_l^{j_l}}{a_l}\right) &\exp\left(\pi i \sum_{l=1}^{3} \frac{\alpha_l + N}{m_l}(1 - m_l)\right) \exp\left(2\pi i \sum_{l=1}^{3} \frac{\alpha_l + N}{m_l} j_l\right).
\end{aligned} \tag{6.9}$$

We set

$$g_{(\alpha+N)(\beta+N)(\zeta+N)}^{N+1,m} = f_{\alpha\beta\zeta}^{N+1,m} \exp\left(\pi i \sum_{l=1}^{3} (\alpha_l + N)(1/m_l - 1)\right), \tag{6.10}$$

if $-N \le \alpha_l \le N$, $l = 1, 2, 3$, and zero otherwise.

Let $k_l = \alpha_l + N$. Then Eq. (6.10) becomes

$$g^{N+1,m}_{k_1k_2k_3} = f^{N+1,m}_{\alpha\beta\zeta} \exp\left(\pi i \sum_{l=1}^{3} (1/m_l - 1)k_l\right), \tag{6.11}$$

if $0 \leq k_l \leq 2N$, $l = 1, 2, 3$, and zero otherwise.

Inserting Eqs. (6.9) and (6.11) in Eq. (6.8) results in

$$\begin{aligned} &W_n(x^{j_1}, y^{j_2}, z^{j_3}) \\ &\quad = \exp\left(-N\pi i \sum_{l=1}^{3} x_l^{j_l}/a_l\right) \sum_{\alpha,\beta,\zeta=-N}^{N} g^{N+1,m}_{k_1k_2k_3} \exp\left(2\pi i \sum_{l=1}^{3} (\alpha_l + N) j_l/m_l\right) \\ &\quad = \exp\left(-N\pi i \sum_{l=1}^{3} x_l^{j_l}/a_l\right) \sum_{k_1,k_2,k_3=0}^{2N} g^{N+1,m}_{k_1k_2k_3} \exp\left(2\pi i \sum_{l=1}^{3} j_l k_l/m_l\right). \end{aligned}$$

We can extend the upper limit of k_l from $2N$ to $m_1 - 1$, because $g^{N+1,m}_{k_1k_2k_3}$ is zero if one or more k_l are larger than $2N$. Consequently,

$$\begin{aligned} &W_n(x^{j_1}, y^{j_2}, z^{j_3}) \\ &\quad = \exp\left(-N\pi i \sum_{l=1}^{3} x_l^{j_l}/a_l\right) \sum_{k_1=0}^{m_1-1} \sum_{k_2=0}^{m_2-1} \sum_{k_3=0}^{m_3-1} g^{N+1,m}_{k_1k_2k_3} \exp\left(2\pi i \sum_{l=1}^{3} j_l k_l/m_l\right) \\ &\quad = m_1 m_2 m_3 \exp\left(-N\pi i \left(x^{j_1}/a + y^{j_2}/b + z^{j_3}/c\right)\right) \mathrm{DFT}^{-1}_{j_1,j_2,j_3}(\boldsymbol{g}^{N+1,m}_{\boldsymbol{k}}; \boldsymbol{m}), \end{aligned}$$

so all $W_n(x^{j_1}, y^{j_2}, z^{j_3})$ can be calculated simultaneously, using the three-dimensional inverse FFT. This is done in Code A.3.

6.1.5 Plotting the results

We must be able to display the solution in the interior of the product. This can be done by defining a plane that cuts through the product, thereby defining a cross-sectional surface. The plane can be defined by one of its points, $\boldsymbol{x}_0 = (x_0, y_0, z_0)$ (which positions it in space) and a normal vector $\boldsymbol{n} = (n_1, n_2, n_3)$ (which orients it, by pointing upwards from the plane). The plane then consists of the points satisfying

$$f(x, y, z) = \boldsymbol{n} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) = n_1(x - x_0) + n_2(y - y_0) + n_3(z - z_0) = 0.$$

If $I_{\Omega_1} = 0$ defines the product surface, plotting the points that satisfy

$$\min\{I_{\Omega_1}(x, y, z), f(x, y, z)\} = 0, \tag{6.12}$$

ensures that:

- The surroundings are not plotted, because $I_{\Omega_1} < 0$.
- The part of the product that lies below the cross-sectional plane is not plotted, because $f < 0$.
- The interior of the product above the plane is not plotted, because $f, I_{\Omega_1} > 0$.
- The part of the product surface that lies above the cross-sectional plane is plotted, because $I_{\Omega_1} = 0 < f$.
- The intersection of the product and the plane (i.e. the cross-section) is plotted, because $f = 0 \leq I_{\Omega_1}$.

The next section contains plots created by plotting the points that satisfy Eq. (6.12). The plotting is implemented in Code A.1.

6.2 Simulations of cod AFD

To test the usefulness of the framework, we have simulated the AFD experiments on pieces of cod fillet in [5], using background data from those experiments. In this section we present the results and parameter values from those simulations. The results are discussed in Section 7.1.

The physical parameters are slightly different in the two simulations, so they are given separately in Tables 6.2 and 6.3. Some geometrical parameters, such as the radii (r_x, r_y, r_z), are given in the source codes Code A.6 and Code A.7. In both simulations, the product rests on a perforated plate, with circular holes of diameter 3 mm. For the cod slab given by Eq. (2.2), with $p = 10$, $a = b = 0.01$ and $c = 0.0025$, the GDFs for the holes are

$$I_{\text{holes},1}(x,y,z) = 1 - \left(\frac{\sin\left(\frac{\pi x}{0.005}\right)^2}{\sin\left(\frac{\pi 0.0015}{0.005}\right)^2} + \frac{\sin\left(\frac{\pi y}{0.00866}\right)^2}{\sin\left(\frac{\pi 0.0015}{0.00866}\right)^2} + (1.25 + 100z)^{20} \right), \tag{6.13}$$

$$I_{\text{holes},2}(x,y,z) = 1 - \left(\frac{\sin\left(\frac{\pi(x-0.0025)}{0.005}\right)^2}{\sin\left(\frac{\pi 0.0015}{0.005}\right)^2} + \frac{\sin\left(\frac{\pi(y-0.00433)}{0.00866}\right)^2}{\sin\left(\frac{\pi 0.0015}{0.00866}\right)^2} + (1.25 + 100z)^{20} \right). \tag{6.14}$$

The product and the perforated plate are plotted in Figure 6.1. **In the source codes, the GDFs are slightly different, to account for the actual average size of the real samples, including shrinkage.**

The air flow in the simulations is a laminar flow over plane sides, so the Sherwood number is given by [1, Eq. (11.64)]:

$$Sh_y = 0.332\ Sc^{1/3} Re_y^{1/2} = 0.332\ Sc^{1/3} \left(\frac{V_{\text{air}}\ y}{\nu_{\text{air}}} \right)^{1/2},$$

where Sc is the Schmidt number. By [1, Eqs. (11.65)-(11.66)], this implies that

$$h_{\text{m,max}}(y) = \frac{\nu_{\text{air}} Sh_y}{y\ Sc} = 0.332\ Sc^{-2/3} \left(\frac{V_{\text{air}}\ \nu_{\text{air}}}{y} \right)^{1/2} \tag{6.15}$$

in the ambient air. Since $y \in (-b, b)$ can become negative, we use the expression

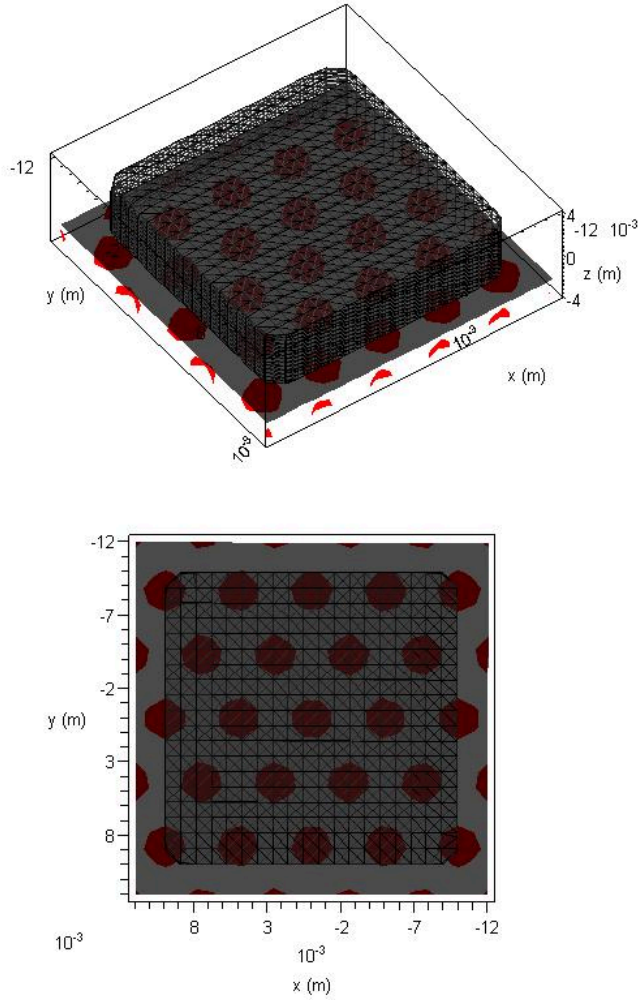


Fig. 6.1: Maple 11 plots of a cod slab on a perforated plate with holes given by Eqs. (6.13)-(6.14).

$$h_{\mathrm{m,max}}(y,t) = 0.332\ Sc^{-2/3} \left(\frac{V_{\mathrm{air}}\ \nu_{\mathrm{air}}}{2b + y - r_y(t)} \right)^{1/2}$$

instead, to avoid taking the square root of a negative number. We set $h_{\mathrm{m}} = 0$ in the metal of the plate, and $h_{\mathrm{m}} = 0.80 \cdot h_{\mathrm{m,max}}$ in the air in the holes (to account for the fact that the thickness of the plate causes the underside of the cod and the airstream beneath the plate to be separated by a small distance). The resulting expression for

h_{m} becomes

$$h_{\mathrm{m}}^k(x,y,z,t) = h_{\mathrm{m,max}}(y,t) \cdot \big[(\chi_{\{z>-r_z(0)\},k}(z) - \chi_{\{I_{\mathrm{prod}}>0\},k}(x,y,z,t)) \\ +0.80 \cdot (\chi_{\{I_{\mathrm{holes},1}>0\},k}(x,y,z) + \chi_{\{I_{\mathrm{holes},2}>0\},k}(x,y,z))\big] .$$

Furthermore, the permeabilities become

$$K_l^k(\boldsymbol{x},t,U(\boldsymbol{x},t)) = \overline{\chi}_{\Omega_{\mathrm{ice}},k}(U(\boldsymbol{x},t))\ \chi_{\{I_{\mathrm{prod}}>0\},k}(\boldsymbol{x},t)\ K_l(\boldsymbol{x},t).$$

For the saturation density of water vapor, we use the Goff-Gratch equation for vapor over pure ice:

$$\rho_{\mathrm{sat}}(T) = \frac{610.71}{R_{\mathrm{H_2O}}T}\left(\frac{273.16}{T}\right)^{-3.56654} \cdot 10^{-9.09718(273.16/T-1)+0.876793(1-T/273.16)},$$

where $R_{\mathrm{H_2O}}$ (= 461.52364 J/(kg·K), [9]) is the specific gas constant for water vapor. We calculate the wet bulb temperature from Eq. (4.10) and γ from Eq. (4.4).

