A computational fluid dynamic model of heat and moisture transfer during beef chilling

Author:
Trujillo, Francisco Javier

Publication Date:
2004

DOI:
https://doi.org/10.26190/unsworks/5934

School of Chemical Engineering and Industrial Chemistry
The University of New South Wales

A COMPUTATIONAL FLUID DYNAMIC MODEL OF HEAT AND MOISTURE TRANSFER DURING BEEF CHILLING

by

Francisco Javier Trujillo Silvestre

A thesis submitted in fulfillment of the
requirements for the degree of

Doctor of Philosophy

February 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this submission is my own work and to the best of my knowledge it contains no material previous published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree of diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I am worked at UNSW or elsewhere, is explicitly acknowledged in this thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

ABSTRACT

The heat and moisture transfer processes during the chilling of beef carcasses were modeled using Computational Fluid Dynamics (CFD). A two- dimensional representation of the beef leg and a full three-dimensional carcass model were considered. A fully coupled heat and mass transfer processes inside the meat and on the air phase was developed on the 2D model. In the air, turbulent flow was modeled with the RNG $k-\varepsilon$ model and the boundary layer was fully solved using the FLUENT enhanced wall treatment. The turbulence model (RNG $k-\varepsilon$ model) and the near wall treatment approach (FLUENT enhanced wall treatment) used on the modelling were selected after comparing the heat and mass transfer coefficients obtained on water evaporation experiments on a circular cylinder.

The complex 3-D geometry of a beef carcass was constructed as a function of weight and fatness. A steady state CFD simulation was firstly conducted to analyze the heat (h_t) and mass transfer (h_m) coefficients around the beef surface. CFD showed that there are important local variations on the heat and mass transfer coefficients around the beef surface caused by the development of the momentum, heat and mass boundary layer. The forced heat and mass transfer coefficients around leg, loin and shoulder were correlated as a function of Reynolds and the turbulence intensity. It was found that the effects of buoyancy at air velocities higher than 0.54 m/s can be ignored given that its contribution to

the heat and mass transfer coefficient is less than 5.3% at the beginning of the chilling stage.

The simultaneous heat and mass transfer process on the air and meat phases was modeled on the 3-dimensional carcass model. However, at this stage of technology, it is impractical to do full 3D transient simulations. Thus, this work used a three-step method that consists of a steady state simulation of the flow field, determination of the local heat and mass transfer coefficients, and finally the simultaneous heat and mass transfer process simulation in the meat only.

The water diffusivity and moisture sorption isotherm of beef meat, which are physical properties that must be known in order to model the mass transfer process, were experimentally determined.

LIST OF PUBLICATIONS

1. Trujillo F. J. and Pham Q. T. (2003). Modeling the chilling of the leg, loin and shoulder of beef carcasses using an evolutionary method. International Journal of Refrigeration, 26, 224 – 231.

2. Trujillo, F. J., Yeow, P. C., Pham, Q. (2003) Moisture sorption isotherm of fresh lean beef and external beef fat. Journal of Food Engineering, 60, 357-366.

3. Trujillo F. J. and Pham Q. T. (2003). CFD modeling of heat and moisture transfer on a two dimensional model of a beef leg. 21th International Congress of Refrigeration, 17-22/8/2003, Washington DC, USA, Paper 160.

4. Trujillo, F.J., Q.T. Pham, S.J. Lovatt, M.B. Harris & J. Willix 2003 CFD Modelling of the Heat and Mass Transfer Process during the Evaporation of Water around a cylinder. 3rd International Conference on CFD in the Minerals & Process Industries, 10-12 December, 2003, Melbourne.

5. Trujillo F.J., Wiangkaew C., Pham Q.T. (2004). Comparison of three methods of estimating the effective diffusivity of moisture in meat from drying data. International Conference Engineering and Food. 8-10-March, Montpellier, France.

6. Trujillo F. J. and Pham Q. T. (2004). Drying Modeling and water diffusivity in beef meat. Paper submitted to the Journal of Food Engineering.

ACKNOWLEGEMENTS

I would like to express my deepest gratitude to my supervisor Professor Tuan Pham for giving me the wonderful opportunity of doing this PhD. It is a dream come true. I also want to thank him for his guidance, advice and help throughout all this research.

I also would like to thank:

Simon Lovatt, Jim Willix and all the staff in MINIRZ for their kindness and help in Hamilton, New Zealand, and for letting me use the experimental data obtained in their laboratories;

and Phillip Moore for reading this document helping me to correct the spelling.

TABLE OF CONTENTS

NOMENCLATURE

Symbols

a	constant of the Peleg equation
a_w	Water activity
$a_{w,a}$	Water activity of the air
$a_{w,a,s}$	Water activity of the air on the wall surface
$a_{w,m}$	Water activity of the meat
$a_{w,m,s}$	Water activity of the meat on the surface
A	Area (m^2); constant on the empirical Nusselt equation
$\overline{A}$	Average Area of the element (m^2)
$\vec{A}$	Surface area vector (m^2)
b	Constant of the Peleg equation
Bi	Biot number
c	Constant of the Peleg equation
c_p	Heat capacity (J/Kg.K)
$c_{p,m}$	Meat heat capacity (J/Kg.K)
C	Constant of the empirical Nusselt equation
C_2	Constant of the empirical Nusselt equation
C_g	Constant of the GAB equation
C_υ	Constant on the RNG $k-\varepsilon$ model

C_μ	Constant on the $k-\varepsilon$ models
d	Diameter (m); constant of the Peleg equation
d_d	Diffusion time (s)
ds	Distance between the face and cell centroids on the volume element next to the wall (m)
dx	longitudinal distance (m)
D	Diffusivity (m^2/s); distance on Boltzman's equation (m); constant of the empirical Sherwood equation
D_2	Constant of the empirical Sherwood equation
D_a	Diffusivity of water in the air (m^2/s)
D_{crit}	Critical difference between experimental and literature values
$D_{i,m}$	Diffusion coefficient for species i in the mixture.
$D_{i,m,eff}$	Effective diffusivity of component i in the mixture m (m^2/s)
D_m	Diffusivity of water in meat (m^2/s)
D_o	Arrhenius factor (m^2/s)
E	Specific energy of the fluid on the energy conservation equation (J/Kg); empirical constant on the standard wall function; constant on the empirical Sherwood equation; geometric factor.
E_o	Activation energy for water diffusion (KJ / mol)
f_{conv}	Slop of a plot of $\ln(Y_{ma})$ vs Fo
F	Constant of the Lewicki equation
Fo	Fourier number

F_t	F - statistic
$\vec{g}$	Gravitation force vector (m^2/s)
G	Constant of the Lewicki equation
G_b	Generation of turbulence kinetic energy due to buoyancy
G_k	Generation of turbulence kinetic energy due to the mean velocity gradients.
h_c	Combined heat transfer coefficient (convection + vaporization) ($W / m^2.K$)
h_{eff}	Effective heat transfer coefficient taking in account convection, radiation and evaporation ($W / m^2.K$)
h_m	Local mass transfer coefficient ($Kg /s.m^2$) or (Kg dry air / m^2. s)
h'_m	Local mass transfer coefficient ($Kg /s.m^2.Pa$)
$\bar{h}_m$	Average mass transfer coefficient ($Kg /s.m^2$)
h_j	Enthalpy of the specie j (J/Kg)
h_r	Equivalent radiation heat transfer coefficient ($W / m^2.K$)
h_t	Local heat transfer coefficient ($W / m^2.K$)
$\bar{h}_t$	Average heat transfer coefficient ($W / m^2.K$)
H	Constant of the Lewicki equation
i	Integer number that identify the nodes or volume elements
I	Unit tensor
J_{conv}	Lag factor or intercept of a plot of $\ln(Y_{ma})$ vs Fo

$\vec{J}_j$	Diffusion flux of species j (Kg /s.m^2)
J_w	Water mass flux on the wall (meat-air interface) (Kg / m^2.s)
$\bar{J}_w$	Average water mass flux on the wall (meat-air interface) (Kg/m^2.s)
k	Turbulent kinetic energy; thermal conductivity (W/m.K)
	mass transfer coefficient (kg/m^2 . Pa.s)
k_a	Thermal conductivity of the air (W/m.K)
k_{eff}	Effective thermal conductivity (W/m.K)
k_m	Thermal conductivity of the meat (W/m.K)
k_p	Turbulent kinetic energy on the cell centroid of the volume
	element next to the wall
K	Ratio of heat to mass transfer; constant of the GAB equation;
	Shape factor
K_b	Woltzman constant
L	Full length of the beef side (m), thickness of the meat sample (m)
Le	Lewis number
m	Constant on the Nusselt empirical equation
M	Total mass of the beef side (Kg)
M_w	Molecular weight of mixture
MSI	Moisture sorption isotherm
n	Constant used on the Lewis relationship; constant on the empirical
	Nusselt equation, normal unit vector
Nu_c	Nusselt number of combined convection

Nu_f	Nusselt number of forced convection
$\overline{Nu_t}$	Average Nusselt number
Nv	Total number of nodes or volume elements
p	Pressure (Pa)
P	Constant on the thermal standard wall function;
	Perimeter (m)
P_a	Vapor pressure of the air (Pa)
P_c	Constant on the mass standard wall function
P_h	Saturated water pressure at the wet bulb temperature T_h (Pa)
Pr	Prandtl number
Pr	Turbulent Prandtl number
P_s	Vapor pressure in the air-meat interface (Pa)
P_T	Total pressure (Pa)
P_V	Vapor pressure of pure water (Pa)
P_{ws}	Vapor pressure of pure water at the air-meat interface temperature
	(Pa). $P_s = a_w P_{ws}$
q	Heat flux on the wall interface (W/m^2)
q_{conv}	Convective heat flux on the wall interface (W/m^2)
q_{evap}	Evaporative heat flux on the wall interface (W/m^2)
q_{rad}	Radiative heat flux on the wall interface (W/m^2)
Q_{Exp}	Total experimental heat load of the beef (J)

Q_{FD}	Total heat load of the beef calculated with the FD model (J)
Q_{Max}	Total heat lost after slaughter (J)
r	Constant on the Sherwood empirical equation
R	Ideal gas constant = 8.314 (KJ $mol^{-1}K^{-1}$); cylinder radius (m)
Re	Reynolds number
R_{eq}	Radius of the equivalent cylinder
RH	Relative humidity
RMS	Average Root mean square percentage error
s	Constant on the empirical Sherwood equation
S	Source term on the general transport equation; product surface area (m^2)
$\overline{Sh_m}$	Average Sherwood number
Sc	Schmidt number
Sc_t	Turbulent Schmidt number
S_ε	User-defined source term for k
S_k	User-defined source terms for ε
S_ϕ	Source of ϕ per unit volume
t	Time (s)
t_c	Chilling center time (s)
T_h	Wet bulb temperature (K)
t_m	Chilling mass average time (s)
t_{shape}	Cooling time of a particular shape (s)

t_{slab}	Cooling time of the slab (s)
T	Temperature (K)
T_a	Temperature of the air (K)
$T_{a,s}$	Temperature of the air on the wall surface (K)
T_c	Temperature of the air on the cell centroid of the volume element next to the wall (K)
T_{Death}	Temperature of the beef after death (K)
T_{eq}	Equilibrium temperature (K)
T_f	Temperature on the wall interface centroid of the volume element next to wall (K)
T_{film}	Film temperature (K)
T_h	Wet bulb temperature (K)
T_m	Temperature of the meat (K)
$T_{m,s}$	Temperature of the meat on the interface (K)
T_o	Initial temperature (K)
T_r	Temperature at any point along the radius (K)
T_w	Wall temperature (K)
$\overline{T}_w$	Average wall temperature (K)
Tu	Turbulence intensity
v	Velocity (m/s), specific volume of the sample (m^3 / kg dry material)
$\vec{v}$	Velocity vector (m/s)

$\vec{\bar{v}}$	Mean average velocity vector (m/s)
$\vec{\bar{v}}^T$	Transpose mean average velocity vector (m/s)
$\bar{v}_j$	j component of the mean average velocity vector (m/s)
V	Air velocity at the wind tunnel inlet (m/s), volume (m^3)
$V_{initial}$	Initial volume of the meat sample (m^3)
V_{final}	Final volume of the meat sample (m^3)
x	Position (m)
X	Water content of the meat (kg water / kg dry material); characteristic dimension (m)
X_{eq}	Equilibrium water content (kg water / kg dry material)
X_{EXP}	Experimental value of the water content (kg water / kg dry material)
X_{lit}	Literature value of the water content (kg water / kg dry material)
X_m	Average water content (kg water / kg dry material)
X_{mean}	Mean water content X (average of the replicates)
X_{mg}	Constant of the GAB equation
X_{MODEL}	Value of X obtained by a mathematical model
X_o	Initial water content (kg water / kg dry material)
y^*	Non-dimensional distance to the wall
y^+	Non-dimensional distance to the wall

y_c^*	Mass sub-layer thickness
$\bar{y}$	Water mol fraction (mol air / mol total)
Y	Water weight composition (Kg water/Kg total)
Y_a	Moisture content of the air (Kg water / Kg dry air); moisture content of the air in equilibrium with the meat surface (Kg water / Kg dry air)
Y_c	Water weight composition on the cell centroid of the volume element next to the wall (Kg water/Kg total)
Y_{cc}	Fractional unaccomplished center temperature
Y_f	Water weight composition on the wall interface centroid of the volume element next to wall (Kg water/Kg total)
Y_i	Water weight fraction composition of i (Kg i/Kg total)
Y_m	Water weight composition of the meat (Kg water/Kg total)
Y_{ma}	Fractional unaccomplished mass average temperature change
Y_M	Contribution of the fluctuating dilatation in compressible turbulence to the overall dissipation rate
Y_s	Moisture content of the air in equilibrium with the meat surface (Kg water / Kg dry air)
Y_w	Water weight composition at the wall (Kg water/ Kg total)
$\bar{Y}_w$	Average water weight composition at the wall (Kg water/ Kg total)
Y^*	Non-dimensional water composition on the near wall cell

Greek symbols

α	Thermal diffusivity (m^2/s)
α_ε	Inverse effective Prandtl numbers for ε
α_k	Inverse effective Prandtl numbers for k
β	Volume-shrinkage coefficient
ΔH_{vap}	Water heat of vaporization (J /Kg).
Δ_t	Time interval (s)
Δx	Longitude of the control volume element (m)
ε	*D*issipation rate of turbulence
ε_r	Radiation emissivity
ϕ	Field variable
κ	Von Karman constant
λ	Thermal conductivity (W/m.K)
μ	Viscosity (Kg/m.s)
μ_{eff}	Effective viscosity (Kg/m.s)
μ_t	Turbulent viscosity (Kg/m.s)
ρ	Density (Kg /m^3)
ρ_{bo}	Bulk density of the sample at zero value of moisture content (Kg dry material / m^3)
ρ_m	Density of the meat (Kg/ m^3)
ρ_s	Concentration of dry solids (kg dry material/m^3)

σ	Stefan-Boltzmann constant; standard deviation
Γ	Diffusivity (m^2/s)
τ	Characteristic time for molecular movement
$\bar{\tau}_{eff}$	Effective stress tensor (N/m^2)
$\hat{\upsilon}$	$= \mu_{eff}/\mu$

Subscripts

CV	Control volume
e	East face
E	East node
o	Initial
w	West face
W	West node
P	Central node of the element

Superscripts

t	Present time (s)
$t+1$	Future time (s)

1. INTRODUCTION

Refrigeration is the most widely used method for preserving the quality of fresh meat. After slaughter, meat carcasses are stored in refrigerated ventilated chillier rooms. This process operation involves the simultaneous transfer of heat and moisture. The temperature of the product is reduced by heat conduction within the meat while heat and water exchange with the air takes place at the carcass interface. The intensity of these phenomena determines the cooling time, weight loss, temperature profiles and water activity on the surface. The first two are economically important criteria of process efficiency (Mirade *et al.*, 2002). The last two, temperature profile and surface water activity, are of great importance to food safety through their effect on microbial growth.

Many simplified models, which are very useful for design, have been developed to predict heat load during chilling and weight loss (Lovatt *et al.*, 1993; Kuitche and Daudin, 1996; Chuntranuluck *et al.*, 1998; Davey and Pham, 1997, 2000). A common shortcoming of these methods is that they use empirical equations to determine the heat and mass transfer coefficients. These equations have been developed for air flow around simple geometries but they do not necessarily represent the complex flow pattern around real carcasses. They also use average heat transfer coefficients over the surface, neglecting local variations.

Additionally, in order to model the mass transfer process, the water diffusivity on meat and the moisture sorption isotherm must accurately be known. However, there is a lack of reliable data on these two mass transfer-related properties.

Therefore, the aim of the present work is to conduct a CFD modeling of the heat and moisture transfer processes during the chilling of beef carcasses, and to experimentally determine the meat water diffusivity and the moisture sorption isotherm.

The literature review and the objectives of the project are contained on chapter 2. The meat moisture sorption isotherm, which was experimentally determined in collaboration with honours student Pei Ching Yeow (Yeow, 2001), is reported on chapter 3. Chapter 4 contains the meat water diffusivity. The experiments were conducted with the help of coursework master student Chaiyan Wiangkaew (Wiangkaew, 2003).

The selection of the turbulence model and the near wall treatment approach used on CFD modelling may drastically affect the accuracy and reliability of the results. No single turbulence model is universally accepted as being superior for all classes of problems and most of the widely known models, such as the standard κ-ε model, have been developed for fully turbulent flows. Turbulent flows are also significantly affected by the presence of walls. The modelling of flows near wall significantly impacts the reliability of numerical solutions.

Chapter 5 contains some experimental validations, conducted on a circular cylinder under similar conditions of meat chilling, which were used to find the turbulent model and the wall treatment that best represent the beef chilling process. The experiments were fully designed and conducted by Simon Lovatt, Mark Harris and Jim Willix in MINIRZ, Hamilton, New Zealand.

Chapter 6 contains a numerical simulation of the simultaneous heat and mass transfer in a 2D ellipse model of a beef leg. It is established the mathematical model for the unsteady state simulation. The complex 3-D geometry of a beef carcass is constructed on chapter 7. Then, a steady state simulation was conducted to analyze the heat (h_t) and mass transfer (h_m) coefficients around the beef surface. To study the influence of free convection, the CFD was run both with and without buoyancy effects.

The unsteady heat and mass transfer processes in beef chilling on the complex 3D geometry is established on chapter 8. A three step method was used to simplify and accelerate the simulation. Three runs were completed and the heat load, weight loss, and surface and centre leg, loin and shoulder temperatures were compared with experimental data. Conclusions of the work and recommendations are contained in chapter 9.

2. LITERATURE REVIEW

2.1 Introduction

Refrigeration is the most widely used method for preserving the quality of fresh meat (Pham, 2001b). To ensure that refrigeration is effective, it is necessary to calculate processing times, product temperatures, heat loads and water diffusion into and out of the product. Early models concentrated on the prediction of chilling time based on the centre leg temperature, which is the thickest part of beef carcasses and therefore the place where temperature keeps at higher levels. The variation of heat load with time is important to beef processing and other food cooling processes as it comprises a large part of the total processing cost. Heat load determines the size of the refrigeration system and hence affects the capital and operation costs. If the system is incorrectly designed, product quality problems such as uncontrolled microbial growth, bone taint and excess weight loss can occur. Weight loss has also been a mayor concern of researchers. For instance, weight loss through evaporation (typically of the order of 2% of carcass weight) is an economic problem of great concern to the beef industry. It has been estimated that several hundred millions dollars are lost each year through evaporation from meat alone, in Australia, during refrigeration.

The chilling of fresh meat involves the simultaneous transfer of heat and moisture. Because food quality and safety (microbial growth and spoilage) depend on temperature and surface moisture, understanding the heat and

moisture transfer is of great importance to public health. Therefore, heat transfer and weight loss during beef processing has been the target of research for decades and a number of numerical models have been built.

Davey (1998) made a very complete literature review amount the different beef chilling models that have been proposed. Pham (2001a, 2001b) detailed explains the most popular methods and techniques of prediction of cooling, freezing, thawing and heat load on food processing. Thus, the aim of this new literature review is to show some of the models that clearly represent the evolution and progresses in this research area in the last decade (sections 2.2 to 2.4).

Additionally, to successfully model the heat and mass transfer during beef chilling, the transport properties and other physical properties of beef meat must be established. The thermal and calorimetric properties of meat are well known (Pham, 1989, 1996). However, there is a lack of reliable data on mass transfer-related properties, namely moisture sorption isotherm (water activity vs. water content) and water diffusivity. A review on these two important properties will fully covered on sections 2.5 and 2.6 respectively. Based on this examination, the objectives of this research are proposed on section 2.7. Sections 2.8 and 2.9 concentrated on a brief review of CFD modelling and turbulence models respectively.

2.2 Simplified methods based on analytical solutions

A number of simplified dynamic models predicting product heat load during food cooling have been developed. Marshall and James (1975) modelled a continuous vegetable freezing tunnel by using a lumped parameter system with an apparent product specific heat capacity which varied through the process to account for latent heat. The product heat load was described by:

$$q = V \cdot c_P \frac{dT_m}{dt} = h_t A(T_a - T_m) \tag{2.1}$$

The model was accurate as long as the refrigeration plant was working correctly (Lovatt *et al*, 1993), but it was difficult to extend to other situations. Further, equation (2.1) implicitly requires that $T_s = T_m$, which is only true if the Biot number $Bi = 0$. Models such as this assume that there is no internal temperature gradient. Given that:

$$Bi = \frac{h_t \cdot X}{k_m} \tag{2.2}$$

The model is true when the characteristic dimension X is very small, the thermal conductivity k_m is large and the heat transfer coefficient h_t is small. This model is expected to be accurate only under low Biot number conditions (typically less than 5% error on the heat load below $Bi = 0.1$). This approach is therefore inaccurate for most food freezing situations (where the Biot number is more commonly between 1 and 10).

2.2.1 Models based on analytical solutions of simple geometries

Seeking a model applicable to a wide range of Biot numbers, but avoiding a numerical solution, Cleland and Earle (1982) calculate solutions for simple reference shapes, and use a geometric factor (E) to estimate solutions for more complex shapes from that reference. The general equation for one dimensional transient heat conduction is given by:

$$\frac{1}{x^n}\frac{\partial}{\partial x}\left(x^n \frac{\partial T}{\partial x}\right) = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.3}$$

where n = 0, 1, 2, for an infinite slab, infinite cylinder and sphere, respectively. For these three cases, there are exact analytical solutions. Lovatt *et al.* (1993) used the sphere as a reference shape arguing that it is the most appropriate of the simple shapes to use as a reference due to the method that was used by them to calculate the geometric factor (E).

For a sphere, the solution for the fractional unaccomplished mass average temperature change, Y_{ma}, for cooling under constant conditions is:

$$Y_{ma} = \frac{T_m - T_a}{T_o - T_a} = \sum_{i=1}^{\infty} \frac{6Bi^2}{\beta_i^2\left[\beta_i^2 + Bi(Bi-1)\right]}\exp\left(-\beta_i^2 Fo\right) \tag{2.4}$$

Where β_i^2 is the root of:

$$\beta \cot \beta + (Bi - 1) = 0 \tag{2.5}$$

During all but the initial period of the cooling process, the terms where i>1 are insignificant. In that case, the slope of the ln (Y_{ma}) ought to equal the slope of the integrated equation (2.1) when expressed in terms of Y_{ma}. This implies that:

$$h_t A = \frac{E}{3}\frac{V\beta_1^2 k_m}{X^2} \tag{2.6}$$

Substituted into equation (2.1), this gives:

$$q = Vc_P \frac{dT_m}{dt} = \frac{E}{3}\frac{V\beta_1^2 k_m}{X^2}(T_a - T_m) \tag{2.7}$$

This is not an exact solution, although the importance of the implied assumption that $T_s = T_m$ is considerably reduced compared with equation (2.1). A corollary of equation (2.6) implies that:

$$\frac{6Bi^2}{\beta_i^2\left[\beta_i^2 + Bi(Bi-1)\right]} = 1 \tag{2.8}$$

This is true with less than 10% error for Bi<3.5, indicating that better heat load estimates may be expected from equation (2.7) than from equation (2.1).

Cleland and Earle (1982) defined the geometric factor as:

$$t_{shape} = \frac{t_{slab}}{E} \tag{2.9}$$

They suggested empirical formulas to calculate (E) for some shapes. Hossain *et al.* (1992) developed and analytical method to estimate the geometry factor (E), but it must be noted that this method is strictly derived only for the phase change case (thawing or freezing). E values for chilling must be somewhat different for freezing. In the case of carcasses, with far more complex shapes, it is impossible to define a shape factor on a theoretical basis and it must be adjusted based on experimentation.

The heat load predicted by this method will underestimate the actual load at the start of the chilling process and this error will increase as Bi $\rightarrow \infty$. Lovatt *et al.* (1993) tested this method against finite difference (FD) calculations under a wide range of conditions for those product shapes to which FD methods may be applied. It was found to predict the product heat load to within 10% of the FD estimate for all cases, except at the start and at the end of the cooling process. According to them, the method is capable of extension to shapes not easily handled by FD methods and it requires much fewer computational resources than FD.

Lin *et al.* (1996a) developed a simple method to calculated chilling time, mass average and centre temperature applicable to two dimensional irregular shapes. The method was based on the first term approximation to the analytical solution for convective cooling of a sphere in conjunction with two geometric parameters determined by empirical equations. The real geometric shape is related to an equivalent infinite ellipse using dimensional measurements of the shape. The equations are:

$$Y_{cc} = L_c \exp\left(-\frac{E}{3}\alpha^2 \frac{k_m t_c}{\rho c_p R^2}\right) \tag{2.10}$$

$$Y_{ma} = L_m \exp\left(-\frac{E}{3}\alpha^2 \frac{k_m t_m}{\rho c_p R^2}\right) \tag{2.11}$$

for the fractional unaccomplished centre and mass average temperature respectively, or:

$$t_c = \frac{3\rho c_p R^2}{\alpha^2 k_m E} \ln\left(\frac{L_c}{Y_{cc}}\right) \quad (2.12$$

$$t_m = \frac{3\rho c_p R^2}{\alpha^2 k_m E} \ln\left(\frac{L_m}{Y_{ma}}\right) \quad (2.13)$$

for the chilling centre and mass average time respectively. The constants E, L_c and L_m are calculated with the empirical equations:

$$E = \frac{Bi^{4/3} + 1.85}{\frac{Bi^{4/3}}{E_\infty} + \frac{1.85}{E_0}} \quad (2.14)$$

$$L_c = \frac{Bi^{1.35} + \frac{1}{\beta}}{\frac{Bi^{1.35}}{L_\infty} + \frac{1}{\beta}} \quad (2.15)$$

$$L_m = \mu_l L_c \quad (2.16)$$

Where

β = (second shortest dimension of object)/(shortest dimension) (2.17)

$$E_0 = \left(1 + \frac{1}{\beta}\right)\left[1 + \left(\frac{\beta - 1}{2\beta + 2}\right)^2\right] \quad (2.18)$$

$$E_\infty = 0.75 + 1.01\left[\frac{1}{\beta^2} + 0.01\exp\left(\beta - \frac{\beta^2}{6}\right)\right] \quad (2.19)$$

$$L_\infty = 1.271 + 0.305\exp\left(0.172\beta - 0.115\beta^2\right) \quad (2.20)$$

$$\mu_l = \left(\frac{1.5 + 0.69Bi}{1.5 + Bi}\right)^2 \quad (2.21)$$

The major benefits of Lin's model is the simplicity of the algebraic calculations and it accuracy for predicting chilling rates at the themal center position. The method was extended to three dimensional irregular shapes (Lin *et al.*, 1996b)

The main disadvantage of these simplified models is that they do not take in account the water evaporation which causes weight loss. In the case of beef carcasses, this loss is around 1.5 – 2.3% of weight and represents around 20 times the cost of the refrigeration process. Surface water evaporation also considerably modifies the transfer phenomena around the product, as the heat for evaporation joins the sensitive heat at the product surface.

