NTNU – Trondheim
Norwegian University of
Science and Technology

TMA4850 Mathematics in applications

Simulation of temperature and water concentration distributions for various preparations of beef

Group 2 project report

Mina Sørensen Bratvold
Sverre Myhre Lien
Julius Mihkkal Eriksen Lindi
Erik Liu
Vemund Tjessem
Svein Åmdal

October 19, 2021

Abstract

Numerical simulations for coupled heat and mass transfer during cooking of beef has been performed. The goal was to generate distributions of temperature and moisture within the beef for some chosen cooking methods, and verify them. Temperature measurements were performed in a basic oven-cooking experiment.

The simulations were based on implementation of finite difference methods, using forward difference in time and central difference in space. In general the obtained results showed expected physical behaviour. However, the numerical results deviated slightly from experimental values for cooking in a convection oven. This might be due to uncertainty in the model parameters for thermal properties of meat, numerical error or experimental uncertainties. The temperature and water concentration distributions for simulating some models of preparation methods are shown; convection oven cooking, sous-vide cooking and frying pan cooking. Their respective influences on temperature and water concentration are briefly discussed.

Cooking of meat is a common process in the entire world, and mathematical modeling is of interest to reduce the dependency on experienced judgement for a good result. A possible extension of the project is therefore to use the numerical model to optimize parameters for cooking of beef. The model can also be adapted to account for other types of meat or other cases with coupled heat and mass transfer, such as thermal insulation for construction.

Preface

This report is a result of a teamwork project related to the course TMA4850, Experts in teamwork, Mathematics in Applications at the Norwegian University of Science and Technology (NTNU). The aim of the course is to provide insights into interdisciplinary collaboration, and that the students get a new perspective on their own expertise.

The project is related to coupled heat and mass transfer during cooking of beef, and the theoretical model is based on the formulation given in [1]. A system of coupled heat and mass transfer were discretized using finite difference and implemented in *Python* with extensive use of the *NumPy* and *SciPy* libraries. A complimentary plotting library was also implemented using the *Matplotlib* library.

We want to thank our course supervisor Trond Kvamsdal for providing support and feedback during the project work. We also want to thank the group facilitators Jenny Aune Forbord and Atle Johan Gundersen Øvestad.

Trondheim, October 19, 2021

Mina Bratvold

Sverre Lien

Julius Lindi

Erik Liu

Vemund Tjessem

Svein Åmdal

Contents

1 Introduction

Meat is a common food in every part of the world, and spoilage due to improper processing or conservation is a widespread problem [2]. Cooking of meat is an important process, as it can extend storage life and prevent damages to the consumer by destructing microorganisms.

Cooking of beef is one of many cases where coupling between heat and mass transfer is relevant, and models of coupled mass and heat transfer during cooking of meat have been previously established. One of the first contributors was Obuz et. al [3]. In that paper, water evaporation is considered to be driven by the moisture difference between the air and the meat surface. Latent heat of evaporation is included in the boundary condition for the energy balance by adding a term where it is multiplied by the surface mass flow, and thereby coupling the mass and energy balances. Van der Sman [4] estimated the swelling pressure by applying a linearization of the Flory-Rehner theory, and argued that it is the driving force in Darcy's law. The swelling pressure was expressed as proportional to the difference between the moisture content and the water holding capacity. The paper does not consider shrinking and deformations, evaporation is assumed to only happen at the meat surface and the permeability of meat is assumed to be constant [2]. In a further work by Van der Sman and Ruud [5], the model was applied to industrial tunnel ovens. In 2012 Datta's group from Cornell University [6] described beef patties as a multicomponent and multiphase system which experiences evaporation and flow of moisture and fat in a hygroscopic porous medium. On the other hand, Goni and Salvadori [7] focused on the difference between loss due to dripping and loss due to evaporation, and measured the losses experimentally. This was further investigated by Kondjoyan et al. [8], which studied different processes and meat pieces.

Since experimental studies of meat have proven to be difficult, mathematical modeling of the cooking process is particularly useful. Experiments are difficult because the meat material is variable, and because meat will loose fluid and shrink when subjected to heat [9]. In that way, computer simulations can eliminate the variability related to materials and equipment.

The aim of this project was to develop a numerical model which would generate distributions of heat and moisture within beef samples during cooking. The heat and moisture distributions within the meat will be dependent as a result of coupling. Furthermore, we wanted to study how the distributions are affected by using different cooking methods. The methods that are covered by the project are cooking in oven, frying pan and in water bath (so-called sous-vide cooking). However, the main focus will be on oven cooking. Tuning of the process parameters is important for obtaining an optimal result, in terms of for example juiciness [1]. The initial plan was to perform some kind of optimization, but this was adjusted to qualitatively compare the distributions for some selected methods of cooking. Optimization of the cooking process is therefore not covered by this project, but is however a natural possible extension of the project.

The project theme is chosen based on the overall theme "Mathematics in applications". The task covers all main areas of the MAC-model; mathematics, application and computer science. The project consists of a mathematical model in the form of differential equations. The model is used for a specific application (cooking of beef) and is implemented in *Python*. Several group members had a background in thermodynamics, so cooking of beef was chosen as a natural application.

During cooking of a food product, as described in [10] regarding cod loin and [9] regarding cooking in general, heat is transported from the hot air to the product surface by convection and radiation. Furthermore, heat is transported from the surface to the core by convection and conduction. Mass transfer through the product is driven by diffusion and convection, and there is a mass loss in the form of evaporation at the surface.

For the modelling of cooking of beef in this project, some assumptions are made. They include:

- Only transport of pure water is considered. Impurities, solvents and transportation of other substances, such as fat, is neglected.
- The transporting fluid (water) is assumed to be incompressible (divergenceless).
- Internal heat generation and heat transport via radiation are neglected, on the assumption that they are small compared to transport from fluid convection[1].
- The energy consumed during denaturing of proteins is negligible compared to energy transferred from the oven.
- The crust is assumed to be thin, allowing water to be transported to the surface, and otherwise not interfering with evaporation.
- The shape of the beef is assumed to be constant, i.e. neglecting shrinkage.
- The beef is assumed to be isotropic and an uniformly ideal porous medium.
- Traditionally constant quantities such as heat capacity and diffusion constants are assumed to be constant.

2 Theory

2.1 The heat equation

Following Bird et. al.[11, p.336], by demanding conservation of energy, we have that

$$\underbrace{\frac{\partial}{\partial t}\rho_m U}_{(i)} = \underbrace{-\left(\nabla\cdot\rho_w U\boldsymbol{u_w}\right)}_{(ii)} \underbrace{-\left(\nabla\cdot\boldsymbol{q}\right)}_{(iii)} \underbrace{-p(\nabla\cdot\boldsymbol{u_w})}_{(iv)} \underbrace{-\left(\boldsymbol{\tau}:\nabla\boldsymbol{u_w}\right)}_{(v)}, \tag{2.1}$$

for viscous heat flow in a medium, where each term refers to (all per unit volume):

(i) The rate of increase in internal energy.

(ii) The net rate of additional internal energy by convective transport (i.e. water flow).

(iii) The rate of internal energy addition by heat conduction.

(iv) The reversible rate of internal energy increase by compression.

(v) The irreversible rate of internal energy by viscous dissipation.

In Equation 2.1, U means internal energy per unit volume, ρ_m and ρ_w are the (constant) meat and water densities, $\boldsymbol{u_w}$ is the water velocity, $\boldsymbol{q}$ is heat flux density, p is pressure and $\boldsymbol{\tau}$ is the material (meat) stress tensor.

We assume the beef is isotropic, i.e. $\tau_{ij} = \tau\delta_{ij}$. Insert this into the definition of the notation in (v) to obtain

$$(\boldsymbol{\tau}:\nabla\boldsymbol{u_w}) := \sum_{i,j}\tau_{ij}\frac{\partial u_{w,i}}{\partial x_j} \to \sum_i \tau\frac{\partial u_{w,i}}{\partial x_i} = \tau(\nabla\cdot\boldsymbol{u_w}). \tag{2.2}$$

We make the common assumption that water is incompressible, meaning

$$\nabla\cdot\boldsymbol{u_w} = 0, \tag{2.3}$$

which removes terms (iv) and (v) from Equation 2.1.

By the definition of heat capacity at constant pressure,

$$c_p := \left(\frac{\partial U}{\partial T}\right)_p, \tag{2.4}$$

we can insert more explicit quantities in terms (i) and (ii). Namely in (i),

$$\frac{\partial U}{\partial t} = c_p\frac{\partial T}{\partial t} \implies (i) = \rho_m c_{pm}\frac{\partial T}{\partial t}. \tag{2.5}$$

We may vary the spatial parameters at constant pressure, and find that

$$\nabla U := \sum_i \left(\frac{\partial U}{\partial x_i}\right)_p = \sum_i c_p \frac{\partial T}{\partial x_i} := c_p \nabla T, \tag{2.6}$$

and by the product rule, we have for (ii)

$$(ii) = -\rho_w \left[U(\underbrace{\nabla \cdot \boldsymbol{u_w}}_{=0}) + \boldsymbol{u_w} \cdot (\nabla U) \right] = -\rho_w c_{pw} \boldsymbol{u_w} \cdot (\nabla T). \tag{2.7}$$

Note that term (i) relates to the internal energy of the material (meat), and we write c_{pm} for the meat heat capacity at constant pressure. The term (ii) refers to the water's internal energy, and we therefore must use the heat capacity of water, c_{pw}.

Finally, by Fourier's law of heat conduction, the heat flux density is given by

$$\boldsymbol{q} = -k_m \nabla T, \tag{2.8}$$

where k_m is the heat diffusion constant for the meat.

Inserting from Equation 2.5 , Equation 2.7 and Equation 2.8, we find that Equation 2.1 reduces to

$$\rho_m c_{pm} \frac{\partial T}{\partial t} + \nabla \cdot (-k_m \nabla T) + \rho_w c_{pw} \boldsymbol{u_w} \cdot \nabla T = 0, \tag{2.9}$$

which we recognize from [1]. An alternative way to arrive at this equation is to assume there are two contributions to the heat flux $\boldsymbol{q}$. One being Fourier's law, seen in Equation 2.8, and one being the "density" of heat energy $\rho_w c_{pw}$ carried along with the water velocity $\boldsymbol{u_w}$ (due to the finite water temperature). This relation may be inserted in a continuity equation for heat flow, which yields the same equation. Any other contributions to the heat flux is then neglected.