When estimating the mass fraction of ice, we use Table 9.3 and Eq. (9.5) in the *2006 ASHRAE Handbook - Refrigeration*. The initial freezing point for cod is -2.2 °C, and the initial product temperature -20 °C.

Some chosen non-physical parameters are given in Table 6.1.

Parameter	**Matlab name**	**Value**
Geometry approximation parameter	n_app	10^{10}
Ice indication threshold (K_{ice})	ice_level	0.99
Number of basis functions	N	7
Number of grid points	m	[64,64,64]
Relative convection strength, holes	hole_fac	0.80
Semi-axes (a,b,c)	a,b,c	125 % of initial radii
Small parameter (ε)	epsilon	10^{-30}

Tab. 6.1: Chosen parameter values in both -5 °C and -10 °C cod slab simulations.

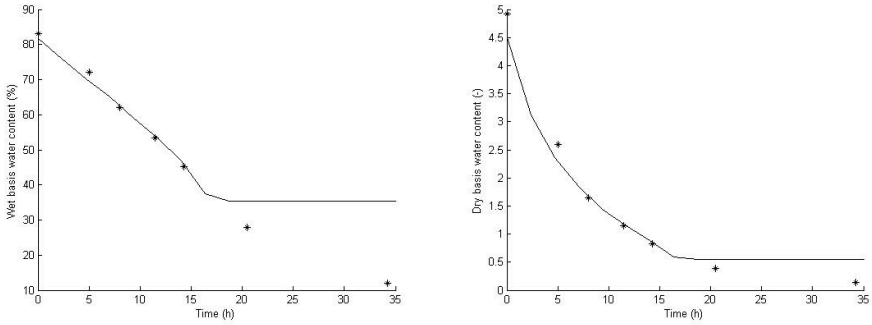
6.2.1 Results for -5 °C

We have simulated 35 h of drying at -5 °C (300 time steps of length 7 min). Physical parameter values for this simulation are given in Table 6.2.

The measured and calculated drying curves are shown in Figure 6.2, and the calculated moisture concentration evolution is shown in Figures 6.3-6.5.

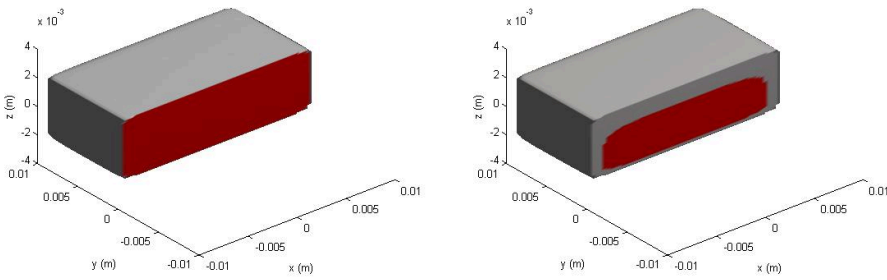
Parameter	**Matlab name**	**Value**	**Source**
Air velocity	V_a	3.3 m/s	[5]
Ambient relative humidity	RH	0.40 1	[5]
Ambient temperature	T_surr	268.15 K	[5]
c_p/R	c_p_R	4.0262 1	[7, Table 2] (estimated)
$(\gamma P_0 K_1)/(\mu_{vap}\rho_{vap,0}^{\gamma})$ $(\gamma P_0 K_2)/(\mu_{vap}\rho_{vap,0}^{\gamma})$ $(\gamma P_0 K_3)/(\mu_{vap}\rho_{vap,0}^{\gamma})$	C_fac_1 C_fac_2 C_fac_3	0.037 $m^{3\gamma+2}/(s\ kg^{\gamma})$	fitted value
Initial total mass	m_init	$1.886 \cdot 10^{-3}$ kg	[5]
Kinematic viscosity, air	nu_a	$1.2883 \cdot 10^{-5}$ m^2/s	[12, Table A-1] (estimated)
Latent heat of sublimation	Delta_H_sub	2835607 J/kg	[9, Table 4] (estimated)
Porosity	phi	0.5583 1	[5, Table 1]
Prandtl number	Pr	0.7185 1	[12, Table A-1] (estimated)
Schmidt number	Sc	0.60 1	[12, Table A-18]
Thermal conductivity, air	k_a	0.023687 W/(m·K)	[12, Table A-1] (estimated)

Tab. 6.2: Physical parameter values in -5 °C cod slab simulations.

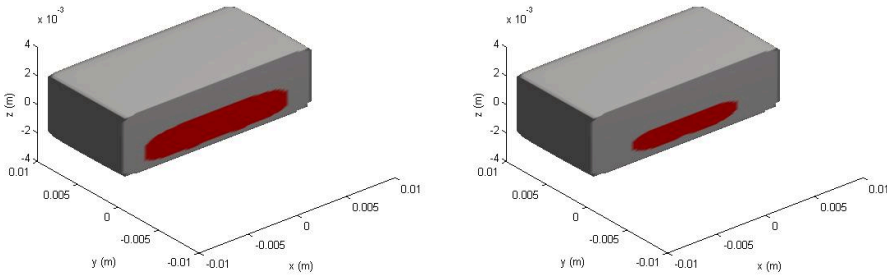


(a) Wet basis drying curve (b) Dry basis drying curve

Fig. 6.2: The measured (points) and calculated (curve) drying curves for -5 °C.

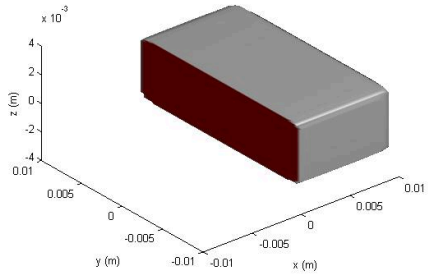


(a) Initially (b) After 4 h 40 min

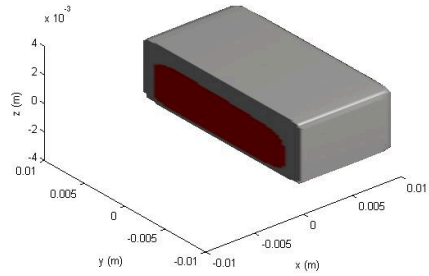


(c) After 9 h 20 min (d) After 14 h

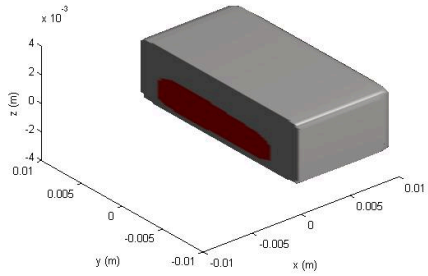
Fig. 6.3: The calculated moisture concentration profile at different times for -5 °C. The cross-section shown is defined by $\boldsymbol{x}_0 = (0, 0, 0)$ and $\boldsymbol{n} = (0, 1, 0)$.



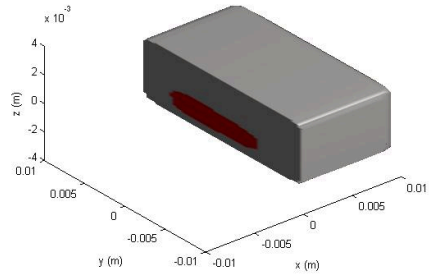
(a) Initially



(b) After 4 h 40 min

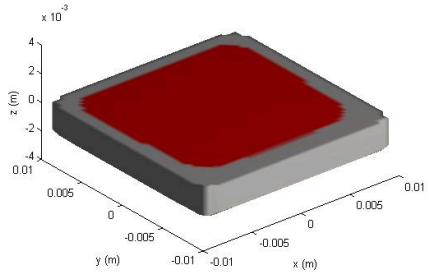


(c) After 9 h 20 min

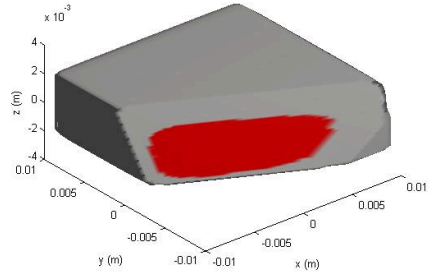


(d) After 14 h

Fig. 6.4: The calculated moisture concentration profile at different times for -5 °C. The cross-section shown is defined by $\boldsymbol{x}_0 = (0, 0, 0)$ and $\boldsymbol{n} = (1, 0, 0)$.



(a) $\boldsymbol{x}_0 = (0, 0, 0), \boldsymbol{n} = (0, 0, -1)$



(b) $\boldsymbol{x}_0 = (0, -0.5r_y, 0), \boldsymbol{n} = (1, 2, -2)$

Fig. 6.5: The calculated moisture concentration profile after 4 h 40 min for -5 °C. Alternative cross-sections.

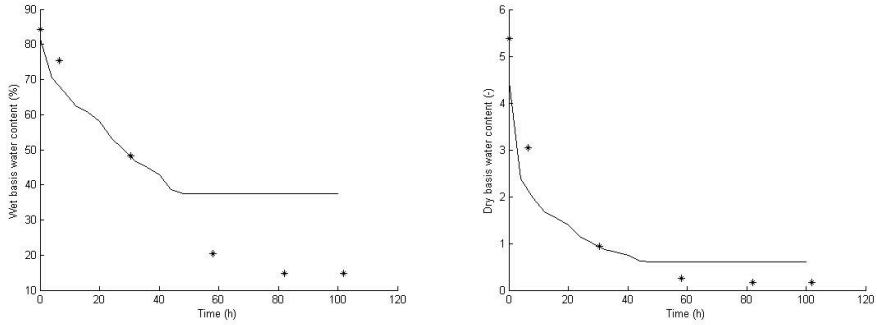
6.2.2 Results for -10 °C

We have simulated 100 h of drying at -10 °C (450 time steps of length 13 min 20 s). Physical parameter values for this simulation are given in Table 6.3.