These models also assume constant air conditions: air temperature, air velocity, humidity etc. However, it appears that industrial systems, of whatever type (batch or continuous), used in meat chilling operate under variable air conditions (Kuitche *et al.*, 1996a). This variations can be voluntary or not and has significant influence on product chilling.

2.2.2 A simple model of temperature and weight loss kinetics for time variable conditions

Kuitche *et al.* (1996a) developed a simplified mathematical model, similar to the previous one, but that accounts for: (1) product surface water evaporation, and (2) the variable chilling conditions that exist in industrial chillers. The model was tested and compared to measurements carried out on wet plaster cylinders of

homogeneous composition (Kuitche *et al.*, 1996b). The same testing process was then applied to a mould of the hindquarters of pork carcasses of homogeneous composition, while adjusting a shape factor. Finally the calculations were compared to chilling kinetics measured on pork carcasses in order to assess the effect of composition heterogeneity (Daudin and Kuitche, 1996).

Kuitche *et al.* (1996a) modelled the carcass hindquarter as an infinite cylinder. Analytical solution of the equation (2.3) for the cylinder shape are available in the literature (equation similar to 2.4) but it does not allow for the variable chilling conditions observed in industrial systems, mainly changes in temperature and air velocity. Moreover, the solution is given for purely convective boundary that not takes in account evaporation or radiation. Thus, they modelled the boundary condition taking in account surface evaporation and radiation:

$$-k_m\left(\frac{\partial T}{\partial r}\right)_R = (h_c + h_r)(T_s - T_a) + h'_m \Delta H_{vap}(P_s - P_a) \tag{2.22}$$

Where h_r is the equivalent radiation heat transfer coefficient defined as:

$$h_r = 4\sigma\varepsilon T_a^3 \tag{2.23}$$

The coupling of the heat and water transfer means that the product temperature tends towards an equilibrium temperature instead of that of air T_a. The equilibrium temperature of a product having a surface water activity equal to one, if radiation transfer is negligible, is equivalent to the wet bulb temperature T_h. During meat chilling, surface drying (water activity less than 1) and the presence

of radiation flow imply an equilibrium temperature higher than T_h but smaller than T_a. However, in practice, relative air humidity is high, and surface drying only occurs at the beginning, followed by a re-wet process of the meat surface. Therefore, for simplicity, they considered the equilibrium temperature equal to T_h. Thus, equation (2.22) can be expressed as:

$$-k_m\left(\frac{\partial T}{\partial r}\right)_R = h_{eff}(T_s - T_h) \tag{2.24}$$

Where h_{eff} is the effective heat transfer coefficient taking in account convection, radiation and evaporation.

$$h_{eff} = h_c\left[1+\frac{h_r}{h_c}+\left(\frac{h'_m \Delta H_{vap}}{h_c}\frac{P_s - P_h}{T_s - T_h}\right)\right] \tag{2.25}$$

The ratio $h'_m \Delta H_{vap}/h_c$ is calculated with the Lewis relationship. An effective Biot number can be defined:

$$Bi_{eff} = \frac{h_{eff} R}{k_m} \tag{2.26}$$

The effective Biot number depends on cooling medium property and product surface; it varies all around the chilling process. This variation, due to the heat mass coupling, is incompatible with the analytical solution usually employed, even with constant chilling conditions.

To make an analytical solution possible, Kuitche *et al.* (1996a) made the time discrete. The elementary calculation is done on a time step Δt, during which Bi_{eff} is assumed constant. This consists of an analytical prediction of product

temperature distribution at the end of the increment Δt; the initial one being known. Carslaw and Jaeger (1959) gave an analytical solution for any given initial temperature distribution f(r):

$$T_r = T_h + \frac{2}{R^2}\sum_{n=1}^{n=\infty}(Y_n X_n) - 2Bi_{eff}T_h\sum_{n=1}^{n=\infty}Z_n \tag{2.27}$$

Where Y_n, X_n and Z_n are function of the Fourier and effective Biot numbers, and the temperature distribution at the beginning of the interval of time Δt.

The instantaneous evaporated water per unit time is expressed with the equation:

$$\left(\frac{dM}{dt}\right) = h'_m S(P_s - P_a) \tag{2.28}$$

The surface partial water pressure P_s is dependent on the surface meat water activity (a_w). However, as an approximation and to keep the model simpler, the water activity was assumed to be equal 1.

They test the model against a wet plaster cylinder without using any geometric adjustment factor (Kuitche *et al.*, 1996b), the temperature kinetics at different points within the sample were predicted under constant and time variable conditions with an error margin of 1°C. The relative error in chilling time prediction was 5%. Weight lost at the end of chilling was calculated with an absolute error of 0.1%.

In the case of the hindquarter mould, the radius (R_{eq}) of the cylinder equivalent to the mould was obtained by minimizing the sum of squares of the calculated

and measured temperature differences, for the entire chilling period and for all the experiments. The shape factor was defined as the ratio of the equivalent radius to the mean geometric radius deduced from the perimeter of the measurement section: $K = R_{eq}/(P/2\pi)$. For the mould in question, K was equal to 0.97 which means that it's chilling time, when determined by core temperature, is very close that of an infinite cylinder with the same perimeter. The weight loss profiles were first determined from the calculated surface temperature profiles, using the exchange surface area measured on the mould on equation (2.28). They overestimate weight loss by around 30% and suggested that the difference comes from the fact that the surface temperature of the calculated cylinder, which corresponds well with the mean surface temperature of the thickest section of the sample, is obviously higher than the mean surface temperature of the sample. The saturated water vapour pressure being an increasing function of temperature, the driving force $(P_s - P_a)$ is therefore overestimated. Thus, they calculated an equivalent exchange surface area by the same procedure as described above for R_{eq}. The equivalent exchange surface area was around two thirds of the real surface area. With this area, the difference in prediction the weight loss at the end of the chilling was 0.08% on average.

For real hindquarters Daudin and Kuitche (1996) found a shape factor K equal to 0.92 that is below 0.97, the value obtained for the hindquarter plaster mould. They explain that difference with uncertainties on the thermal diffusivity given the heterogeneity of the meat. The mean diameter was correlated with the pork

carcass weight. The equivalent surface area was adjusted by comparing with the calculated and measured weight loss kinetics. It was expressed as a function of the carcass weight.

This model can be extended to beef carcasses but it has the following problems: (1) the model is valid for the hindquarter but given the complex geometry of the full carcasses it may not be suitable for the full carcass. (2) Multiple experiments are required to adjust the shape factor and the equivalent surface area. (3) The model can not predict local variations in temperature and water activity on the meat surface, both important factors to control the bacterial growth.

2.2.3 A simple chilling time modelling taking in account evaporation

Chuntranuluck *et al.* (1998) proposed a simple method to calculate the time and temperature profiles taking in account the effect of evaporation. Analytical solutions of equation (2.3), when cooling occurs by convection only, are infinite series in which, except at short times, only the first term is important. Most practical chilling processes meet this criterion and the so-called one term approximation is used:

$$Y_{ma} = \frac{(T_m - T_a)}{(T_o - T_a)} = J_{conv} \exp(-f_{Conv} Fo) \tag{2.29}$$

When ln Y_{ma} is plotted against Fo, a straight line of slope - f_{Conv} is obtained, with T_m approaching T_a, the equilibrium temperature as $t \rightarrow \infty$. However, when evaporation is introduced, linear plots are no longer obtained. The curvature of

the plot depends on the true equilibrium temperature reached as $t \rightarrow \infty$ (which is no longer T_a and depends of on both a_w and relative humidity RH). Kuitche *et al.* (1996a) use the wet bulb temperature as the equilibrium temperature. Chuntranuluck *et al.* (1998a) rejected that assumption because it implies that a_w=1. Hence, they defined the equilibrium temperature solving equation (2.22) at steady state and neglecting the heat of radiation:

$$T_{eq} = T_a - \frac{h'_m \Delta H_{vap}}{h_c}(a_w P_{ws} - P_a) \tag{2.30}$$

Because ΔH_{vap} and P_{ws} are function of T_{eq}, equation (2.29) must be solved iteratively. Using T_{eq}, they defined Y_{ma} as:

$$Y_{ma} = \frac{(T_m - T_{eq})}{(T_o - T_{eq})} = J_{evap} \exp(-f_{evap} Fo) \tag{2.31}$$

They found that using T_{eq} from equation (2.30), plots of ln Y_{ma} vs Fo from equation (2.31) became sufficiently linearized. They found the values of $(-f_{evap})$ and (J_{evap}). Finally, they correlated the values of $(-f_{evap})$ and (J_{evap}) to the environmental and product conditions developing empirical equations (reported in Chuntranuluck et al., 1998a). The model is very simple and fast. It was tested against experimental measurements made by chilling cylindrical samples (Chuntranuluck et al., 1998b). The predicted temperature was in agreement with the experiments (within 6%). The agreement of the chilling time prediction was within 11%.

The model has the following problems: (1) it does not predict the weight loss caused by evaporation. (2) It can not predict local variations in temperature and water activity on the meat surface. (3) It has not been tested for meat carcasses chilling where the geometry is irregular and adjusting shape factor might be necessary.

2.3 Modelling the heat and mass transfer using numerical methods

2.3.1 Finite Difference models

Davey and Pham (1997) developed a model for predicting the dynamic heat load and weight loss during beef chilling using the finite difference method. The irregular beef geometry was approximated by a combination of seven cylinders and slabs. The dimensions of each region were defined by empirical correlations based on geometry measurements taken on 71 beef carcasses and correlated with the beef side weight and P8 fatness grade.

The transient heat transfer equation 2.3 in each of the seven regions was solved with a finite difference (FD) approximation and a Crank-Nicholson scheme. The total heat load was then calculated from the changes in product enthalpies over each time step. Each region was divided into 10 nodes. It was assumed there was no axial heat flux. A similar version of equation 2.22 was used to define the heat transfer from product to air during chilling taking in account natural and forced convection, radiation and evaporation. The forced and natural convection

coefficients were calculated using empirical equations for air flow over a slab and a vertical surface respectively.

Slaughter floor air temperature and humidity were measured. Neither air velocity nor the residence time on the slaughter floor were accurately known. Therefore, an air velocity of 0.5 m/s was assumed and a heat balance was used to determine the time on the slaughter floor. The length of the time on the slaughter floor was completed when the total heat remaining in the simulations was within 1% of the total experimental heat loss during chilling. This time was between 1h and 2h. The surface water activity was assumed to be constant and equal to 0.85.

The predicted heat removed during the first 2 hours of chilling was on average 12.6% higher than the experimental value. The average percentage error in the weight loss prediction after 20 hours was 1.25% (% total carcass weight) . The leg temperature was generally over predicted by a few degrees (Davey, 1998), suggesting that the cylinder shape used to represent the leg on the FD model had a larger diameter than the actual beef leg. The loin centre was also over predicted, although not as much as the leg. This was probably due to the fact that the slowest cooling point of the loin is near the vertebrae, and it has heat transfer surfaces on three sides rather than two as in the slab representation in the FD model. The shoulder centre temperatures were not adequately predicted, being well below the experimental values in most cases. According to the Davey, the good heat load and average weight loss predictions shows that even though the

temperature prediction for shoulder is inadequate predicted and the loin temperature is over-predicted, these did not have a significant effect on the overall accuracy of the model for predicting heat load and weight lost. Davey and Pham (1997) suggested that improvements in accuracy would probably have arisen if slaughter floor conditions floor and chillier conditions had been more accurately known.

This model has the following disadvantages: (1) it is not accurate in predicting centre temperature evolution (2) It can not predict local variation in temperature and water activity on the meat surface.

Pham and Karuri (1999) used a finite difference technique to predict the cooling rate, evaporation rate, temperature and surface water activity during the chilling of a beef side. They used the 7 section represented by slabs and cylinders of Davey and Pham (1997). A separated discretization grid was used to solve the heat and mass transport equations since diffusion of water during beef chilling only takes place near the surface. The surface water activity was described by a simple linear sorption isotherm. The model was able to show the reduction and further increase of the surface water activity cause by the drying and re-wetting process.

2.3.2 Finite element models

Malikarjuan and Mittal (1994) develop a two dimensional heat and mass transfer model of beef carcass chilling. The carcasses were divided into five zones and the cross-sectional structure within a zone was considered uniform. The model was solved using finite element methods. Centre temperature and weight loss was recorded and compared with experimental data. The simulated results were in good agreements with the experiments. Simulated temperate and weight loss percentage were within 2.7C and 0.4% (% total carcass weight) respectively.

This model has the following disadvantages: (1) it considered that water activity on the surface was equal to 1. Thus the model can not predict the evolution of a_w vs. time. (2) The mesh is not fine enough close to surface; therefore it can not accurately predict the water content in the surface. (3) It calculate the heat transfer coefficient with empirical equations, thus, it does not take into account changes of the heat transfer coefficients with the boundary layer. It can not predict local variation in temperature and water activity on the meat surface.

Davey and Pham (2000) also developed a finite element model to predict the heat load and weight loss during beef chilling. The beef geometry was approximated by 13 sections, each represented by a two dimensional finite element grid using triangular elements. A modification to the boundary condition used by Davey and Pham (1997) was introduced to account for the thick layer of fat observed on parts of the loin and shoulder sections. This fat layer was modelled as an

additional resistance to the heat transfer. The weight loss was modelled in the same way as established by Davey and Pham (1997). The heat removal during the first 2 h predicted by the model was on average 5.6% higher than the experimental value. The average percentage error in the predicted weight loss after 20 h was 2.3% (% total carcass weight). The leg and loin centre temperature are generally quite well predicted. The shoulder centre temperature was over predicted at the start of chilling and under predicted toward the end probably due to uncertainties in the internal heat generation.

A problem with the above models is that the heat transfer from product to air during chilling occurs by simultaneous convection (on the air) and conduction (on the carcass), but only heat conduction is modelled numerically. The convection heat transfer of air flow over the beef side is not solved simultaneously with the conduction heat transfer in the beef side but is approximated by an empirical heat transfer coefficient relationship, using an empirical relationship for the average heat transfer coefficient for simple geometries such as cylinders or slabs.

Traditionally, only the average surface heat transfer coefficients are determined experimentally or derived from existing correlations for simple geometries (Verboven *et al.*, 1997). Little attention has been paid so far to the study of the variation of the surface heat transfer coefficient around the surface for complex geometries. However, it is known that the value of the surface heat transfer

coefficients varies along the surface of the product, depending on the development of the boundary layer. Kondjoyan and Daudin (1993a) showed that the convective transfer coefficient value varies from -40% to +40% of the mean value around a circular cylinder. Variations in the case of real food geometries are still more complex. Kondjoyan and Daudin (1997) measured the heat and mass transfer coefficient at the surface of a pork hindquarter. They found that the local heat transfer coefficient can be very different from one location to another of the body surface. The effect of turbulence intensity on the mean transfer coefficient is important and different from what happens on a cylinder.

Additionally, while there is a large body of data on the heat transfer coefficient, there is a lack of systematic information on mass transfer coefficients (Tocci and Mascheroni, 1995). The heat transfer coefficient has been usually calculated from the heat transfer coefficient using empirical equations like the Chilton-Colbourn analogy. However, deviations from this analogy have been reported, especially at low air velocities where the radiation effects start becoming important (Daudin and Swain, 1990). Thus, large deviation in the calculated temperature and water activity may result along the meat product surface.

2.4 Modelling using Computational Fluid Dynamics (CFD)

Computational fluid dynamics (CFD) appears as a useful tool to determine local variations on heat and mass transfer coefficients around real and complex food

geometries. CFD has successfully been applied to food processing applications like static mixers, pipe flow, baking ovens, chillers, retail display cabinets etc. (Scott and Richardson, 1997). Nguyen and Pham (1999) pointed in this direction using the CFD package Fluent. The complex geometry of the beef carcasses was for the first time represented in 3-dimensions by a grid of about 100.000 nodes. The turbulence airflow over the beef side was modelled using the RNG (Re Normalisation Group) κ-ε turbulence model. The external convection and the internal conduction heat transfer process were simultaneously modelled. The predicted heat load calculated was lower than the available experimental data; however, the effect of the evaporative heat transfer was not taken into account.

The heat transfer coefficients calculated via CFD can be used in simplified models that are computationally economical and attractive for the industry. Thus, Pham and Nguyen (2000) used the average convective heat transfer coefficient, calculated with the previous model, to predict the heat load with the finite difference model of Davey and Pham (1997).

Hu and Sun (2000) modelled the heat and mass transfer on the air side during the cooling of cylindrical shaped cooked meat. They used the CFD software CFX to calculate the average heat transfer coefficient but they did not predict its local variations. They took in account the convection, radiation and evaporation on the meat surface. To save time, they propose to solve the problem in three steps: (1) steady state simulation of the field flow only. (2) Determination of the average

convective heat transfer coefficient. (3) Simulation of the heat and mass transfer in the meat using the average heat transfer coefficient. The weight loss was modelled in a similar way to Davey and Pham (1997). Water diffusion inside the meat was not modelled. The mass transfer coefficient was calculated from the heat transfer coefficient using the Lewis relationship and the water activity was assumed constant and equal to 0.90.

Hu and Sun (2001) improved the previous model by calculating the local heat transfer coefficient cell by cell around the surface. The same three steps were followed but using the local heat transfer coefficient instead of the global one. The mass transfer was not treated rigorously via CFD modelling. They followed the same approach of Hu and Sun (2000) to determine the weight loss with the only difference that the water activity was assumed to be equal to the air humidity, which is not necessarily true.

Sun and Hu (2002, 2003) developed a CFD simulation to predict heat and mass transfer during vacuum cooling of porous food. The simulation allowed the prediction of temperature distribution, weight loss and moisture content of the meats at low saturation pressure. It accounted for the effects of pressure, temperature, density, water content, thermal shrinkage and anisotropy of the food. As the convective heat loss of the meat is insignificant comparing with the evaporative heat loss, only the meat was modelled via CFD.

Mirade *et al.* (2002) used CFD to model and analyse the performance of a large pork carcass chillier. Two design cases, differing in inlet air direction and flow rate, and two functioning modes, batch and continuous, were analysed. Because modelling the full heat and mass transfer process between the carcasses and the air flow in a big chillier room is not possible given current computer capacity, a four step method was proposed to assess how changes in chillier design and operating conditions affect chilling kinetic, weigh loss and core temperature profiles: (1) 3D calculation of the air velocity field was conducted via CFD. An unstructured 3D mesh of more than 900.000 cells was created to represent the chillier room configuration with 290 pork carcasses distributed inside. Each carcass was model base on a basic 2D layout with a constant thickness of 20 cm. The air flow was assumed to be steady, incompressible and isothermal. (2) With the CFD model, an average velocity was calculated in a volume of interest containing the carcass. This was done all inside the chillier obtaining a velocity map. (3) The heat and mass transfer coefficient was calculated as a function of the air velocity map. The global heat and mass transfer coefficients were calculated using the correlations experimentally developed by Kondjoyan and Daudin (1997) for pork hindquarters. Those correlations express the heat transfer coefficient as a function of free stream velocity and turbulence intensity. The velocity was the one find on the velocity map and the turbulence intensity was assumed to be 30%. (4) With the heat and mass transfer coefficients, the weight loss kinetics and variation of the internal temperature profile were calculated using the procedure developed by Daudin and Kuitche (1996) (See section 2.2.2).

This procedure was originally developed for pork hindquarters but it was adopted for pork carcasses; a shape factor and an equivalent surface area of exchange were fitted from measurements under constant chilling conditions and related to carcasses weight. The advantage of these two factors is that they just not merely account for differences in shape between a cylinder and a carcass, but corrected other simplifications as the heterogeneous composition of the carcasses.

They found strong differences in the main air velocities around the chillier room that, in the batch chillier, leads to very different chilling times and weight losses. They concluded that batch chilling can only be achieved overnight at the cost of overcooling for low weight carcasses and increased weight loss.

2.5 Water activity of beef meat

Water activity is one of the most important factors that affect the transfer of moisture in meat thermal processing operations: drying, chilling, freezing, heating, cooking, storage and transport. It is also important in the preservation and quality of meat. For instance, microbial growth on the surface of foods is mainly controlled by water activity, temperature and pH (Ross, 1999).

When food is exposed to air, the surface water content is the determined by the equilibrium between evaporation and internal water migration (Baucour and Daudin, 2000). For a given food, water activity is function of the water content

(X) and the temperature. Knowledge of this relationship (Moisture sorption isotherm or MSI) is essential for the prediction of evaporative losses and potential for microbial growth during meat chilling.

Unfortunately, there is a lack of information on fresh meat. Previous work has been done to determine the Moisture Sorption Isotherm (MSI) of cooked meats (Iglesias and Chirife, 1982; Delgado and Sun, 2002a). Palnitkar and Heldman (1971) determined the adsorption and desorption isotherm of freeze-dried precooked beef at 21.1°C. Saravacos and Stinchfield (1965) determined the adsorption isotherm of freeze-dried meat in the range -20°C to 50°C. For fresh raw beef, few results has been reported, although Taylor (1961) determined the desorption moisture isotherm of raw beef at 19.5°C. Also, many difficulties are frequently encountered with published moisture sorption data. For example, information on the history and pre-treatment of the food sample is frequently not properly reported.

2.6 Diffusivity of water in meat

Accurate data on the diffusivity (D) of water in meat is very important for predicting the weight loss of beef carcasses during the chilling process. The diffusivity also affects bacterial growth, as it controls the movement of water to the meat surface. However, few works on moisture diffusivity in beef have been found in the literature. Lomauro, Bakshi & Labuza (1990) reported a figure for

ground beef moisture diffusivity and Motarjemi (1988) reported the diffusivity of raw minced beef at various temperatures and moisture content. Merts, Lovatt & Lawson (1998) developed a procedure to measure the drying curve of a cylindrical meat sample and estimated D using several different mathematical techniques. Radford (1976) analysed the possible water transport mechanisms on meat and found that Fickian diffusion is the predominant mechanism. Radford, Herbert & Lovett (1976) used the drying technique to determine the moisture diffusivity. The experimental data was fitted to a heat and mass transfer finite difference model and diffusivity was expressed as a function of temperature and water concentration. Additionally, some good work has recently been done on pork. Gou, Comaposada & Arnau (2002) determined the effect of pH and meat fibre direction on moisture diffusivity. Gou, Comaposada & Arnau (2003) studied the effect of temperature and NaCl on pork ham.

However, since the CSIRO work in the 1970's (Radford *et al.*, 1976, Herbert et al., 1978), there has not been any detailed research on moisture diffusivity on whole beef. Besides, there has been considerable variation between the values of D reported in the literature. For instance Merts *et al.* (1998) reported a diffusivity of 6.54E-11 m2/s at 10°C while the diffusivity obtained with Herbert's equation (Herbert *et al.,* 1978) at the same temperature is 3.26E-10 m^2/s. Merts *et al.* (1998) also reported variations in the calculated diffusivity using different methods. Thus, there is a clear need to carry out some more experimental

diffusivity measurements on beef and to determine whether the method of estimation affects the diffusivity calculation.

2.7 Statement of the objectives of the project

From the literature review is clearly seen that there is still lack of knowledge on the heat and mass transfer coefficients on the chilling process of meat carcasses. Many simplified models, which are very useful for design, have been developed but they generally use empirical equations to determine the heat and mass transfer coefficients. These equations have been developed for air flow around simple geometries but they do not necessarily represent the complex flow pattern around real carcasses. Additionally, simplified models use average heat transfer coefficients neglecting local variations. It has been reported that this variations can be up to 40% in simple geometries as cylinders.

CFD appears as a useful tool to model local differences in temperature and water activity, which are the most important factor affecting the good preservation of meat. CFD started to be used on meat processing recently but there are still many aspects that must be improved, such as: (1) a correct modelling of the coupling of heat and mass transfer of water in the meat to determine the surface water activity and temperature. (2) Determination of the local heat and mass transfer coefficients around the complex carcass geometry. (3) Improved

knowledge on mass transfer-related properties (diffusivity and moisture isotherm). Thus, the objectives of the current work are:

1) To develop a numerical method for calculation temperature and water activity changes during the chilling process of meat carcasses taking in account its complex geometry.

2) To model the external convection and internal conduction of heat and mass transfer by using turbulence models with CFD packages.

3) To determine the water isotherm and water diffusivity of meat.

4) To obtain empirical relationships for the heat transfer and mass transfer coefficients around the carcass surface.

2.8 Computational fluid dynamics

Computational fluid dynamics (CFD) is a numerical technique for the solutions of the equations governing the flow of fluids in defined flow geometry. FLUENT, the CFD package used in this work, is a computer program for modelling fluid flow, heat and mass transfer in complex geometries. The numerical algorithm used is the finite volume method that consists of the following steps:

- Formal integration of the governing equations of fluid flow over all the finite control volumes of the solution domain.
- Discretisation involves the substitution of a variety of finite difference type approximations for the terms in the integrated equation representing flow processes such as convection, diffusion and sources. This converts the integral equations into a system of algebraic equations.
- Solution of the algebraic equations by an iterative method.

The first step, the control volume integration, distinguishes the finite volume from all other CFD techniques. The resulting statements express the conservation of relevant properties for each finite size cell. This clear relationship between the numerical algorithm and the underlying physical conservation principle forms one of the main attractions of the finite volume methods and makes its concepts much more flexible to understand by engineers than finite element and spectral methods (Versteeg & Malalasekera, 1995).

CFD codes contain discretization techniques suitable for the treatment of the key transport phenomena, convection and diffusion as well for the source terms and the rate of change with respect to time. The underlying physical phenomena are complex and non-linear so an iterative solution approach is required.

2.8.1 Governing equations of fluid flow, heat and mass transfer

The governing equations of fluid flow represent mathematical statements of the conservation laws of physics.

- The mass of a fluid is conserved.
- The rate of change of momentum equals the sum of the forces on a fluid particle (Newton's second law).
- The rate of change of energy is equal to the sum of the rate of heat addition and the rate of work done on a fluid particle (first law of thermodynamics).

2.8.1.1 *Mass conservation equation (continuity equation)*

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{2.32}$$

where ρ is density, $\vec{v}$ is the velocity vector and t is time.

2.8.1.2 *Momentum equation*

$$\frac{\partial(\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v}\vec{v}) = -\nabla p + \nabla \cdot (\bar{\tau}) + \rho \vec{g} \tag{2.33}$$

where p is the static pressure, $\bar{\tau}$ is the stress tensor (described below), and $\rho \vec{g}$ is the gravitational body. The stress tensor for Newtonian fluids is given by:

$$\bar{\tau} = \mu \left[\left(\nabla \vec{v} + \nabla \vec{v}^{T} \right) - \frac{2}{3} \nabla \cdot \vec{v} I \right] \tag{2.34}$$

where μ is the molecular viscosity, I is the unit tensor, and the second term on the right hand side is the effect of volume dilation.

2.8.1.3 *Energy equation*

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \vec{J}_j + (\bar{\tau}_{eff} \cdot \vec{v}) \right) \tag{2.35}$$

where k_{eff} is the effective conductivity (k + k_t, where k_t is the turbulent thermal conductivity, defined according to the turbulence model being used), and $\vec{J}_j$ is the diffusion flux of species *j*. E is the specific energy of the fluid defined as:

$$E = h - \frac{p}{\rho} + \frac{v^2}{2}$$

where h is the enthalpy and $v^2/2$ represents the kinetic energy. The first three terms on the right-hand side of equation 2.35 represent energy transfer due to conduction, species diffusion, and viscous dissipation, respectively.