It is assumed that k_m is constant, which simplifies the equation to

$$\rho_m c_{pm} \frac{\partial T}{\partial t} - k_m \nabla^2 T + \rho_w c_{pw} \boldsymbol{u_w} \cdot \nabla T = 0. \tag{2.10}$$

The basic heat equation states that the temporal rate of change in temperature at some point, is proportional to the spatial rate of change between the point and the average of all its neighbours. Rephrasing this statement as the second derivative, and assuming that the proportionality constant may be written $\tilde{\alpha} = \tilde{\alpha}_0 + \tilde{\alpha}_w$, one finds that the basic heat equation may be written

$$\frac{\partial T}{\partial t} = \tilde{\alpha}_0 \nabla^2 T + \tilde{\alpha}_w \nabla(\nabla T). \tag{2.11}$$

We recognize that our calculations have resulted in the familiar heat equation, but the water flow introduces some correction $\tilde{\alpha}_w \nabla$ to the diffusivity.

A simple model for the coupled heat equation may then be obtained by assuming $\tilde{\alpha}_w$ is a constant that is adapted to fit with empirical measurements.

2.2 The diffusion equation

Assuming no fluid sources, the continuity equation for a fluid states that the rate of change of fluid density equals the fluid flux into the unit volume, formally

$$\frac{\partial \rho}{\partial t} = -(\nabla \cdot \boldsymbol{j}), \tag{2.12}$$

where ρ is the fluid density and $\boldsymbol{j}$ is the flux density.

There are two contributions to the water flux. One is the water velocity itself, and one is due to diffusion due to a concentration gradient (known as Fick's law)[11, p.515]. We write

$$\boldsymbol{j} = \rho \boldsymbol{u_w} - \rho D \nabla C, \tag{2.13}$$

where D is the (constant) diffusion coefficient and C is the "mass fraction"[1] or *concentration* of water, meaning the mass of water per unit mass of meat. Given that water has constant density, and constant mass density, we may write $C = \beta \rho_w$ for some constant β.

Inserting Equation 2.13 in Equation 2.12, and inserting $C = \beta \rho_w$, β cancels out, and we obtain

$$\frac{\partial C}{\partial t} + \nabla \cdot (C \boldsymbol{u_w}) = \nabla \cdot (D \nabla C). \tag{2.14}$$

Using the product rule on the second term, and inserting Equation 2.3, we write down the simplified equation

$$\frac{\partial C}{\partial t} + \boldsymbol{u_w} \cdot \nabla C = D \nabla^2 C. \tag{2.15}$$

2.3 Fluid flow in porous media

The beef meat is assumed to be an ideal porous medium. The non-diffusive water flow through a porous media is given by Darcy's law [10]

$$\boldsymbol{u_w} = \frac{-K}{\mu_w} \nabla P, \tag{2.16}$$

[1]Analogous to the molar fraction.

where K is the permeability of the medium, μ_w is the dynamic viscosity of the medium and P is the swelling pressure. The water viscosity depends on the temperature like[1]

$$\mu_w(T) = \exp(-0.0072T - 2.8658), \tag{2.17}$$

where T is the temperature in °C. The swelling pressure P is proportional to the excess moisture, given by the difference between the moisture concentration and the equilibrium water holding capacity

$$P = E(C - C_{eq}), \tag{2.18}$$

where E is the modulus of elasticity and C_{eq} is the equilibrium water holding capacity, both of these are temperature dependent.

We assume C_{eq} is given by

$$C_{eq}(T) = a_1 - \frac{a_2}{1 + a_3 \exp(-a_4(T - T_\sigma))}, \tag{2.19}$$

where $T_\sigma = 52\,°\mathrm{C}$ is the centre of a logistic curve (water holding capacity vs. temperature), and a-parameters are other empirical fitting parameters, given in Appendix A. We also take the elasticity modulus to be given by

$$E(T) = E_0 + \frac{E_m}{1 + \exp(-E_n(T - E_D))}, \tag{2.20}$$

where the E-parameters are also empirical fitting parameters obtained by [1] using experimental data from [12]. Their values are given in Appendix A. Inserting Equation 2.18 into Equation 2.16, one can calculate the water velocity by

$$\boldsymbol{u_w}(T, C) = -\frac{KE(T)}{\mu_w(T)} \nabla(C - C_{eq}(T)), \tag{2.21}$$

where Equation 2.19 and Equation 2.20 may be inserted for the unknown quantities. Because $\boldsymbol{u_w}$ depends on both T and C, we identify that this term gives a coupling between Equation 2.9 and Equation 2.14.

2.4 Boundary conditions

The normal component of convective heat flux from air to the meat's outer surface is given by

$$q = \boldsymbol{q} \cdot \boldsymbol{n} = h(T_O - T_s) \tag{2.22}$$

where h is the convective heat transfer coefficient, which summarizes the non-linear temperature behaviour and any boundary layer and turbulence effects for convection [9]. It is a measure of how effective a temperature difference is converted into a heat flux. T_O is the oven temperature and T_s is the surface temperature of the meat. On the other hand, the inner surface heat flux has two terms. One is the familiar flux from Fourier's law, and the other is the heat energy carried in the normal component of the water flow. Not all the convective heat goes towards heating up the meat, some will dissipate with the evaporating water. The fraction of energy that contributes to water evaporation is denoted as f, a number between 0 and 1. A fraction $(1-f)$ of the energy remains to be transferred between the air and the meat. Thus, we may write

$$(1-f)\boldsymbol{q} = -k_m \nabla T - \boldsymbol{u_w} \rho_w c_{pw} T, \tag{2.23}$$

where $\rho_w c_{pw} T$ may be interpreted as the heat energy density carried by the water. In summary, this equation states that the fraction of the heat flux that does not provide energy for evaporation, is the sum of the Fourier heat diffusion and the water energy density carried by the water velocity. We may eliminate $\boldsymbol{q}$ from the preceding equations. The normal component then defines the boundary condition for heat transfer[1], given by[2]

$$-\boldsymbol{n} \cdot (k_m \nabla T + \boldsymbol{u_w} c_{pw} \rho_w T) = (1-f)h(T_O - T_s). \tag{2.24}$$

In the same fashion, the boundary condition for mass transfer is

$$\boldsymbol{n} \cdot (-D\nabla C + \boldsymbol{u_w} C) = fh\frac{(T_O - T_s)}{H_{\text{evap}}\rho_w}(C - C_{eq}). \tag{2.25}$$

The left side of the equation is the water flux in the meat, which consists of the diffusion predicted by Fick's law, and the water concentration carried along with the water velocity $\boldsymbol{u_w}$. The right hand side is multiplied by f to indicate that only a fraction of the energy is used for evaporation, and it is divided by the evaporation enthalpy H_{evap} to pay the energy cost of the phase transition.

These boundary conditions apply for boundaries 2, 4 and 6 in Figure 1. No vapour is assumed to leak through the bottom, meaning boundary 3 instead has the mass transfer boundary condition

$$\left.\frac{\partial C}{\partial z}\right|_{z=0} = 0. \tag{2.26}$$

Given the reflection symmetry in the planes normal to the bottom plate, we can reduce the computational load by halving the x- and y-dimensions of the beef, and defining the boundaries 2 and 4 in Figure 1 to be symmetric. The full image of the beef object is four similar pieces stitched together at the boundaries 2 and 4, at coordinates L_x and L_y.

Therefore, one way to express the conditions for boundary 2 are

$$\left.\frac{\partial T}{\partial x}\right|_{x=L_x} = 0, \quad \left.\frac{\partial C}{\partial x}\right|_{x=L_x} = 0. \tag{2.27}$$

[2]$\boldsymbol{n}$ is defined to be pointing outwards perpendicular to a given surface of the cube.

The conditions for boundary 4 are

$$\left.\frac{\partial T}{\partial y}\right|_{y=L_y} = 0, \quad \left.\frac{\partial C}{\partial y}\right|_{y=L_y} = 0. \tag{2.28}$$

2.5 Numerical parameters

One approximation for f is a switching function where $f = 0$ at temperatures below 100°C and some value close to 1 at temperatures above (for example 0.88[1]). A more refined model is

$$f(T) = 1 - \frac{1}{1 + e^{\frac{T-f_1}{f_2}}} \tag{2.29}$$

with parameters $f_1 = 47$ and $f_2 = 15$ experimentally adapted for cooking of cod[10]. While not initially intended for beef cooking, we regard Equation 2.29 as a more realistic and interesting model, and choose to make use of it.

The density of the meat is estimated based on the composition of nutrients [1]

$$\rho_m = \frac{1}{\sum \frac{y_i}{\rho_i}} \tag{2.30}$$

where ρ_m, ρ_p, ρ_f, ρ_c and ρ_w are the densities of meat, protein, fat, carbohydrate and water respectively. y_i is the mass fraction of component i, and the composition data are taken from [13].

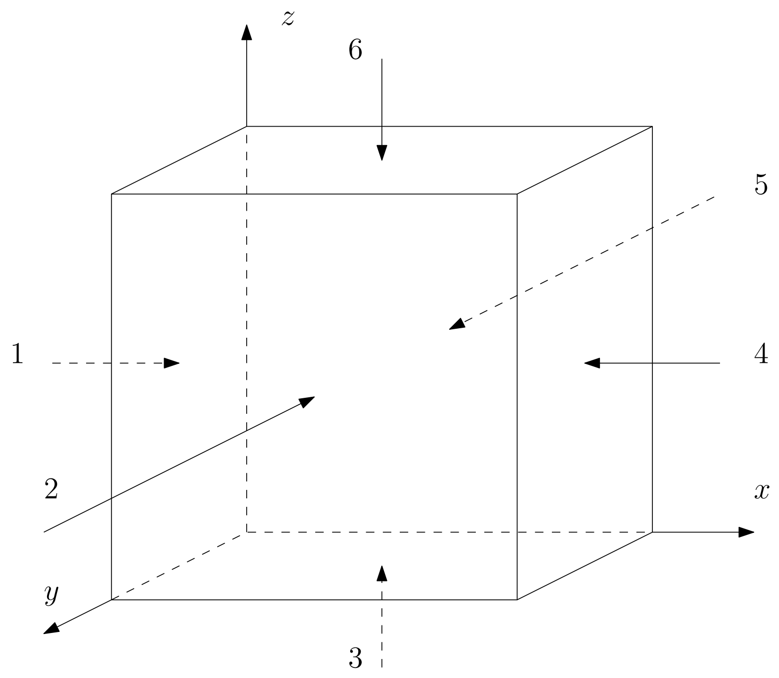


Figure 1: Sketch of the rectangular meat sample. Boundaries 2 and 4 are inside the meat, 1, 5 and 6 are facing air, and 3 is facing the oven plate. The meat sample is assumed to have Cartesian dimensions L_x, L_y and L_z.