The measured and calculated drying curves are shown in Figure 6.6, and the calculated moisture concentration evolution is shown in Figures 6.7 and 6.8.

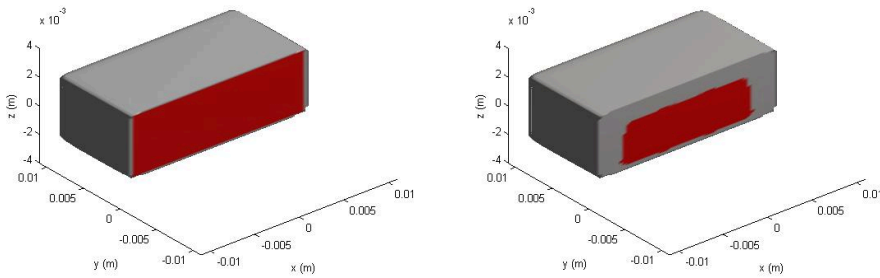
Parameter	**Matlab name**	**Value**	**Source**
Air velocity	V_a	3.3 m/s	[5]
Ambient relative humidity	RH	0.40 1	[5]
Ambient temperature	T_surr	263.15 K	[5]
c_p/R	c_p_R	4.0246 1	[7, Table 2] (estimated)
$(\gamma \mathrm{P_0 K_1})/(\mu_\mathrm{vap}\rho_\mathrm{vap,0}^\gamma)$ $(\gamma \mathrm{P_0 K_2})/(\mu_\mathrm{vap}\rho_\mathrm{vap,0}^\gamma)$ $(\gamma \mathrm{P_0 K_3})/(\mu_\mathrm{vap}\rho_\mathrm{vap,0}^\gamma)$	C_fac_1 C_fac_2 C_fac_3	0.013 $\mathrm{m}^{3\gamma+2}/(\mathrm{s\ kg}^\gamma)$	fitted value
Initial total mass	m_init	$2.180 \cdot 10^{-3}$ kg	[5]
Kinematic viscosity, air	nu_a	$1.2453 \cdot 10^{-5}$ m^2/s	[12, Table A-1] (estimated)
Latent heat of sublimation	Delta_H_sub	2836633 J/kg	[9, Table 4] (estimated)
Porosity	phi	0.7531 1	[5, Table 1]
Prandtl number	Pr	0.72 1	[12, Table A-1] (estimated)
Schmidt number	Sc	0.60 1	[12, Table A-18]
Thermal conductivity, air	k_a	0.023292 W/(m·K)	[12, Table A-1] (estimated)

Tab. 6.3: Physical parameter values in -10 °C cod slab simulations.

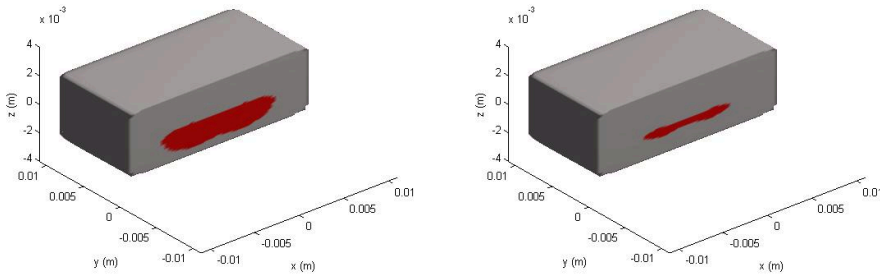


(a) Wet basis drying curve (b) Dry basis drying curve

Fig. 6.6: The measured (points) and calculated (curve) drying curves for -10 °C.

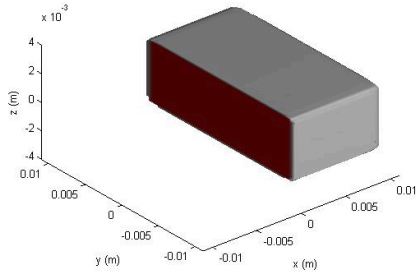


(a) Initially (b) After 8 h

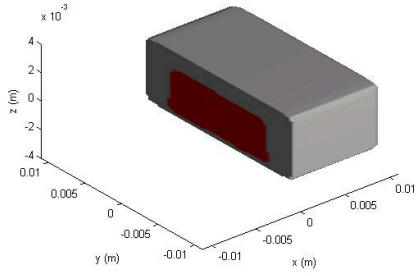


(c) After 24 h (d) After 40 h

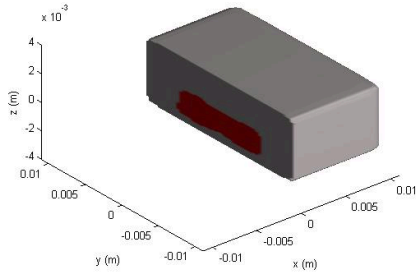
Fig. 6.7: The calculated moisture concentration profile at different times for -10 °C. The cross-section shown is defined by $\boldsymbol{x}_0 = (0, 0, 0)$ and $\boldsymbol{n} = (0, 1, 0)$.



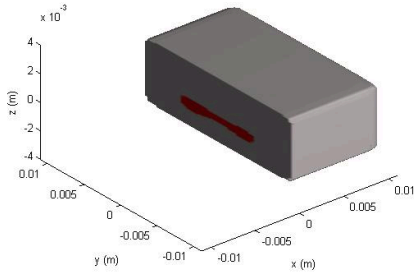
(a) Initially



(b) After 8 h



(c) After 24 h



(d) After 40 h

Fig. 6.8: The calculated moisture concentration profile at different times for -10 °C. The cross-section shown is defined by $\boldsymbol{x}_0 = (0, 0, 0)$ and $\boldsymbol{n} = (1, 0, 0)$.

7. DISCUSSION

In this chapter we discuss different aspects of the framework.

7.1 Simulation results

In the simulations in Chapter 6, the drying curves were fitted to the experimental data by adjusting the parameters

$$\text{C\_fac\_1} = \frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} K_1,$$
$$\text{C\_fac\_2} = \frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} K_2,$$
$$\text{C\_fac\_3} = \frac{\gamma P_0}{\mu_{\text{vap}} \rho_{\text{vap},0}^{\gamma}} K_3,$$

for which we chose the constant values 0.037 (in the -5 °C case) and 0.013 (in the -10 °C case). The resulting diffusivity values became $3.54 \cdot 10^{-11}$ and $3.22 \cdot 10^{-12}$ m^2/s, respectively.

7.1.1 General comments

We observe the following from the figures in Section 6.2:

- We were able to fit the calculated drying curves quite well to the experimental data, down to between 30 and 40 % wet basis water content. At this point, the calculated drying curves stopped decreasing, because all of the ice was gone, and the framework does not model drying of bound water. The initial water contents were somewhat underestimated (see below).

- The calculated drying curves are somewhat jagged in the -10 °C case. This is a result of the mechanism with which we model the movement of the ice front. We have defined the location of the ice front to be where the indicator function $\widehat{\rho}_{\text{ice}}$ intersects a threshold $K_{\text{ice}}\widehat{\rho}_{\text{ice},0}$ (cf. Section 4.2.3). The evolution of the indicator function as time goes by is shown in Figure 7.1. In each of the two figures:

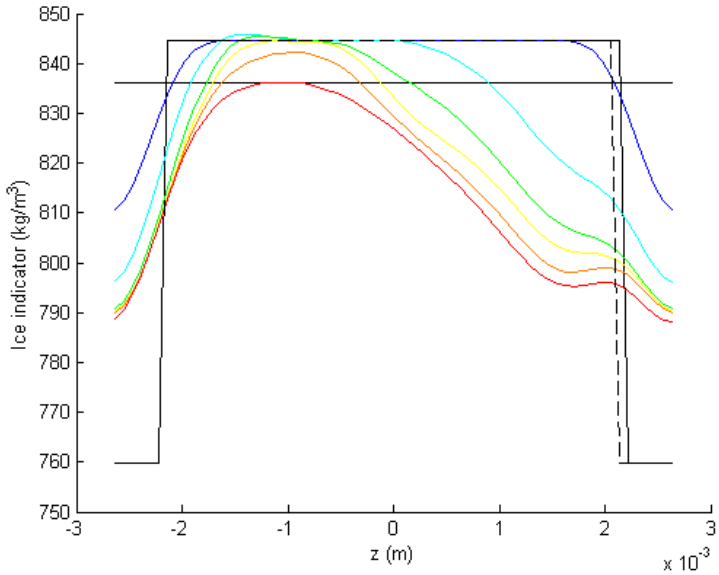
  - The figure shows the value of the ice indicator $\widehat{\rho}_{\text{ice}} = U + \omega\ \rho_{\text{vap,air}}$ along the z axis, i.e. from $\boldsymbol{x} = (0, 0, -c)$ to $\boldsymbol{x} = (0, 0, c)$. The different curves represent different times, from blue (initially) down to red.
  - The solid black square curve is the initial curve $U_0 + \omega\ \rho_{\text{vap,air}}$. It represents the initial surface of the product.
  - The dashed black square curve represents the surface of the product at the last plotted time (i.e. corresponding to the red curve).
  - The horizontal black line is the ice threshold $K_{\text{ice}}\widehat{\rho}_{\text{ice},0}$. The ice is where the ice indicator is above this line.

  We see that as time goes by, the indicator function works its way inwards, representing the retreat of the ice front. The blue curve should approximate the black square curve, but we see that there is a significant round-off error in this approximation. Ideally, the blue curve should intersect the threshold at the product surface, but the intersection takes place just inside the surface in practice. **It is this round-off effect that causes the initial water contents to be underestimated.** Adjusting the blue curve involves finding a good combination of domain dimensions (a, b, c) and number of sigma filtrations.

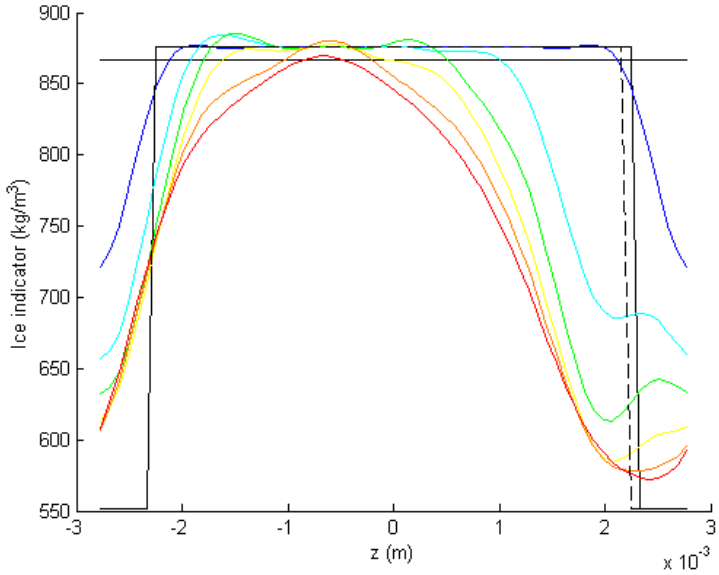
  Increasing the number of basis functions will of course reduce this problem, but this is a very expensive measure, in terms of memory costs.