2.8.1.4 *Species transport equations*

$$\frac{\partial(\rho Y_i)}{\partial t} + \nabla \cdot (\rho \vec{v} Y_i) = -\nabla \cdot \vec{J}_i \tag{2.36}$$

An equation of this form will be solved for *N*-1 species where *N* is the total number of fluid phase chemical species present in the system. Since the mass fraction of the species must sum to unity, the *N*th mass fraction is determined as one minus the sum of the *N*-1 solved mass fractions. In Equation 2.36 $\vec{J}_i$ is the diffusion flux of species *i*, which arises due to concentration gradients. For dilute mixtures the diffusion flux can be approximated as:

$$\vec{J}_i = -\rho D_{i,m} \nabla Y_i \tag{2.37}$$

where $D_{i,m}$ is the diffusion coefficient for species *i* in the mixture.

2.8.2 Overview of Numerical Schemes

FLUENT allows choosing either of two numerical methods:

- segregated solver
- coupled solver

Using either method, FLUENT solves the governing integral equations for the conservation of mass and momentum, energy and other scalars such as turbulence and species. In both cases a control-volume-based technique is used that consists of:

- Division of the domain into discrete control volumes using a computational grid.
- Integration of the governing equations on the individual control volumes to construct algebraic equations for the discrete dependent variables ("unknowns") such as velocities, pressure, temperature, and conserved scalars.
- Linearization of the discretized equations and solution of the resultant linear equation system to yield updated values of the dependent variables.

The two numerical methods employ a similar discretization process (finite-volume), but the approach used to linearize and solve the discretized equations is different.

Using the segregated method, the governing equations are solved sequentially (i.e., segregated from one another). Because the governing equations are non-

linear (and coupled), several iterations of the solution loop must be performed before a converged solution is obtained.

On the other hand, the coupled solver solves the governing equations of continuity, momentum, energy and species transport simultaneously (i.e., coupled together). Governing equations for additional scalars are solved sequentially (i.e., segregated from one another and from the coupled set).

2.8.3 Discretization

FLUENT uses a control-volume-based technique to convert the governing equations to algebraic equations that can be solved numerically. This control volume technique consists of integrating the governing equations about each control volume, yielding discrete equations that conserve each quantity on a control-volume basis.

Discretization of the governing equations can be illustrated most easily by considering the steady-state conservation equation for transport of a scalar quantity ϕ. This is demonstrated by the following equation written in integral form for an arbitrary control volume V as follows:

$$\oint \rho\phi\vec{V}\cdot d\vec{A} = \oint \Gamma_\phi \nabla\phi \cdot d\vec{A} + \int_V S_\phi dV \tag{2.38}$$

where:

$\vec{A}$ = surface area vector

$\nabla\phi$ = gradient of ϕ

S_ϕ = source of ϕ per unit volume

Equation 2.38 is applied to each control volume, or cell, in the computational domain. The two-dimensional, triangular cell shown in Figure 2.1 is an example of such a control volume. Discretization of Equation 2.38 on a given cell yields:

$$\sum_{f}^{N_{faces}} \rho_f \vec{v}_f \phi_f \cdot \vec{A}_f = \sum_{f}^{N_{faces}} \Gamma_\phi (\nabla\phi)_n \cdot \vec{A}_f + S_\phi V \tag{2.39}$$

where

N_{faces} = number of faces enclosing cell

ϕ_f = value of ϕ convected through face f

$\rho_f \vec{v}_f \cdot \vec{A}_f$ = mass flux through the face

$\vec{A}_f$ = area vector of face f

$(\nabla\phi)_n$ = magnitude of $\nabla\phi$ normal to face f

V = cell volume

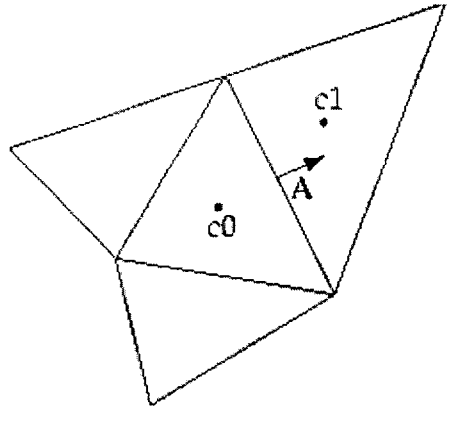


Figure 2.1 Control Volume Used to Illustrate Discretization of a Scalar Transport Equation (Taken from FLUENT Inc, 2001a)

The equations solved by FLUENT take the same general form as the one given above and apply readily to multi-dimensional, unstructured meshes composed of arbitrary polyhedra.

FLUENT stores discrete values of the scalar ϕ at the cell centres (c0 and $c1$ in Figure 2.1). However, face values ϕ_f are required for the convection terms in Equation 2.39 and must be interpolated from the cell centre values. This is accomplished using an upwind scheme. Upwinding means that the face value ϕ_f is derived from quantities in the cell upstream, or "upwind", relative to the direction of the normal velocity v_n. FLUENT allows choosing from several upwind schemes: first-order upwind, second-order upwind, power law, and QUICK. The diffusion terms in Equation 2.39 are central-differenced and are always second-order accurate.

2.8.3.1 *First-Order Upwind Scheme*

The first order upwind scheme is first-order accuracy. Using this scheme quantities at cell faces are determined by assuming that the cell-centre values of any field variable represent a cell-average value and hold throughout the entire cell; the face quantities are identical to the cell quantities. Thus when first-order upwinding is selected, the face value ϕ_f is set equal to the cell-centre value of ϕ in the upstream cell.

Other upwind schemes are describe in Versteeg & Malalasekera (1995) and FLUENT Inc (2003a). After discretization the set of equations 2.39 (one per each discrete volume) can be solved using a solver like the TDMA (tri-diagonal matrix).

2.8.4 Discretization of the Momentum Equation

The discretization scheme described in Section 2.8.3 for a scalar transport equation is also used to discretize the momentum equations. For example, the x-momentum equation can be obtained by setting $\phi = v$.

If the pressure field and face mass fluxes were known, Equation 2.39 could be solved in the manner outlined in Section 2.8.4, and a velocity field obtained. However, the pressure field and face mass fluxes are not known a priori and must be obtained as a part of the solution.

The continuity equation 2.32 can be discretizated setting $\phi = 1$ in equation 2.39:

$$\sum_{f}^{N_{faces}} \rho_f \vec{v}_f \cdot \vec{A}_f = 0 \tag{2.40}$$

In the segregated solver, the momentum and continuity equations are solved sequentially. In this sequential procedure, the continuity equation is used as an equation for pressure. However, pressure does not appear explicitly in Equation 2.40 for incompressible flows, since density is not directly related to pressure. The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations)

family of algorithms (SIMPLE, SIMPLEC, PISO, etc.) is used for introducing pressure into the continuity equation. Detailed information of these methods are found in Versteeg & Malalasekera (1995) and FLUENT Inc (2003a).

2.8.5 Temporal Discretization

For transient simulations, the governing equations must be discretized in both space and time. The spatial discretization for the time-dependent equations is identical to the steady-state case. Temporal discretization involves the integration of every term in the differential equations over a time step Δt. The integration of the transient terms is straightforward, as shown below.

A generic expression for the time evolution of a variable ϕ is given by:

$$\frac{\partial \phi}{\partial t} = F(\phi) \tag{2.41}$$

where the function F incorporates any spatial discretization. If the time derivative is discretized using backward differences, the first-order accurate temporal discretization is given by

$$\frac{\phi^{n+1} - \phi^{n}}{\Delta t} = F(\phi) \tag{2.42}$$

and the second-order discretization is given by

$$\frac{3\phi^{n+1} - 4\phi^{n} + \phi^{n-1}}{2\Delta t} = F(\phi) \tag{2.43}$$

where

ϕ = a scalar quantity

n+1 = value at the next time level, $t + \Delta t$

n = value at the current time level, t

n-1 = value at the previous time level, $t - \Delta t$

Once the time derivative has been discretized, a choice remains for evaluating $F(\phi)$. One method is to evaluate $F(\phi)$ at the future time level:

$$\frac{\phi^{n+1} - \phi^{n}}{\Delta t} = F\left(\phi^{n+1}\right) \tag{2.44}$$

This is referred to as implicit integration since ϕ^{n+1} in a given cell is related to ϕ^{n+1} in neighboring cells through $F\left(\phi^{n+1}\right)$. The advantage of the fully implicit scheme is that it is unconditionally stable with respect to time step size.

A second method is available when the coupled explicit solver is used. This method evaluates $F(\phi)$ at the current time level:

$$\frac{\phi^{n+1} - \phi^{n}}{\Delta t} = F\left(\phi^{n}\right) \tag{2.45}$$

and is referred to as explicit integration since ϕ^{n+1} can be expressed explicitly in terms of the existing solution values, ϕ^{n}.

2.9 Turbulence and its modelling

The Reynolds number of a flow gives a measure of the relative importance of inertia forces (associated with convective effects) and viscous forces (Versteeg &

Malalasekera, 1995). In experiments on fluid systems it is observed that at values below the so called critical Reynolds number Re_{crit} the flow is smooth and adjacent layers of fluid slide past each other in an orderly fashion. If the applied boundary conditions do not change with time the flow is steady. This regime is called laminar flow.

At values of the Reynolds number above Re_{crit} a complicated series of events take place which eventually leads to a radical change of the flow character. In the final state the flow behaviour is random and chaotic. The motion becomes intrinsically unsteady even with constant imposed boundary conditions. The velocity and all other flow properties vary in a random and chaotic way. This regime is called turbulent flow.

Even in flows where the mean velocities and pressures vary in only one or two space dimensions, turbulent fluctuations always have a three dimensional spatial character. Furthermore, visualizations of turbulent flows reveal rotational flow structures, so called turbulent eddies, with a wide range of length scales.

Particles of fluid which are initially separated by a long distance can be brought close together by eddying motions in turbulent flows. As a consequence, heat, mass and momentum are very effectively exchanged. Such effective mixing gives rise to high values of diffusion coefficients for mass, momentum and heat.

The crucial difference between visualizations of laminar and turbulent flows is the appearance of eddying motions of a wide range of length scales in turbulent flows. The computing requirements for the direct solution of the time dependent Navier-Stokes equations of fully turbulent flows at high Reynolds numbers are truly phenomenal and must await major developments in computer hardware.

Meanwhile, engineers need computational procedures which can supply adequate information about the turbulent processes, but which avoid the need to predict the effects of each and every eddy in the flow. Fortunately most part of the CFD engineering applications can be satisfied with information about the time average properties of the flow. The next section examines the effects of appearance of turbulent fluctuations on the mean flow properties.

2.9.1 Reynolds (Ensemble) Averaging

In Reynolds averaging, the solution variables in the instantaneous (exact) Navier-Stokes equations are decomposed into the mean (ensemble-averaged or time-averaged) and fluctuating components. For the velocity components:

$$v_i = \overline{v}_i + v_i' \tag{2.46}$$

where $\overline{v}_i$ and v_i' are the mean and fluctuating velocity components (i = 1, 2, 3). Likewise, for pressure and other scalar quantities:

$$\phi = \overline{\phi} + \phi' \tag{2.47}$$

where ϕ denotes a scalar such as pressure, energy, or species concentration. Substituting expressions of this form for the flow variables into the instantaneous continuity and momentum equations and taking a time (or ensemble) average yields the ensemble-averaged momentum equations. They can be written in Cartesian form as:

$$\frac{\partial \rho}{\partial t} + \sum_j \frac{\partial}{\partial x_j}(\rho \bar{v}_i) = 0 \tag{2.48}$$

$$\frac{\partial}{\partial t}(\rho \bar{v}_i) + \sum_j \frac{\partial}{x_j}(\rho \bar{v}_i \bar{v}_j) = -\frac{\partial p}{\partial x_i} + \sum_j \left[\mu \left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i} - \frac{2}{3}\delta_{ij} \sum_l \frac{\partial \bar{v}_l}{\partial x_l} \right) \right] + \sum_j \frac{\partial}{\partial x_j}\left(-\rho \overline{v_i' v_j'}\right) \tag{2.49}$$

Where Equations 2.48 and 2.49 are called Reynolds-averaged Navier-Stokes (RANS) equations. They have the same general form as the instantaneous Navier-Stokes equations, with the velocities and other solution variables now representing ensemble-averaged (or time-averaged) values. Additional terms now appear that represent the effects of turbulence. These Reynolds stresses, $-\rho \overline{v_i' v_j'}$, must be modelled in order to solve Equation 2.49.

Similar extra turbulent transport terms arise when it is derived a transport equation for an arbitrary scalar quantity. The time average transport equation for a scalar ϕ is:

$$\frac{\partial}{\partial t}(\bar{\phi}) + \sum_j \frac{\partial}{x_j}(\bar{\phi} \bar{v}_j) = +\sum_j \Gamma_\phi \frac{\partial \bar{\phi}}{\partial x_j} + \sum_j \frac{\partial}{\partial x_j}\left(-\rho \overline{v_i' v_j'}\right) \tag{2.50}$$

Where Γ_ϕ is the diffusivity of the scalar ϕ.

2.9.2 Turbulence models

Turbulence needs to be modelled to solve the Navier-Stokes equations. For most engineering purposes it is unnecessary to resolve the details of the turbulence fluctuations. Only the effects of the turbulence on the mean flow are usually sought. In particular, expressions for the Reynolds stress in equations 2.49 and 2.50 are usually needed. For turbulence model to be useful in general purpose CFD code it must have wide applicability, be accurate, simple and economical to run.

Classical models, like the mixing length, $k-\varepsilon$, $k-\omega$, or the Reynolds Stress Transport Models use the Reynolds average Navier-Stokes equations (2.48 – 2.49) and form the basis of turbulence calculations in current available commercial CFD codes. Large eddy simulations (LES) are turbulence models where the time dependent flow equations are solved for the mean flow and the largest eddies. Large eddy simulations are at present at the research stage and the calculations are too costly to merit consideration in general computation at the present (Versteeg and Malalasekera, 1995). Of the classical models the mixing length and $k-\varepsilon$ models are presently by far the most widely used and validated. They are based on the presumption that there exists an analogy between the action of viscous stresses and Reynolds stresses on the mean flow (Boussinesq approach).

2.9.3 Boussinesq Approach vs. Reynolds Stress Transport Models

The Reynolds-averaged approach to turbulence modelling requires that the Reynolds stresses in Equation 2.49 can be appropriately modelled. It was proposed by Boussinesq that Reynolds stresses ($\tau_{ij}^{R} = -\rho\overline{v_i'v_j'}$) can be linked to mean rates of deformation analogously to the equation 2.34:

$$-\rho\overline{v_i'v_j'} = \tau_{ij}^{R} = \mu_t\left(\frac{\partial\overline{v}_i}{\partial x_j} + \frac{\partial\overline{v}_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\sum_l \frac{\partial\overline{v}_l}{\partial x_l}\right) \tag{2.51}$$

The Boussinesq hypothesis is used in the Spalart-Allmaras model, the $k-\varepsilon$ models, and the $k-\omega$ models. The advantage of this approach is the relatively low computational cost associated with the computation of the turbulent viscosity, μ_t In the case of the Spalart-Allmaras model, only one additional transport equation (representing turbulent viscosity) is solved. In the case of the $k-\varepsilon$ and $k-\omega$ models, two additional transport equations (for the turbulence kinetic energy, k, and either the turbulence dissipation rate, ε, or the vorticity fluctuation of turbulence, ω) are solved, and μ_t is computed as a function of k and ε. The disadvantage of the Boussinesq hypothesis as presented is that it assumes that μ_t is an isotropic scalar quantity, which is not strictly true.

The alternative approach, embodied in the RSM (Reynolds stress models), is to solve transport equations for each of terms in the Reynolds stress tensor. An additional scale-determining equation (normally for ε) is also required. This

means that five additional transport equations are required in 2D flows and seven additional transport equations must be solved in 3D.

In many cases, models based on the Boussinesq hypothesis perform very well, and the additional computational expense of the Reynolds stress model is not justified. However, the RSM is clearly superior for situations in which the anisotropy of turbulence has a dominant effect on the mean flow. Such cases include highly swirling flows and stress-driven secondary flows. The design of Reynolds stress equation models is an area of vigorous research and the models have not been validated as widely as the mixing length and $k-\varepsilon$ model.

2.9.4 The Standard $k-\varepsilon$ Model

The standard $k-\varepsilon$ model of Launder and Spalding (1974), is a semi-empirical model based on model transport equations for the turbulence kinetic energy (k) and its dissipation rate (ε). The model transport equation for k is derived from the exact equation, while the model transport equation for ε was obtained using physical reasoning and bears little resemblance to its mathematically exact counterpart.

In the derivation of the $k-\varepsilon$ model, it was assumed that the flow is fully turbulent, and the effects of molecular viscosity are negligible. The standard $k-\varepsilon$ model is therefore valid only for fully turbulent flows.

The turbulence kinetic energy, k, and its rate of dissipation, ε, are obtained from the following transport equations:

$$\frac{\partial}{\partial t}(\rho k)+\sum_j \frac{\partial}{\partial x_j}(\rho k \bar{v}_j)=\sum_j \frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]+G_k+G_b-\rho\varepsilon-Y_M+S_k \tag{2.52}$$

and

$$\frac{\partial}{\partial t}(\rho \varepsilon)+\sum_j \frac{\partial}{\partial x_j}(\rho \varepsilon \bar{v}_j)=\sum_j \frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right]+C_{1\varepsilon}\frac{\varepsilon}{k}(G_k+C_{3\varepsilon}G_b) \tag{2.53}$$
$$-C_{2\varepsilon}\rho\frac{\varepsilon^2}{k}+S_\varepsilon$$

In these equations, G_k represents the generation of turbulence kinetic energy due to the mean velocity gradients. G_b is the generation of turbulence kinetic energy due to buoyancy. Y_M represents the contribution of the fluctuating dilatation in compressible turbulence to the overall dissipation rate $C_{1\varepsilon}, C_{2\varepsilon}$ and $C_{3\varepsilon}$ are constants. σ_k and σ_ε are the turbulent Prandtl numbers for k and ε, respectively. S_k and S_ε are source terms.

The turbulent (or eddy) viscosity, μ_t, is computed by combining k and ε as follows:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{2.54}$$

where C_μ is a constant. The effective viscosity can be defined as

$$\mu_{eff} = \mu + \mu_t \tag{2.55}$$

2.9.5 The RNG k- ε Model

The RNG-based k- ε turbulence model is derived from the instantaneous Navier-Stokes equations, using a mathematical technique called "renormalization group" (RNG) methods (Yakhot V and Orszag S. A., 1986). The analytical derivation results in a model with constants different from those in the standard k- ε model, and additional terms and functions in the transport equations for k and ε. The RNG k- ε model has a similar form to the standard k- ε model:

$$\frac{\partial}{\partial t}(\rho k)+\sum_j \frac{\partial}{\partial x_j}(\rho k \bar{v}_j)=\sum_j \frac{\partial}{\partial x_j}\left[\alpha_k \mu_{eff}\frac{\partial k}{\partial x_j}\right]+G_k+G_b-\rho\varepsilon-Y_M+S_k \tag{2.56}$$

and

$$\frac{\partial}{\partial t}(\rho \varepsilon)+\sum_j \frac{\partial}{\partial x_j}(\rho \varepsilon \bar{v}_j)=\sum_j \frac{\partial}{\partial x_j}\left[\alpha_\varepsilon \mu_{eff}\frac{\partial \varepsilon}{\partial x_j}\right]+C_{1\varepsilon}\frac{\varepsilon}{k}(G_k+C_{3\varepsilon}G_b)-C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{2.57}$$
$$-R_\varepsilon+S_\varepsilon$$

The quantities α_k and α_ε are the inverse effective Prandtl numbers for k and ε, respectively. S_k and S_ε are user-defined source terms.

The scale elimination procedure in RNG theory results in a differential equation for turbulent viscosity:

$$d\left(\frac{\rho^2 k}{\sqrt{\varepsilon\mu}}\right)=1.72\frac{\hat{\upsilon}}{\sqrt{\hat{\upsilon}^3-1+C_\upsilon}}d\hat{\upsilon} \tag{2.58}$$

where:

$\hat{\upsilon}=\mu_{eff}/\mu$

$C_\upsilon \approx 100$

Equation 2.58 is integrated to obtain an accurate description of how the effective turbulent transport varies with the effective Reynolds number, allowing the model to better handle low-Reynolds-number and near-wall flows.

2.9.6 Convective Heat and Mass Transfer Modelling in the k- ϵ Models

In FLUENT, turbulent heat transport is modelled using the concept of Reynolds' analogy to turbulent momentum transfer. The "modelled" energy equation is thus given by the following:

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \vec{J}_j + (\bar{\tau}_{eff} \cdot \vec{v}) \right) \tag{2.59}$$

where E is the total energy, k_{eff} is the effective thermal conductivity, and $\bar{\tau}_{eff}$ is the stress tensor, defined as

$$(\tau_{ij})_{eff} = \mu_{eff} \left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \sum_l \frac{\partial \bar{v}_l}{\partial x_l} \right) \tag{2.60}$$

The term involving $\bar{\tau}_{eff}$ represents the viscous heating. For the standard k- ε model, the effective thermal conductivity is given by:

$$k_{eff} = k + \frac{c_p \mu_t}{\mathrm{Pr}_t} \tag{2.61}$$

where k, in this case, is the thermal conductivity. The default value of the turbulent Prandtl number is 0.85. For the RNG k- ε model, the effective thermal conductivity is:

$$k_{eff} = \alpha C_p \mu_{eff} \tag{2.62}$$

where α is calculated from

$$\left|\frac{\alpha - 1.3923}{\alpha_0 - 1.3929}\right|^{0.6321} \left|\frac{\alpha - 2.3923}{\alpha_0 - 2.3929}\right|^{0.3679} = \frac{\mu}{\mu_{eff}} \tag{2.63}$$

where $\alpha_0 = 1/\Pr = k/\mu c_p$

The fact that α varies with μ/μ_{eff} is an advantage of the RNG k- ε model. It is consistent with experimental evidence indicating that the turbulent Prandtl number varies with the molecular Prandtl number and turbulence. Equation 2.63 works well across a very broad range of molecular Prandtl numbers, from liquid metals to paraffin oils, which allows heat transfer to be calculated in low-Reynolds-number regions. Equation 2.63 smoothly predicts the variation of effective Prandtl number from the molecular value ($\alpha = 1/\Pr$) in the viscosity-dominated region to the fully turbulent value ($\alpha = 1.393$) in the fully turbulent regions of the flow.

Turbulent mass transfer is treated similarly. For the standard k- ε models, the default turbulent Schmidt number is 0.7. For the RNG model, the effective turbulent diffusivity for mass transfer is calculated in a manner that is analogous to the method used for the heat transport. The value of α_0 in Equation 2.63 is $\alpha_0 = 1/Sc$, where Sc is the molecular Schmidt number.

2.9.7 The Standard k- ω Model

The standard k- ω model in FLUENT is based on the Wilcox k- ω model, which incorporates modifications for low-Reynolds-number effects, compressibility, and shear flow spreading. The Wilcox model predicts free shear flow spreading

rates that are in close agreement with measurements for far wakes, mixing layers, and plane, round, and radial jets, and is thus applicable to wall-bounded flows and free shear flows. A variation of the standard k- ω model called the SST k- ω model is also available in FLUENT. The shear-stress transport (SST) k- ω model was developed to effectively blend the robust and accurate formulation of the k- ω model in the near-wall region with the free-stream independence of the k- ε model in the far field. To achieve this, the k- ε model is converted into a k- ω formulation. These features make the SST k- ω model more accurate and reliable for a wider class of flows (e.g., adverse pressure gradient flows, airfoils, transonic shock waves) than the standard k- ω model.

2.9.8 Near Wall modelling

Turbulent flows are significantly affected by the presence of walls. Obviously, the mean velocity field is affected through the no-slip condition that has to be satisfied at the wall. However, the turbulence is also changed by the presence of the wall in non-trivial ways. Very close to the wall, viscous damping reduces the tangential velocity fluctuations, while kinematic blocking reduces the normal fluctuations. Toward the outer part of the near-wall region, however, the turbulence is rapidly augmented by the production of turbulence kinetic energy due to the large gradients in mean velocity.

The near-wall modelling significantly impacts the fidelity of numerical solutions, inasmuch as walls are the main source of mean vorticity and turbulence. After

all, it is in the near-wall region that the solution variables have large gradients, and the momentum and other scalar transports occur most vigorously. Therefore, accurate representation of the flow in the near-wall region determines successful predictions of wall-bounded turbulent flows.

The k- ε models, the RSM, and the LES model are primarily valid for turbulent core flows (i.e., the flow in the regions somewhat far from walls). Consideration therefore needs to be given as to how to make these models suitable for wall-bounded flows. The Spalart-Allmaras and k-ω models were designed to be applied throughout the boundary layer, provided that the near-wall mesh resolution is sufficient.

Numerous experiments have shown that the near-wall region can be largely subdivided into three layers. In the innermost layer, called the "viscous sub layer", the flow is almost laminar, and the (molecular) viscosity plays a dominant role in momentum and heat or mass transfer. In the outer layer, called the fully-turbulent layer, turbulence plays a major role. Finally, there is an interim region between the viscous sub layer and the fully turbulent layer where the effects of molecular viscosity and turbulence are equally important.

2.9.8.1 Wall Functions vs. Near-Wall Model

Traditionally, there are two approaches to modelling the near-wall region. In one approach, the viscosity-affected inner region (viscous sub layer and buffer layer)

is not resolved. Instead, semi-empirical formulas called "wall functions" are used to bridge the viscosity-affected region between the wall and the fully-turbulent region. The use of wall functions obviates the need to modify the turbulence models to account for the presence of the wall.

In another approach, the turbulence models are modified to enable the viscosity-affected region to be resolved with a mesh all the way to the wall, including the viscous sub layer. This is termed the "near-wall modelling" approach.

In most high-Reynolds-number flows, the wall function approach substantially saves computational resources, because the viscosity-affected near-wall region, in which the solution variables change most rapidly, does not need to be resolved. The wall function approach is popular because it is economical, robust, and reasonably accurate. It is a practical option for the near-wall treatments for industrial flow simulations.

The wall function approach, however, is inadequate in situations where the low-Reynolds-number effects are pervasive in the flow domain in question, and the hypotheses underlying the wall functions cease to be valid. Such situations require near-wall models that are valid in the viscosity-affected region and accordingly integrable all the way to the wall.

2.9.8.2 Standard Wall Functions

The standard wall functions in FLUENT are based on the proposal of Launder and Spalding (1974), and have been most widely used for industrial flows.

2.9.8.2.1 Momentum

The law-of-the-wall for mean velocity yields:

$$U^* = \frac{1}{\kappa}\ln(Ey^*) \tag{2.64}$$

where

$$U^* = \frac{U_p C_\mu^{1/4} k_p^{1/2}}{\tau_w/\rho} \tag{2.65}$$

$$y^* = \frac{\rho C_\mu^{1/4} k_p^{1/2} y_p}{\mu}$$

κ = von Kármán constant (= 0.42)

E = empirical constant (= 9.81)

U_P = mean velocity of the fluid at point P

k_P = mean turbulent kinetic energy fluid at point P

y_P = distance from point P to the wall

μ = dynamic viscosity of the fluid

The logarithmic law for mean velocity is known to be valid for y^* > about 30 to 60. In FLUENT, the log-law is employed when y^* > 11.225.