The heat capacity of meat is taken to be given by[1]

$$C_{pm} = (1.6y_c + 2y_p + 2y_f + 4.2y_w) \cdot 10^3, \tag{2.31}$$

where all parameters are again specified in Appendix A.

The meat permeability K has a somewhat uncertain value, but it is in the range 10^{-19} to 10^{-17} m^2 [1]. For explicit calculation, we used 10^{-17}m^2 [14], as it also provides the largest contribution to the coupling term. Then, the effect of the coupling term is more interesting while still retaining all parameters within literature values.

The value of the convective heat transfer coefficient, h, is taken from [1], where it is determined experimentally using the lumped heat transfer method. There is some uncertainty regarding the exact value.

2.6 Summary of the governing equations

In summary, the coupled differential equations are:

$$\rho_m c_{pm} \frac{\partial T}{\partial t} = k_m \nabla^2 T - \rho_w c_{pw} \boldsymbol{u_w} \cdot \nabla T \tag{2.32}$$

$$\frac{\partial C}{\partial t} = D \nabla^2 C - \boldsymbol{u_w} \cdot \nabla C \tag{2.33}$$

The equations have corresponding boundary conditions given by

$$-\boldsymbol{n} \cdot (k_m \nabla T + \boldsymbol{u_w} c_{pw} \rho_w T) = (1-f) h (T_{oven} - T_s) \tag{2.34}$$

$$\boldsymbol{n} \cdot (-D \nabla C + \boldsymbol{u_w} C) = f h \frac{(T_{oven} - T_s)}{H_{evap} \rho_w} (C - C_{eq}) \tag{2.35}$$

which are simplified on some boundaries. The coupling term, $\boldsymbol{u_w}$, is given by Equation 2.21. These equations are the basis for numerical implementation.

3 Numerics

The problem is solved using finite difference method (FDM). Second order central difference is used for both the first and second order derivatives in space, while first order forward difference is used in time. The approach is somewhat inspired by Hellevik[15]. Generally we have for some function $f(t_n, x_i) \approx f_i^n$ that the first order derivative in space for some spatial coordinate x at time t_n and position x_i is

$$\frac{\partial f}{\partial x}(t_n, x_i) = \frac{f_{i+1}^n - f_{i-1}^n}{2\Delta h} + \mathcal{O}(\Delta h^2). \tag{3.1}$$

The second order derivative in space is

$$\frac{\partial^2 f}{\partial x^2}(t_n, x_i) = \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{\Delta h^2} + \mathcal{O}(\Delta h^2). \tag{3.2}$$

For the time derivative we have

$$\frac{\partial f}{\partial t}(t_n, x_i) = \frac{f_i^{n+1} - f_i^n}{\Delta t} + \mathcal{O}(\Delta t). \tag{3.3}$$

3.1 Numerical simplifications and assumptions

To simplify the boundary conditions, we consider a beef with a rectangular box shape. This captures the realistic behaviour, but is perhaps less interesting for solving the exact states for realistic beefs. Due to our simplifications of the geometry of the beef we opted to use difference methods rather than the more involved element methods. This also extends to the chosen difference scheme, which we chose to make it easier to construct the iterating matrices that were necessary to implement the boundary conditions.

Due to the simple geometry of the beef we will implement a uniform spacial mesh-size i.e. we will let $\Delta h := \Delta x = \Delta y = \Delta z$.

Another simplification is assuming that the coupling term in Equation 2.21 may be calculated with the state (T, C) from the previous iteration step. This is justified if the time discretization Δt is sufficiently small, so we know the change in state is marginal.

3.2 Discretizing the governing equations

Both Equation 2.32 and Equation 2.33 are on the same form. This is a simplified version of the convection-diffusion equation

$$a\frac{\partial Y}{\partial t} = b\nabla^2 Y + c\boldsymbol{u} \cdot \nabla Y \tag{3.4}$$

with Y being either T or C. These types of equations can be discretized on the form

$$\begin{aligned} Y_{i,j,k}^{n+1} =& Y_{i,j,k}^{n} + \frac{\Delta t}{a}\Big(C_{1,x}Y_{i+1,j,k}^{n} + C_{2,x}Y_{i-1,j,k}^{n} + C_{1,y}Y_{i,j+1,k}^{n} \\ &+ C_{2,y}Y_{i,j-1,k}^{n} + C_{1,z}Y_{i,j,k+1}^{n} + C_{2,z}Y_{i,j,k-1}^{n} - C_{3}Y_{i,j,k}^{n}\Big), \end{aligned} \tag{3.5}$$

where the superscript n denotes the time step and the subscripts i, j, k denote the spatial position on the grid. The constants are defined as

$$C_{1,d} = \frac{b}{(\Delta h)^2} + \frac{cu_{wd,i,j,k}}{2\Delta h}, \quad d \in \{x, y, z\} \tag{3.6}$$

$$C_{2,d} = \frac{b}{(\Delta h)^2} - \frac{cu_{wd,i,j,k}}{2\Delta h}, \quad d \in \{x, y, z\} \tag{3.7}$$

$$C_{3} = \frac{6b}{(\Delta h)^2}, \tag{3.8}$$

where $u_{wd,i,j,k}$ is the d-component of $\boldsymbol{u_w}$ in the position labeled by indices (i, j, k).

A more detailed derivation of Equation 3.5 is shown in Appendix B. The velocity shown in Equation 2.21 must also be discretized. Since it is a vector each component must be discretized separately

$$u_{wx,i,j,k} = -\frac{KE_{i,j,k}}{2\mu_{w,i,j,k}\Delta h}(C_{i+1,j,k}^{n} - C_{i-1,j,k}^{n} - C_{eq,i+1,j,k}^{n} + C_{eq,i-1,j,k}^{n}) \tag{3.9}$$

$$u_{wy,i,j,k} = -\frac{KE_{i,j,k}}{2\mu_{w,i,j,k}\Delta h}(C_{i,j+1,k}^{n} - C_{i,j-1,k}^{n} - C_{eq,i,j+1,k}^{n} + C_{eq,i,j-1,k}^{n}) \tag{3.10}$$

$$u_{wz,i,j,k} = -\frac{KE_{i,j,k}}{2\mu_{w,i,j,k}\Delta h}(C_{i,j,k+1}^{n} - C_{i,j,k-1}^{n} - C_{eq,i,j,k+1}^{n} + C_{eq,i,j,k-1}^{n}) \tag{3.11}$$

3.3 Implementing the boundary conditions

By discretizing using the first order central difference, the general robin boundary condition;

$$\boldsymbol{n} \cdot (\alpha \nabla Y + \beta \boldsymbol{u} Y) = \gamma, \qquad \alpha, \beta, \gamma \in \mathbb{R} \tag{3.12}$$

becomes[3]:

$$T^n_{-1,j,k} = T^n_{1,j,k} + \frac{2\Delta h\beta u_{wx,0,j,k}}{\alpha} T^n_{0,j,k} + \frac{2h\gamma}{\alpha} \tag{3.13}$$

$$T^n_{i,-1,k} = T^n_{i,1,k} + \frac{2\Delta h\beta u_{wy,i,0,k}}{\alpha} T^n_{i,0,k} + \frac{2h\gamma}{\alpha} \tag{3.14}$$

$$T^n_{i,j,-1} = T^n_{i,j,1} + \frac{2\Delta h\beta u_{wz,i,j,0}}{\alpha} T^n_{i,j,0} + \frac{2h\gamma}{\alpha} \tag{3.15}$$

$$T^n_{I+1,j,k} = T^n_{I-1,j,k} - \frac{2\Delta h\beta u_{wx,I,j,k}}{\alpha} T^n_{I,j,k} + \frac{2h\gamma}{\alpha} \tag{3.16}$$

$$T^n_{i,J+1,k} = T^n_{i,J-1,k} - \frac{2\Delta h\beta u_{wy,i,J,k}}{\alpha} T^n_{i,J,k} + \frac{2h\gamma}{\alpha} \tag{3.17}$$

$$T^n_{i,j,K+1} = T^n_{i,j,K-1} - \frac{2\Delta h\beta u_{wz,i,j,K}}{\alpha} T^n_{i,j,K} + \frac{2h\gamma}{\alpha} \tag{3.18}$$

Example of a modified equation on the boundary $x = 0$, where Equation 3.13 is inserted into Equation 3.5:

$$\begin{aligned} Y^{n+1}_{0,j,k} = Y^n_{0,j,k} + \frac{\Delta t}{a}\Big(& \\ &+ (C_{1,x} + C_{2,x})Y^n_{1,j,k} \\ &+ C_{1,y}Y^n_{0,j+1,k} + C_{2,y}Y^n_{0,j-1,k} \\ &+ C_{1,z}Y^n_{0,j,k+1} + C_{2,z}Y^n_{0,j,k-1} \\ &- (C_3 - C_{2,x}u_{wx,0,j,k}\frac{2h\beta}{\alpha})Y^n_{0,j,k} \\ &+ C_{2,x}\frac{2h\gamma}{\alpha}\Big) \end{aligned} \tag{3.19}$$

More examples are given in Appendix C.

3.4 Iteration method

Equation 3.5 can be rewritten to the form:

$$\mathbf{Y}^{n+1} = \mathbf{Y}^n + \Delta t(\mathbf{A}\mathbf{Y}^n + \mathbf{b}) \tag{3.20}$$

Where:

- $\mathbf{Y}^{n+1}$: Vector of values at the next time step
- $\mathbf{Y}^n$: Vector of values at the current time step
- $\mathbf{A}$: Band matrix of coefficients for $\mathbf{Y}^n$
- $\mathbf{b}$: Vector with values bundled with γ

In order to use the iteration scheme defined in Equation 3.20, the indexing scheme needs to be changed. The previous coordinate indexing scheme (i, j, k) is now uniquely mapped to a single index[4] $n = k + jK + iJK$. For instance, with a cube with shape $(I = 5, J = 6, K = 7)$, the new index of $T_{2,4,3}$ becomes T_{115}.