- The calculated moisture concentration profiles seem reasonable. The evolution of the profiles agree qualitatively with the SEM images in Section 3.2.3, but the growth of the dry region is faster in the simulations than in the images. Of course, the quantitative aspects of the SEM images are very uncertain, since only a few samples were studied. We do however note that the calculated drying curves in Figure 6.2 suggest that the ice disappeared at between 16 and 18 h of drying. The SEM images suggest that this happened at some time between 14 h 15 min and 20 h 30 min.

- The plotting procedure in Section 6.1.5 works well.



(a) -5 °C, three times sigma filtered



(b) -10 °C, twice sigma filtered

Fig. 7.1: Indicator function curves along the z axis at different times.

We have also tested the effect of changing just one parameter in the above simulation (to test the qualitative properties of the framework). The results are shown in Table 7.1. Note that in the case of increased ambient temperature, only the temperature itself was altered, although increasing it in reality changes the values of most of the other physical parameters as well, as shown in Tables 6.2 and 6.3. Note that the large difference in the fitted diffusivity factors C_fac_1, C_fac_2 and C_fac_3 indicate that the diffusivity depends significantly on the ambient temperature.

Change in single parameter	**Change in drying rate**
Increased ambient temperature	Marginal increase
Decreased ambient humidity	Significant increase
Increased air velocity	Moderate increase
Increased diffusivity	Significant increase
Increased hole diameter	Moderate increase
Decreased hole convection strength	Marginal decrease
Increased final porosity	Significant increase

Tab. 7.1: The effect of altering one parameter at a time in the -5 °C simulation.

7.1.2 Numerical stability

The results in Table 7.1 confirm that the framework reacts correctly to moderate changes in single parameter values. But if we change some of the parameters too much, we end up in trouble. A too high increase in ambient humidity, or a too high decrease in porosity, causes the boundary value (4.16) to become larger than the initial ice concentration, causing U to become negative and making the framework produce incorrect results.

Although the framework is in principle independent of the choice of numerical method, the chosen method has had a major influence on its design. It turns out that this numerical method has a low tolerance for unfavorable parameter values. This makes it difficult to construct a robust and reliable framework, so we have had to make certain sacrifices (cf. the next section).

There are a few alternatives to the numerical methods in [2]-[3], but most of them use the weak formulation of the equation system, so they end up with a scheme that is similar to ours. Perhaps a finite difference method, such as the one proposed in [20], would work better.

7.2 Other issues

Before we conclude, we briefly consider three other issues:

7.2.1 Modeling of temperature

The drying process is clearly a coupled heat and mass transfer problem, so we should definitely model heat transfer. Unfortunately, we have not been able to do this in a satisfactory manner. There are two reasons for this:

1. It has turned out to be very difficult to model the sublimation heat sink at the ice front by equations that fit the system in [2]-[4]. The main cause for this problem is the fact that the front moves.
2. It turns out that if we try to model thermal convection in the same manner as we have modeled mass convection, we get divergence. The numerical method is simply not able to handle the heat flux, and there is a very simple reason why. The basis functions are simply too few to be able to represent very small temperature gradients. So, when heat is supplied to the product surface, the basis functions are not able to sufficiently even out the temperature difference, causing it (and thus the heat flux) to remain artificially high. As a result, the local temperature starts to oscillate more and more, and get out of control.

The modeling of mass convection works well with the current framework, because the convective mass transfer coefficient h_{m}^k becomes divided by the large parameter ω (Eq. (4.19)). This effectively works as a very strong natural underrelaxation. Without this effect, we would have similar problems with the mass transfer as with the heat transfer. Stabilizing the thermal convection requires either a very strong artificial underrelaxation or a miniscule time step.

The omission of local temperature modeling forces us to make assumptions on the temperature profile (Asmp. 7). Any model of the internal temperature profile would be hard to verify anyway, since it is very hard to measure the internal temperature profile inside the product during drying.

7.2.2 Modeling of bound moisture removal

This framework does not consider the removal of bound water. This is an obvious weakness, but in terms of proportions, bound water makes up only a small fraction

	Fraction of total mass	Fraction of water
Ice, -5 °C	73.88 %	88.88 %
Bound water, -5 °C	9.25 %	11.12 %
Dry matter, -5 °C	16.88 %	
Ice, -10 °C	74.92 %	88.88 %
Bound water, -10 °C	9.38 %	11.12 %
Dry matter, -10 °C	15.70 %	

Tab. 7.2: Calculated initial mass fractions from the simulations.

of the total initial mass of water in the product (cf. Table 7.2). The wet basis drying curves give an exaggerated impression of the amount of bound water in the product, but the corresponding dry basis curves give a better impression.

There are two reasons why we do not model removal of bound water:

- We do not want a second source of vapor. If both ice and bound water were removed simultaneously, they would both form vapor, and we would then have to model what fraction of the vapor each of them were responsible for.
- By assuming removal of just ice, we have boiled down our problem to just tracking the ice front. A similar strategy might not work with bound water, and its concentration is of the same magnitude as that of ice, so the stability issues concerning modeling of phase change (cf. Section 4.2.3) are present for bound water as well.

Being able to model drying of bound water would be a definite advantage. Even though most of the water in the product is ice, much (if not most) of the drying time is usually spent on removing the bound water at the end of the process.

7.2.3 Modeling of shrinkage

We have chosen to model shrinkage through predefined, time-dependent functions. It might seem strange that we must specify the shrinkage beforehand, when the usual way to model shrinkage is to adjust the dimensions of the product according to the current water content. Unfortunately, this has not been an option.

The reason why we cannot use the current water content to adjust for shrinkage is that it requires knowledge of the current moisture concentration throughout the

product. Our equation is a partial differential equation, i.e. the coefficients are functions of *local* information only. Including water content as an argument in the coefficients requires a different type of equation, e.g. a functional differential equation. There is some theory on quasilinear functional differential equations, but the numerical methods are not as well developed for these types of equations as for partial differential equations.

It is possible to base shrinkage on water content, and still use this framework. This can be done by dividing the simulation into smaller parts. Instead of running a simulation with prescribed shrinkage for the entire period $(0, \tau)$, we can run several smaller simulations, where we adjust the product geometry between each one. We start the first simulation with a certain geometry and with the ordinary initial condition. When it stops, we calculate the water content and adjust the geometry accordingly. Then we start a new simulation, using the end result of the first one as initial condition. We repeat this process for all the remaining simulations.

7.3 Conclusions

Lastly, we present some thoughts on using the framework, and on further development.

7.3.1 Modeling AFD with the framework

The framework has not been extensively tested, but the test performed on the cod example did feature some interesting geometrical aspects. It showed that the implicit geometrical modeling works well. Not only were we able to model the three-dimensional geometry of the cod slab itself (including shrinkage). We were also able to easily model the perforated plate on which the cod slab rested, and the framework did detect the effect of these perforations (cf. Table 7.1). This is important, as the product rests on some sort of base in many types of drying. Being able to model the effect of the structure of the base is highly relevant. **In general, just about any geometrical feature could in principle be modeled, as long as we can find a GDF for it.** However, complicated geometrical features make the fluid dynamics needed to model the convective mass transfer coefficient more difficult.

We did not try variable diffusivities in the test, but simply used fitted constant values. Since the fitted values were so much higher in the -5 °C case than in the

-10 °C case, it seems clear that the diffusivities should depend on ambient temperature in some way. Determining what such a dependence should look like mathematically, requires many more tests on data for other temperatures.

In general, the framework needs comprehensive background data. There are many physical and non-physical parameters, and finding suitable values for these generally involves a lot of trial and error. Part of the difficulty in finding suitable values is that the three-dimensional nature of the framework means that memory demands quickly become very large, so we are typically not able to use as many basis functions as we would like. This limits the accuracy and stability of the framework.

However, when the necessary background data is available, and the framework is well calibrated, the cod test showed that the framework can be a useful modeling tool.

7.3.2 Improvement and further development

When the work on the framework started, the initial idea was to make it as sophisticated as possible. As expected, more and more simplifying assumptions became necessary as work progressed. In the end, the framework has essentially reduced a complicated physical problem into a pure front-tracking problem, through the RIF assumption and the omission of bound water from the modeling. Since it has become so focused on tracking the ice front, it is difficult to see how the current framework can be adapted to include many other physical mechanisms and phenomena. Finding a formulation that works both physically, mathematically and numerically has been a surprisingly hard challenge.

However, if such an attempt is made, there are some basic considerations to make. If new phenomena (like temperature and/or bound water) are added to the current framework, these must be modeled by separate, new equations. These equations must fit the general equation system (4.1)-(4.3). Since we have already developed a numerical scheme for this type of system, we can easily add new equations to the framework. However, implementing the framework in another numerical scheme may perhaps provide more stability, and thus allow model formulations that in practice do not work with the current scheme.

One natural extension of the current formulation would be to develop a framework for modeling VFD. Since there is no convection in VFD, the convection-related stability problems mentioned above would not exist, and temperature could be modeled.

In fact, temperature would have to be modeled, as the AFD assumption of wet bulb temperature at the ice front is clearly not applicable in VFD.

Finally, even though the complete framework is tailored to modeling AFD, some of its features might perhaps be applicable in totally different areas. This is in particular true for the implicit geometrical modeling.