When the mesh is such that $y^* < 11.225$ at the wall-adjacent cells, FLUENT applies the laminar stress-strain relationship that can be written as:

$$U^* = y^* \tag{2.66}$$

2.9.8.2.2 Energy

Reynolds' analogy between momentum and energy transport gives a similar logarithmic law for mean temperature. As in the law-of-the-wall for mean velocity, the law-of-the-wall for temperature employed in FLUENT comprises the following two different laws:

- Linear law for the thermal conduction sub layer where conduction is important.
- Logarithmic law for the turbulent region where effects of turbulence dominate conduction.

The thickness of the thermal conduction layer is, in general, different from the thickness of the (momentum) viscous sub layer, and changes from fluid to fluid. The law-of-the-wall implemented in FLUENT has the following composite form:

$$T^* = \frac{(T_w - T_p)\rho c_p C_\mu^{1/4} k_p^{1/2}}{\dot{q}} = \begin{cases} \Pr y^* & (y^* < y_T^*) \\ \Pr_t \left[\frac{1}{\kappa} \ln(Ey^*) + P \right] & (y^* > y_T^*) \end{cases} \tag{2.67}$$

where P is computed by using the formula:

$$P = 9.24 \left[\left(\frac{\alpha}{\alpha_t} \right)^{3/4} - 1 \right] \left[1 + 0.28 e^{-0.007 \alpha / \alpha_t} \right] \tag{2.68}$$

and

k_f = thermal conductivity of fluid

ρ = density of fluid

c_p = specific heat of fluid

$\dot{q}$ = wall heat flux

T_p = temperature at the cell adjacent to wall

T_w = temperature at the wall

Pr = molecular Prandtl number ($\mu c_p / k$)

Pr_t = turbulent Prandtl number (0.85 at the wall)

κ = 0.4187 (von Kármán constant)

E = 9.793 (wall function constant)

The non-dimensional thermal sub layer thickness, y_T^*, in Equation 2.67 is computed as the y^* value at which the linear law and the logarithmic law intersect, given the molecular Prandtl number of the fluid being modelled.

The procedure of applying the law-of-the-wall for temperature is as follows. Once the physical properties of the fluid being modelled are specified, its molecular Prandtl number is computed. Then, given the molecular Prandtl number, the thermal sub layer thickness, y_T^*, is computed from the intersection of the linear and logarithmic profiles, and stored. During the iteration, depending on the y^* value at the near-wall cell, either the linear or the logarithmic profile in

Equation 2.67 is applied to compute the wall temperature T_w or heat flux $\dot{q}$ (depending on the type of the thermal boundary conditions).

2.9.8.2.3 Species

When using wall functions for species transport, FLUENT assumes that species transport behaves analogously to heat transfer. Similarly to Equation 2.67, the law-of-the-wall for species can be expressed for constant property flow with no viscous dissipation as:

$$Y^* = \frac{(Y_{i,w} - Y_i)\rho C_\mu^{1/4} k_p^{1/2}}{J_{i,w}} = \begin{cases} Scy^* & (y^* < y_c^*) \\ Sc_t\left[\frac{1}{\kappa}\ln(Ey^*) + P_c\right] & (y^* > y_c^*) \end{cases} \tag{2.69}$$

where Y_i is the local species mass fraction, Sc and Sc_t are molecular and turbulent Schmidt numbers, and $J_{i,w}$ is the diffusion flux of species i at the wall. Note that P_c and y_c^* are calculated in a similar way as P and y_T^*, with the difference being that the Prandtl numbers are always replaced by the corresponding Schmidt numbers.

The standard wall functions work reasonably well for a broad range of wall-bounded flows. However, they tend to become less reliable when the flow situations depart too much from the ideal conditions that are assumed in their derivation.

2.9.8.3 Enhanced Wall Treatment

Enhanced wall treatment is a near-wall modelling method that combines a two-layer model with enhanced wall functions. If the near-wall mesh is fine enough to be able to resolve the laminar sub layer (typically $y^+ \approx 1$), then the enhanced wall treatment will be identical to the traditional two-layer zonal model (see below for details). However, the restriction that the near-wall mesh must be sufficiently fine everywhere might impose too large a computational requirement. Ideally, then, one would like to have a near-wall formulation that can be used with coarse meshes (usually referred to as wall-function meshes) as well as fine meshes (low-Reynolds-number meshes). In addition, excessive error should not be incurred for intermediate meshes that are too fine for the near-wall cell centroid to lie in the fully turbulent region, but also too coarse to properly resolve the sub layer.

To achieve the goal of having a near-wall modelling approach that will possess the accuracy of the standard two-layer approach for fine near-wall meshes and that, at the same time, will not significantly reduce accuracy for wall-function meshes, FLUENT combines the two-layer model with enhanced wall functions.

2.9.8.3.1 Two-Layer Model for Enhanced Wall Treatment

In FLUENT's near-wall model, the viscosity-affected near-wall region is completely resolved all the way to the viscous sub layer. The two-layer approach is an integral part of the enhanced wall treatment and is used to specify both ε

and the turbulent viscosity in the near-wall cells. In this approach, the whole domain is subdivided into a viscosity-affected region and a fully-turbulent region. The demarcation of the two regions is determined by a wall-distance-based, turbulent Reynolds number, Re^y, defined as:

$$\mathrm{Re}_y = \frac{\rho y \sqrt{k}}{\mu} \tag{2.70}$$

where y is the normal distance from the wall at the cell centres. In FLUENT, y is interpreted as the distance to the nearest wall. In the fully turbulent region ($\mathrm{Re}_y > 200$), the k- ε models are employed.

In the viscosity-affected near-wall region ($\mathrm{Re}_y < 200$), the one-equation model of Wolfstein (1969) is employed. In the one-equation model, the momentum equations and the k equation are retained as described for the k- ε. However, the turbulent viscosity, μ_t, is computed from:

$$\mu_{t,2layer} = \rho C_\mu l_\mu \sqrt{k} \tag{2.71}$$

where $\mu_{t,2layer}$ is the turbulent viscosity inside the inner layer and l_μ is the length scale:

$$l_\mu = y c_l \left(1 - e^{-\mathrm{Re}_y / A_\mu}\right) \tag{2.72}$$

The two-layer formulation for turbulent viscosity is used as a part of the enhanced wall treatment, in which the two-layer definition is smoothly blended with the high-Reynolds-number μ_t defined from the outer region:

$$\mu_{t,enhanced} = \lambda_\varepsilon \mu_t + (1 - \lambda_\varepsilon)\mu_{t,2layer} \tag{2.73}$$

where λ_ε is a blending function defined in such a way that it is equal to unity far from walls and is zero very near to walls. More details of the method are found in Fluent Inc. (2003).

3. MOISTURE SORPTION ISOTHERM OF BEEF

The moisture sorption isotherm (MSI) of lean beef and fat beef was experimentally determined. The experimental procedure used was that of the COST 90 project with some modifications to accelerate equilibration. The procedure was validated with the standard reference material microcrystalline cellulose. The MSI of the beef at the highest humidity range was obtained by accelerating equilibration with changes of salts, using a low water activity salt for some time. This procedure was reliable for beef samples but not for the fat samples. No significant changes were found for lean beef in the temperature range 5°C to 40°C. Three models, GAB, Peleg and Lewicki, were used to fit the experimental data. The best fit was obtained with the GAB equation. The fat MSI was determined at 5°C, 15°C and 25°C and it was best fitted with the Lewicki model.

3.1 Introduction

The water activity of the meat was determined in the range of $a_w = 0.3 - 0.98$. The method of the COST 90 project (Spiess and Wolf, 1987) was modified to accelerate the process and avoid spoilage or fungal growth. Determining the MSI at high values of a_w (over 0.90) presents a big problem, since a long time is required to reach the equilibrium (due to the low driving force and large moisture content to be evaporated) and spoilage becomes inevitable (Baucour and Daudin,

2000; Wolf, Spiess and Jung, 1985). However, this range must be considered because it is important in the chilling of beef carcasses where the meat surface is nearly fully wetted for considerable periods (Herbert, 1978).

There are several equations available to describe the MSI of food materials (Berg and Bruins; 1981). However, No single model fits the whole range of a_w accurately. The obtained data was fitted using the following models: GAB (Timmermann *et al.*, 2001), Peleg (Peleg, 1992) and Lewicki (Lewicki, 1998).

3.2 Materials and methods

3.2.1 Previous techniques of sorption measurements

Current methods to determine the MSI require the food samples to reach equilibrium with the surrounding atmosphere and then the water content X of the sample is obtained by weighing. The most accepted and standardized method is that of the COST 90 project (Spiess and Wolf, 1987), which uses a standard sorption apparatus, procedure and reference material (microcrystalline cellulose or MCC) (Wolf, Spiess and Jung, 1985).

The COST 90 equipment consists of a 1 litre sorption flask containing a weighing bottle which contains 1 g of food. At the bottom of the flask is slurry of analytical grade salts with known a_w. The time recommended by the COST 90 project to reach equilibrium is 4 days for MCC powder (Spiess and Wolf, 1987)

and at least 14 days for real food materials (Wolf, Spiess and Jung, 1985). Singh *et al.* (2001) reached the equilibrium of smoked chicken sausages in 2 – 3 weeks following the COST 90 project. With a perishable food such as raw beef we cannot afford to take such long equilibration times because of bacterial and fungal degradation. To avoid spoilage, it has been suggested to add small amounts of fungicides like phenyl mercury acetate and thymol; however the consequences on the product on the isotherm are difficult to assess (Baucour and Daudin, 2000). Therefore, some modifications of the technique were made to accelerate the mass transfer process and reduce the equilibration time.

3.2.2 Our Modifications to the COST 90 method

The geometry of the sorption container, the distance between the sample and the liquid surface and the sample geometry all affect the rate of mass transfer.

According to Lang, McCune and Steinberg (1981), in an equilibrium environment between a food product and saturated salt slurry, the driving force is the difference in vapour pressure. Thus, the faster the vapour space reaches the equilibrium with the saturated salt slurry, the quicker the maximum driving force for water absorption will be applied to the sample. A reduction in the size of the sorption container would reduce the diffusion distance and accelerate the process, since according to Boltzman's equation (Labuza and Hyman; 1998), the effective diffusion time (d_d) decreases inversely with the square of distance (D):

$$D = \frac{K_b . T^{\tau}}{3 \Pi \mu d_d^{2}} \tag{3.1}$$

The container size should be just enough to accommodate a single sample to ensure the highest possible area-volume ratio (Lang, McCune and Steinberg, 1981), and the container height should be approximately equal to container diameter (Spiess and Wolf, 1987). Thus the sorption jars used in this experiment were 5.2 cm tall by 5.3 cm diameter (figure 3.1).

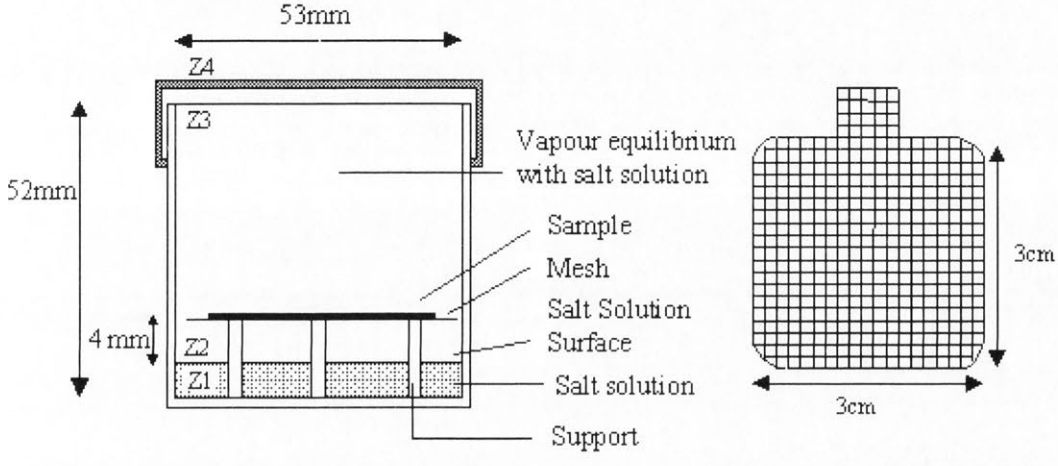


Figure 3.1 Schematic diagram of sorption jar and sample holder used in the experiments.

If the jar is not hermetic, water vapour loss will cause a humidity gradient. Yeow (2001) made some preliminary experiments to determine whether this would cause a significant humidity difference between the salt surface and the sample location. The rate of weight loss from the whole jar was determined, and the air between the salt slurry surface and the lid of the jar was modelled as a stagnant film (Bird *et al.*, 1960), from which the humidity gradient can be calculated. To

avoid the sample getting wet, it was placed 4 mm above the liquid surface. Calculations showed that at this distance, the deviation of the vapour pressure from equilibrium is less than 1% and can be neglected.

In the COST 90 method, the sample is held in a weighing bottle that resembled a miniature kettle. The weighing bottle is placed above the saturated salt in a closed container and is weighed regularly to detect the first sign of equilibration. During weighing the bottle is closed to minimize water adsorption. The design of the weighing bottle imposes great resistance to the mass transfer of the desorption process and requires a long time to equilibrate with the salt. Hence, in our experiments a mesh was used to hold the sample, and thus minimize the external resistance (Figure 3.1).

The preparation of the sample also affects the mass transfer rate. The diffusion process may be accelerated by increasing the external area and reducing the thickness of the sample. Thus, the samples of meat were cut into thin strips.

3.2.3 Measurement of the Moisture Sorption Isotherm

The COST 90 method was followed with the modifications described before. Samples of eye fillet and external beef fat were used on the test. The eye fillet was obtained from the butcher the same day that the beef was slaughtered. Pham and Karuri (1999) reported that during chilling, beef carcasses lose moisture only in the first few centimetres below the surface. Thus, slices were cut from the

inside of the piece of beef to obtain samples with high initial water content. The slices were cut into thin strips of approximately 2 mm x 2 mm x 25 mm along the direction of the fibres to avoid rupturing the fibres. 0.3 g samples were taken from the strips to conduct the experiments.

The cut samples were put on stainless steel mesh and placed into the jar described in Figure 3.1. Ten analytical grade salts (Table 3.1) were used to make saturated slurries of known a_w (Greenspan, 1976). The slurries were placed into a water bath to keep the temperature at 25°C and 40°C ± 1°C. A refrigerator with on-off controller was used to keep the temperature in 5°C and 15°C ± 2°C. The samples were weighed at regular intervals of time until equilibrium was reached. The criteria to determine the equilibrium were when the slope of curve of weight vs. time reached a value close to zero and the difference between three consecutive weightings were less than 0.5 mg. The tests were done in triplicate and the ash, protein, fat and moisture content of the batch of meat were determined following the procedure of Greenfield *et al.* (1998).

The samples of fat were of about 1.0 g and were cut in small slices because it was not possible to cut it in small strips. The temperature was fixed at 5°C, 15°C or 25°C. It was not possible to carry out experiments at 40°C because the fat melted.

Salt	T = 5		T = 15		T = 25		T = 40	
	aw	X	aw	X	aw	X	aw	X
K_2SO_4	0.9842	1.2842 ± 0.1288	0.9789	-	0.9730	1.1681 ± 0.0355	0.9541	0.9949 ± 0.0101
KNO_3	0.9627	0.7670 ± 0.0335	0.9541	0.6209 ± 0.0328	0.9358	0.5170 ± 0.0573	0.8903	0.3691 ± 0.0418
KCl	0.8767	0.3534 ± 0.0386	0.8592	0.3039 ± 0.0283	0.8434	0.2763 ± 0.0226	0.8232	0.2775 ± 0.0116
KBr	0.8509	0.3070 ± 0.0152	0.8262	0.3173 ± 0.0825	0.8089	0.2337 ± 0.0100	0.7942	0.2304 ± 0.0070
NaCl	0.7565	0.2073 ± 0.0334	0.7561	0.2095 ± 0.0210	0.7529	0.2024 ± 0.0267	0.7468	0.2062 ± 0.0083
KI	0.7330	0.1884 ± 0.0273	0.7098	0.1629 ± 0.0321	0.6886	0.1574 ± 0.0246	0.6609	0.1470 ± 0.0034
NaBr	0.6351	0.1371 ± 0.0219	0.6068	0.1015 ± 0.0318	0.5757	0.1023 ± 0.0435	0.5317	0.0923 ± 0.0048
$Mg(NO_3)_2$	0.5886	0.1230 ± 0.0223	0.5587	0.1002 ± 0.0101	0.5289	0.0999 ± 0.0184	0.4842	0.0724 ± 0.0101
NaI	0.4242	0.0955 ± 0.0162	0.4088	0.0807 ± 0.0042	0.3817	0.0545 ± 0.0148	0.3288	0.0437 ± 0.0029
$MgCl_2$	0.3360	0.0861 ± 0.0213	0.3330	0.0792	0.3278	0.0517 ± 0.0237	0.3160	-

Table 3.1 Experimental moisture sorption isotherms of lean beef at 5C, 15C, 25C and 40C (means and standard deviations). The a_w values were obtained from Greenspan (1976).

3.2.4 Equilibration of High Humidity Samples

Our preliminary tests confirmed previous reports (Baucour and Daudin, 2000; Wolf, Spiess and Jung, 1985) that, at high humidities, the mass transfer is very slow, making it difficult to reach equilibrium in the range 0.9 – 1.0 a_w. Moreover, at these high a_w fungal and bacterial growth proceeds quickly. For that reason it was necessary to accelerate the process on this range. Because the driving force is the a_w difference between the surrounding air and the sample, the first stage of the desorption process was accelerated by putting the sample in a salt slurry of $MgCl_2$, which have the lowest humidity (a similar procedure was used by Delgado and Sun, 2002b; on chicken meat to accelerate the equilibrium process). The sample was weighed at constant time and then it was transferred to a high-humidity jar when its weight was within about 12% of the expected equilibrated value.

3.3 Results and discussion

3.3.1 Validation of the method with MCC

The method was validated with MCC (Avicel™ PH101 50 micron powder) following the recommendations of Spiess and Wolf (1987). Because the MCC came as a micron powder and the mesh could not hold such fine powder, the mesh was lined with aluminium foil.

With our reported modifications to the method, the time to reach the equilibrium for MCC was reduced from 4 days to about 20 hours, except the sample at a_w = 0.9710 which took 40 hours (See Figure 3.2). Thus, we have accelerated the equilibration rate by 3 to 4 times compared to the COST 90 method.

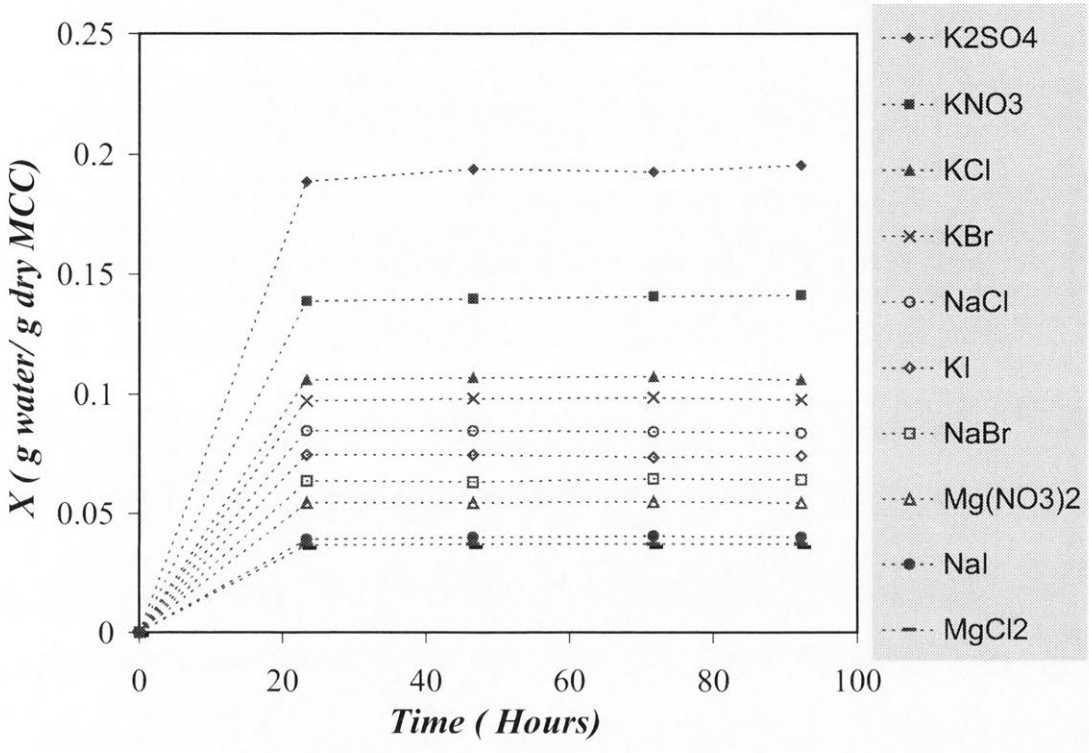


Figure 3.2 Kinetic of the water adsorption of MCC.

The four replicate experimental results of MCC sorption isotherm were compared to the 32 collaborative participating laboratories COST 90 project (Figure 3.3). The literature results given by COST 90 were reported in the form of the GAB equation (Spiess and Wolf, 1987).

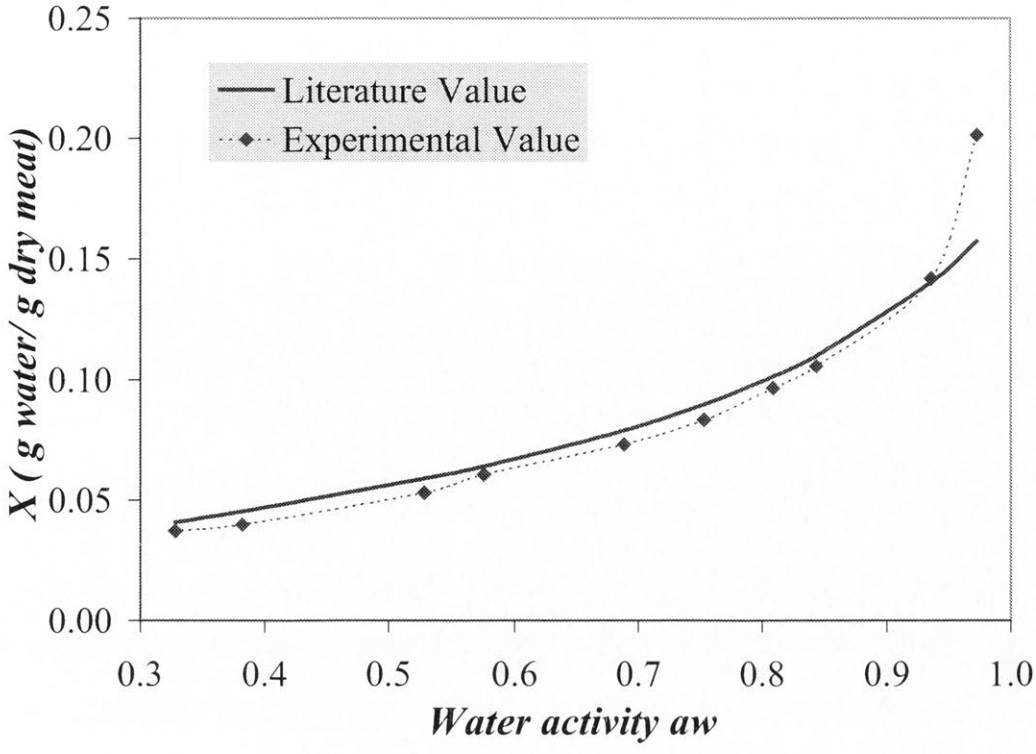


Figure 3.3 Moisture sorption Isotherm of MCC.

According to Spiess and Wolf (1987), if n replicate determinations are performed, then good agreement, on a 95% probability level, is obtained if the difference $|X_{mean}- X_{lit}|$ between the result and the reference value is equal to or smaller than a critical difference $D_{crit} = 0.498$ (Spiess and Wolf defined D_{crit} as a function of n). Our MCC results for the 9 salts with the lower a_w meet the criterion of $|X_{mean}-X_{lit}|<D_{crit}$, therefore, the experimental results obtained with this method are within the 95% confidence interval. The highest a_w (K_2SO_4, a_w = 0.9730) did not meet that criteria but this was attributable to the fact that the GAB literature equation was fitted only in the interval 0.1115 – 0.9026. It has been reported that this equation gives a good fit of food MSI in the interval 0.10

$< a_w < 0.90$ (Timmerman *et al.* 2001). Thus, the reported equation for MCC is not reliable for $a_w > 0.9026$.

3.3.2 Experimental Results for Meat Moisture Isotherm

Figures 3.4 and 3.5 show the change of the water content of meat with time at 25^0C and 5^0C. All samples reached equilibrium within 30 to 80 hours, except at high air humidities (K2SO4 at 25^0C, K2SO4 and KNO3 at 5^0C). However, when desorption was accelerated by first exposing the sample to low humidity (over MgCl2 for two hours), equilibration was achieved within 20 hours (Figure 3.6).

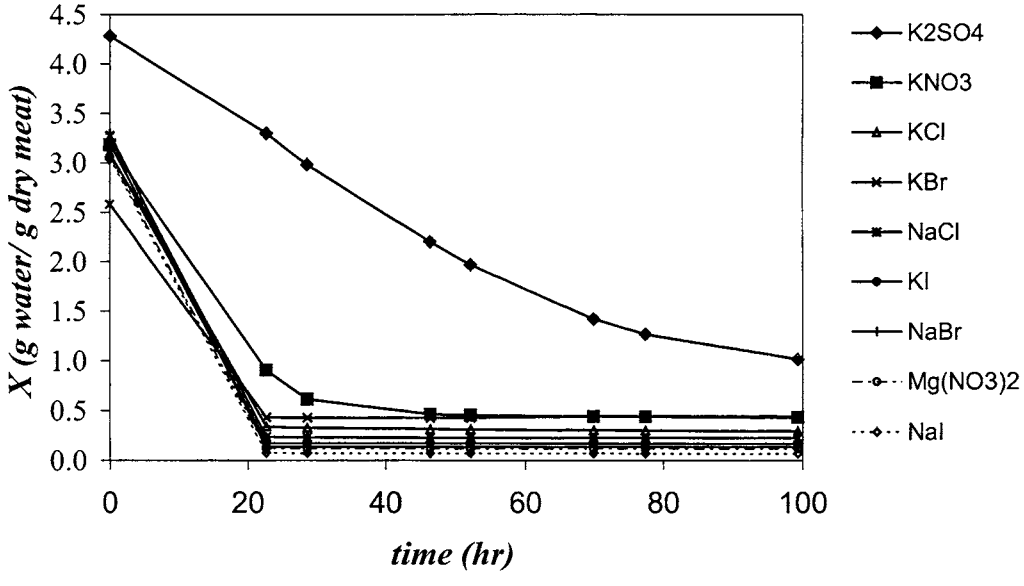


Figure 3.4 Change of the water content with time at 25^0C with change of salt. The sample at high humidity (K2SO4) could not reach the equilibrium but the others did.