[3]Note that **n** becomes -1 on boundaries where i, j or k equal 0. As the **n** is defined to be outwards perpendicular from the cube.

[4]Row-major indexing, $n \in \{0, 1, \ldots, IJK - 1\}$

3.5 Code implementation and optimization

The programming language of the project was chosen to be *Python*, for its intuitive nature, ease of use, and familiarity for our team-members with differing experiences with programming.

Modelling the coupled temperature and moisture development of a 3 dimensional beef over time is incredibly computationally expensive and in order to meet deadlines for obtaining actual results, minimizing the run-time of the program has been a crucial task.

First of all, the implementation of the numerics in has been heavily reliant on using fast numerical libraries that are both implemented in either *C/C++* or *Fortran* and still easy to use, namely *NumPy* and *SciPy*.

However, merely importing these libraries to the project files is not enough. We also needed to make sure that the most optimized routines are used for the calculation operations. For instance, the $\mathbf{A}$ band matrix in Equation 3.5 was constructed as a sparse diagonal matrix by using the scipy.sparse library. As opposed to just making a dense $\mathbf{A}$ matrix, this not only makes the memory usage substantially less but also speeds up the matrix multiplication with $\mathbf{Y}^n$ by a huge margin since the sparse matrix library has overloaded its own accelerated methods for this operation.

Another important step in the effort of increasing the performance has been to compile Numpy and Scipy from source with *OpenBLAS*[5]. While the default BLAS installation in Numpy is already fairly fast, the main advantage of OpenBLAS is that it supports multi-threading in several linear algebra operations. In other words, we are getting parallelism for free by doing this. Compiling in itself is a long and tedious process but we were lucky in the sense that the lab computers that we ended up using for processing were already compiled with OpenBLAS.

An overview of the main-program workflow, and the code repository can be seen in Appendix D.

3.6 Error and stability analysis

By estimating the functions using central and forward difference approximations there will be some amounts of numerical error. Using Taylor expansions from Equation 3.1, Equation 3.2, and Equation 3.3 we get that the numerical error should be proportional to the sum of the time-step and the two times the square of the spacial-step: $\Delta T = \mathcal{O}(\Delta t) + 2\mathcal{O}(\Delta h^2)$

This means that after each step i we get some error $\epsilon_i > 0$ such that after n-th time-step computation we get the error $\|T(\vec{s}, t_n) - T_{approx}(\vec{s}, t_n)\| = \sum_{i=0}^{n} \epsilon_i$ which should in theory depend on the mesh-size of both time and spacial grid, i.e. we should get that as $\Delta t \to 0$ and $\Delta h \to 0$ then $\epsilon_i \to 0$. But as long as the total variation $\epsilon_{tot} = \sum_{i=0}^{n} \epsilon_i$ of the numerical solution at a fixed time remains bounded as the step size goes to zero, we should get a stable scheme that converges to the desired values.

Furthermore we also have the Lax equivalence theorem that states that for a consistent finite difference method for a well-posed linear initial value problem, the method is convergent if and only if it is stable. It is very important to note that our set of equations is

[5]An open-source implementation of *Basic Linear Algebra Subprograms* (BLAS)

not only an *initial value problem (IVP)* but also a *boundary value problem (BVP).* This means that the following section does not completely capture the problem, Lax equivalence theorem does not even necessarily hold for BVPs. It could however give a sense of the range of the conditions.

To check for stability there are two often used standard procedures: *Lax–Richtmyer stability test* and *Von Neumann stability analysis.*

Lax–Richtmyer stability test (LRS) says that some iterating scheme $\vec{a}_{n+1} = A(\vec{a}_n)$ is stable if the iterating function A has 2-norm $\|A\|_2 < 1$.

Von Neumann stability analysis is a little bit more involved. Given some partial differential equation with coefficients $\{a_i\}$ we have that the difference scheme is stable if $g(\{a_i\}, \Delta t, \Delta h) \leq 1$ where g is some function determined by the differential equation. The Von-Neumann condition for the coupled system is rather difficult to compute, but it is possible to make an educated guess by looking at the conditions for the stability of the two uncoupled equations

$$\rho_m c_{pm} \frac{\partial T}{\partial t} - k_m \nabla^2 T + \rho_w c_{pw} \sum_{i=1}^{3} (\nabla T)_i = 0, \qquad \frac{\partial C}{\partial t} - D \nabla^2 C + \sum_{i=1}^{3} (\nabla C)_i = 0.$$

Due to the Hartman–Grobman theorem this should be sufficient, as it shows that for non-linear problems a local, linearized stability analysis will lead to the necessary conditions.

We can use the computations already laid out in Hirsch [16] where we get that the necessary conditions for the stability of the Forward-Time Central-Space scheme of a general equation;

$$\alpha \frac{\partial F}{\partial t} - \beta \nabla^2 F + \mathbf{u} \cdot \nabla F = 0$$

is given by:

$$\frac{\beta \Delta t}{\alpha} \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2} \right) \leq \frac{1}{3}, \qquad \frac{\Delta t}{\beta} (\mathbf{u}_1^2 + \mathbf{u}_2^2 + \mathbf{u}_3^2) \leq 3. \tag{3.21}$$

In our case this simplifies to

$$\frac{\beta \Delta t}{\alpha \Delta h^2} \leq \frac{1}{9}, \qquad \frac{u^2 \Delta t}{\beta} \leq 1, \tag{3.22}$$

where we can substitute α, β and u for the appropriate values. Hence the maximum allowable time step is given by

$$\Delta t \leq \min \left(\frac{\alpha \Delta h^2}{9\beta}, \frac{\beta}{u^2} \right). \tag{3.23}$$

Also observe that the second condition of **??** is independent of the mesh-size Δh.

We compared the results to the LRS-test which was calculated for the iterating matrices $\frac{\Delta t}{2 c_{pm} \rho_m (\Delta h)^2} A$ and $\frac{\Delta t}{2(\Delta h)^2} B$ for temperature and concentration respectively. While letting

$\frac{\Delta t}{\Delta h} = 0.01$ and sampling at $t = 5$ we got, that the matrix norms $\left\| \frac{\Delta t}{2c_{pm}\rho_m(\Delta h)^2} A \right\|_2$ and $\left\| \frac{\Delta t}{2(\Delta h)^2} B \right\|_2$ hovered around 10^{-5}, which is well within reason. [6]

[6]This was calculated using the standard Frobenius matrix-norm [17] method for sparse matrices already implemented in the scipy library.

4 Test of convergence

To verify the numerical results, we compare the numeric solution to an exact solution with increasingly reduced discretization. Temperature and concentration errors of a selection of discretization constants are shown to give the general trend.

The analytical (dummy) equation is the uncoupled heat equation

$$\frac{\partial T}{\partial t} = \tilde{\alpha}\nabla^2 T, \tag{4.1}$$

where the heat transfer coefficient $\tilde{\alpha}$ is set to 10^{-4}[m^2/s] to make the relation of the coefficients in front of the terms be somewhat close to the realistic value[7], but at the same time let the solution change substantially over a relatively short time period. This is preferable due to run-time limitations of the numerics. We assume the beef has dimensions L_x, L_y and L_z in the corresponding Cartesian directions, and has zero temperature on the boundaries. Also assume[8] it has the initial state

$$T(x,y,z,0) = 3\xi(x)\eta(y)\zeta(z), \tag{4.2}$$

where the functions

$$\xi(x) = \sin\frac{2\pi}{L_x}x, \qquad \eta(y) = \sin\frac{2\pi}{L_y}y, \qquad \zeta(z) = \sin\frac{4\pi}{L_z}z \tag{4.3}$$

have been defined. This convenient choice of initial state yields an analytic solution of only a single Fourier component, namely

$$T(x,y,z,t) = 3e^{-4\tilde{\alpha}\pi^2\left(\frac{1}{L_x^2}+\frac{1}{L_y^2}+\frac{4}{L_z^2}\right)t}\xi(x)\eta(y)\zeta(z). \tag{4.4}$$

After iterating to a time of 10 units (seconds), the mean square error between the calulcated and analytical beef temperature distribution were calculated. The dimensions of the convergence test sample was set to be equal to those of the numerically calculated beef samples, being given in Appendix A.

Conveniently, the uncoupled concentration diffusion equation is exactly the same as the heat equation, only using different coefficients:

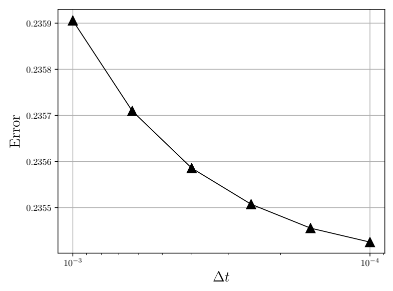
$$\frac{\partial C}{\partial t} = D\nabla^2 C. \tag{4.5}$$

The diffusion coefficient D is assumed to be constant, and several orders of magnitude smaller than $\tilde{\alpha}$ from Equation 4.1[9][m^2/s]. The solution is still given by Equation 4.4 with $T \to C$ and $\tilde{\alpha} \to D$. It should suffice to test the convergence properties of T.

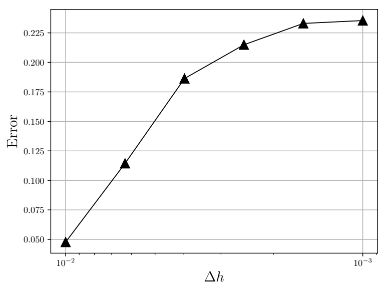
[7]The realistic value is really of the order 10^{-7}, but the chosen values illustrate the point of the coefficient a in Equation 3.4 being much larger than b.

[8]This is a purely mathematical construction, so we don't care about the possible physical implications of zero and negative temperatures.

[9]D is of the order 10^{-10}



(a) This plot is calculated with $\Delta h = 10^{-3}$ [m], and Δt varying along the abscissa.



(b) This plot is calculated with $\Delta t = 10^{-4}$ [s], and Δh varying along the abscissa.

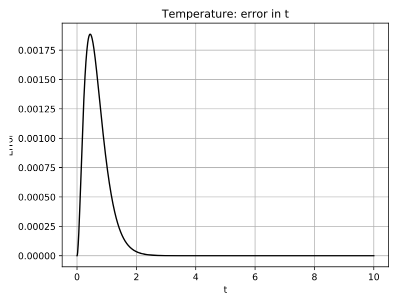
Figure 2: The numerical mean square error in the solution of the uncoupled heat equation after iterating for 10 seconds.