APPENDIX

A. MATLAB SOURCE CODE

Code A.1 (*Plotting a product with cross-section*)

The file Plot3D.m, *which creates a three-dimensional plot of the moisture concentration in the part of the product lying above a plane given by one of its points and a normal vector.*

```
function Plot3D(I_prod,rho_hat_w,x,y,z,n_1,n_2,n_3,x_0,y_0,z_0,ts_p1,t)
% Plots 3D moisture concentration in a cross section
%
% INPUT
%
% I_prod = GDF for the product surface
% rho_hat_w = matrix containing moisture concentration values
% x = matrix containing the x values of the grid points
% y = matrix containing the y values of the grid points
% z = matrix containing the z values of the grid points
% n_1 = x component of normal vector of cross-sectional plane
% n_2 = y component of normal vector of cross-sectional plane
% n_3 = z component of normal vector of cross-sectional plane
% x_0 = x value of point in cross-sectional plane
% y_0 = y value of point in cross-sectional plane
% z_0 = z value of point in cross-sectional plane
% ts_p1 = time step number + 1
% t = actual time (h)

% Calculating cross-sectional GDF values
c_s = n_1*(x - x_0) + n_2*(y - y_0) + n_3*(z - z_0);
c_s(:,:,:) = min(c_s(:,:,:),I_prod(x,y,z,t/3600)); % Needs time in s, not h

% Plotting C_w for time t
figure('Name',['Moisture concentration (kg/m^3) after ' num2str(t) ' h'])
isosurface(x,y,z,c_s(:,:,:),0,rho_hat_w(:,:,:,ts_p1))
shading interp
```

Code A.2 (*Calculating Fourier integrals*)

The file Inverarity.m, *which calculates one or more given three-dimensional Fourier integrals.*

```
function integrals = Inverarity(f,grid_sp,m,N)
% Evaluates the integrals of f(x,y,z)*exp(i*pi*[kappa*x/a lambda*y/b xi*z/c])
% for -a<=x<=a, -b<=y<=b, -c<=z<=c and kappa,lambda,xi = -N,...,N
%
% OUTPUT
%
% integrals = the integrals
%
% INPUT
```

```
10 %
11 % f = function values at the grid points
12 % grid_sp = grid spacings = 2*[a/m(1) b/m(2) c/m(3)]
13 % m = vector containing the number of grid points in each direction
14 % N = kappa,lambda,xi bounds
15
16 [kappa,lambda,xi] = ndgrid(-N:N);
17 k1 = ((1/m(1)) - 1)*kappa(:);
18 k2 = ((1/m(2)) - 1)*lambda(:);
19 k3 = ((1/m(3)) - 1)*xi(:);
20 k123 = k1 + k2 + k3;
21
22 k1 = (kappa(:) > 0)*m(1) - kappa(:);
23 k2 = (lambda(:) > 0)*m(2) - lambda(:);
24 k3 = (xi(:) > 0)*m(3) - xi(:);
25 v_perm = m(1)*m(2)*k3 + m(1)*k2 + k1 + 1;
26
27 f = fftn(f);
28 integrals = prod(grid_sp)*(exp(1i*pi*k123).*f(v_perm));
```

Code A.3 (*Evaluating point values*)

The file FourierEval.m, *which calculates point values for a three-dimensional Fourier expansion, given grid points and Fourier coefficients.*

```
1  function W = FourierEval(v,N,x,y,z,a,b,c)
2  % For p,q,r = -N,...,N, evaluates the sum of
3  % v_(p,q,r)*exp(pi*i*(p*x/a + q*y/b + r*z/c))
4  %
5  % OUTPUT:
6  %
7  % W = the values at different points (x,y,z)
8  %
9  % INPUT:
10 %
11 % v = a vector of size (2*N + 1)^3 x 1
12 % N = summation bounds
13 % x = x values in (-a,a)
14 % y = y values in (-b,b)
15 % z = z values in (-c,c)
16 % a = x bounds
17 % b = y bounds
18 % c = z bounds
19
20 m = size(x);
21 K = 2*N + 1;
22
23 g = zeros(size(x));
24 g(1:K,1:K,1:K) = reshape(v,K,K,K);
25 [k_1,k_2,k_3] = ndgrid(0:(m(1) - 1),0:(m(2) - 1),0:(m(3) - 1));
26 g = g.*exp(pi*1i*((1/m(1) - 1)*k_1 + (1/m(2) - 1)*k_2 + (1/m(3) - 1)*k_3));
27 W = prod(m)*exp(-N*pi*1i*(x/a + y/b + z/c)).*ifftn(g);
```

Code A.4 (*Calculating predefined tables*)

The file CreateTables.m, *which creates matrices used by Code A.5. Only depends on the problem size N.*

```
function CreateTables(N)
% Creates predefined tables for CannonEwing3D.m
%
% INPUT
%
% N = number of basis functions (2*N in each direction)

%#ok<*NASGU>

% Useful constants
K1 = 2*N + 1;
K2 = K1^3;
K3 = 4*N + 1;
K4 = floor(0.5*K2);
K5 = 2*N;
K6 = 4*N^3;
K7 = ceil(0.5*K3^3) + K3^3;

% Defining index vectors
[kappa,lambda,xi] = ndgrid(-N:N);
kappa = kappa(:);
lambda = lambda(:);
xi = xi(:);

% Lanczos sigma factors
sigma_fun = @(p,q,r) sinc(p./(N + 1)).*sinc(q./(N + 1)).*sinc(r./(N + 1));
v_sigma = sigma_fun(kappa,lambda,xi);

% Creating permutation matrices
p = repmat(kappa.',K4,1) - repmat(kappa(1:K4),1,K2);
q = repmat(lambda.',K4,1) - repmat(lambda(1:K4),1,K2);
r = repmat(xi.',K4,1) - repmat(xi(1:K4),1,K2);

pos = @(p,q,r) ceil(0.5*K3^3) + (K3^2)*r + K3*q + p;
M_perm = tril(pos(p(:,1:K4),q(:,1:K4),r(:,1:K4)),-1);
M_perm = M_perm ...
    + triu(fliplr(pos(p(:,(K4 + 2):end),q(:,(K4 + 2):end),r(:,(K4 + 2):end))));
M_perm = M_perm + (K3^3)*eye(K4);
if max(max(M_perm)) <= intmax('uint16')
    M_perm = uint16(M_perm);
else
    M_perm = uint32(M_perm);
end

% Creating basis matrices
K3 = K5^3;
G = zeros(K2,K3);
pos = @(p,q,r) (K1^2)*r + K1*q + p + K4 + 1;
[nz,~] = find(prod([kappa lambda xi],2));
p = kappa(nz).';
q = lambda(nz).';
r = xi(nz).';
G(sub2ind(size(G),pos(p,q,r),1:K3)) = 1;
G(sub2ind(size(G),(K4 + 1)*ones(size(p)),1:K3)) = (-1).^(p + q + r + 3);
G(sub2ind(size(G),pos(p,q,0),1:K3)) = (-1).^(r + 1);
G(sub2ind(size(G),pos(p,0,r),1:K3)) = (-1).^(q + 1);
G(sub2ind(size(G),pos(0,q,r),1:K3)) = (-1).^(p + 1);
```

```
58 G(sub2ind(size(G),pos(p,0,0),1:K3)) = (-1).^(q + r + 2);
59 G(sub2ind(size(G),pos(0,q,0),1:K3)) = (-1).^(p + r + 2);
60 G(sub2ind(size(G),pos(0,0,r),1:K3)) = (-1).^(p + q + 2);
61 G = sparse(G);
62
63 G_hat_inv = G(:,1:K5).'*G(:,1:K5)/4;
64 G_hat_inv = G_hat_inv\eye(K5);
65
66 Q = [speye(K6) 1i*speye(K6); fliplr(speye(K6)) -1i*fliplr(speye(K6))]/sqrt(2);
67 GQ = G*Q;
68 GQHLGQ = real(GQ'*GQ);
69 GQ = GQ(1:K4,:);
70
71 opts.SYM = true;
72 opts.POSDEF = true;
73
74 clear K1 K2 K3 K6 sigma_fun nz p q r pos
75 whos
76
77 save(['Tables_' num2str(N)]);
```

Code A.5 (*Solving the linear system*)

The file CannonEwing3D.m, *which solves the linear system* (5.23).

```
1  function U = CannonEwing3D(C,B,U_init,h_m,a,b,c,x,y,z,Delta_t, ...
2      t_saved,t_all,N,m,grid_sp,N_sigma)
3  % Calculates ice indicator
4  %
5  % OUTPUT
6  %
7  % U = matrix containing ice indicator values
8  %
9  % INPUT
10 %
11 % C = cell array containing diffusivity functions
12 % B = cell array containing convection functions
13 % U_init = initial values
14 % h_m = convective mass transfer coefficient
15 % a = x bounds
16 % b = y bounds
17 % c = z bounds
18 % x = matrix containing the x values of the grid points
19 % y = matrix containing the y values of the grid points
20 % z = matrix containing the z values of the grid points
21 % Delta_t = time step
22 % t_saved = times to be saved
23 % t_all = all times
24 % N = number of basis functions (2*N in each direction)
25 % m = vector containing the number of grid points in each direction
26 % grid_sp = grid spacings = 2*[a/m(1) b/m(2) c/m(3)]
27 % N_sigma = number of times to apply sigma filter
28
29 %#ok<*NODEF>
30
31 %-------------------------------------------------------------------------
32 % PREPARATIONS
33 %-------------------------------------------------------------------------
34
35 load(['Tables_' num2str(N)]);
36 K8 = length(t_saved);
```

```
37  v_sigma = v_sigma.^N_sigma;
38
39  kappa = kappa*pi/a;
40  lambda = lambda*pi/b;
41  xi = xi*pi/c;
42  kappa_sh = kappa(1:K4);
43  lambda_sh = lambda(1:K4);
44  xi_sh = xi(1:K4);
45  D_1_sh_sq = spdiags(kappa_sh.^2,0,K4,K4);
46  D_2_sh_sq = spdiags(lambda_sh.^2,0,K4,K4);
47  D_3_sh_sq = spdiags(xi_sh.^2,0,K4,K4);
48
49  U = zeros([m K8]);
50
51  % Calculating initial basis coefficients
52  Fourier_0 = conj(Inverarity(U_init,grid_sp,m,N))/(8*a*b*c);
53  w_n = reshape(kron(G_hat_inv,G_hat_inv)*(reshape(G'*Fourier_0,4*N^2,2*N) ...
54      *G_hat_inv'),8*N^3,1);
55  w_0 = w_n;
56
57  % Calculating initial point values
58  U(:,:,:,1) = real(FourierEval(v_sigma.*(G*w_0),N,x,y,z,a,b,c));
59  U(:,:,:,K8) = U(:,:,:,1);
60
61  Delta_t_abc = 0.5*Delta_t/(8*a*b*c);
62
63  timeind = 1;
64  timeref = 0;
65  disp('Preprocessing done!')
66  tic;
67
68  %-------------------------------------------------------------------------
69  % CALCULATING SOLUTION
70  %-------------------------------------------------------------------------
71
72  for t_point=1:length(t_all)
73
74      %---------------------------------------------------------------------
75      % Preparations
76      %---------------------------------------------------------------------
77
78      t_nph = 0.5*(t_all(max(t_point - 1,1)) + t_all(max(t_point,2)));
79      arg_U = 0.5*(3*U(:,:,:,K8) - U(:,:,:,1));
80      arg_h = h_m(x,y,z,t_nph);
81
82      %---------------------------------------------------------------------
83      % Calculating basis coefficients
84      %---------------------------------------------------------------------
85
86      disp('Calculating coefficients...')
87
88      z_n = -conj(1i*(kappa ...
89          .*Inverarity(B{1}(x,y,z,t_nph,arg_U,arg_h),grid_sp,m,N) + ...
90          lambda.*Inverarity(B{2}(x,y,z,t_nph,arg_U,arg_h),grid_sp,m,N) ...
91          + xi.*Inverarity(B{3}(x,y,z,t_nph,arg_U,arg_h),grid_sp,m,N)));
92      z_n = real(Q'*(G'*(Delta_t_abc*z_n + G*w_n)));
93
94      v_int = Delta_t_abc*kron([1 0.5], ...
95          Inverarity(C{1}(x,y,z,t_nph,arg_U),grid_sp,m,K5));
96      M_int = bsxfun(@times,bsxfun(@times,v_int(M_perm),kappa_sh),kappa_sh.');
97      v_1 = v_int(K7);
98
```

```
99      v_int = Delta_t_abc*kron([1 0.5], ...
100         Inverarity(C{2}(x,y,z,t_nph,arg_U),grid_sp,m,K5));
101     M_int = M_int ...
102         + bsxfun(@times,bsxfun(@times,v_int(M_perm),lambda_sh),lambda_sh.');
103     v_2 = v_int(K7);
104
105     v_int = Delta_t_abc*kron([1 0.5], ...
106         Inverarity(C{3}(x,y,z,t_nph,arg_U),grid_sp,m,K5));
107     M_int = M_int + bsxfun(@times,bsxfun(@times,v_int(M_perm),xi_sh),xi_sh.');
108     v_3 = v_int(K7);
109
110     R_n = 2*real(GQ'*((tril(M_int,-1) + v_1*D_1_sh_sq + v_2*D_2_sh_sq ...
111         + v_3*D_3_sh_sq)*GQ - triu(M_int)*conj(GQ)));
112     R_n = (R_n + R_n.') + GQHLGQ;
113
114     w_n = 2*Q*linsolve(R_n,z_n,opts) - w_n;
115
116     %-------------------------------------------------------------------------
117     % Calculating point values
118     %-------------------------------------------------------------------------
119
120     disp('Calculating point values...')
121
122     if t_point == 1 % Predictor-corrector method for the first time step
123         U(:,:,:,1) = 2*U(:,:,:,1) ...
124             - real(FourierEval(v_sigma.*(G*w_n),N,x,y,z,a,b,c));
125         w_n = w_0;
126         continue % Skipping point value calculations
127     end
128
129     % Storing old values
130     U(:,:,:,1) = U(:,:,:,K8);
131
132     % New values
133     U(:,:,:,K8) = real(FourierEval(v_sigma.*(G*w_n),N,x,y,z,a,b,c));
134
135     if ismember(t_all(t_point),t_saved)
136         timeind = timeind + 1;
137         U(:,:,:,timeind) = U(:,:,:,K8);
138     end
139
140     timeref = toc - timeref;
141     disp(['Time step ' int2str(t_point - 1) ' done in ' num2str(timeref) ...
142         ' seconds!'])
143     timeref = toc;
144
145     if timeind == K8
146         break % Exit loop if no more results are to be saved
147     end
148 end
149
150 % Reinstating initial values
151 U(:,:,:,1) = real(FourierEval(v_sigma.*(G*w_0),N,x,y,z,a,b,c));
152
153 disp('Solution calculations done!')
```