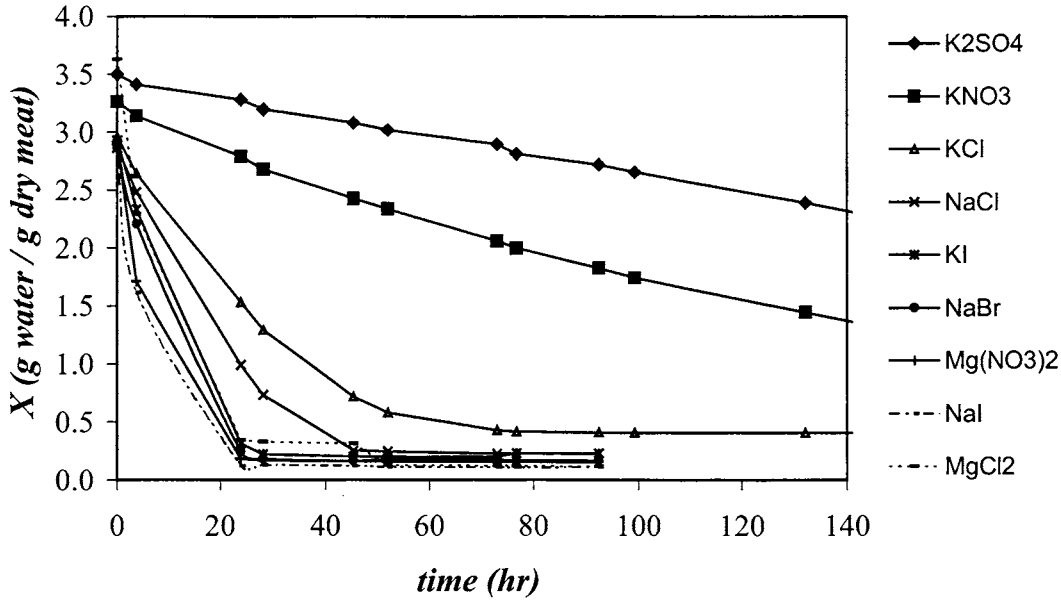


Figure 3.5 Change of the water content with time at 5^0C with no change of salt. The samples reach the equilibrium except at the two higher humidity values (K2SO4 and KNO3).

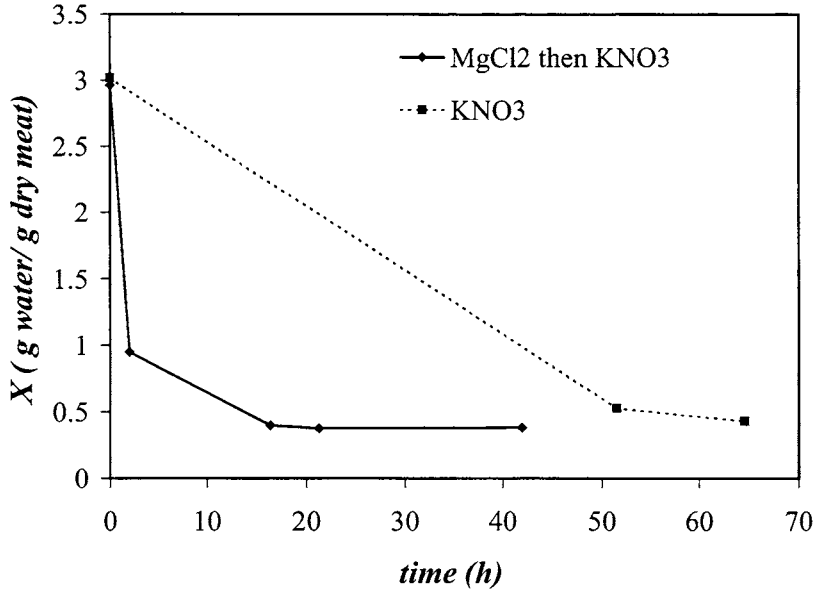


Figure 3.6 Accelerated desorption process of meat. MgCl2 used during the fist 2 hours then KNO_3.

	Units	Average	σ
Fat	g fat/ g wet sample	0.156	0.282
Ash	g ash/ g wet sample	0.018	0.018
Water content	g water/ g dry material	2.779	0.171
Protein	% protein of dry mass basis	77.87	4.54

Table 3.2 Average Composition of the lean meat.

Salt	Temp.	With Change of salt		Without Change of salt	
		X	σ	X	σ
KCl	25	0.2736	0.00016	0.2903	0.00013
KBr	25	0.2170	0.00149	0.2574	0.00000
Mg(NO3)2	25	0.0949	0.00009	0.1051	0.00005
KNO_3	40	0.3825	0.00167	0.3558	0.00216

ANOVA

	SS	Dof	MSS	Ft	F critic	Result
Treatment	6.163e-4	1	6.163e-4	0.7802	4.38	No Sig.
Salts	0.2271	3	0.0756917	95.834	3.13	Sig.
Error	0.0150	19	7.898e-4			
Total	0.2427	23				

Table 3.3 Effect of the change of salts on the MSI of lean beef and ANOVA table.

To see if this acceleration method affected the equilibrium moisture content, the final moisture content of meat samples was determined using KCl, KBr and $Mg(NO_3)_2$ at 25°C and KNO_3 at 40°C. For each salt, three replicate meat samples were allowed to reach the equilibrium without acceleration and another three

replicates were placed first over $MgCl_2$ during the first hours. Analysis of variance showed that there is no significant difference at 95% confidence between the two treatments (Table 3.3). Therefore the acceleration method was considered acceptable.

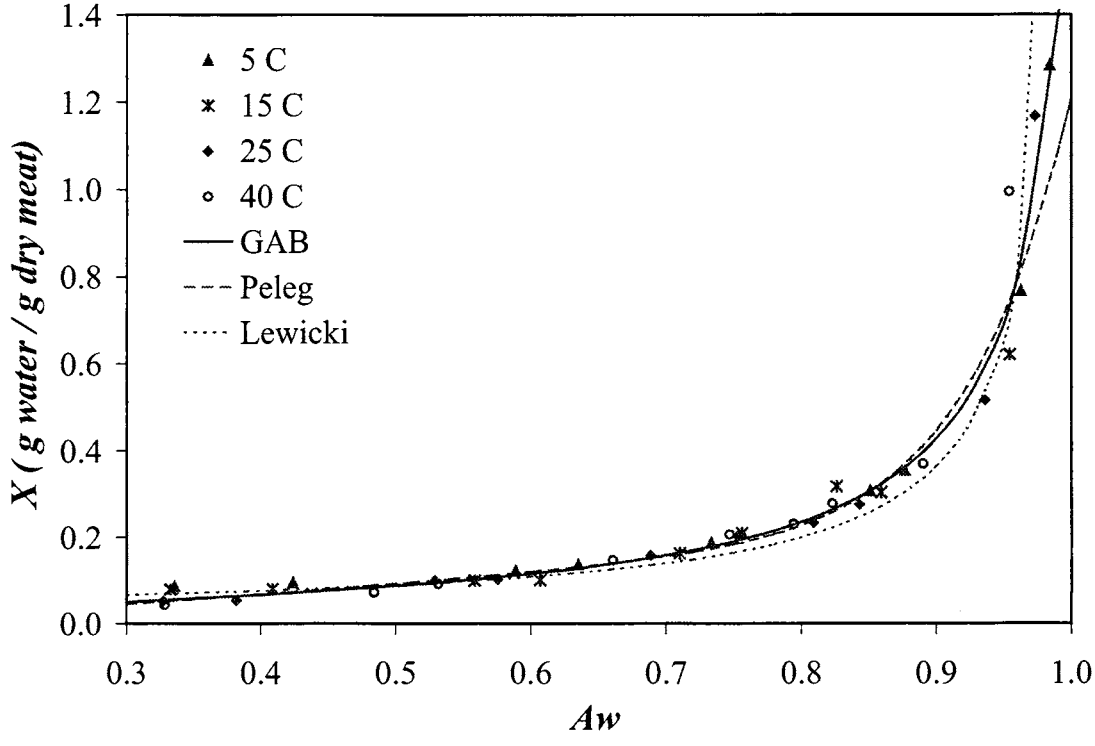


Figure 3.7 Moisture sorption isotherm of beef in the range 5°C – 40°C.

The desorption isotherm of lean meat, at 5°C, 15°C, 25°C and 40°C, are shown in Table 3.1 and Figure 3.7. The influence of salts, sample composition and temperature on the MSI was subjected to analysis of variance. No significant difference was found for sample composition at a confidence level of 95%. Table 3.2 shows the average composition of the meat. The factor that most affects the variance is the type of salt i.e. water activity. No effect was found for temperature. Thus, it was assumed that the MSI of lean beef is independent of

temperature in the range 5°C– 40°C. Saravacos and Stinchfield (1965) found that the adsorption moisture isotherm of freeze dried beef shows a maximum between 10°C and 20°C. Their isotherms at 0°C, 10°C and 20°C overlap in most of the range of humidity.

3.3.3 Curve fitting the experimental results

Several equations are available on the literature to fit MSI data (Berg and Bruin S, 1981). The following three were used:

Peleg:

$$X = a.a_w^{\ b} + c.a_w^{\ d} \tag{3.2}$$

GAB:

$$X = \frac{X_{mg}.C_g.K.a_w}{(1-K.a_w)[1+(C_g-1)K.a_w]} \tag{3.3}$$

Lewicki:

$$X = \frac{F}{(1-a_w)^G} - \frac{F}{1+a_w^{\ H}} \tag{3.4}$$

where:

X = Water content in solid.

a_w = Water activity.

and all other symbols represent constants in the equations.

The Peleg equation can predict both sigmoidal and non-sigmoidal isotherms. According to Peleg (1992), this model fitted as well as or better than the GAB model but its constants have no physical meaning.

The GAB equation is a semi-theoretical multilayer sorption model with a physical meaning of the constants (Timmermann *et al.* 2001). The model is applicable to a wide range of a_w (0.1 – 0.9) but it has been reported that the error increased sharply for values of a_w above 0.9. At higher water activities, the GAB plot presents a downward deviation due to the appearance of a third sorption stage (Timmerman and Chirife; 1991), an effect that determines the upper limit of application of the GAB equation. The main difficulty of GAB model for practical purpose is to extend its use up to a_w of 1 (Rahmna *et al.* 1998).

Both the GAB model and most of its modifications as well as the Peleg's four parameter model have the same weakness in that they predict a finite adsorption at water activity 1 (Lewicki, 1998).

The Lewicki equation was developed to make it applicable to the high range of a_w. It was assumed that $X = 0$ when $a_w = 0$, and $X = \infty$ when $a_w = 1$ (Lewicki, 1998). This equation fits well the MSI data at high humidities and predicts that the water content X tends to ∞ when $a_w \rightarrow 1$, condition that have been found in many fresh tissue foods.

The three models were used to fit the experimental MSI data and their constants were calculated minimizing the root mean square percent error (% RMS) defined as:

$$\%RMS = \frac{\sqrt{\Sigma\left(\frac{X_{EXP} - X_{MODEL}}{X_{EXP}}\right)^2}}{n-1}.100\% \tag{3.5}$$

GAB	T = 5C	T = 15C	T = 25C	T = 40C	T. Indep. model
X_{mg}	0.0522	0.0497	0.0540	0.0527	0.0565
C_g	4.269E+13	3.233E+13	3.8212	3.0732	4.2396
κ	0.9721	0.9688	0.9750	0.9878	0.9692
RMS %	0.2956	1.3919	0.8208	1.6601	1.8778
Peleg					
a	1.0848	1.0410	1.9641	2.6280	0.9650
b	13.3198	9.6905	31.0390	28.9194	13.3039
c	0.2011	0.1153	0.3137	0.3456	0.2427
d	0.8855	0.4031	2.0065	1.9638	1.3886
RMS %	2.0627	3.5093	6.2678	0.9974	2.3377
Lewicki					
F	0.1209	0.1398	0.0787	0.0428	0.0488
G	0.5816	0.5474	0.7386	1.0235	0.8761
H	0.5273	2.7044	1.4549	-10.5667	-34.7794
RMS %	2.0165	5.3498	5.3018	0.0249	4.1227

Table 3.4 Parameter of the GAB, Peleg and Lewicki equations for lean meat.

Table 3.4 shows the fitted parameter of the models on lean beef at each temperature and for all temperatures. Overall the GAB model fitted the experimental data best, followed by the Peleg and Lewicki Models. However, the Lewicki model shows a better fit in the high a_w range, although it is the poorest

on the intermediate range (0.60 – 0.90). The only disadvantage of the GAB model, as well as the Peleg, is that it predicts a finite water content X (X = 1.82) at water activity 1. Figure 3.7 shows the fitting of the three equations. Figure 3.8 shows the distribution of residual errors. It shows higher residual deviation at higher water activities probably caused by the inadequacy of the models to fit the full range of a_w. For instance the GAB equation is not recommended for values of a_w above 0.9. The Lewicki model exhibits the best residual distribution in the range of a_w between 0.9-1, but shows a negative deviation on the range 0.85-0.90.

	T = 5°C		T = 15°C		T = 25°C	
Salt	Aw	X	aw	X	Aw	X
K_2SO_4	0.9842	-	0.9789		0.9730	0.0260
KNO_3	0.9627	-	0.9541	0.0218 ± 0.0061	0.9358	0.0170 ± 0.0183
KCl	0.8767	0.0502 ± 0.0340	0.8592	0.0182 ± 0.0129	0.8434	0.0089 ± 0.0096
KBr	0.8509	0.0223 ± 0.0042	0.8262	0.0075 ± 0.0015	0.8089	0.0068 ± 0.0070
NaCl	0.7565	0.0139 ± 0.0025	0.7561	0.0083 ± 0.0025	0.7529	0.0093 ± 0.0078
KI	0.7330	0.0192 ± 0.0136	0.7098	0.0064 ± 0.0033	0.6886	0.0069 ± 0.0075
NaBr	0.6351	0.0123 ± 0.0056	0.6068	0.0063 ± 0.0043	0.5757	0.0056 ± 0.0058
$Mg(NO_3)_2$	0.5886	0.0094 ± 0.0084	0.5587	0.0041 ± 0.0026	0.5289	0.0062 ± 0.0075
NaI	0.4242	0.0084 ± 0.0063	0.4088	0.0064 ± 0.0041	0.3817	0.0100 ± 0.0116
$MgCl_2$	0.3360	0.0060 ± 0.0044	0.3330	0.0039 ± 0.0003	0.3278	0.0058 ± 0.0096

Table 3.5 Experimental moisture sorption isotherms of external fat at 5°C, 15°C and 25°C (means and standard deviations).

The mean repeatability and the standard deviation of the lean beef moisture content X are 0.14 and 0.30 respectively, which are smaller than those of Wolf, Spiess and Jung (1985) under the COST 90 project (mean repeatability of 0.28 and standard deviation of 0.52), even though the latter measured the isotherm at only one temperature. The temperature-independent GAB equation produced a RMS error of only 1.88%. Thus, it can be concluded that the lean beef data can be accurately fitted with a model independent of temperature.

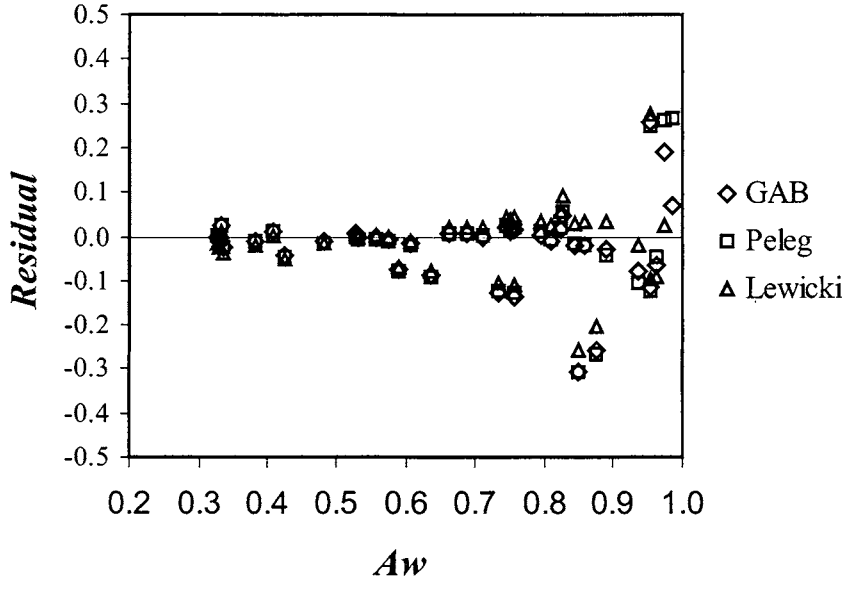


Figure 3.8 Distribution of residual errors for GAB, Peleg and Lewicki models for lean beef isotherm at all temperatures.

3.3.4 Experimental Results of the External Fat Moisture Isotherm

The desorption isotherm of beef fat at 5°C, 15°C and 25°C are shown in Table 3.5. The data was statistically analysed and, as in the case of the lean meat, no significant difference was found with changes of composition at a confidence level of 95%. The factor that mostly affects the variance is also the change of

salt. Temperature was found to have a significant effect. As it seen in Figure 3.9, there is no clear difference between 15°C and 25°C but the MSI at 5°C is clearly higher than the others.

Because fat is less susceptible to bacteria degradation than meat, it was possible to obtain the moisture content of the samples in equilibrium with KNO_3 at 15°C and 25°C. The two high humidity points, with K_2SO_4 and KNO_3, were accelerated with the salt change technique. However, the three replicates of the fat in KNO3 using the acceleration process were statistically lower than the three replicates with the standard procedure. Thus, this procedure can not be considered reliable in the case of fat.

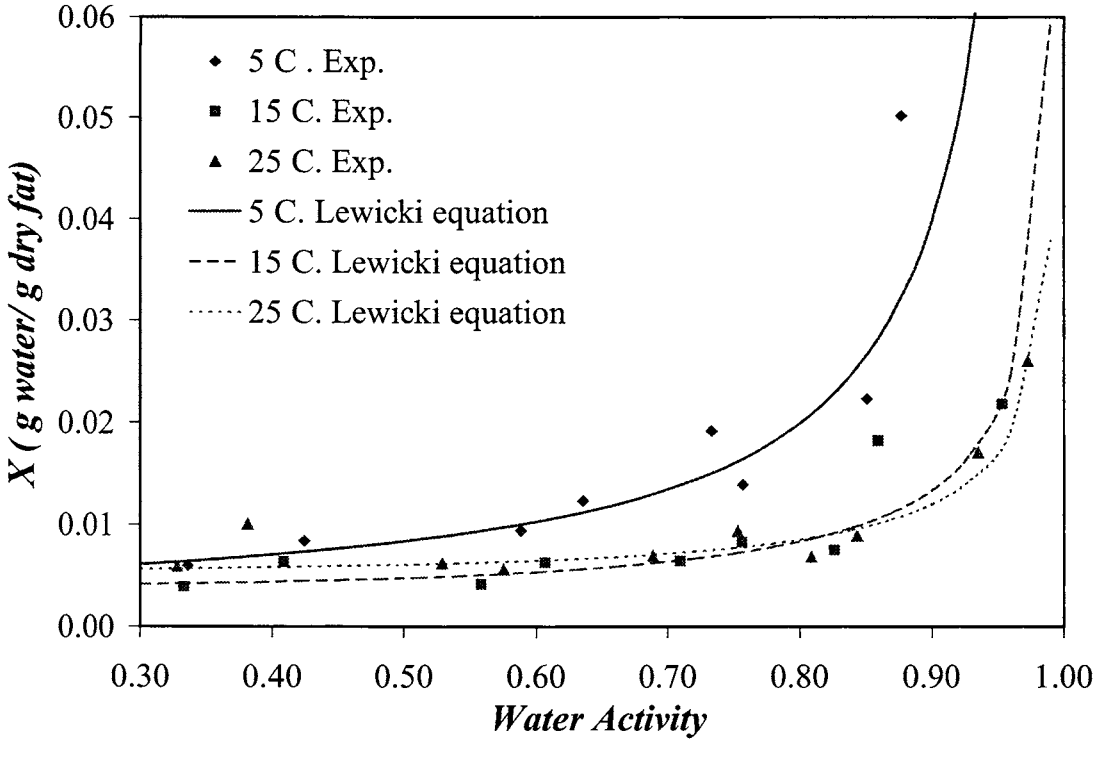


Figure 3.9 Moisture Sorption Isotherm of fat at 5°C, 15°C and 25°C.

The different behaviour of fat under accelerated desorption may be attributed to the hydrophobic character and resulting low moisture content of the fat. Under accelerated desorption at the beginning of the process, the external part of the sample loses most or all its moisture due to the low surrounding humidity. After changing to the second (high humidity) salt, moisture migrates from the centre to the surface but cannot be re-absorbed by the desiccated layers of fat because these has become denatured and lost hydrophilic sites to which water molecules can be attached.

The three previous models were also used to fit the fat data. The Lewicki model fitted the data better than the others. Table 7 shows the fitted constants of the GAB, Lewicki and Peleg equations at 5°C, 15°C and 25°C. The Lewicki equation was expressed as a function of temperature to make it more general. This modification was made by correlating the parameters of the equation as an exponential function of the temperature, similar to the modification of the GAB model as a function of temperature (Weisser, 1985). The Lewicki constants as function of temperature are:

$$F = 6.3828\times10^{2}\, e^{\left(\frac{-3.3153\times10^{3}}{T}\right)}$$
$$G = 7.7994\times10^{12}\, e^{\left(\frac{7.1235\times10^{3}}{T}\right)} \qquad (3.6)$$
$$H = 9.0851\times10^{9}\, e^{\left(\frac{-1.5453\times10^{7}}{T}\right)}$$

The RMS error using these equations is 6.616 which mean that the fitted Lewicki equations at each temperature (table 3.4) are better than the model including temperature (equations 3.6).

Although the GAB model does not fit the data as well as the Lewicki model, it is interesting to note that the C_g value of the GAB equation is very high. Thus, the GAB equation can be reduced to a two parameter equation:

$$X = \frac{X_{mg}}{(1 - K.a_w)} \tag{3.7}$$

3.4 Conclusions

- The moisture sorption isotherm of fresh lean beef and external beef fat has been successfully measured by using improved techniques to accelerate equilibration and avoid spoilage at high relative humidities.

- The experimental procedure used, based on COST 90 with some geometrical modifications (sorption container geometry, position of the sample and geometry of the sample) was validated with the standard reference material MCC.

- The moisture sorption isotherm (MSI) of the beef at the two highest humidities was obtained by accelerating equilibration with a change of salt. This procedure was reliable for beef samples but not for the fat

samples, possibly caused by strong hysteresis due to the hydrophobic characteristics of the fat.

- In both cases, lean beef and fat, no significant differences of the MSI with changes in composition was found.

- The MSI of the beef was measured at 5°C, 15°C, 25°C and 40°C. The experimental data was accurately fitted using a model independent of temperature. The GAB equation fitted the data best, although it predicted a constant value of moisture content at $a_w = 1$. The Lewicki model does not fit well the data in the range (0.60 – 0.90) but gives the best fit at high humidity (0.90 -1.00) and predict that water content approaches infinity when $a_w \rightarrow 1$.

- The MSI of fresh beef surface fat was measure at 5°C, 15°C and 25°C. The experimental water content of the MSI at 5°C is higher than that at 15°C and 25°C. However, no significant difference in the MSI was found between 15°C and 25°C. The experimental data was fitted using the GAB, Lewicki and Peleg models. The best fit was obtained with the Lewicki equation and the parameters were expressed as a function of temperature to make the equation more general, at a slight cost in accuracy.

4. DIFFUSIVITY OF WATER IN MEAT

4.1 Introduction

Accurate data on the diffusivity (D) of water in meat is very important for predicting the weight loss of beef carcasses during the chilling process. The diffusivity also affects bacterial growth, as it controls the movement of water to the meat surface. However, few works on moisture diffusivity in beef have been found in the literature, as it was established on section 2.6. Thus, there is a clear need to carry out some more experimental measurements.

4.2 Material and methods

The drying method was selected among different procedures reported in the literature to determine the moisture diffusivity. According to Zogzas *et al.* (1994) this is the most popular technique because drying process are of major importance in the industry and the recorded data can be used with confidence for scaling up industrial driers. Another and maybe the most important reason is that the equipment settlement to do experiments can be easily constructed and controlled.

This method consist of drying a meat sample (of a standard geometry as a slab or cylinder) in a drying tunnel or a controlled environmental chamber where the air velocity can be controlled through a fan, relative humidity through the humidifier, and dry bulb temperature through a heater. The dry and wet bulb temperature, as well as the weight of the sample, can be recorded at specific time intervals via I/O system connected to the computer. The recorded data can be used to determine the diffusivity of the sample.

4.2.1 Sample preparation

Lean beef samples were obtained from the local butchery. The samples were cut and placed into a cylindrical plastic container of 38 mm diameter X 10 mm height. The used container was open on the top and closed on the bottom, allowing the moisture to diffuse only from the bottom to the top. Any meat excess was trimmed to perfectly fit the sample into the dish. Then, the sample was removed from the plastic dish and the lateral face of the sample was coated with Vaseline to avoid any moisture diffusion in that direction. The sample was fitted again in the plastic dish covered with wrapping plastic and it was put inside the environmental chamber for 40 minutes to get temperature equilibrium before starting the drying process. The plastic dish with the meat sample was thermally isolated with expanded polystyrene to avoid thermal diffusion in the lateral direction.

4.2.2 Equipment

An environmental chamber with temperature and humidity control was used. Inside the chamber, three sets of experimental components were placed: the first, to dry the sample and record the weight; the second, to measure the dry and wet bulb temperatures; the third, to determine the equilibrium moisture content of the sample.

Figure 4.1 and 4.2 show the drying equipment. The meat sample, in a plastic dish insulated by expanded polystyrene, sat on an electronic balance with precision ±0.00005 g. A CPU cooling type fan was placed 5 cm over the meat sample to reduce the external resistance to mass transfer, in order to assure that the moisture transport is governed by diffusion and thus that errors in the external mass transfer coefficient do not lead to large errors in measured diffusivity. Once the drying process started, the weight was electronically recorded every 10 seconds and it was stored in the PC running the software Windmill. The air velocity was not recorded, but additional experiments were conducted to determine the external mass transfer coefficient. Due to the activity of the fan and the high sensitivity of the balance, the weight curve is somewhat noisy and has to be pre-treated. Occasional spikes were removed from the weight curves and they were smoothed by a rolling average technique. The force exerted by the airflow on the sample was determined from the baseline shift of the weight curve and subtracted from the recorded weight.

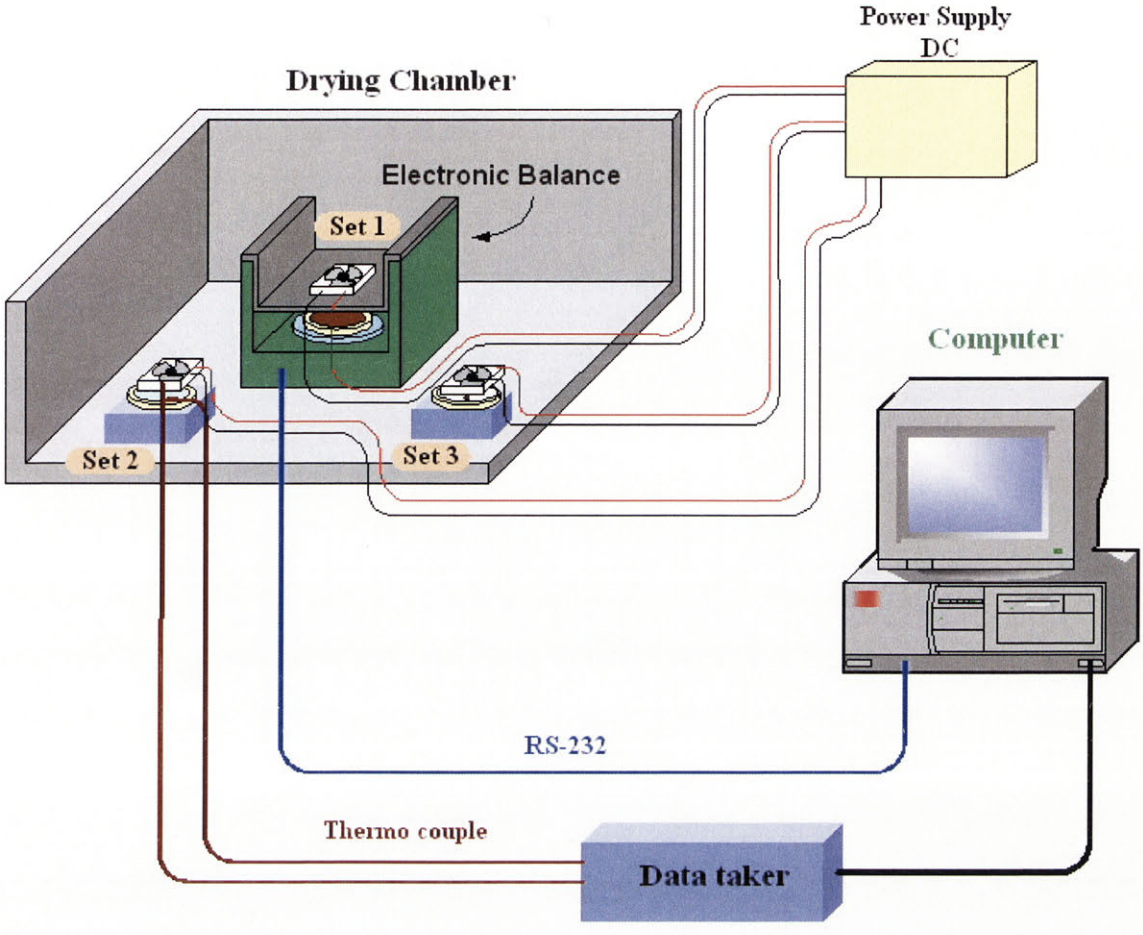


Figure 4.1 Schematic diagram of drying equipment (Taken from Wiangkaew, 2003).