The testing is limited to the uncoupled case, as we don't know of any analytic solution of the coupled cases, even with simplifications. We suspect that none exists, but our endeavours to investigate further is cut short by time constraints. The finest discretizations are not tested, and the gap between testing points is perhaps unpleasantly large. Run-time constraints have relegated the convergence test to only display the general, qualitative trend of the error made by our numerical implementation.

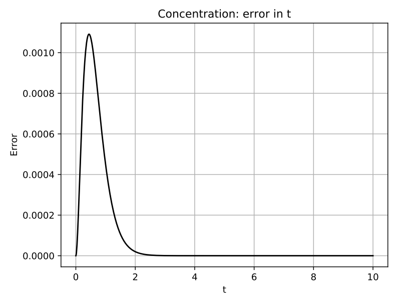
The convergence test is shown in Figure 2. It is seen that a reduction in the Δt yields a reduction in the iteration error. The error upon reduction of Δh appears to increase in Figure 2b, but at least the error also appears to be bounded. The increasing error might be a result of instability that the Δt isn't sufficiently lower than Δh as described in Equation 3.23.

The plots in Figure 2 only show the error at a given time step. What is perhaps more interesting is to see the error plotted with respect of time instead. An example is given in Figure 3, which is calculated with the same configuration but which different dimensions.

We set $\Delta t = 10^{-4}$ and $\Delta h = 10^{-3}$ for the remaining simulations.



(a) Error in temperature for the uncoupled test case.



(b) Error in concentration for the uncoupled test case.

Figure 3: The numerical mean square error in the solution of the uncoupled heat and concentration dummy equations upon iteration for 10 seconds. The dimensions of the object are unity, and the discretization is $\Delta x = 0.1$, $\Delta t = 10^{-2}$. The error decrease is due to the convergence of the dummy equations.

5 Experimental results

To validate our solution, we also compare the computed results to experimental data.

We had initially planned to have a collaboration with a local restaurant (To Rom og Kjøkken). They would supply us with their procedure for cooking the beefs, as well as external and internal temperatures, and cooking-times. We would then try to replicate the procedure using the beef simulator and compare results. This has been impossible due to the current pandemic situation. We instead produced our own results.

Beef was bought and stored in the fridge before preparation to avoid moisture loss. The beef lengths was measured. The oven temperature was 175 °C, with fan setting. Next, a thermometer was placed in the center of the beef and the beef sample was placed in the middle of the oven on a baking tray. The oven was opened and closed quickly when placing the beef. During cooking the temperature was recorded manually every 10 s, until the temperature had reached 65 °C. The results for two samples of beef are presented below.

Table 1: Dimensions of two beef samples cooked in oven at $T_O = 175\,°\text{C}$.

Sample	Dimensions [mm · mm · mm]
A	$93 \cdot 50 \cdot 16$
B	$100 \cdot 46 \cdot 17$

The numerical simulations of convection-oven cooking (subsection 6.1) replicated the dimensions of beef A. The comparison is shown in Figure 4. The uncertainty of the temperature measurements are unknown. Still, it seems likely that the deformation of the beef required to place the thermometer could influence the results somewhat. The actual geometric shape may also have deviated slightly from the ideal case.

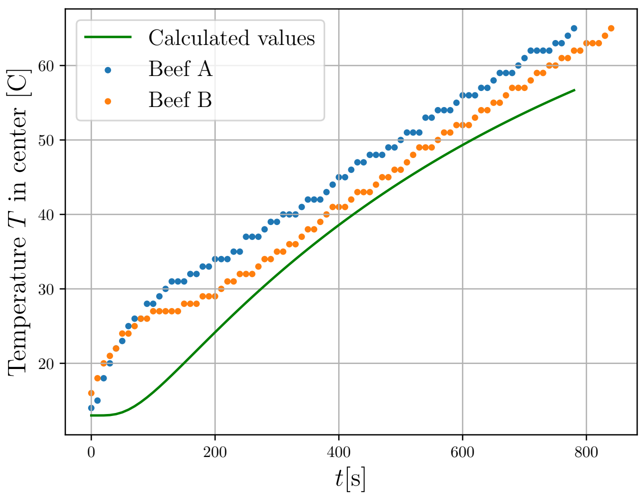


Figure 4: Temperature in the core of two beef samples during preparation in an oven with temperature 175 °C. Numerical results using the dimensions of beef A are also shown.

6 Results, plots and discussion

6.1 Convection oven

The temperature and concentration distributions for a simulation of an oven-cooked beef is shown in Figure 5. The geometrical parameters of the simulated beef are equal to those of beef A given in Table 1 from the empirical measurements. The simulation was run to its final state at 13 minutes, in line with the duration of the empirical measurements.

We make note of the fact that only a quarter of the beef has been simulated. The full beef object is supposed to be mirrored about the planes of maximal x- and y-values. Then the symmetry of the physical state would be equal the geometric symmetry. We also make note of the fact that the overall shape's temperature and concentration distributions adhere to what we would expect from using "common sense", so we take that as an indication that the simulation is working correctly. For example, the temperature gradient within the beef decreases as time passes as expected, while the water concentration is reduced near the edges due to vaporization.

6.2 Comparison with experimental data

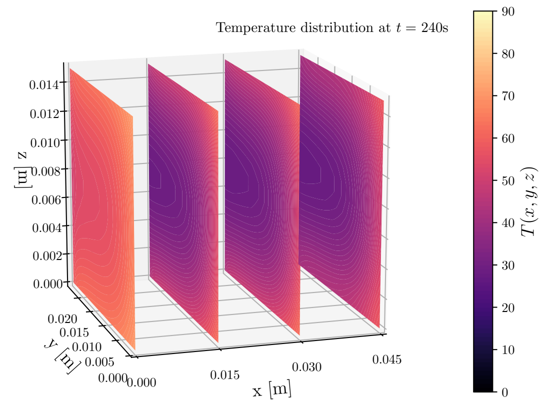
Comparing the numerical results to the experiment in Figure 4, the deviation is greatest in the first period of cooking. After $t = 200\,\mathrm{s}$ the experimental temperature increases approximately linearly. Although the numerical temperature development is slightly curved, the slopes are similar in this period.

One likely cause of the quick increase in centre temperatures for the experimental cases is geometric deformation of the beef and heat conducted by the thermometer itself. Other causes of empirical uncertainty may also contribute to the offset, as previously noted. The numerical calculations may additionally vary due to the uncertainty of the values of h and K. There is limited available data in the literature for the permeability of whole meat. K and h could have been adjusted to make the numerical results fit the experimental data, but this was not done due to time limitations. A similar extension would be to empirically determine an effective correction to the diffusivity in Equation 2.11.

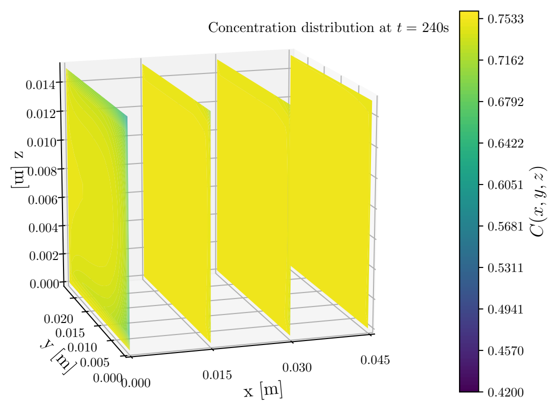
6.3 Comparison with uncoupled case

It may be interesting to observe the effect of the coupling term on the temperature distribution. In the uncoupled heat equation, we set $\boldsymbol{u_w} = 0$. The final temperature distribution after a calculated solution of the uncoupled heat equation is shown in Figure 6.

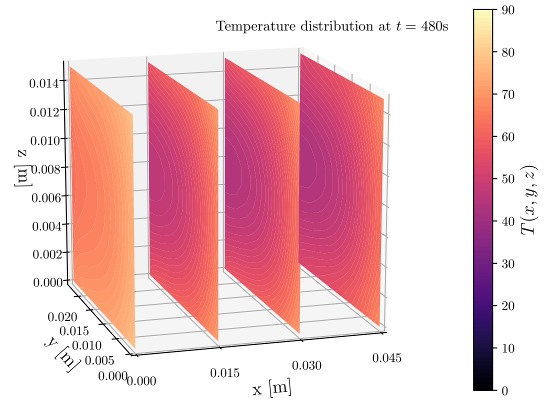
The difference between the simulation of the coupled and uncoupled equations is taken in two different points in the beef object. One in the centre, and one at the middle of the corner vertex between the x- and y-facing edges. The differences are shown in Figure 7. The overall trend is that the temperature increase on the boundary is slowed down by the coupling term, resulting in marginally lower recorded temperatures. The opposite is true in the centre, although it is noted that the difference is quite small. Therefore, we disregard the coupling term as being a significant contribution to the disparity between calculated and measured centre temperatures.



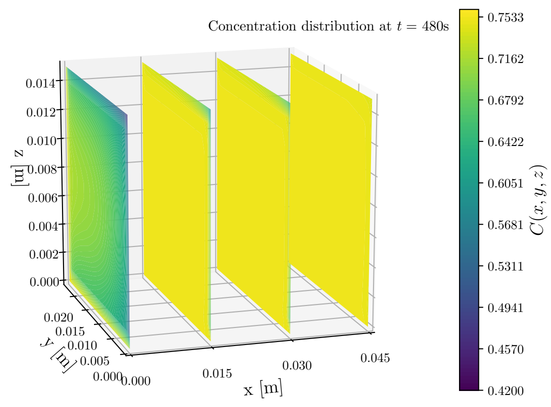
(a) Temperature distribution after 240 seconds.



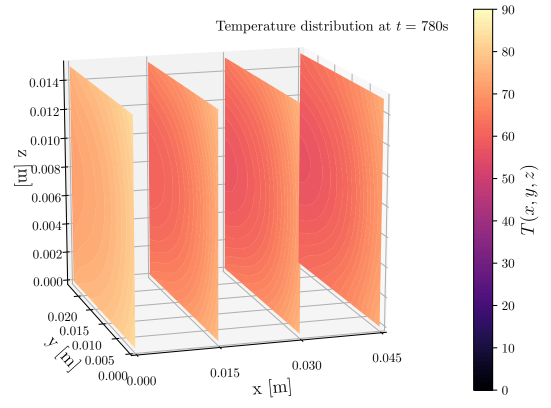
(b) Concentration distribution after 240 seconds.