Code A.6 (*Script for -5 °C*)

The file Cod_5.m, *which simulates AFD for slabs of cod fillet at -5 °C.*

```
% Script simulating drying of a cod slab at -5 C.

% METHOD PARAMETERS:

% APPROXIMATION PARAMETERS:
N = 7;
n_app = 1e10;
epsilon = 1e-30;

% CALCULATION DOMAIN:
% Product size (radii in m)
r_x = @(t) 0.001*0.5*(19.485 + (18.858 - 19.485)*(t/(34.25*3600))^(1/3));
r_y = @(t) 0.001*0.5*(19.758 + (19.325 - 19.758)*(t/(34.25*3600))^(1/3));
r_z = @(t) 0.001*0.5*(4.285 + (3.895 - 4.285)*(t/(34.25*3600))^(1/3));
Vol = @(t) 8*r_x(t)*r_y(t)*r_z(t); % Product volume
% Product surface GDF
I_prod = @(x,y,z,t) (1 - ((x./r_x(t)).^20 + (y./r_y(t)).^20 ...
    + ((z + r_z(0) - r_z(t))./r_z(t)).^20));
% Hole GDFs
I_holes_1 = @(x,y,z) (1 - ((sin(pi*x/(5e-3)).^2)/sin(pi*(1.5e-3)/(5e-3))^2 ...
    + (sin(pi*y/(8.66e-3)).^2)/sin(pi*(1.5e-3)/(8.66e-3))^2 ...
    + (1 + 100*(z + r_z(0))).^20));
I_holes_2 = @(x,y,z) (1 ...
    - ((sin(pi*(x - 2.5e-3)/(5e-3)).^2)/sin(pi*(1.5e-3)/(5e-3))^2 ...
    + (sin(pi*(y - 4.33e-3)/(8.66e-3)).^2)/sin(pi*(1.5e-3)/(8.66e-3))^2 ...
    + (1 + 100*(z + r_z(0))).^20));
% Unit normal for product surface = -grad(I_prod)/|grad(I_prod)|
I_prod_dx = @(x,t) -20*(x.^19)./(r_x(t).^20);
I_prod_dy = @(y,t) -20*(y.^19)./(r_y(t).^20);
I_prod_dz = @(z,t) -20*((z + r_z(0) - r_z(t)).^19)./(r_z(t).^20);
abs_grad_I_prod = @(x,y,z,t) realsqrt(I_prod_dx(x,t).^2 + I_prod_dy(y,t).^2 ...
    + I_prod_dz(z,t).^2);
n = {@(x,y,z,t) -I_prod_dx(x,t)./abs_grad_I_prod(x,y,z,t), ...
    @(x,y,z,t) -I_prod_dy(y,t)./abs_grad_I_prod(x,y,z,t), ...
    @(x,y,z,t) -I_prod_dz(z,t)./abs_grad_I_prod(x,y,z,t)};
% Indicator functions
chi_ihb = @(I) 0.5*(1 + tanh(n_app.*I));
chi_ohb = @(I) 0.5*(1 - tanh(n_app.*I));
% Subset indicator functions
chi = cell(4,1);
chi{1} = @(x,y,z,t) chi_ihb(I_prod(x,y,z,t)); % Inside the product
chi{2} = @(z) chi_ihb(z + r_z(0)); % Above the plate
chi{3} = @(x,y,z) chi_ihb(I_holes_1(x,y,z)); % Holes
chi{4} = @(x,y,z) chi_ihb(I_holes_2(x,y,z)); % Holes
m = [64 64 64];
a = 1.25*r_x(0); % 25 % larger than initial radius
b = 1.25*r_y(0); % 25 % larger than initial radius
c = 1.25*r_z(0); % 25 % larger than initial radius

% TIME:
% Number of time steps
N_t = 300;
% Total time is 35 h (s)
t_tot = 3600*35;
% Time step length (s)
Delta_t = t_tot/N_t;
% Number of time steps between each stored time step
N_save = 20; % Store every 20th time step
```

```
59 % Stored time steps to be plotted
60 t_plot = [1 3 5 6 7 8];
61
62 % PHYSICAL PARAMETER VALUES:
63
64 % MISCELLANEOUS:
65 gamma = 4.0262/(4.0262 - 1);
66 R_H2O = 461.52364;
67 rho_sat = @(T) ((610.71./(R_H2O.*T)).*(10.^(-9.09718.*(273.16./T - 1) ...
68     + 0.876793.*(1 - T/273.16))).*(273.16./T).^(-3.56654));
69 rho_sat_dT = @(T) (4e-17)*(1.785400132e14 - (2.30617171e8)*(T.^2) ...
70     + (8.008367955e10)*T)./(R_H2O*T.^0.56654);
71 Delta_H_sub = 2835607;
72 % Measured water content, with times (h)
73 X_wb_meas = [83.1214329 72.15796098 62.10824762 53.45253889 45.30650736 ...
74     27.99496966 11.93179059];
75 X_db_meas = X_wb_meas./(100 - X_wb_meas);
76 t_wb_meas = [0 5 8 11.5 14.25 20.5 34.25];
77
78 % COD:
79 phi = 0.5583;
80 m_init = 1.886e-3;
81 % Initial mass of moisture (kg)
82 m_w_init = X_wb_meas(1)*m_init/100;
83 % Initial mass fraction of ice (%)
84 x_ice = 1.105*X_wb_meas(1)/(1 + 0.7138/log(1 + (-2.2) - (-20)));
85 % Initial mass fraction of bound water (%)
86 x_bw = X_wb_meas(1) - x_ice;
87 % Mass of bound water (kg)
88 m_bw = x_bw*m_init/100;
89 % Mass of dry matter (kg)
90 m_dry = m_init - m_w_init;
91 % Initial ice concentration (kg/m^3)
92 rho_hat_ice_init = (x_ice*m_init/100)/Vol(0);
93
94 % SURROUNDINGS:
95 T_surr = 268.15;
96 RH = 0.40;
97 rho_surr = RH*rho_sat(T_surr);
98 V_a = @(t) 3.3;
99 nu_a = 1.2883e-5;
100 k_a = 0.023687;
101 Pr = 0.7185;
102 Sc = 0.60;
103 % Convective mass transfer coefficient (m/s)
104 h_m_max = @(x,y,z,t) ...
105     0.332*(Sc^(-2/3))*realsqrt(V_a(t).*nu_a./(2*b + y - r_y(t)));
106 hole_fac = 0.80;
107 h_m = @(x,y,z,t) h_m_max(x,y,z,t).*(chi{2}(z) - chi{1}(x,y,z,t) ...
108     + hole_fac*(chi{3}(x,y,z) + chi{4}(x,y,z)));
109 % Wet bulb temperature (K)
110 syms Twb
111 T_wb = double(solve((Delta_H_sub*nu_a/(k_a*Pr))*(rho_surr - rho_sat(Twb)) ...
112     + T_surr - Twb,Twb));
113 % Concentration ratio
114 omega = rho_hat_ice_init/(phi*rho_sat(T_wb));
115 % Modified moisture concentration in the surroundings (kg/m^3)
116 rho_hat_surr = omega*rho_surr;
117
118 %--------------------------------------------------------------------------
119 % PREPARATIONS
120 %--------------------------------------------------------------------------
```

```
121
122 % Generating grid
123 grid_sp = 2*[a/m(1) b/m(2) c/m(3)]; % Grid spacings
124 [x,y,z] = ndgrid((-a + 0.5*grid_sp(1)):grid_sp(1):a, ...
125     (-b + 0.5*grid_sp(2)):grid_sp(2):b,(-c + 0.5*grid_sp(3)):grid_sp(3):c);
126
127 % Generating times
128 N_t = N_t + 1;
129 t_all = zeros(N_t,1);
130 t_saved = zeros(ceil(N_t/N_save),1);
131 K1 = length(t_saved);
132 j = 2;
133 for k=1:(N_t - 1)
134     t_all(k + 1) = Delta_t*k;
135     if mod(k,N_save) == 0
136         t_saved(j) = Delta_t*k;
137         j = j + 1;
138     end
139 end
140
141 % Ice bound
142 K_ice = 0.99;
143 K_ice = K_ice*rho_hat_ice_init;
144
145 % Defining diffusivities
146 C_fac_1 = @(x,y,z,t) 0.037;
147 C_fac_2 = @(x,y,z,t) 0.037;
148 C_fac_3 = @(x,y,z,t) 0.037;
149 C_fac = @(U) ((rho_sat(T_wb)/rho_hat_ice_init)^(gamma + 1)) ...
150     *(((U + rho_hat_surr).^gamma).*chi_ohb(U + rho_hat_surr - K_ice));
151
152 C_ice_1 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_1(x,y,z,t).*C_fac(U) + epsilon;
153 C_ice_2 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_2(x,y,z,t).*C_fac(U) + epsilon;
154 C_ice_3 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_3(x,y,z,t).*C_fac(U) + epsilon;
155
156 C = {C_ice_1, C_ice_2, C_ice_3};
157
158 B = {@(x,y,z,t,U,h_m) - h_m.*U.*n{1}(x,y,z,t)/omega, ...
159     @(x,y,z,t,U,h_m) - h_m.*U.*n{2}(x,y,z,t)/omega, ...
160     @(x,y,z,t,U,h_m) - h_m.*U.*n{3}(x,y,z,t)/omega};
161
162 % Initial condition. Although this expression does not vanish at the
163 % boundary, its modified Fourier approximation will.
164 U_init = (rho_hat_ice_init - rho_hat_surr)*ones(size(x));
165
166 %-------------------------------------------------------------------------
167 % SOLVING THE EQUATION
168 %-------------------------------------------------------------------------
169
170 rho_hat_ice = CannonEwing3D(C,B,U_init,h_m,a,b,c,x,y,z,Delta_t, ...
171     t_saved,t_all,N,m,grid_sp,3);
172
173 % Remaining mass of water (ignoring vapor) (kg)
174 m_w = zeros(K1,1);
175 rho_hat_w = rho_hat_ice;
176 for k=1:K1
177     rho_hat_ice(:,:,:,k) = rho_hat_ice(:,:,:,k) + rho_hat_surr;
178     rho_hat_w(:,:,:,k) = (rho_hat_ice(:,:,:,k) >= K_ice).*rho_hat_ice(:,:,:,k);
179     m_w(k) = m_bw + Inverarity(rho_hat_w(:,:,:,k) ...
180         .*chi{1}(x,y,z,t_saved(k)),grid_sp,m,0);
181     rho_hat_w(:,:,:,k) = rho_hat_w(:,:,:,k) + m_bw/Vol(t_saved(k));
182 end
```

```
183
184 %-------------------------------------------------------------------------
185 % DISPLAYING RESULTS
186 %-------------------------------------------------------------------------
187
188 % Calculating water content
189 X_wb_calc = 100*m_w./(m_w + m_dry);
190 X_db_calc = X_wb_calc./(100 - X_wb_calc);
191
192 t_saved = t_saved/3600; % Converting from seconds to hours (for plotting)
193
194 % Plotting drying curves
195 figure('Name','Wet basis water content (%) at different times (h)')
196 hold on
197 plot(t_wb_meas,X_wb_meas,'*b')
198 plot(t_saved,X_wb_calc,'r')
199 hold off
200 figure('Name','Dry basis water content at different times (h)')
201 hold on
202 plot(t_wb_meas,X_db_meas,'*b')
203 plot(t_saved,X_db_calc,'r')
204 hold off
205
206 % Plotting ice concentration
207 for k=1:length(t_plot)
208     Plot3D(I_prod,permute(rho_hat_w,[2 1 3 4]),permute(x,[2 1 3 4]), ...
209         permute(y,[2 1 3 4]),permute(z,[2 1 3 4]),0,1,0,0,0,0, ...
210         t_plot(k),t_saved(t_plot(k)))
211         Plot3D(I_prod,permute(rho_hat_w,[2 1 3 4]),permute(x,[2 1 3 4]), ...
212         permute(y,[2 1 3 4]),permute(z,[2 1 3 4]),1,0,0,0,0,0, ...
213         t_plot(k),t_saved(t_plot(k)))
214 end
215
216 % Drying statistics
217 m_rem = 1000*(m_w(1) - m_w(K1));
218 disp(['Mass of water removed: ' num2str(m_rem) ' g (' ...
219     num2str(0.1*m_rem/m_w(1)) ' % of initial mass of water removed)'])
220 E_phase = m_rem*Delta_H_sub*(1e-6);
221 disp(['Energy spent on sublimation: ' num2str(E_phase) ' kJ'])
222
223 % Time usage
224 TU = toc;
225 Time_usage.Min = floor(TU/60);
226 Time_usage.Sec = mod(TU,60);
227 disp(['Time spent: ' num2str(Time_usage.Min) ' minute(s) and ' ...
228     num2str(Time_usage.Sec) ' seconds'])
```