The second set, consisting of a plastic dish, a fan and 2 thermocouples, was used to calculate the dry and wet bulb temperatures. The plastic dish was thermally isolated with expanded polystyrene and filled with distilled water (9 mm deep). The water evaporates and cools to wet bulb temperature, which was measured by one of the thermocouples. A fan was placed 5 cm over the plastic dish. The dry bulb temperature was measured with a thermocouple placed besides the fan. The

dry and wet bulb temperatures were recorded and used to calculate the air humidity in the chamber.

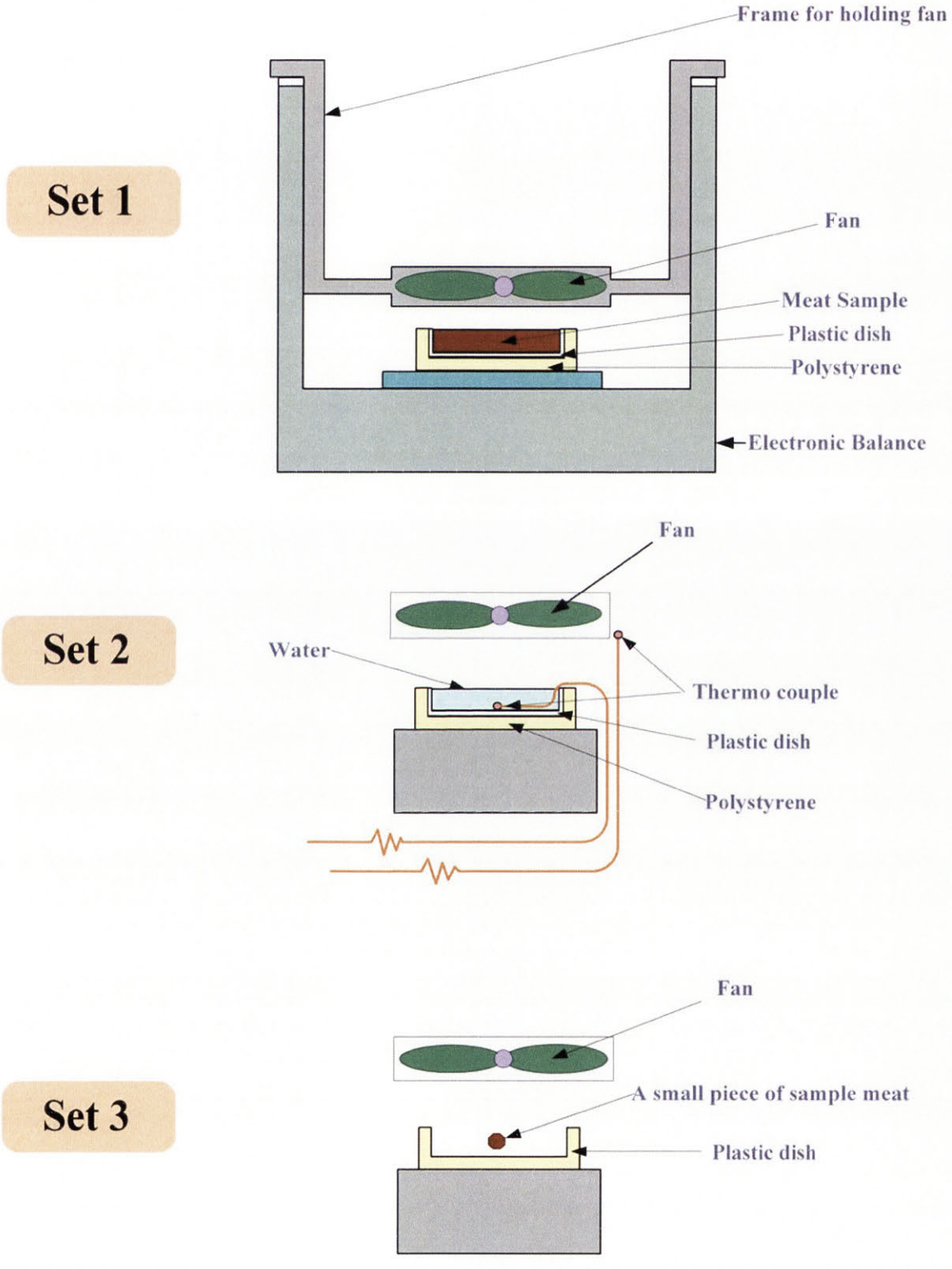


Figure 4.2 Details of the equipment (Taken from Wiangkaew, 2003).

The third set was used to experimentally determine the equilibrium moisture content of the meat under the temperature and humidity conditions of the environmental chamber. A small piece of meat of approximately 5 X 5 X 5 mm was dried at the chamber conditions, placing it on a plastic dish and under a fan. The equilibrium moisture content was determined separately from the dry experiment of the first set because drying a big sample of meat (first set) takes longer time to equilibrate. The experiments were carried out in triplicate at 4 different temperatures as established in Table 4.1.

Temperature (°C)	6.8	19.9	27.9	40.4
σ	0.10	0.10	0.03	0.10
Humidity (%)	92.0%	79.6%	77.6%	78.1%
σ	2.6	0.2	1.4	0.2

Table 4.1 Temperature and humidity conditions of the drying experiments.

4.2.3 Determination of the mass transfer coefficient

The external mass transfer coefficient was experimentally determined by evaporating water using set 1 under the same conditions as for meat. Figure 4.3 shows the change of weight of the water vs. time at a temperature of 40.41°C and 78.1% of Humidity. It is clearly seen that the rate of evaporation, calculated with the slope of the line weight vs. time, is constant. The mass flux is obtained by dividing the rate of evaporation by the transversal area. The external mass transfer coefficient was calculated with the equation:

$$h_m = \frac{J_w}{(Y_s - Y_a)} \tag{4.1}$$

Where J_w (Kg water/ m2 / s) is the water flux, h_m (Kg dry air / m^2. s) is the external mass transfer coefficient, Y_s (Kg water / Kg dry air) the moisture content of the air in equilibrium with the meat surface and Y_a (Kg water / Kg dry air) is the moisture content of the air. The experiments were done by triplicate. Table 4.2 summarizes the obtained results.

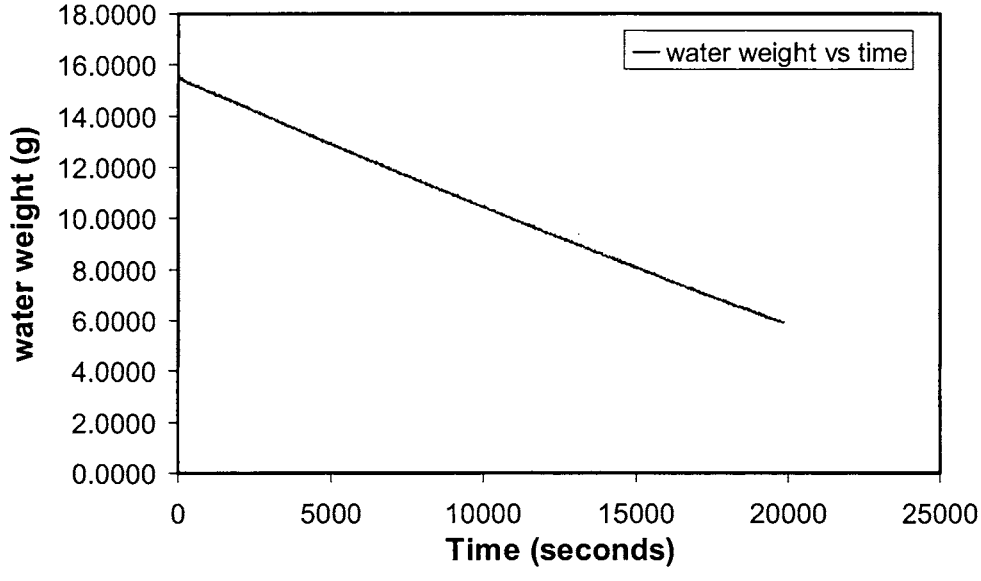


Figure 4.3 Water evaporation experiment at 40.41C and 78.1% of humidity.

Temperature	6.8	19.9	27.9	40.4
Humidity	92.0%	79.6%	77.6%	78.1%
Replicate 1	0.1774	0.1023	0.1160	0.3115
Replicate 2	0.1686	0.1055	0.1216	0.3341
Replicate 3	0.1820	0.1055	0.1196	0.3069
hm average	0.1760	0.1044	0.1191	0.3175
SD	0.0068	0.0018	0.0029	0.0146
% SD	3.9%	1.8%	2.4%	4.6%

Table 4.2 Experimental mass transfer coefficients.

4.3 Mathematical model

The mechanism of water diffusion within the porous solids is complex. Different mechanisms for moisture transportation have been proposed (e.g. molecular diffusion, capillary motion, and diffusion through solid pores). However, none of these prevails throughout the total drying process (Zogzas & Maroulis, 1996). Thus, an empirical effective diffusivity is often used to represent the combination of mechanisms. To calculate the effective diffusivity, a mathematical model of the drying process must be used. Modellers often make simplifying assumptions concerning the variations in heat transfer and temperature, the boundary conditions, the dependence of diffusivity on moisture and temperature, and the shrinkage of the product. According to Mulet (1994), the effective diffusivity could lose physical meaning if the model is not carefully written and solved. For instance, if the chosen model is wrong, the calculated diffusivity may be very different from its real value even though the model's prediction fits the experimental data. In this work, three models with different levels of complexity will be compared.

The moisture diffusivity is estimated from the Fick's law according to the equation:

$$\frac{\partial(\rho_s \cdot X)}{\partial t} = \nabla(D \cdot \rho_s \cdot \nabla X) \tag{4.2}$$

Where X is the moisture content (Kg water/ Kg dry material), ρ_s is the concentration of dry solids (Kg solids/m^3), t is the time (s) and D is the moisture diffusivity (m^2/s).

4.3.1 Simplified model (Model A)

This model assumes constant moisture diffusivity without volume change, uni-dimensional moisture movement, uniform initial moisture distribution, negligible external resistances and an isothermal process. With this simplifications equation 4.2 can be solved analytically for the case of an infinity slab (Crank, 1995):

$$X^* = \frac{X_m - X_{eq}}{X_o - X_{eq}} = \frac{8}{\pi^2}\sum_{n=0}^{\infty}\frac{1}{(2n+1)^2}e^{\left(-(2n+1)^2\frac{\pi^2 Dt}{L^2}\right)} \tag{4.3}$$

where X^* is the dimensionless moisture content, X_m is the average water content of the material at time t, X_o is the initial water content, X_{eq} is the equilibrium water content and L is the thickness of the slab. In practice, only the first fifty terms of the infinite series were used, which is much more than needed.

Even though the process is considered isothermal, the diffusivity can be considered as an Arrhenius type temperature function given that the experiments were conducted at 4 temperature levels.

$$D = D_o e^{\left(-\frac{E_o}{RT}\right)} \tag{4.4}$$

In this expression D_o is the Arrhenius factor (m^2/s), E_o is the activation energy for moisture diffusion (KJ / mol), R is the ideal gas constant (KJ/mol/K) and T is temperature (K).

4.3.2 Constant volume model (Model B)

This model assumes constant moisture diffusivity without volume change, uni-dimensional moisture movement and uniform initial moisture distribution but it takes in account temperature changes caused by the evaporation and the effects of the external resistance on the drying process.

The water flux from the solid's surface to the surroundings air stream is calculated by the equation:

$$J_w = h_m (Y_a - Y_s) \tag{4.5}$$

Where h_m is the external mass transfer coefficient (which was experimentally measured above) Y_a is the humidity of the air (Kg water / Kg dry air) and Y_s is the humidity of the air in equilibrium with the meat surface. Y_s is calculated with the following procedure:

- The water activity of the sample at the interface is calculated with the GAB equation (Trujillo *et al.,* 2003) knowing the water content in the surface.
- The vapour pressure of pure water is calculated with a vapour pressure equation knowing the temperature in the interface.

- The vapour pressure in the air-interface (P_s) is calculated as the product of the water activity (a_w) and vapour pressure of pure water (P_V).
- The water mol fraction on the interface ($\bar{y}_s$) is calculated with the equation:

$$\bar{y}_s = \frac{P_s}{P_T} = \frac{a_w P_V}{P_T} \tag{4.6}$$

  Where P_T is the total pressure.

- Y_s is easily calculated with $\bar{y}_s$ and the molecular weight of air and water.

A correction factor was introduced in the GAB equation to make it match the experimental equilibrium value in each of the different samples.

The temperature of the sample was calculated assuming that it reaches a "pseudo wet bulb temperature" following an adiabatic saturation process. The procedure to calculate this temperature is similar to the one used to calculate the wet bulb temperature giving the dry temperature and the humidity:

- From the surface water content of the sample, the water activity is calculated using the GAB equation.
- From the water activity and the dry bulb temperature (air temperature), the "pseudo wet bulb temperature" is estimated following an adiabatic saturation process until the meat sample reaches equilibrium with the humidity of the air.

The diffusivity was also considered as an Arrhenius type temperature function. With these changes equation 4.2 has to be solved numerically. A finite volume method was used to solve equation 4.2.

4.3.3 Variable volume model (Model C)

This model, unlike the previous one, takes in account the shrinkage of the meat sample. Shrinkage was observed on the experimental trials. The meat samples volume reduced up to 30% of the original volume at the end of the drying process. Even though shrinkage should not be considered to calculate the weight loss on beef carcasses, it may strongly affect the calculation of the diffusivity on these drying experiments. According to May and Perre (2002) neglecting the shrinkage on drying modelling is only valid for products that have a very small shrinkage coefficient. The current drying samples showed an average shrinkage coefficient of 70.3%. The shrinkage coefficient is defined with the equation:

$$shrinkage\ coefficient = \left[1 - \frac{V_{final}}{V_{initial}}\right] \times 100 \tag{4.7}$$

Where V_{final} is the final volume and $V_{initial}$ is the initial volume of the sample. V_{final} and $V_{initial}$ where determined with the average diameter and thickness of the meat samples.

Model C assumes that the specific volume, the surface area and the thickness all decrease during the drying process. Uni-dimensional moisture movement,

uniform initial moisture distribution and constant moisture diffusivity are also assumed. Changes in temperature and the effects of the external resistance are calculated using the same procedure followed in model B.

The specific volume of the sample was assumed to depend on moisture content (Zogzas and Moroulis, 1996) according to the equation:

$$v = \frac{1 + \beta \cdot X}{\rho_{bo}} \tag{4.8}$$

where v is the specific volume of the sample (m^3 / kg dry material), β is the volume-shrinkage coefficient, and ρ_{bo} is the bulk density of the sample at zero value of moisture content (Kg dry material / m^3). Equation 4.8, which is a linear model, is adequate to describe material and process conditions leading to a negligible porosity during the drying process, corresponding to a linear decrease of volume in the whole range of humidity (Mayor and Sereno, 2004).

A finite volume method was used to model the process because it can be easily adapted to take shrinkage into account. The first step in this method is to divide the domain into discrete control volume (Versteeg and Malalasekera, 1995). The second step is the integration of the governing equations over a control volume to yield a discretized equation at each nodal point. The last step is to solve the resulting system of linear equations to obtain the distribution of the concentration at nodal points.

The shrinkage was calculated in each volume element as a function of the water content of the element. It is assumed that the shrinkage is isotropic, i.e. the diameter and the thickness decrease in the same proportion. However, the water content does not drop all at once in the meat. After the drying process starts a concentration profile is established inside the meat sample. The water content close to the meat surface is lower than inside the meat. Thus, the model shows a higher shrinkage close to meat surface than the shrinkage at the middle (Figure 4.4).

The level of isotropy of the shrinkage process is defined by the shrinkage isotropicity:

$$Shrinkage\ isotropicity = \frac{(d_o - d)/d_o}{(L_o - L)/L_o} \tag{4.9}$$

where d and L are the diameter and the thickness respectively. The subindices "o" refers to the initial values. A shrinkage isotropicity of 1 means that process is isotropic. The measured shrinkage isotropicity of the meat samples was 1.04 showing a good isotropicity.

Knowing the initial density (ρ_{so}) or the initial specific volume (v_o) when $t = t_o$ and $X = X_o$, equation (4.8) can be expressed as:

$$v = v_o \left(\frac{1 + \beta \cdot X}{1 + \beta \cdot X_o} \right) = v_o \cdot Fx \tag{4.10}$$

or

$$\rho = \rho_{so}\left(\frac{1+\beta \cdot X_o}{1+\beta \cdot X}\right) = \frac{\rho_{so}}{Fx} \tag{4.11}$$

where:

$$Fx = \left(\frac{1+\beta \cdot X}{1+\beta \cdot X_o}\right) \tag{4.12}$$

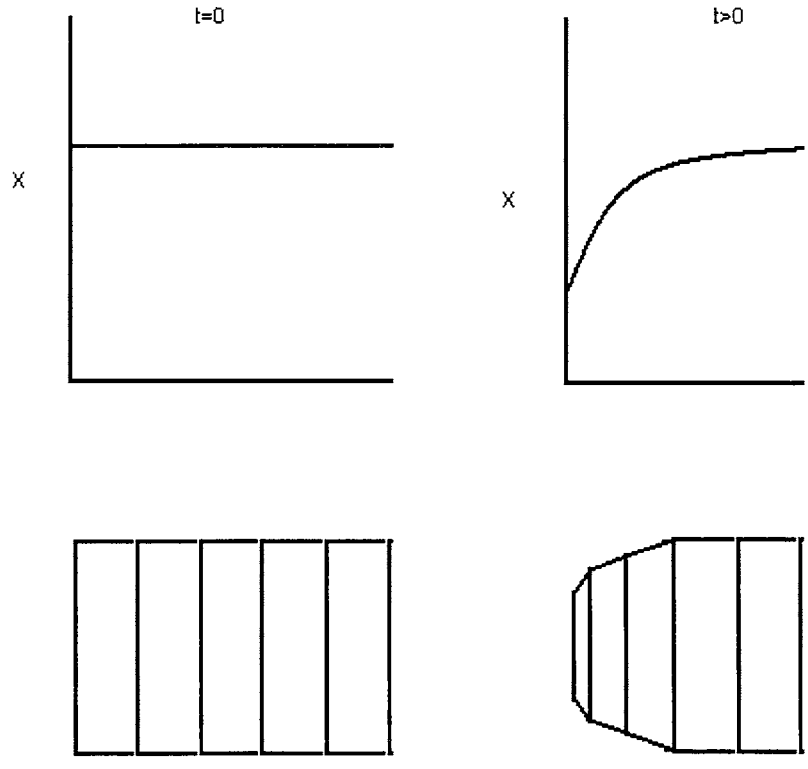


Figure 4.4 Shrinkage process.

The initial volume (V_o) of a volume element is:

$$V_o = A_o \cdot L_o \tag{4.13}$$

where A_o is the initial area and L_o is the initial thickness of the volume element.

From equation (4.10) is clearly seen that the volume at any time is:

$$V = V_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right) = A_o\cdot L_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right) \tag{4.14}$$

Because the shrinkage is isotropic:

$$V = V_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right) = A_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right)^{2/3}\cdot L_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right)^{1/3} = A\cdot L \tag{4.15}$$

$$A = A_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right)^{2/3} = A_o\cdot Fx^{2/3} \tag{4.16}$$

$$L = L_o\left(\frac{1+\beta\cdot X}{1+\beta\cdot X_o}\right)^{1/3} = L_o\cdot Fx^{1/3} \tag{4.17}$$

Where A is the area and L is the thickness expressed as a function of the water content.

Model C incorporate two new constants: ρ_{bo} and β. The former was experimentally determined as ρ_{bo} = 1050.21 ± 9.3% (Kg dry meat/ m^3). β can be determined from equation 4.8 knowing ρ_{bo}, the initial specific volume and the initial water content. The obtained value of β is 0.9360 +-7.5%.

4.3.4 Finite volume method considering shrinkage

The key step of the finite volume method (Versteeg and Malalasekera, 1995) is the integration of the governing equation 4.2 over a control volume and over a finite time step (Δt). Replacing the volume integral of the diffusive term with surface integrals by using the Gauss' divergence theorem equation 4.2 is transformed in:

$$\int_{CV}\int_{t}^{t+\Delta t}\left(\frac{\partial(\rho_s \cdot X)}{\partial t}dt\right)dV = \int_{t}^{t+\Delta t}\left(\int_{A} n\cdot(D\cdot\rho_s\cdot\nabla X)dA\right)dt \tag{4.18}$$

Considering the one-dimensional control volume in figure 4.5

$$\rho_{sP}\cdot\overline{A}_P\left(X_P^{t+1}-X_P^{t}\right)dx = \int_{t}^{t+\Delta t}\left[(A\cdot D\cdot\rho_s\cdot\nabla X)_e-(A\cdot D\cdot\rho_s\cdot\nabla X)_w\right]dt \tag{4.19}$$

Applying the Crank-Nicholson scheme, knowing that:

$$A = A_o\cdot Fx^{2/3}$$

$$\nabla X_e = \frac{X_E - X_P}{dx_{PE}}$$

$$\nabla X_w = \frac{X_P - X_W}{dx_{WP}}$$

and with some algebraic simplifications (Versteeg and Malalasekera, 1995), it is found the following set of equations for central volume (i = 2, 3 ..., Nv-1), the first volume (i = 1) and the last volume (i = N):

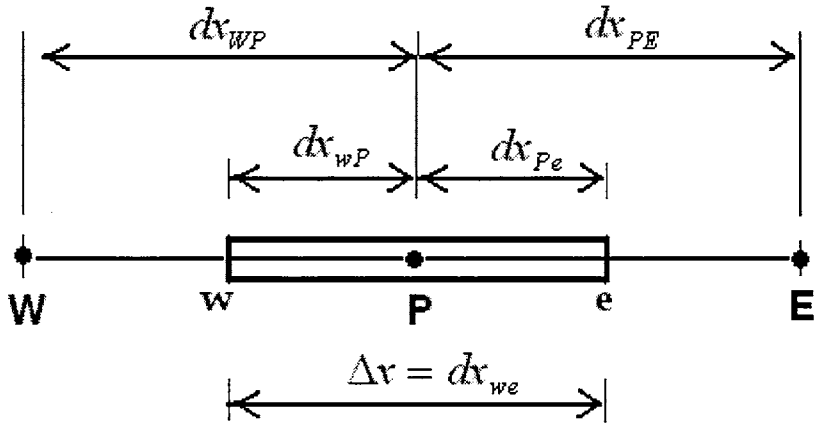


Figure 4.5 Finite Volume grid. Nodal point are identified with upper case letter i.e. W, P, E. Boundaries or faces are positioned midway between adjacent nodes and are identified with lower case letters i.e. w, e.

4.3.4.1 *Central volume*

$$-\frac{1}{2}a_W \cdot X_W^{t+1} + a_P \cdot X_P^{t+1} - \frac{1}{2}a_E \cdot X_E^{t+1} = S_U \tag{4.20}$$

where:

$$a_W = \frac{\left(Fx_w^t\right)^{2/3} \cdot \rho_{sw}^{\ t} \cdot D_w^t}{\left(Fx_P^t\right)^{2/3} \cdot dx_{WP}^t \cdot \rho_{sP}^{\ t}} \tag{4.21}$$

$$a_E = \frac{\left(Fx_e^t\right)^{2/3} \cdot \rho_{se}^{\ t} \cdot D_e^t}{\left(Fx_P^t\right)^{2/3} \cdot dx_{PE}^t \cdot \rho_{sP}^{\ t}} \tag{4.22}$$

$$a_P = a_{Po} + \frac{1}{2}\left(a_E + a_W\right) \tag{4.23}$$

$$a_{Po} = \frac{\Delta x^t}{\Delta t} \tag{4.24}$$

$$S_U = \frac{1}{2}a_E \cdot X_E^t + \frac{1}{2}a_W \cdot X_W^t + \left[a_{Po} - \frac{1}{2}\left(a_E + a_W\right)\right] \cdot X_P^t \tag{4.25}$$

The water content on the volume boundary faces e and w is calculated as:

$$X_e^t = \frac{1}{2}\left(X_P^t + X_E^t\right) \tag{4.26}$$

$$X_w^t = \frac{1}{2}\left(X_W^t + X_P^t\right) \tag{4.27}$$

Equation 4.12 is used to calculate $(Fx)^{2/3}$, which can be taken as a correction factor caused by the shrinkage of the surface area:

$$Fx_w^t = \left(\frac{1+\beta \cdot X_w^t}{1+\beta \cdot X_o}\right)$$

$$Fx_e^t = \left(\frac{1+\beta \cdot X_e^t}{1+\beta \cdot X_o}\right)$$

$$Fx_P^t = \left(\frac{1+\beta \cdot X_P^t}{1+\beta \cdot X_o}\right)$$

Equation 4.11 is used to calculate the density as:

$$\rho_{sw}^{t} = \rho_{so} \cdot \left(\frac{1+\beta \cdot X_o}{1+\beta \cdot X_w^t} \right)$$

$$\rho_{se}^{t} = \rho_{so} \cdot \left(\frac{1+\beta \cdot X_o}{1+\beta \cdot X_e^t} \right)$$

$$\rho_{sP}^{t} = \rho_{so} \cdot \left(\frac{1+\beta \cdot X_o}{1+\beta \cdot X_P^t} \right)$$

From equation 4.17 it clearly seen that:

$$dx = dx_o \left(\frac{1+\beta \cdot X}{1+\beta \cdot X_o} \right)^{1/3} \tag{4.28}$$

Thus, the above equation is used to calculate the shrinkage on dx as:

$$dx_{WP}^{t} = dx_{WPo} \left(\frac{1+\beta \cdot X_w^t}{1+\beta \cdot X_o} \right)^{1/3}$$

$$dx_{PE}^{t} = dx_{PEo} \left(\frac{1+\beta \cdot X_e^t}{1+\beta \cdot X_o} \right)^{1/3}$$

$$\Delta x^{t} = \Delta x_o \left(\frac{1+\beta \cdot X_P^t}{1+\beta \cdot X_o} \right)^{1/3}$$

where dx_{WPo}, dx_{PEo} and Δx_o are the initial values of dx_{WP}, dx_{PE} and Δx respectively.