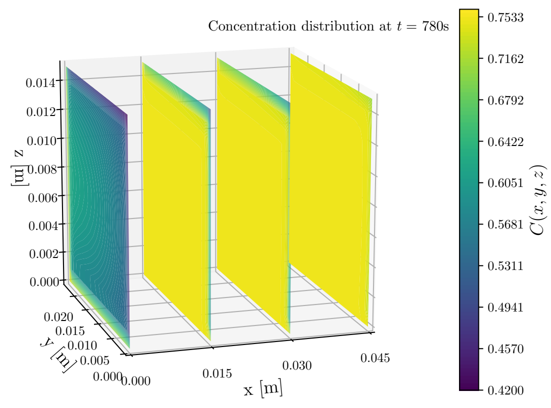
(c) Temperature distribution after 480 seconds.



(d) Concentration distribution after 480 seconds.



(e) Temperature distribution after 780 seconds.



(f) Concentration distribution after 780 seconds.

Figure 5: Time development of temperature and concentration in the simulation of convection-oven cooked beef. The geometrical dimensions equal those of the experimental measurements.

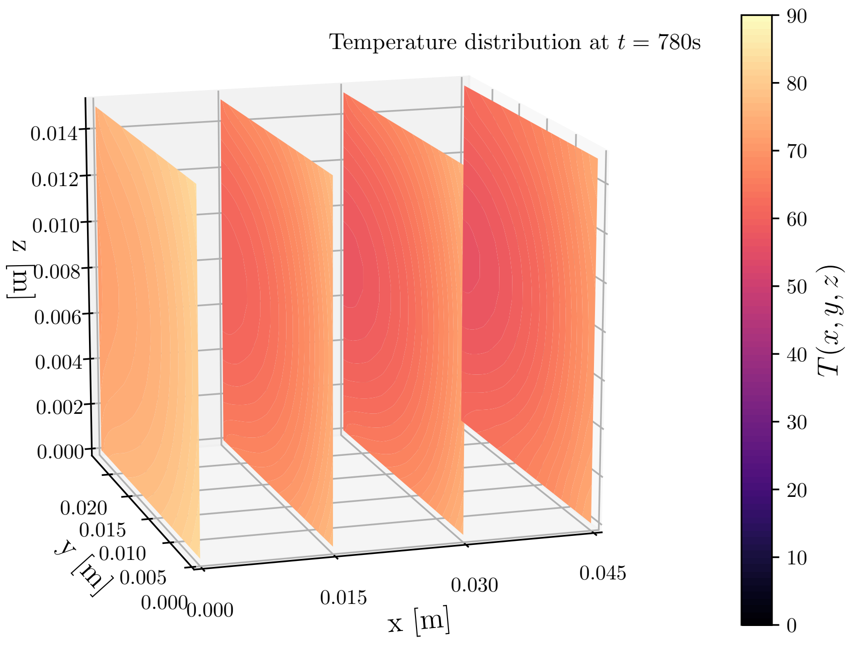


Figure 6: Temperature distribution at $t = 780\,\mathrm{s}$ for the solution of the uncoupled heat equation.

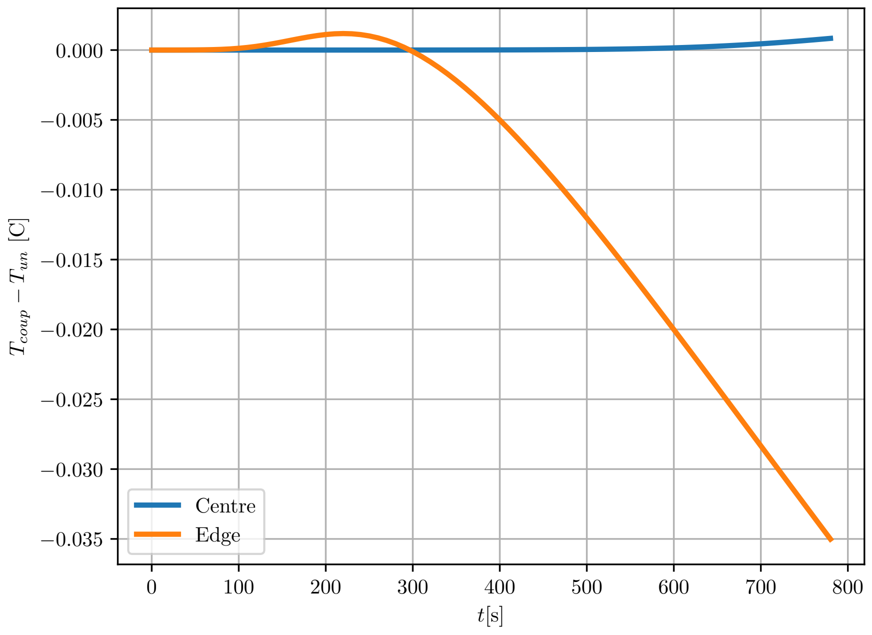
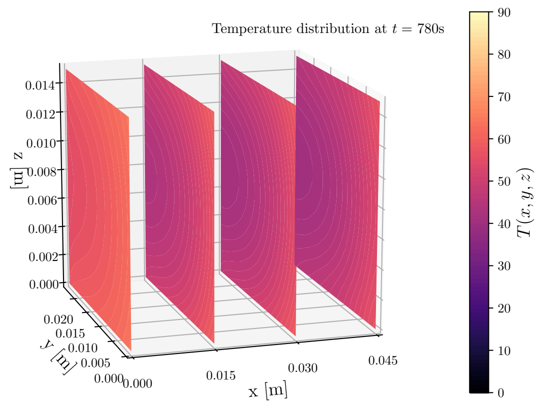
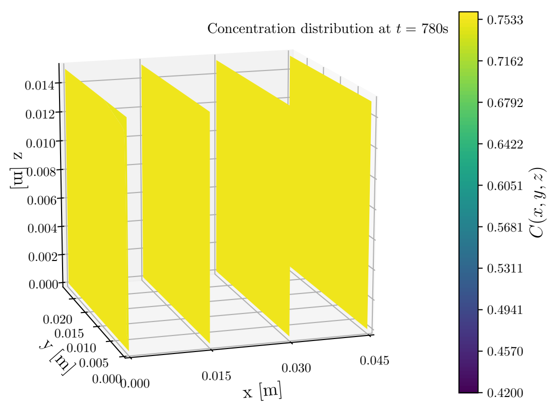


Figure 7: The temperature difference between the simulations of the coupled and uncoupled case, shown for the centre and for the middle of a corner vertex.



(a) Temperature distribution at $t = 780\,\text{s}$ for the simulation of sous-vide cooked beef.



(b) Concentration distribution at $t = 780\,\text{s}$ for the simulation of sous-vide cooked beef.

Figure 8: The final state of the simulated sous-vide cooked beef. The geometrical dimensions equal those of the experimental measurements.

6.4 Sous-vide cooking

Sous-vide is a cooking method where the food sample is placed in a vacuumized plastic container and cooked at a precisely controlled temperature [18]. The temperature of the water bath is set to the desired end temperature of the center of the meat sample. In that way, the method gives better control over doneness and texture than traditional cooking methods. The purpose is to cook the meat evenly to ensure that the inside is cooked properly without overcooking the outside, while at the same time retain moisture.

The temperature and concentration distributions of the final state of the simulation of sous-vide cooking are shown in Figure 8. The water bath temperature for the sous-vide simulation is 70°C.

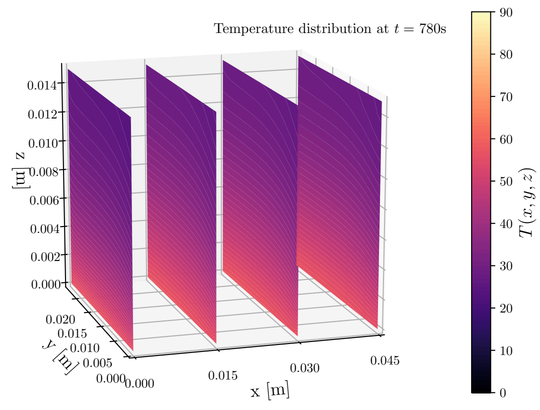
In the final state of the sous-vide cooked beef, the temperature is lower than for convection oven, which is as expected. It is difficult to determine conclusively if the temperature gradient is smaller or not. The concentration is constant throughout the beef and has remained unchanged (within numerical uncertainty) from the initial state. The reason for this is that our model has suppressed all evaporation of water at the boundaries (the water evaporation fraction f, previously defined by Equation 2.29, has been set to 0). Evidently, this model is too simple to accurately model the concentration distribution of sous-vide cooking, but we have no doubt that the concentration should, in fact, be greater for this preparation.

6.5 Single side frying pan

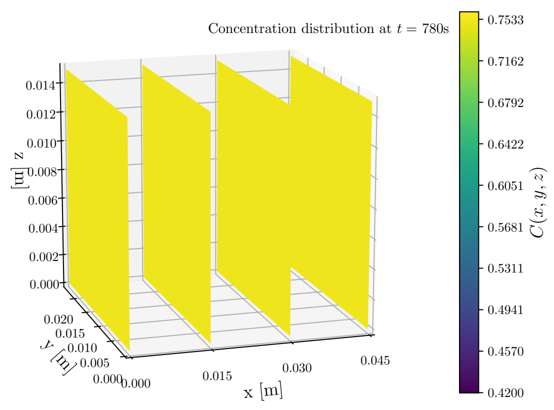
As another example of the simulating capabilities of the numerical solver, we consider a case resembling a frying pan. The bottom side is exposed to the oven temperature, while the other sides are exposed to a room temperature of 20 °C.

The results of the simulation are shown in Figure 9.

Like expected, the highest temperature is in the bottom where the beef is in direct



(a) Final temperature distribution for single side frying pan cooked beef.



(b) Final concentration distribution for single side frying pan cooked beef

Figure 9: The final state of the simulated single side frying pan cooked beef. The geometrical dimensions equal those of the experimental measurements.

contact with the frying pan. The temperature gradually decreases along the z-axis. [10] The concentration is relatively unchanged. This is because the surrounding temperature is assumed to be low (room temperature) so the amount of evaporation on the edges is also low. And we assume no water can escape through the bottom, so almost no concentration is lost anywhere.

The current model is evidently too simple for an exact description of this situation too. A simple extension to the model would be to let the air temperature have a gradient in the negative z-direction.