Code A.7 (*Script for -10 °C*)

The file Cod_10.m, *which simulates AFD for slabs of cod fillet at -10 °C.*

```
% Script simulating drying of a cod slab at -10 C.

% METHOD PARAMETERS:

% APPROXIMATION PARAMETERS:
N = 7;
n_app = 1e10;
epsilon = 1e-30;

% CALCULATION DOMAIN:
% Product size (radii in m)
r_x = @(t) 0.001*0.5*(20.4 + (18.988 - 20.4)*(t/(102*3600))^(1/3));
r_y = @(t) 0.001*0.5*(20.273 + (17.315 - 20.273)*(t/(102*3600))^(1/3));
r_z = @(t) 0.001*0.5*(4.51 + (4.08 - 4.51)*(t/(102*3600))^(1/3));
Vol = @(t) 8*r_x(t)*r_y(t)*r_z(t); % Product volume
% Product surface GDF
I_prod = @(x,y,z,t) (1 - ((x./r_x(t)).^20 + (y./r_y(t)).^20 ...
    + ((z + r_z(0) - r_z(t))./r_z(t)).^20));
% Hole GDFs
I_holes_1 = @(x,y,z) (1 - ((sin(pi*x/(5e-3)).^2)/sin(pi*(1.5e-3)/(5e-3))^2 ...
    + (sin(pi*y/(8.66e-3)).^2)/sin(pi*(1.5e-3)/(8.66e-3))^2 ...
    + (1 + 100*(z + r_z(0))).^20));
I_holes_2 = @(x,y,z) (1 ...
    - ((sin(pi*(x - 2.5e-3)/(5e-3)).^2)/sin(pi*(1.5e-3)/(5e-3))^2 ...
    + (sin(pi*(y - 4.33e-3)/(8.66e-3)).^2)/sin(pi*(1.5e-3)/(8.66e-3))^2 ...
    + (1 + 100*(z + r_z(0))).^20));
% Unit normal for product surface = -grad(I_prod)/|grad(I_prod)|
I_prod_dx = @(x,t) -20*(x.^19)./(r_x(t).^20);
I_prod_dy = @(y,t) -20*(y.^19)./(r_y(t).^20);
I_prod_dz = @(z,t) -20*((z + r_z(0) - r_z(t)).^19)./(r_z(t).^20);
abs_grad_I_prod = @(x,y,z,t) realsqrt(I_prod_dx(x,t).^2 + I_prod_dy(y,t).^2 ...
    + I_prod_dz(z,t).^2);
n = {@(x,y,z,t) -I_prod_dx(x,t)./abs_grad_I_prod(x,y,z,t), ...
    @(x,y,z,t) -I_prod_dy(y,t)./abs_grad_I_prod(x,y,z,t), ...
    @(x,y,z,t) -I_prod_dz(z,t)./abs_grad_I_prod(x,y,z,t)};
% Indicator functions
chi_ihb = @(I) 0.5*(1 + tanh(n_app.*I));
chi_ohb = @(I) 0.5*(1 - tanh(n_app.*I));
% Subset indicator functions
chi = cell(4,1);
chi{1} = @(x,y,z,t) chi_ihb(I_prod(x,y,z,t)); % Inside the product
chi{2} = @(z) chi_ihb(z + r_z(0)); % Above the plate
chi{3} = @(x,y,z) chi_ihb(I_holes_1(x,y,z)); % Holes
chi{4} = @(x,y,z) chi_ihb(I_holes_2(x,y,z)); % Holes
m = [64 64 64];
a = 1.25*r_x(0); % 25 % larger than initial radius
b = 1.25*r_y(0); % 25 % larger than initial radius
c = 1.25*r_z(0); % 25 % larger than initial radius

% TIME:
% Number of time steps
N_t = 450;
% Total time is 100 h (s)
t_tot = 3600*100;
% Time step length (s)
Delta_t = t_tot/N_t;
% Number of time steps between each stored time step
N_save = 18; % Store every 18th time step
```

```
59 % Stored time steps to be plotted
60 t_plot = [1 3 5 7 9 11];
61
62 % PHYSICAL PARAMETER VALUES:
63
64 % MISCELLANEOUS:
65 gamma = 4.0246/(4.0246 - 1);
66 R_H2O = 461.52364;
67 rho_sat = @(T) ((610.71./(R_H2O.*T)).*(10.^(-9.09718.*(273.16./T - 1) ...
68     + 0.876793.*(1 - T/273.16))).*(273.16./T).^(-3.56654));
69 rho_sat_dT = @(T) (4e-17)*(1.785400132e14 - (2.30617171e8)*(T.^2) ...
70     + (8.008367955e10)*T)./(R_H2O*T.^0.56654);
71 Delta_H_sub = 2836633;
72 % Measured water content, with times (h)
73 X_wb_meas = [84.3 75.302145 48.35722565 20.34261018 14.74214693 14.69997103];
74 X_db_meas = X_wb_meas./(100 - X_wb_meas);
75 t_wb_meas = [0 6.5 30.5 58 82 102];
76
77 % COD:
78 phi = 0.7531;
79 m_init = 2.180e-3;
80 % Initial mass of moisture (kg)
81 m_w_init = X_wb_meas(1)*m_init/100;
82 % Initial mass fraction of ice (%)
83 x_ice = 1.105*X_wb_meas(1)/(1 + 0.7138/log(1 + (-2.2) - (-20)));
84 % Initial mass fraction of bound water (%)
85 x_bw = X_wb_meas(1) - x_ice;
86 % Mass of bound water (kg)
87 m_bw = x_bw*m_init/100;
88 % Mass of dry matter (kg)
89 m_dry = m_init - m_w_init;
90 % Initial ice concentration (kg/m^3)
91 rho_hat_ice_init = (x_ice*m_init/100)/Vol(0);
92
93 % SURROUNDINGS:
94 T_surr = 263.15;
95 RH = 0.40;
96 rho_surr = RH*rho_sat(T_surr);
97 V_a = @(t) 3.3;
98 nu_a = 1.2453e-5;
99 k_a = 0.023292;
100 Pr = 0.72;
101 Sc = 0.60;
102 % Convective mass transfer coefficient (m/s)
103 h_m_max = @(x,y,z,t) ...
104     0.332*(Sc^(-2/3))*realsqrt(V_a(t).*nu_a./(2*b + y - r_y(t)));
105 hole_fac = 0.80;
106 h_m = @(x,y,z,t) h_m_max(x,y,z,t).*(chi{2}(z) - chi{1}(x,y,z,t) ...
107     + hole_fac*(chi{3}(x,y,z) + chi{4}(x,y,z)));
108 % Wet bulb temperature (K)
109 syms Twb
110 T_wb = double(solve((Delta_H_sub*nu_a/(k_a*Pr))*(rho_surr - rho_sat(Twb)) ...
111     + T_surr - Twb,Twb));
112 % Concentration ratio
113 omega = rho_hat_ice_init/(phi*rho_sat(T_wb));
114 % Modified moisture concentration in the surroundings (kg/m^3)
115 rho_hat_surr = omega*rho_surr;
116
117 %--------------------------------------------------------------------------
118 % PREPARATIONS
119 %--------------------------------------------------------------------------
120
```

```
% Generating grid
grid_sp = 2*[a/m(1) b/m(2) c/m(3)]; % Grid spacings
[x,y,z] = ndgrid((-a + 0.5*grid_sp(1)):grid_sp(1):a, ...
    (-b + 0.5*grid_sp(2)):grid_sp(2):b,(-c + 0.5*grid_sp(3)):grid_sp(3):c);

% Generating times
N_t = N_t + 1;
t_all = zeros(N_t,1);
t_saved = zeros(ceil(N_t/N_save),1);
K1 = length(t_saved);
j = 2;
for k=1:(N_t - 1)
    t_all(k + 1) = Delta_t*k;
    if mod(k,N_save) == 0
        t_saved(j) = Delta_t*k;
        j = j + 1;
    end
end

% Ice bound
K_ice = 0.99;
K_ice = K_ice*rho_hat_ice_init;

% Defining diffusivities
C_fac_1 = @(x,y,z,t) 0.013;
C_fac_2 = @(x,y,z,t) 0.013;
C_fac_3 = @(x,y,z,t) 0.013;
C_fac = @(U) ((rho_sat(T_wb)/rho_hat_ice_init)^(gamma + 1)) ...
    *(((U + rho_hat_surr).^gamma).*chi_ohb(U + rho_hat_surr - K_ice));

C_ice_1 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_1(x,y,z,t).*C_fac(U) + epsilon;
C_ice_2 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_2(x,y,z,t).*C_fac(U) + epsilon;
C_ice_3 = @(x,y,z,t,U) chi{1}(x,y,z,t).*C_fac_3(x,y,z,t).*C_fac(U) + epsilon;

C = {C_ice_1, C_ice_2, C_ice_3};