4.3.4.2 *First volume*

On the west face of the first volume a convective boundary condition applies. Thus:

$$\left(D \cdot \rho_s \cdot \nabla X\right)_w = mass\_flux \tag{4.29}$$

where:

$$mass\_flux = h_m\left(Y_a - Y_s\right) \tag{4.30}$$

Thus, equation 4.20 is transformed in:

$$a_P \cdot X_P^{t+1} - \frac{1}{2} a_E \cdot X_E^{t+1} = S_U \tag{4.31}$$

where:

$$a_W = 0 \tag{4.32}$$

$$S_U = \frac{1}{2} a_E \cdot X_E^t + \left[a_{Po} - \frac{1}{2}\left(a_E\right)\right] \cdot X_P^t + b \tag{4.33}$$

$$b = \frac{mass\_flux}{\rho_{sP}^t \cdot \left(Fx_P^t\right)^{2/3}} = \frac{h_m\left(Y_a - Y_s\right)}{\rho_{sP}^t \cdot \left(Fx_P^t\right)^{2/3}} \tag{4.34}$$

4.3.4.3 *Last volume*

The east face of the last volume is isolated. There is not mass flux, thus:

$$\left(D \cdot \rho_s \cdot \nabla X\right)_e = 0$$

Equation 4.20 is transformed in:

$$-\frac{1}{2} a_W \cdot X_W^{t+1} + a_P \cdot X_P^{t+1} = S_U \tag{4.35}$$

where:

$$a_E = 0 \tag{4.36}$$

4.3.5 Determination of the Diffusivity

The diffusivity was determined by minimizing the square of differences between the experimental data and the prediction by the model. Pham's (1994) evolutionary algorithm was chosen to minimize the function. This method, unlike the classical minimization methods that advance in the gradient direction, starts with a "population" of trial points chosen randomly, which evolve towards the optimum using various mechanisms such as "reproduction", "mutation" and "selection". Reproduction is the creation of a new and hopefully better trial point, starting from two or more existing trial points, using extrapolation, interpolation, local exploration and other techniques. Mutation is the creation of a new trial point by randomly modifying some features of one existing trial point. Selection is the elimination of one or more trial points in order to get a "fitter", or more optimal, population. All these steps are highly randomized. The advantage of stochastic methods such as evolutionary algorithms is that they deal better with objective functions that contain random errors arising out of measurements or numerical calculations. Classical methods generally follow a single search path, thus, they may end up at a local minimum instead of the desired global one and are easily misled by small errors in estimating the objective function. The method was programmed in Visual Basic. Appendix A1 shows the sub routine that executes the finite volume method with shrinkage.

4.4 Results and Analysis

Figures 4.6 to 4.9 show drying curves at 6.8 °C, 19.9 °C, 27.9 °C and 40.4 °C respectively and compare the experimental with the simulated data using the model A, B and C. It is clearly seen that Model C (shrinkage – finite volume) gets the better fit followed by model B (convective boundary - finite difference). Model A (constant diffusivity) gets the worst fit.

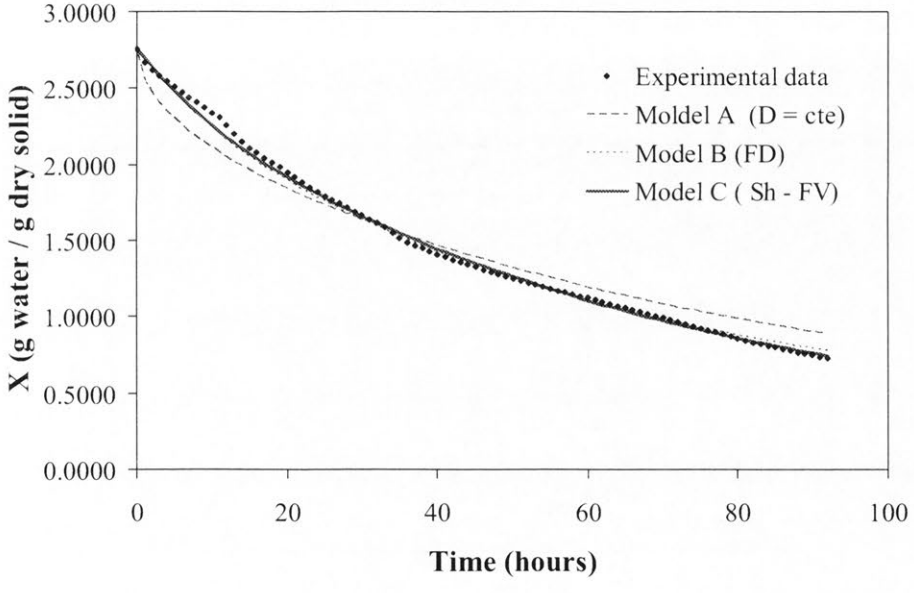


Figure 4.6 Comparison of drying models A, B and C at 6.8 °C, 92 % RH, Sample 1.

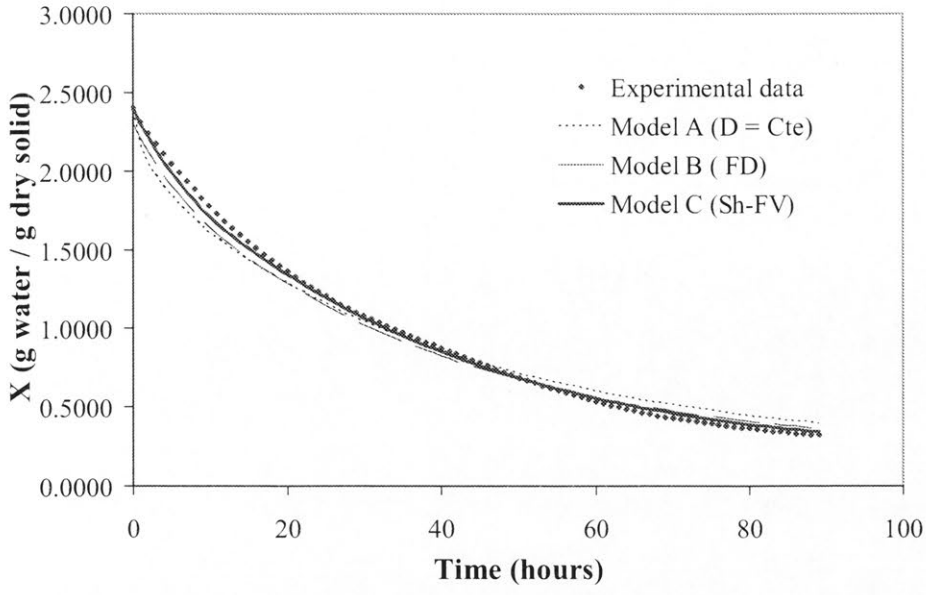


Figure 4.7 Comparison of drying models A, B and C at 19.9 °C, 79.6 % RH, Sample 3.

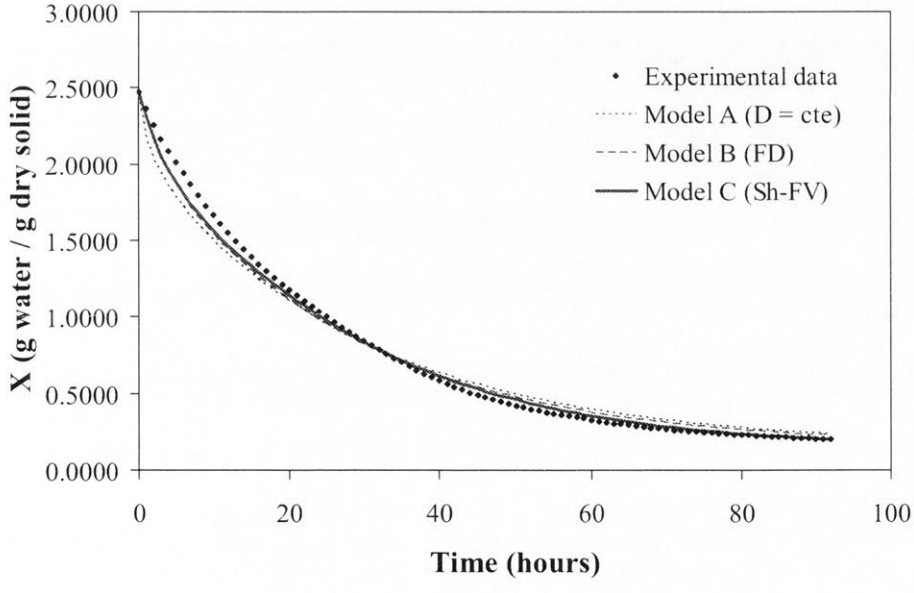


Figure 4.8 Comparison of drying models A, B and C at 27.9 °C, 77.6 % RH, Sample 1.

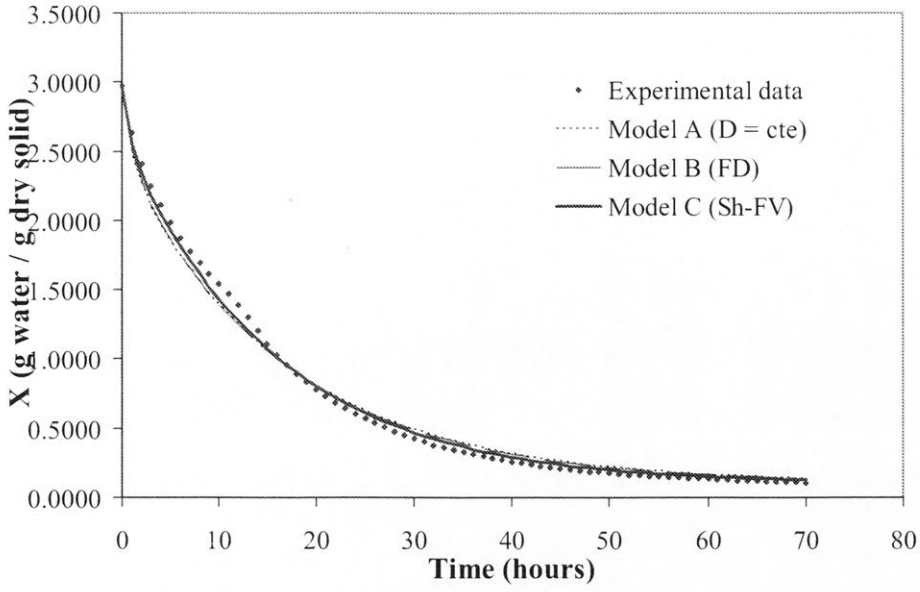


Figure 4.9 Comparison of drying models A, B and C at 40.4 °C, 78.1 % RH, Sample 3.

Table 4.3 shows the average calculated diffusivity using each one of the models at the four temperature levels. Three replicates were made at each temperature level. The standard deviation is smaller than 13% for all the cases, showing good reproducibility of the experiments.

	Temperature (^{0}C)	6.8	19.9	27.9	40.4
Model A	Diffusivity (m^2/s)	1.62E-10	2.82E-10	3.68E-10	7.04E-10
	% SD	3.41%	8.79%	7.36%	11.44%
Model B	Diffusivity (m^2/s)	2.06E-10	3.21E-10	3.99E-10	7.16E-10
	% SD	3.82%	9.68%	9.17%	12.53%
Model C	Diffusivity (m^2/s)	1.41E-10	2.13E-10	2.56E-10	3.73E-10
	% SD	3.98%	10.14%	10.77%	6.13%

Table 4.3 Average calculated diffusivity and % Standard deviation (%SD) using models A, B and C.

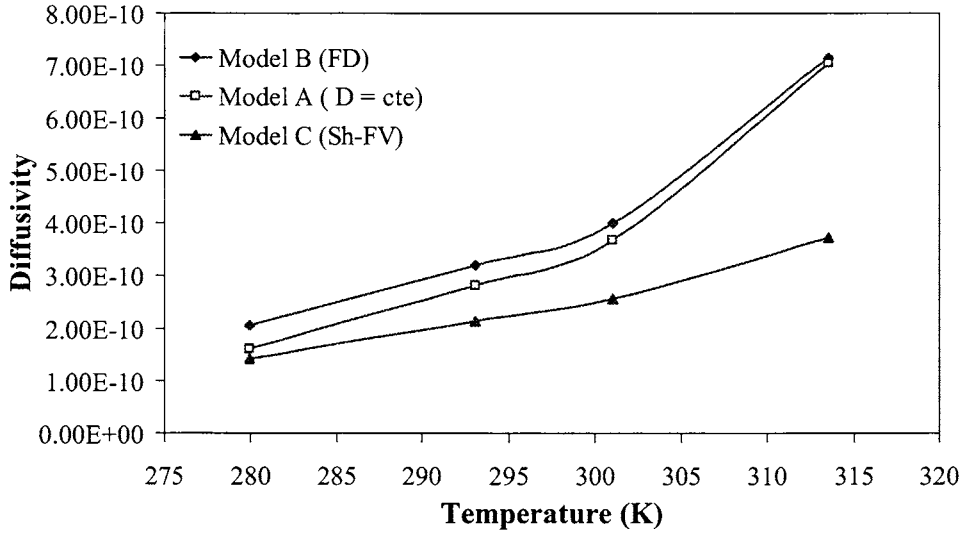


Figure 4.10 Calculated diffusivity using models A, B and C vs. temperature.

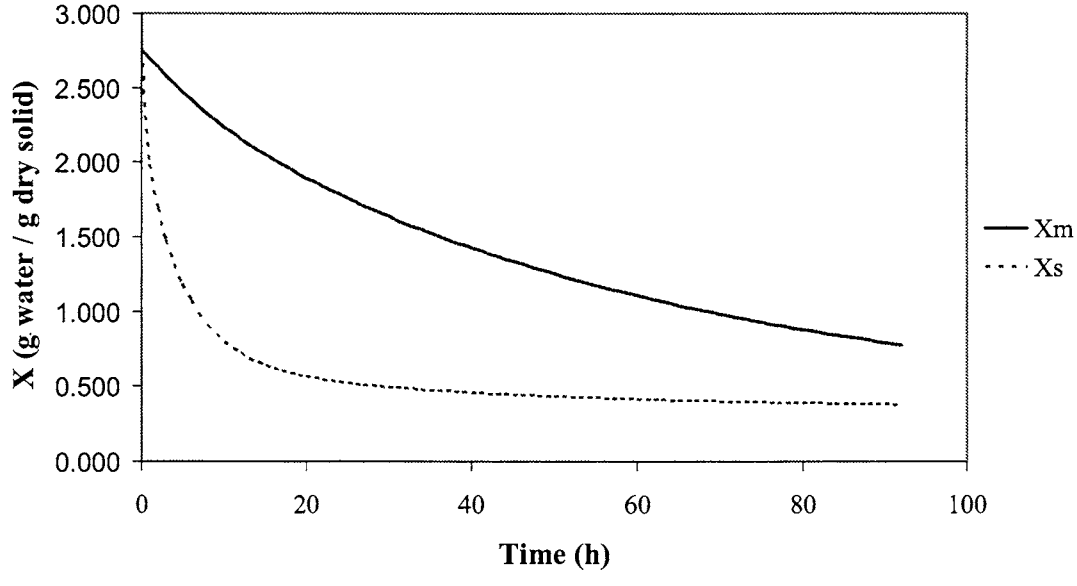


Figure 4.11 X average (X_m) and X surface (X_s) vs. time calculated with model B at 6.8 °C, 92 % RH, Sample 1.

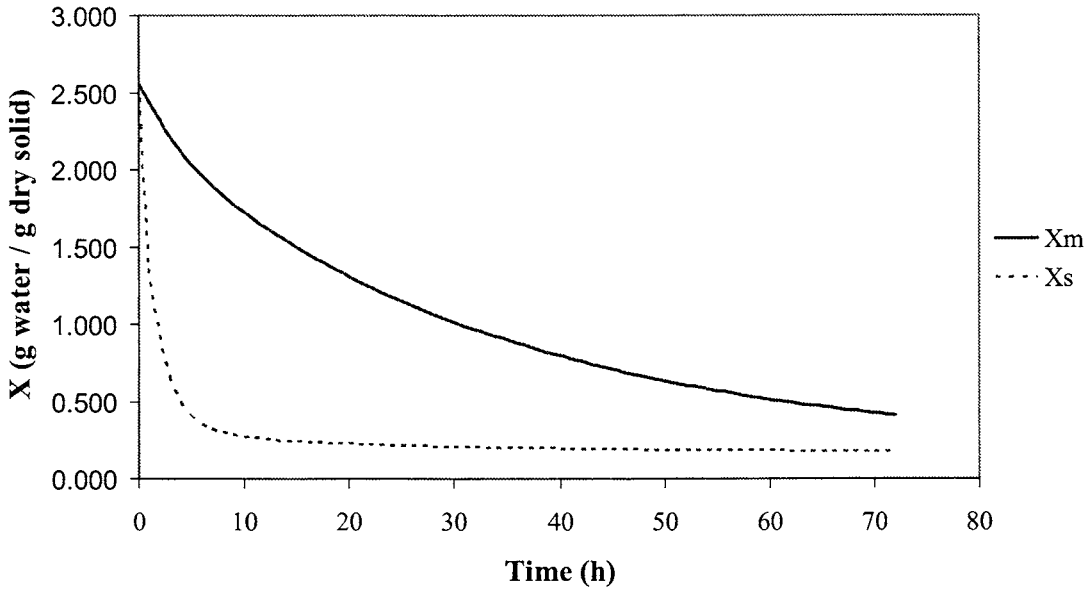


Figure 4.12 X average (X_m) and X surface (X_s) vs. time calculated with model B at 19.9°C, 79.6 % RH, Sample 1.

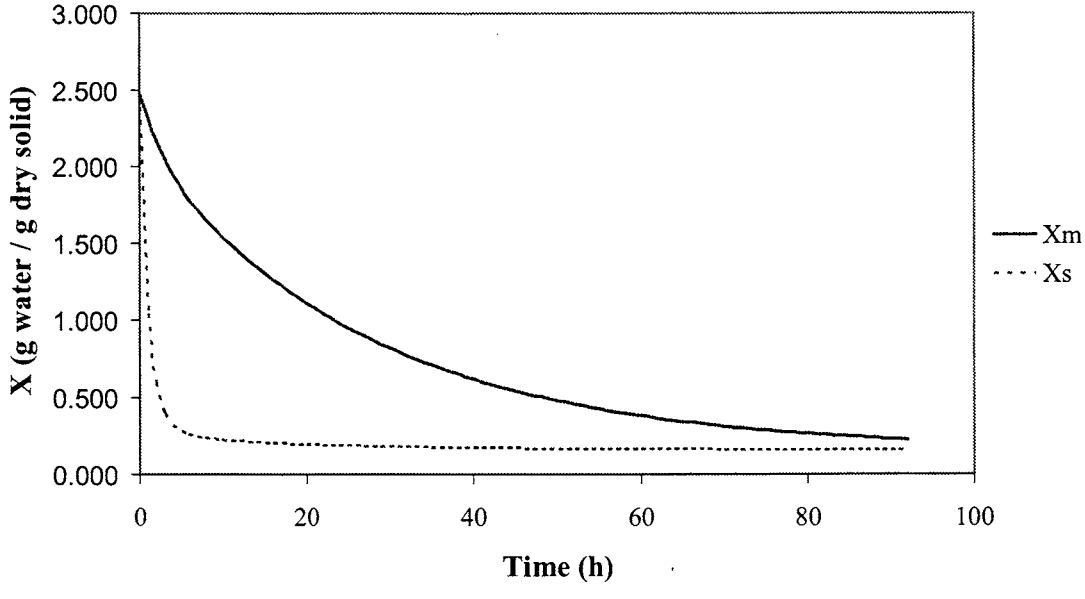


Figure 4.13 X average (X_m) and X surface (X_s) vs. time calculated with model B at 27.9 °C, 77.6 % RH, Sample 1.

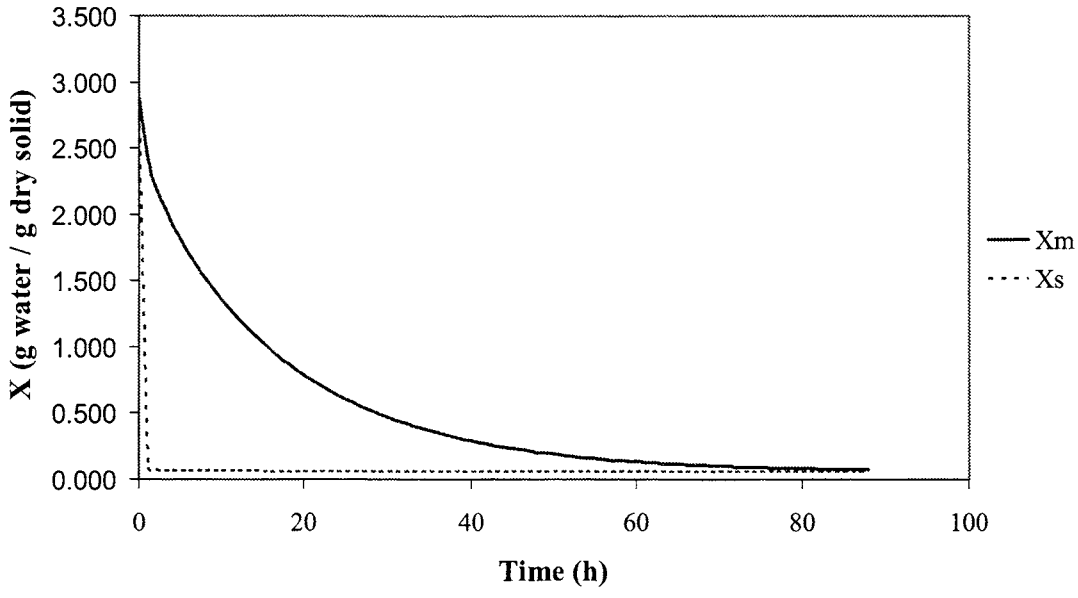


Figure 4.14 X average (X_m) and X surface (X_s) vs. time calculated with model B at 40.4 °C, 78.1 % RH, Sample 1.

Figure 4.10 plots the calculated diffusivity as a function of temperature. The values obtained with model C are much smaller (down to 50%) than those calculated with models A or B, which is as expected since the diffusion path is much shorter in Model C. Mulet (1994) reported similar differences for carrots when taking shrinkage into account. In the graph, it is seen that the difference between models A and B is bigger at low temperatures but decreases at higher temperatures, getting to be negligible at the highest temperature. That can be explained with figures 4.11 to 4.14 where the average (X_m) and surface (X_s) water content obtained with model B are plotted as a function of time. In these graphics it is seen that at the lowest temperature (6.8°C) the surface water content

reaches the equilibrium rather slowly (figure 4.11), meaning that convection on the boundary is important. That explains the higher difference comparing with model A (that assumes that the surface is always in equilibrium with the air). On the other hand, at the highest temperature (40.4 °C) the surface water activity drops to the equilibrium almost immediately after starting the drying process starts (figure 4.14). Therefore, the calculated diffusivity with models A and B are similar.

Figures 4.15 to 4.17 plot the surface water activity Vs time showing that at higher temperatures (Figure 4.17) the sample surface reaches the equilibrium faster than at lower temperatures (Figure 4.15).

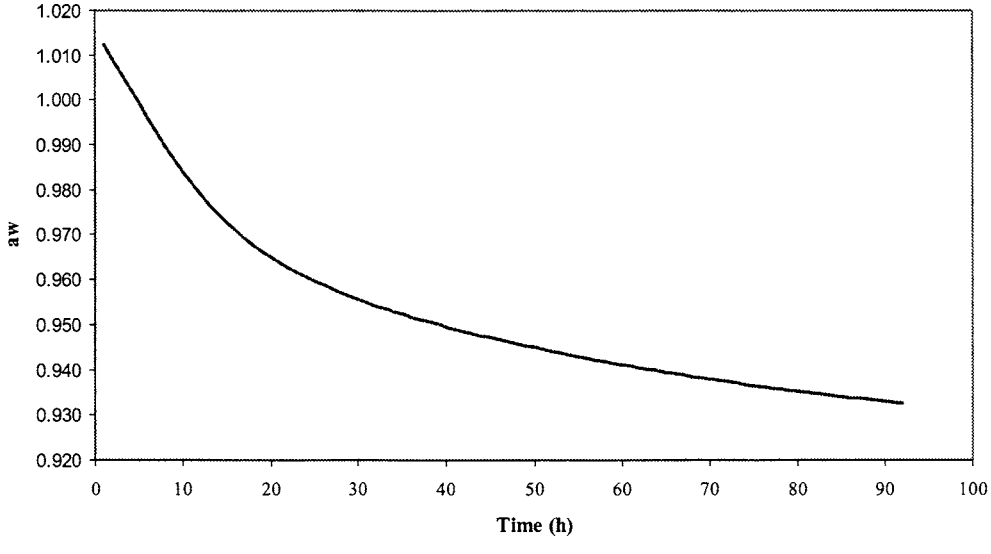


Figure 4.15 Water activity (aw) on the meat surface vs. time calculated with model A at 6.8 °C, 92 % RH, Sample 1.

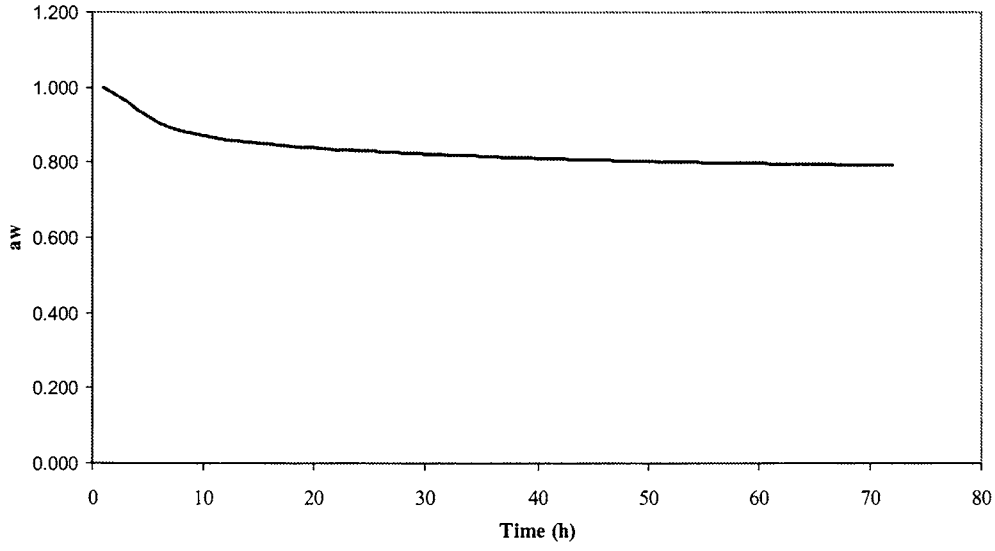


Figure 4.16 Water activity (aw) on the meat surface vs. time calculated with model A at 19.9°C, 79.6 % RH, Sample 1.