[10]When cooking beef using this method, the meat is usually turned upside down at least once (if not several times) to get the temperature more evenly distributed. This also provides a crust that might alter the boundary conditions (we suspect it might reduce the vapour that escapes at the boundaries).

7 Application of competence

Mina has contributed with finding data in the literature regarding relevant parameters for the process of cooking beef. She has also read up on previous work within the field and different methods of cooking meat. Mina carried out the experiments that are presented in the report.

Svein has contributed with some programming knowledge and ideas for implementation, as well as setting a high standard for data presentation. He researched and wrote large parts of the theory section, and was involved in running convergence tests.

Vemund has contributed with the understanding of the theory and equations. He was involved with the discretization and with creating some figures. He has also raised concerns about numerical stability.

Sverre has contributed with some programming of the implementation of the solver, mostly with regards to coupling the equations as well as some minor refactorization. He was involved with the discretizing of the problem and finalizing the stability analysis.

Erik has contributed with developing the backbone of the solver and the logic behind the boundary conditions. In addition to frequent refactorization of the code base and maintaining the GitHub repository.

Julius has developed the custom data display (plotting) routine from the Matplotlib package, done a large amount of run-time optimization and has been responsible for running the data iteration on NTNU computers.

Evidently, the project is multi-faceted, and beyond the competence of any single individual in the group. Thus, we conclude that we have been able to make use of more than one set of academic background knowledge.

7.1 Personal reflections

- **Erik**: From the weeks of interdisciplinary work, I have realized that proper conventions in the programming world need to be properly conveyed, and not to assume that the others already know about those. As in the earlier phase of the project, I needlessly tried to get everyone on the same page on the implementation of the solver, while in reality I should have just explained how to use the solver as a black box. Which I, at last, did in the later phase of the project. This project has indeed been quite the difference from the projects that I normally do, where the code base have a high degree of modularity. While in this project everything was bottlenecked by the main solver implementation, which had rather low degree of modularity. It pushed all the responsibility to the few that worked on it. I should still have tried to split the implementation of the solver into modules to lessen the responsibility and bottleneck of the solver and included other for parallel implementation of the solver. But at the time, the project looked so innocent and straightforward.
- **Julius**: After writing code for solving numerical physics problems for most of my academic career, it has become apparent to me during this project that I have been stuck in my own bubble in relation to proper coding conventions in programming. The other team-members have been able to burst my bubble by giving me a great

insight into good programming practices that not only makes sure the actual program works as intended, but also makes the code understandable for people other than myself. In addition, the disparity of educational backgrounds has made me a lot more aware of the importance of explaining and simplifying scientific concepts when conveying them to others.

Despite not being able to directly parallelize the code, I am still happy that I got to apply some of my knowledge from my parallel computing course, especially the more subtle aspects of memory management such as caching and the principle of locality. These are concepts that are not immediately obvious to the layman engineer since it is very dependent on the conventions of the programming language, but are easy enough to make use of without knowing the technical details. Therefore, in most of the cases related to optimization, it was sufficient to show others how it is done rather than explaining the details.

- **Mina**: As a way of sharing information obtained using my competence, I made a document summarizing information about relevant parameters for beef and experimental data. I could however have made a greater effort making sure that all information was taken into consideration from the beginning when setting up the cases to run. Prior to this project work I had some experience with programming, but by experiencing the expertise present in the group, I have become more aware of what I lack of knowledge.

- **Svein**: After writing a very comprehensive section, group members of different backgrounds complained that it was difficult to understand. This summarizes some of what I learned during the project. Concepts and conventions that seem trivial to some, may be completely unknown to others. I am sure the same applies to all the others, and I am grateful that they "dumb-ed it down" for me upon request. Disparities in preconceived notions of correct practice has arisen, and is the cause of a small conflict that may go unnoticed for a long time. I am certain the reader will encounter differing standards within this report, regarding the (practical vs theoretical) approach to explanation, explanation length (extensiveness vs brevity), vocabulary, notation, frequency and incorporation of references, etc. Overall, I have been made much more aware of belonging to a certain "clique" of physicist's academic conventions and background knowledge. I am also more appreciative of the fact that other conventions exists, all valid for their respective fields. I, along with other physicists, could communicate our knowledge better if we rid ourselves of those assumptions and conventions.

- **Sverre**: The project has made one thing abundantly clear: Abstraction does not necessarily equal better. My previous tendencies of regarding every element of a problem as some representative of some abstract class rather than an element in its own right, does not really work when working with such explicit problems. The switch has been challenging but at the same time rewarding. Even though my wish to make use of my more theoretical background did not come to fruition, I do hope that I was of some use during the project. One thing that did not pose any notable problem was technical language, as everyone was more or less on the same page with only a few differences that were easily interpreted using context.

- **Vemund**: This project has given me a valuable experience in what it is like if you don't have a thorough understanding of all parts of the project. It has also given me an insight into my own habits when it comes to programming both alone and in groups. In previous group project I've usually been the only one responsible for the programming so I have perhaps adapted some bad habits. It was a somewhat new experience when using GitHub and collaborating. This after a while resulted in that I let the more experienced programmers handle that and shifted my focus towards other things. In retrospect we should probably have tried to find a project where the needed competence was more evenly matched with the members of our group. The experience from EiT has made me reevaluate my own behaviour in regards to other projects, and made me realize that I should be better at communicating what I am doing.

8 Conclusion and suggested further work

In conclusion, the numerical simulation was capable of producing well-defined temperature and concentration distributions, solving the coupled heat and mass transfer equations that describe heat transfer in a beef. The trend of the results coincided with experimental data to some extent. The disparity may be explained by experimental uncertainty, or inaccurate choice of some model parameters. The effect of the coupling on the final temperature distribution was no larger than 0.1°C at any of the measured points, not a significant contribution. The simulation was displayed for convection-oven cooking, sous-vide cooking and frying pan cooking, but it is evident that the model does not capture the intricacies of the concentration distribution in the latter two cases.

A natural extension of the project is using the numerical model to optimize different parameters for cooking of beef. An example of an optimization problem would be: *Given a minimum value of temperature within the beef and a time limit, which cooking method will optimize the juiciness?* Such a problem seems of interest for restaurants. Optimization will produce even more realistic results if some improvements to the model are made. Firstly, the value of h can be adjusted to make the numerical model fit experimental data. For this to be useful, one would also have to generate more experimental data to reduce the uncertainty in the data.

In reality the permeability will be affected by the change in microstructure caused by pore formation [1]. This is not considered in our model, which uses a constant value of K. The model can therefore be improved by obtaining information about the porosity and its distribution within the product.

The results can also be improved by implementing higher order precision of finite difference or finite element methods. An advantage of the finite element method is that it is well adapted for studying complex geometry, and could therefore enable more realistic shapes of the beef samples.

The model can easily be adapted to account for other types of meat or produce. Also, the model can be adapted to represent other cases besides food where there is coupled heat and mass transfer. Examples are drying of wood and insulation in buildings, and many more possible applications exist. One would then have to use parameters that are relevant for the specific case of interest and also adapt the boundary conditions. One can then investigate cases that are perhaps of an even larger interest when it comes to societal utility.

8.1 Societal utility

As optimization has not been the focus of our project, the direct results of the project may have limited benefit. However, by viewing the project in a larger perspective, there is no doubt that a mathematical model for coupled heat and mass transfer, and for cooking of beef in particular, can be useful. Roasting in a convection oven is a common cooking method both in households, professional kitchens and in the ready-meal industry [1]. Although this method of cooking goes way back, the determination of process parameters is still to a large degree determined by the cook or operator. A good outcome does therefore require experienced judgement and skills. The dependency on experience also makes it difficult to upscale the process and make predictions when using new equipment or new process parameters. This justifies the need of a mathematical model, which can make automated process control possible.

An advantage of our numerical model is that it is not limited to one specific cooking method, which makes it more applicable. To simulate different cooking methods one only needs to change the boundary conditions, or apply different boundaries successively.

References

[1] Feyissa, A. H., Gernaey, K. V. & Adler-Nissen, J. 3d modelling of coupled mass and heat transfer of a convection-oven roasting process. *Meat Science* **93**, 810 – 820 (2013). URL http://www.sciencedirect.com/science/article/pii/S0309174012004111.

[2] Papasidero, D., Pierucci, S., Manenti, F. & Piazza, L. Heat and mass transfer in roast beef cooking. temperature and weight loss prediction. *Chemical Engineering Transactions* **43**, 151–156 (2015).

[3] Obuz, E., Powell, T. & Dikeman, M. Simulation of cooking cylindrical beef roasts. *Lwt - Food Science and Technology* **35**, 637–644 (2002).

[4] Van der Sman, R. G. M. Soft condensed matter perspective on moisture transport in cooking meat. *AIChE Journal* **53**, 2986–2995 (2007). URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.11323.

[5] Van der sman, R. Modeling cooking of chicken meat in industrial tunnel ovens with the flory-rehner theory. *Meat science* **95** (2013).

[6] Dhall, A., Halder, A. & Datta, A. Multiphase and multicomponent transport with phase change during meat cooking. *Journal of Food Engineering* **113**, 299–309 (2012).

[7] Goñi, S. & Salvadori, V. Prediction of cooking times and weight losses during meat roasting. *Journal of Food Engineering* **100**, 1–11 (2010).

[8] Kondjoyan, A., Oillic, S., Portanguen, S. & Gros, J.-B. Combined heat transfer and kinetic models to predict cooking loss during heat treatment of beef meat. *Meat science* **95**, 336–344 (2013).

[9] McGee, H., Mclnerney, J. & Harms, A. Modeling heat transfer in the kitchen. *Physics Today* **52**, 30–36 (1999). URL https://doi.org/10.1063/1.882728.

[10] Blikra, M. J., Skipnes, D. & Feyissa, A. H. Model for heat and mass transport during cooking of cod loin in a convection oven. *Food Control* **Volume 102**, Pages 29–37 (2019). URL http://www.sciencedirect.com/science/article/pii/S0956713519301008.

[11] Bird, R. B., Stewart, W. E. & Lightfoot, E. N. *Transport phenomena* (John Wiley & Sons, inc, 2002), 2 edn.

[12] Tornberg, E. Effects of heat on meat proteins – implications on structure and quality of meat products. *Meat Science* **Volume 70**, Pages 493–508 (2005). URL http://www.sciencedirect.com/science/article/pii/S0309174005000434.