B = {@(x,y,z,t,U,h_m) - h_m.*U.*n{1}(x,y,z,t)/omega, ...
    @(x,y,z,t,U,h_m) - h_m.*U.*n{2}(x,y,z,t)/omega, ...
    @(x,y,z,t,U,h_m) - h_m.*U.*n{3}(x,y,z,t)/omega};

% Initial condition. Although this expression does not vanish at the
% boundary, its modified Fourier approximation will.
U_init = (rho_hat_ice_init - rho_hat_surr)*ones(size(x));

%-------------------------------------------------------------------------
% SOLVING THE EQUATION
%-------------------------------------------------------------------------

rho_hat_ice = CannonEwing3D(C,B,U_init,h_m,a,b,c,x,y,z,Delta_t, ...
    t_saved,t_all,N,m,grid_sp,2);

% Remaining mass of water (ignoring vapor) (kg)
m_w = zeros(K1,1);
rho_hat_w = rho_hat_ice;
for k=1:K1
    rho_hat_ice(:,:,:,k) = rho_hat_ice(:,:,:,k) + rho_hat_surr;
    rho_hat_w(:,:,:,k) = (rho_hat_ice(:,:,:,k) >= K_ice).*rho_hat_ice(:,:,:,k);
    m_w(k) = m_bw + Inverarity(rho_hat_w(:,:,:,k) ...
        .*chi{1}(x,y,z,t_saved(k)),grid_sp,m,0);
    rho_hat_w(:,:,:,k) = rho_hat_w(:,:,:,k) + m_bw/Vol(t_saved(k));
end
```

```
183 %-------------------------------------------------------------------------
184 % DISPLAYING RESULTS
185 %-------------------------------------------------------------------------
186
187 % Calculating water content
188 X_wb_calc = 100*m_w./(m_w + m_dry);
189 X_db_calc = X_wb_calc./(100 - X_wb_calc);
190
191 t_saved = t_saved/3600; % Converting from seconds to hours (for plotting)
192
193 % Plotting drying curves
194 figure('Name','Wet basis water content (%) at different times (h)')
195 hold on
196 plot(t_wb_meas,X_wb_meas,'*b')
197 plot(t_saved,X_wb_calc,'r')
198 hold off
199 figure('Name','Dry basis water content at different times (h)')
200 hold on
201 plot(t_wb_meas,X_db_meas,'*b')
202 plot(t_saved,X_db_calc,'r')
203 hold off
204
205 % Plotting ice concentration
206 for k=1:length(t_plot)
207     Plot3D(I_prod,permute(rho_hat_w,[2 1 3 4]),permute(x,[2 1 3 4]), ...
208         permute(y,[2 1 3 4]),permute(z,[2 1 3 4]),0,1,0,0,0,0, ...
209         t_plot(k),t_saved(t_plot(k)))
210         Plot3D(I_prod,permute(rho_hat_w,[2 1 3 4]),permute(x,[2 1 3 4]), ...
211         permute(y,[2 1 3 4]),permute(z,[2 1 3 4]),1,0,0,0,0,0, ...
212         t_plot(k),t_saved(t_plot(k)))
213 end
214
215 % Drying statistics
216 m_rem = 1000*(m_w(1) - m_w(K1));
217 disp(['Mass of water removed: ' num2str(m_rem) ' g (' ...
218     num2str(0.1*m_rem/m_w(1)) ' % of initial mass of water removed)'])
219 E_phase = m_rem*Delta_H_sub*(1e-6);
220 disp(['Energy spent on sublimation: ' num2str(E_phase) ' kJ'])
221
222 % Time usage
223 TU = toc;
224 Time_usage.Min = floor(TU/60);
225 Time_usage.Sec = mod(TU,60);
226 disp(['Time spent: ' num2str(Time_usage.Min) ' minute(s) and ' ...
227     num2str(Time_usage.Sec) ' seconds'])
```

BIBLIOGRAPHY

[1] Bejan, A. 1993. *Heat Transfer*. Wiley.

[2] Cannon, J.R., & Ewing, R.E. 1977. Galerkin procedures for systems of parabolic partial differential equations related to the transmission of nerve impulses. *Pages 24–52 of:* Fitzgibbon (III), W.E., & Walker, H.F. (eds), *Nonlinear diffusion*. Research notes in mathematics, no. 14. London: Pitman Publishing Limited.

[3] Cannon, J.R., & Ewing, R.E. 1980. A Galerkin Procedure for Systems of Differential Equations. *Calcolo*, **17**(1), 1–23.

[4] Cannon, J.R., Ford, W.T., & Lair, A.V. 1976. Quasilinear Parabolic systems. *Journal of Differential Equations*, **20**, 441–472.

[5] Claussen, I.C., & Strømmen, I. 2007. Physical Parameters of Atmospheric Freeze Dried Cod and Turnip Cabbage. *In: Atmospheric Freeze Drying - Physiochemical Parameters during Drying*. NTNU.

[6] Claussen, I.C., Ustad, T.S., Strømmen, I., & Walde, P.M. 2007. Atmospheric Freeze Drying - A Review. *Drying Technology*, **25**(6), 957–967.

[7] Cooper, J.R. 1982. Representation of the Ideal-Gas Thermodynamic Properties of Water. *International Journal of Thermophysics*, **3**(1), 35–43.

[8] Danilatos, G.D. 1993. Introduction to the ESEM Instrument. *Microscopy Research and Technique*, **25**(5–6), 354–361.

[9] Feistel, R., & Wagner, W. 2007. Sublimation pressure and sublimation enthalpy of H_2O ice Ih between 0 and 273.16 K. *Geochimica et Cosmochimica Acta*, **71**(1), 36–45.

[10] Golub, G.H., & Van Loan, C.F. 1996. *Matrix Computations*. 3 edn. Johns Hopkins University Press.

[11] Inverarity, G.W. 2002. Fast Computation of Multidimensional Fourier Integrals. *SIAM Journal on Scientific Computing*, **24**(2), 645–651.

[12] Kays, W.M., & Crawford, M.E. 1993. *Convective Heat and Mass Transfer*. 3 edn. McGraw-Hill International Editions.

[13] Kudra, T., & Mujumdar, A.S. 2009. *Advanced Drying Technologies*. 2 edn. CRC Press.

[14] Lanczos, C. 1966. *Discourse on Fourier Series*. Edinburgh and London: Oliver & Boyd.

[15] Lee, A. 1980. Centrohermitian and Skew-Centrohermitian Matrices. *Linear Algebra and its Applications*, **29**, 205–210.

[16] Reichelt, R. 2007. Scanning Electron Microscopy. *Chap. 3, pages 133–272 of:* Hawkes, P.W., & Spence, J.C.H. (eds), *Science of Microscopy*, vol. I. Springer.

[17] Serra, S. 1994. Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems. *BIT*, **34**(4), 579–594.

[18] Strømmen, I., & Kramer, K. 1994. New applications of heat pumps in drying processes. *Drying Technology*, **12**(4), 889–901.

[19] Sun, Y., & Beckermann, C. 2007. Sharp interface tracking using the phase-field equation. *Journal of Computational Physics*, **220**, 626–653.

[20] Yuan, G., & Zhou, Y. 1998. Stability of Difference Schemes for Nonlinear Parabolic Systems with Three Dimensions. *Acta Mathematicae Applicatae Sinica*, **14**(3), 284–299.