[13] Heldman, D. R. & Lund, D. B. *Handbook of food engineering* (Taylor & Francis group, 2007), 2 edn.

[14] Datta, A. Hydraulic permeability of food tissues. *International Journal of Food Properties* **9**, 767–780 (2006).

[15] Hellevik, L. R. Numerical methods for engineers. https://folk.ntnu.no/leifh/teaching/tkt4140. Accessed: 2020-04-21.

[16] Hirsch, C. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics* (Butterworth-Heinemann, 2007), 2 edn.

[17] Weisstein, E. W. Frobenius norm. From MathWorld – a Wolfram web resource. `https://mathworld.wolfram.com/FrobeniusNorm.html`. Accessed: 2020-04-16.

[18] Baldwin, D. E. Sous vide cooking: A review. *International Journal of Gastronomy and Food Science* **1**, 15–30 (2012). URL `https://www.sciencedirect.com/science/article/pii/S1878450X11000035`.

A Symbol list

Symbol	Description	Value	Unit
a_1	Empirical fitting parameter 1 for C_{eq}	0.745	
a_2	Empirical fitting parameter 2 for C_{eq}	0.345	
a_3	Empirical fitting parameter 3 for C_{eq}	30	
a_4	Empirical fitting parameter 4 for C_{eq}	0.25	1/°C
C_0	Initial moisture concentration	0.75	kg/kg
c_{pm}	Specific heat of beef		J/kg°C
c_{pw}	Specific heat of water	4170	J/kg°C
D	Moisture diffusion coefficient	4×10^{-10}	m^2/s
E_0	Elasticity modulus parameter 1	12	kPa
E_m	Elasticity modulus parameter 2	83	kPa
E_n	Elasticity modulus parameter 3	0.3	1/°C
E_D	Elasticity modulus parameter 4	60	°C
f_1	Evaporation fraction parameter 1	47	°C
f_2	Evaporation fraction parameter 2	15	°C
h	Heat transfer coefficient	33.4	W/m^2°C
H_{evap}	Latent heat of water vaporization	2.3×10^6	J/kg
K	Permeability of meat	10^{-17}	m^2
k_m	Thermal conductivity of meat	0.4	W/m°C
L_x	Half x-length of beef sample	45	mm
L_y	Half y-length of beef sample	24	mm
L_z	Full z-length of beef sample	15	mm
ρ_c	Density of carbohydrate	1600	kg/m^3
ρ_f	Density of fat	920	kg/m^3
ρ_p	Density of protein	1320	kg/m^3
ρ_w	Density of water	988	kg/m^3
T_0	Initial temperature	13	°C
T_O	Oven temperature	175	°C
T_σ	Empirical fitting parameter for C_{eq}	52	°C
y_c	Carbohydrate composition	0	kg/kg
y_f	Fat composition	0.036	kg/kg
y_p	Protein composition	0.21	kg/kg
y_w	Water composition	0.74	kg/kg
μ_w	Viscosity of water		kg/ms
$\mathbb{R}$	Set of real numbers		

B More detailed discretization

The full derivation of the discretization of Equation 3.4 given in Equation 3.5 will be shown here. The first step is inserting the discretizations of $\frac{\partial Y}{\partial t}$, $\nabla^2 Y$ and $\mathbf{u_w} \cdot \nabla Y$, which gives

$$\begin{aligned}Y_{i,j,k}^{n+1} = Y_{i,j,k}^{n} + \frac{\Delta t}{a}\bigg(&\frac{b}{\Delta h^2}(Y_{i+1,j,k}^{n} + Y_{i-1,j,k}^{n} + Y_{i,j+1,k}^{n} + Y_{i,j-1,k}^{n} + Y_{i,j,k+1}^{n}\\ &+ T_{i,j,k-1}^{n} - 6Y_{i,j,k}^{n}) + \frac{c}{2\Delta h}(u_x(Y_{i+1,j,k}^{n} - Y_{i-1,j,k}^{n}) + u_y(Y_{i,j+1,k}^{n} - Y_{i,j-1,k}^{n})\\ &+ u_z(Y_{i,j,k+1}^{n} - Y_{i,j,k-1}^{n}))\bigg).\end{aligned} \tag{B.1}$$

Gathering the terms for the different temperatures gives

$$\begin{aligned}Y_{i,j,k}^{n+1} = Y_{i,j,k}^{n} + \frac{\Delta t}{a}\Bigg(&\left(\frac{b}{\Delta h^2} + \frac{cu_x}{2\Delta h}\right)Y_{i+1,j,k}^{n} + \left(\frac{b}{\Delta h^2} - \frac{cu_x}{2\Delta h}\right)Y_{i-1,j,k}^{n}\\ &+ \left(\frac{b}{\Delta h^2} + \frac{cu_y}{2\Delta h}\right)Y_{i,j+1,k}^{n} + \left(\frac{b}{\Delta h^2} - \frac{cu_y}{2\Delta h}\right)Y_{i,j-1,k}^{n}\\ &+ \left(\frac{b}{\Delta h^2} + \frac{cu_z}{2\Delta h}\right)Y_{i,j,k+1}^{n} + \left(\frac{b}{\Delta h^2} - \frac{cu_z}{2\Delta h}\right)Y_{i,j,k-1}^{n}\\ &- \frac{b}{\Delta h^2}6Y_{i,j,k}^{n}\Bigg).\end{aligned} \tag{B.2}$$

Defining the constants

$$C_{1,d} = \frac{b}{(\Delta h)^2} + \frac{cu_{wd,i,j,k}}{2\Delta h}, \quad d \in \{x, y, z\} \tag{B.3}$$

$$C_{2,d} = \frac{b}{(\Delta h)^2} - \frac{cu_{wd,i,j,k}}{2\Delta h}, \quad d \in \{x, y, z\} \tag{B.4}$$

$$C_3 = \frac{6b}{(\Delta h)^2} \tag{B.5}$$

gives the equation

$$\begin{aligned}Y_{i,j,k}^{n+1} =& Y_{i,j,k}^{n} + \frac{\Delta t}{a}\left(C_{1,x}Y_{i+1,j,k}^{n} + C_{2,x}Y_{i-1,j,k}^{n} + C_{1,y}Y_{i,j+1,k}^{n}\right.\\ &\left.+C_{2,y}Y_{i,j-1,k}^{n} + C_{1,z}Y_{i,j,k+1}^{n} + C_{2,z}Y_{i,j,k-1}^{n} - C_3Y_{i,j,k}^{n}\right)\end{aligned} \tag{B.6}$$

C More examples on inserted border conditions

Boundary $i = 0$

$$\begin{aligned} Y_{0,j,k}^{n+1} = Y_{0,j,k}^{n} + \frac{\Delta t}{a}\Bigg(& (C_{1,x} + C_{2,x})Y_{1,j,k}^{n} \\ & + C_{1,y}Y_{0,j+1,k}^{n} + C_{2,y}Y_{0,j-1,k}^{n} \\ & + C_{1,z}Y_{0,j,k+1}^{n} + C_{2,z}Y_{0,j,k-1}^{n} \\ & - \left(C_3 - C_{2,x}u_{wx,0,j,k}\frac{2h\beta}{\alpha}\right) Y_{0,j,k}^{n} \\ & + C_{2,x}\frac{2h\gamma}{\alpha}\Bigg) \end{aligned} \tag{C.1}$$

Boundary $i = 0$, $j = J$

$$\begin{aligned} T_{0,J,k}^{n+1} = T_{0,J,k}^{n} + \frac{\Delta t}{a}\Bigg(& (C_{1,x} + C_{2,x})T_{1,J,k}^{n} \\ & + (C_{1,y} + C_{2,y})T_{0,J-1,k}^{n} \\ & + C_1 T_{0,J,k+1}^{n} - C_2 T_{0,J,k-1}^{n} \\ & - \left(C_3 + (-C_{2,x}u_{wx,0,j,k} + C_{1,y}u_{wy,0,J,k})\frac{2h\beta}{\alpha}\right) T_{0,J,k}^{n} \\ & + (C_{2,x} + C_{1,y})\frac{2h\gamma}{\alpha}\Bigg) \end{aligned} \tag{C.2}$$

Boundary $i = 0$, $j = J$, $k = K$

$$\begin{aligned} T_{0,J,K}^{n+1} = T_{0,J,K}^{n} + \frac{\Delta t}{a}\Bigg(& (C_{1,x} + C_{2,x})T_{1,J,K}^{n} \\ & + (C_{1,y} + C_{2,y})T_{0,J-1,K}^{n} \\ & + (C_{1,z} + C_{2,z})T_{0,J,K-1}^{n} \\ & - \left(C_3 + (-C_{2,x}u_{wx,0,j,k} + C_{1,y}u_{wy,0,J,k} + C_{1,z}u_{wz,0,J,K})\frac{2h\beta}{\alpha}\right) T_{0,J,K}^{n} \\ & + (C_{2,x} + C_{1,y} + C_{1,z})\frac{2h\gamma}{\alpha}\Bigg) \end{aligned} \tag{C.3}$$

D Pseudocodes

Algorithm 1 Coupled Solver

```
 1: procedure SOLVER($T^0$, $C^0$, Configuration)
 2:     $T^1, T^0 \leftarrow 0, T^0$
 3:     $C^1, C^0 \leftarrow 0, C^0$
 4:     $dt \leftarrow$ time step-length
 5:     $dx \leftarrow$ spacial step-length
 6:     $t \leftarrow t_0$
 7:     while $t \leq t_{\max}$ do
 8:         Calculate $u_w$ from $T_0$ and $C_0$
 9:         $T^1 \leftarrow$ SOLVENEXT($T_0, Temperature$)
10:         Save $T^1$ to disk
11:         $T^1, T^0 \leftarrow 0, T^1$
12:         $C^1 \leftarrow$ SOLVENEXT($C_0, Concentration$)
13:         Save $C^1$ to disk
14:         $C^1, C^0 \leftarrow 0, C^1$
15:         $t \leftarrow t + dt$

16: procedure SOLVENEXT($\vec{U}$, PDE)
17:     $A, \vec{b} \leftarrow$ MakeAb(PDE)
18:     $\vec{U} \leftarrow \vec{U} + \frac{dt}{a}(A \cdot \vec{U} + \vec{b})$
19:     $\vec{U} \leftarrow$ ApplyDirichletBoundaries($\vec{U}, PDE$)
20:     return $\vec{U}$
```

The GitHub repository can be found here:
`https://github.com/waffelroffel/BeefSimulator`