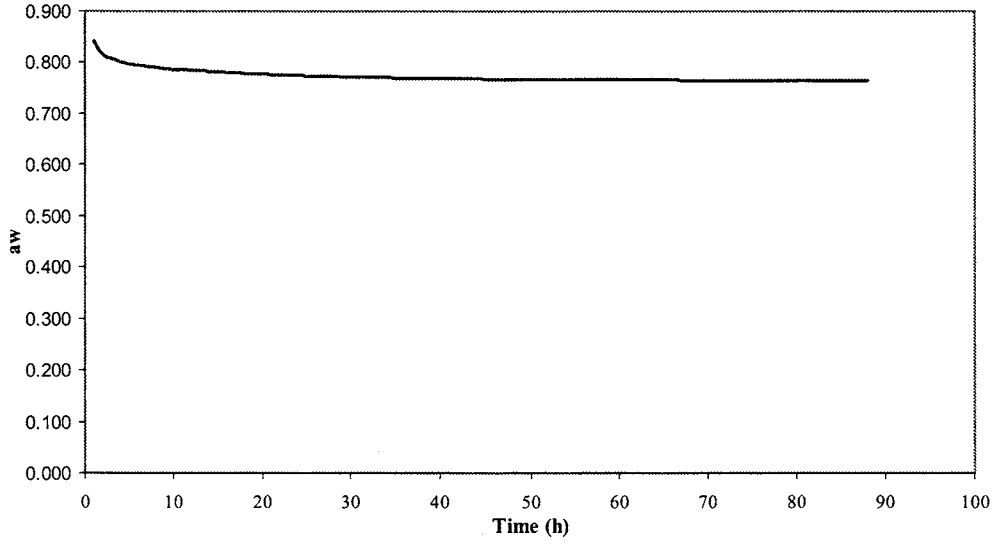


Figure 4.17 Water activity (aw) on the meat surface vs. time calculated with model C at 40.4 °C, 78.1 % RH, Sample 1.

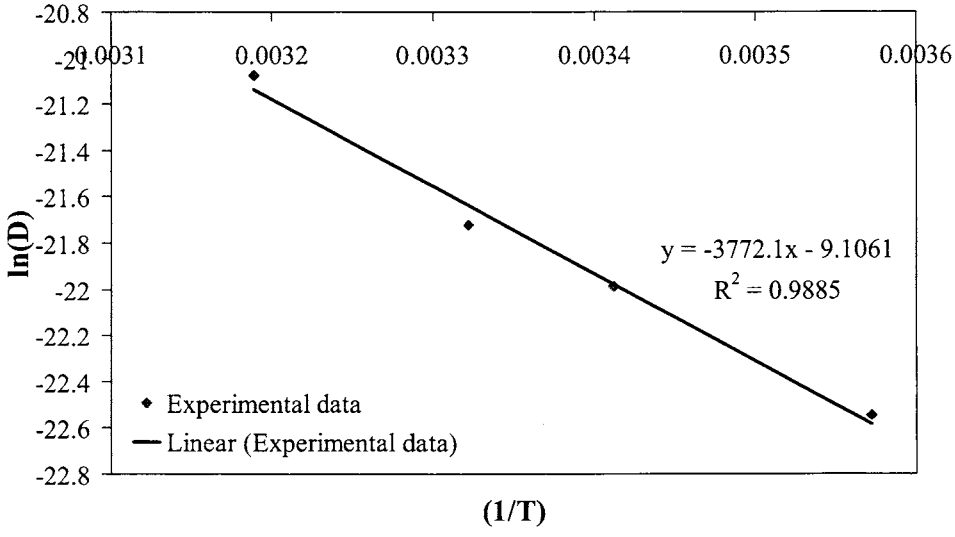


Figure 4.18 Ln (D) Vs (1/T) of diffusivities calculated with model A.

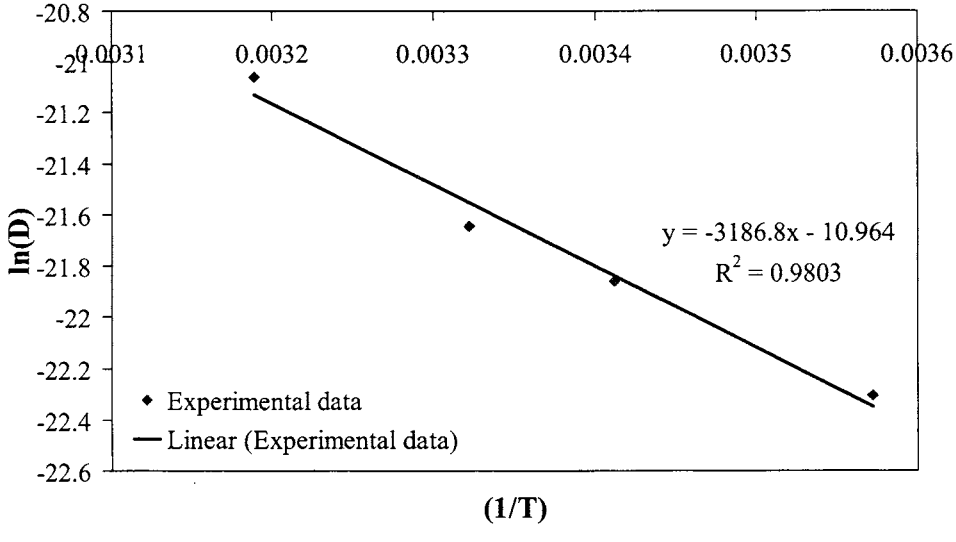


Figure 4.19 Ln (D) Vs (1/T) of diffusivities calculated with model B.

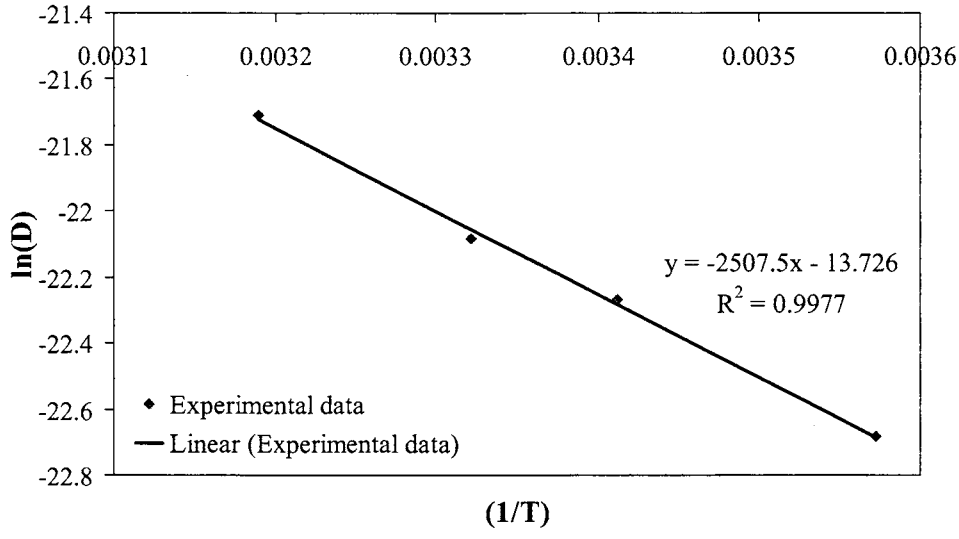


Figure 4.20 Ln (D) Vs (1/T) of diffusivities calculated with model C.

	model A	**Model B**	**Model C**
Slope	-3772.14	-3186.82	-2507.52
E_a	-31361.53	-26495.24	-20847.5
Intercept	-9.11	-10.96	-13.73
D_O	1.11E-04	1.73E-05	1.09E-06
Correlation Factor	0.9885	0.9803	0.9977

Table 4.4 Parameters of the Arrhenius equation (4.5) calculated with model A, B, and C.

Model	**%RMS Average**
Model A	1.269%
Model B	0.874%
Model C	0.564%

Table 4.5 Average %RMS percent error fitting the experimental data with models A, B and C.

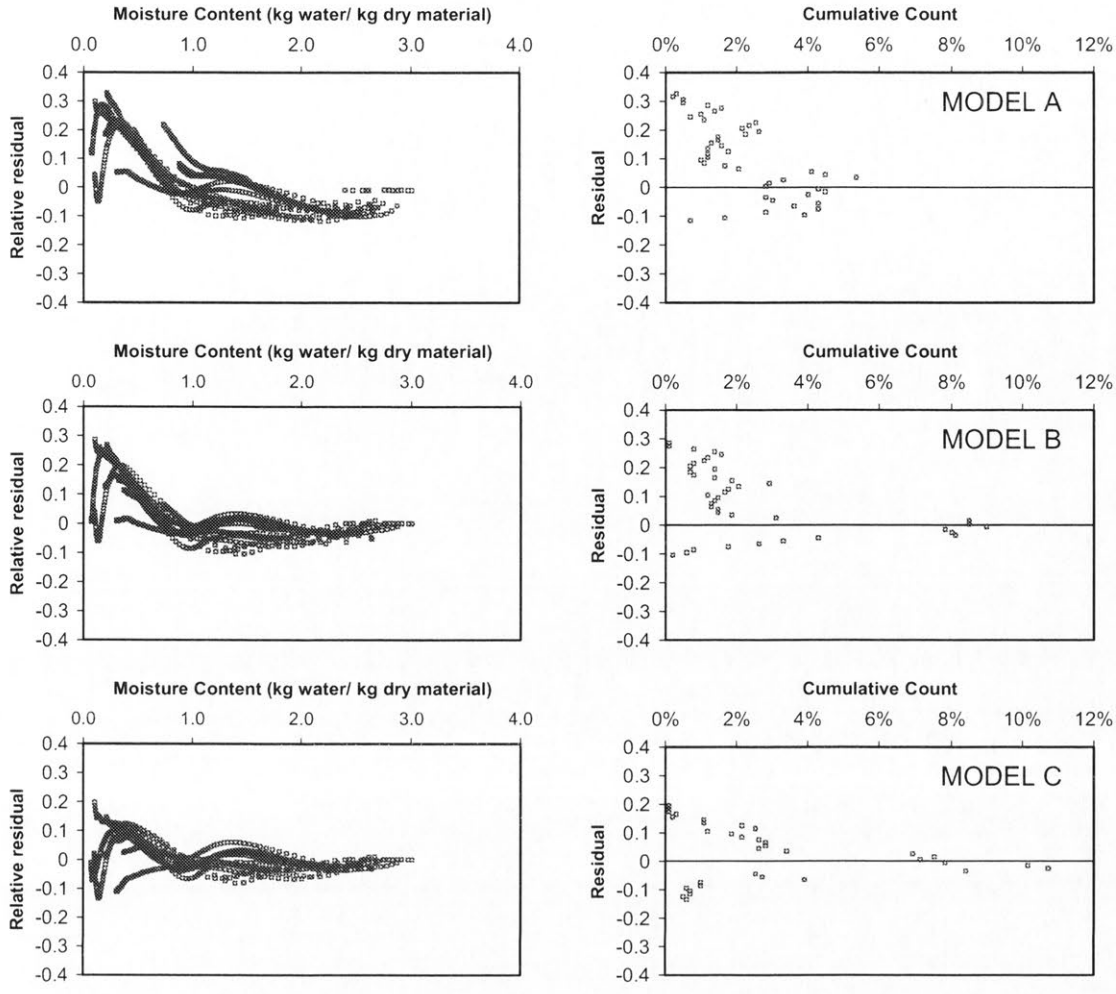


Figure 4.21 Relative value residual vs. moisture content (left) and relative residual cumulative distribution (right).

The diffusivity–temperature data was used to express the diffusivity as an Arrhenius type relation (equation 4.4). Figures 4.18, 4.19 and 4.20 plot the natural logarithm of the diffusivity (ln (D)) as a function of the inverse of temperature (1/T) with models A, B and C respectively. These graphics show a good correlation coefficient. Thus, it is concluded that equation 4.4 fits the data well. The best correlation factor was obtained with the diffusivity calculated with

model C (Shrinkage – Finite Volume). Table 4.4 shows the constants E_a and D_O of the three models.

Table 4.5 shows the average root mean squares percent error (%RMS) using models A, B and C. On the table it is seen that the %RMS reduces from model A to C, showing that the model that fit the data best is the model C. Figure 4.21 shows the relative value residual (RVR) vs. the moisture content and the relative residual cumulative distribution of the models. Models A and B exhibit a strong trend to positive RVR values and can not be considered of normal shape. Meaning that they over-predict the water content comparing with the experimental values. Model C exhibits the best behaviour, since it shows the closer normal shape population, even though it shows a slight trend to positive RVR values.

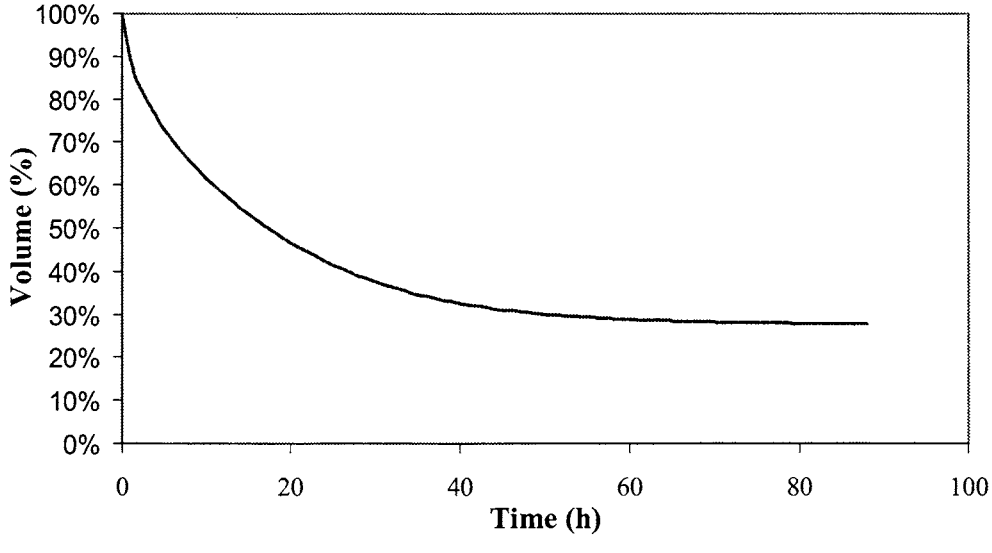


Figure 4.22 Change of volume, expressed as percentage, during drying at 40.4 C, 78.1 % RH, Sample 1 (values obtained with model C).

The better behaviour of model C is expected since the meat samples suffered strong shrinkage during the drying (experimental shrinkage coefficient = 70.3%). Figure 4.22 shows the shrinkage process according to model C. Mulet (1994), who tested several detailed models with different levels of complexity, concluded that the main factor that must be considered to accurately determine the diffusivity is the product shrinkage. According to Hernandez, Pavon & Garcia (2000) food shrinkage is extremely important in diffusional drying because it produces a variation in the distance required for the movement of water molecules. Mayor and Sereno (2004) found that models with shrinkage fits better experimental data than model without shrinkage. They suggested that a model that takes shrinkage into account leads to better predictions of values of the effective diffusivity, moisture content profiles and average values of moisture content during the process.

	Estimated uncertainty	**Absolute % error from estimated uncertainty**
Initial solid density ρ_{so}	$\pm$ 5% ρ_s	3.55%
Sample thickness, (m)	±0.001 m	-0.03%
Mass transfer coeff. h_m	$\pm$ 6% h_m	2.83%
Air RH (0 to 1)	±1% RH	13.08%
Air temperature, (°C)	±1 °C	3.24%
Accumulated error $\left[\sqrt{\left(\sum error^2\right)}\right]$**:**		**14.34%**

Table 4.6 Sensitivity of the calculated diffusivity with model C to uncertainties in the model input values.

The average error of model C was 7.30%. The sensitivity of the diffusivity calculated with model C to the input values was determined for a run at 6.8 C. The model inputs were varied one at a time, by an amount corresponding to their estimated uncertainties. The results are given in table 4.6. It shows that the model has a high sensitivity to the relative humidity. The accumulated error was 14.34%, indicating that the likely uncertainties in the model inputs could explain the 7.30% error of the model.

4.5 Conclusions

- Experimental drying curves obtained at 6.8°C, 19.9°C, 27.9°C and 40.4°C were fitted using three different models: A (constant diffusivity, constant volume, constant temperature and surface moisture), B (convective boundary condition) and C (shrinkage taken into account) to determine the diffusivity of water in beef perpendicular to the fibres.

- The difference between the diffusivity obtained with models A and B decreases when the temperature decreases. This is caused by the slower equilibration of the surface at lower temperatures.

- The model that fits the experimental data best and exhibits the closer normal shape population of the cumulative relative residual value is Model C, which takes shrinkage into account, because it is a better representation of the actual physical phenomena.

- The relation diffusivity-temperature can be accurately expressed with an Arrhenius type equation. The obtained constants using model B are D_o = 1.73E-05 m^2 /s and E_a = -26495.24 J/ mol .K. and those using model C are: D_o = 1.09E-06 m^2 /s and E_a = -20847.5 J/ mol .K.

5. CFD MODELLING OF THE HEAT AND MASS TRANSFER PROCESS DURING THE EVAPORATION OF WATER FROM A CIRCULAR CYLINDER

5.1 Introduction

Heat and mass transfer between air and food products are involved in many food processing operations, such as freezing, drying and chilling. To control and optimize these processes, it is necessary to accurately know the local heat and mass transfer coefficients. They also determine the local temperature and surface water activity, which are two of the most important properties in the control of the bacterial growth.

Heat and mass transfer coefficients depends on the development of the momentum, thermal and mass boundary layers. Thus, they are affected by the geometry, air velocity, temperature and turbulence. These heat and mass transfer coefficients are fundamental inputs to chilling and freezing calculation on beef carcasses, and must therefore be know accurately to enable representative studies (Harris and Willix, 2000).

Computational Fluid Dynamics (CFD) has become an important tool to theoretically determine heat and mass transfer coefficients. It can be used to model the chilling of meat carcasses, operation that used low air velocities (low Reynolds numbers) but may have high turbulence intensity.

Unfortunately, the selected turbulence model and the near wall treatment approach used on the modelling may drastically affect the accuracy and reliability of CFD modelling. For instance, no single turbulence model is universally accepted as being superior for all classes of problems (FLUENT Inc, 2003a). Most of the widely known models, as the standard κ-ε model, have been developed for fully turbulent flows but the chilling of meat carcasses is conducted at low Reynolds numbers. Turbulence flows are also significantly affected by the presence of walls. The near wall modelling significantly impacts the fidelity of numerical solutions. Therefore, some experimental validations, under similar conditions of meat chilling (low Reynolds numbers and turbulence), must be done to find the turbulence model and wall treatment that better model this particular process.

The experiments were fully conducted by the MINIRZ team in New Zealand. They were originally aimed at developing a method for measuring local mass transfer coefficients on carcases shaped objects in different air flow conditions. A wet circular cylinder under cross flow conditions was used to experimentally determine the heat and mass transfer coefficients. The experimental data was

used to validate the CFD modelling and determine which turbulence model and wall treatment fits better the data. Given that the two process, experiments of water evaporation and beef chilling, happens at low Reynolds numbers and are affected by the presence of the wall, the turbulence model and wall treatment approach that better fit the water evaporation experiments are expected to work well on the beef chilling modelling.

The experimental procedure described below (section 5.2) is taken from Harris and Willix (2000) (MINIRZ work). The obtained data was used to validate the CFD modelling.

5.2 Experimental procedure

Kondjoyan and Daudin's (1993a) approach of using a steady-state technique to estimate average surface heat and mass transfer coefficients was followed. Two wet plaster samples were placed into a wind tunnel (one for temperature measurement, one for weight measurement) and held under constant air conditions. Plaster samples were used, as it was claimed that the plaster surface remained fully wet for long periods of time. When steady state was reached, the heat extracted from the sample by evaporation was balanced by that provided from the air stream through convection and radiation to the surface. Under these conditions, the surface of the sample approached the wet-bulb temperature of the air. The average sample heat and mass transfer coefficients were then computed

based on the average surface temperature and the rate of evaporation (weight loss) measured from the samples.

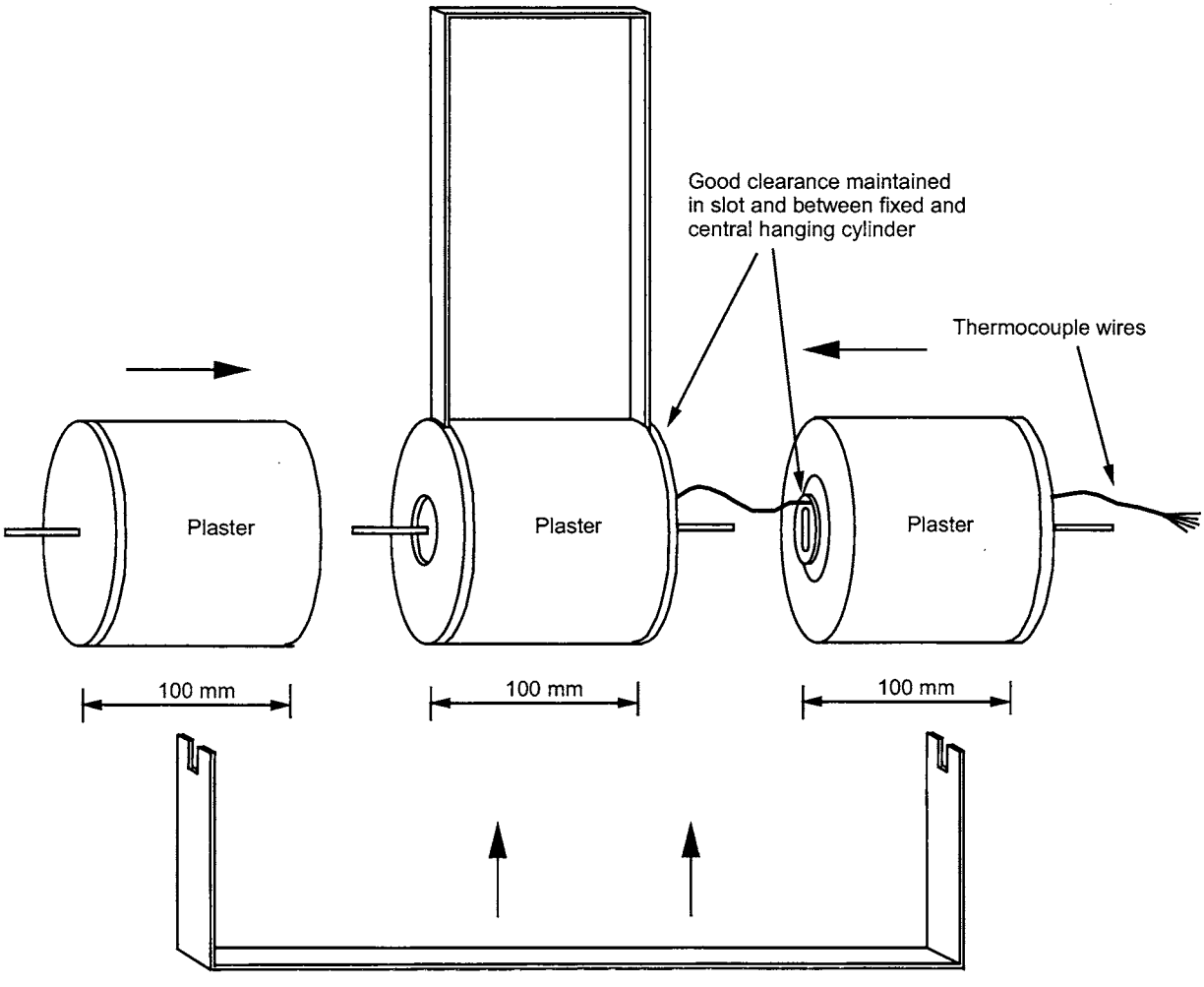


Figure 5.1 Experimental apparatus to minimise aerodynamic, heat and mass transfer edge effects.

The apparatus used comprised a 100mm diameter plaster cylinder, with a length also of 100mm. To reduce aerodynamic edge effects the plaster cylinder was located between two extra wet plasters as it seems in figure 5.1.

The two plaster assemblies were hung in a controlled environment wind tunnel. One of the samples was weighed during the weight loss experiment, while the other had T type thermocouples inserted near the surface of the plaster to measure the surface temperature (figure 5.2). Thus, any influence of the thermocouple wires on sample weight measurement was reduced.

During drying experiments, the air stream temperature, humidity and velocity were maintained at constant pre-determined levels. Plaster surface temperatures, air stream temperature, humidity, and sample weight were recorded versus time.

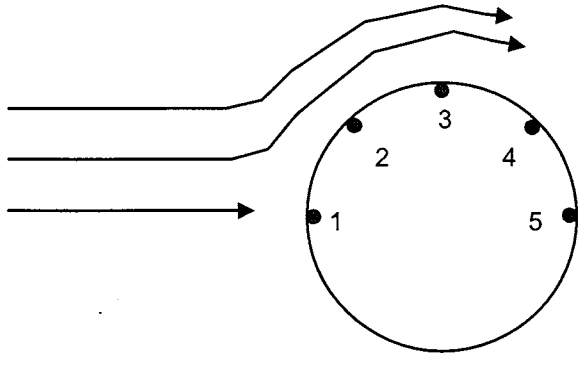


Figure 5.2 Thermocouples inserted just below the surface of the cylinder.

At the start of each run, the plaster surface temperature took a little time to reach a steady state temperature distribution. This value was within 0.2 – 1.8 degrees of the wet bulb temperature, depending on the air velocity over the cylinder. Once this steady temperature was achieved, the rate of evaporation from the cylinder surface remained steady until most of the water in the plaster had evaporated (constant rate dying). After this point the surface temperatures would begin to

rise, as mass transfer within the plaster began to limit the drying rate. The recorded data used was that obtained during the steady state stage.

This method has been used before to determine local heat and mass transfer coefficients on elliptical cylinders (Kondjoyan and Daudin, 1993b) and at the surface of a pork hindquarter (Kondjoyan and Daudin, 1997).

5.2.1 Calculation of experimental heat and mass transfer coefficient

A heat balance over the surface of the plaster during the constant rate drying period may be written:

$$h_m \Delta H_{vap}(Y_a - Y_w) + \varepsilon_r \sigma (T_a^4 - T_w^4) = h_t (T_w - T_a) \tag{5.1}$$

and the rate of mass transfer from the surface, J_w, as

$$J_w = h_m (Y_a - Y_w) \tag{5.2}$$

Rearranging these equations and integrating them over the surface of the plaster cylinder enables the overall heat and mass transfer coefficients to be calculated from measurements of the rate of weight loss, the plaster surface temperatures, air stream temperature and humidity:

$$\overline{h}_m = \frac{\overline{J}_w}{(Y_a - \overline{Y}_w)} \tag{5.3}$$

$$\overline{h}_t = \frac{-\overline{J}_w \Delta H_{vap}}{(T_a - T_w)} - \frac{\varepsilon_r \sigma (T_a^4 - \overline{T}_w^4)}{(T_a - \overline{T}_w)} \tag{5.4}$$

These equations incorporate some small spatial averaging errors. However, these errors were shown to be relatively insignificant by Kondjoyan and Daudin (<3% effect on h and h_m).

The mass transfer coefficients can also be calculated from the heat transfer coefficient if the relationship between heat and mass transfer is known. For laminar flow over flat plates it can be shown that:

$$\frac{h_t}{h_m} = c_p Le^{2/3} \tag{5.5}$$

According to Lewis (1971) Equation 5.5 may be used to predict heat transfer values from mass transfer measurement. The error involved is much less than the deviation caused by such factors as free stream turbulence and wind tunnel blockage. Kestin and Wood (1971) used equation 5.5 to calculate the wall temperature and local mass transfer coefficient before and after the detachment of the boundary layer. This relationship (Chilton-Colburn analogy) has also been shown to hold for more complicated boundary layer flows including turbulent flow over a flat plate, laminar and turbulent flow over cylinders.

Introducing the variable, K, representing the ratio of heat to mass transfer:

$$K = \frac{h_t}{h_m \Delta H_{vap}} = \frac{\overline{h}_t}{\overline{h}_m \Delta H_{vap}} \tag{5.6}$$

the local heat and mass transfer coefficients may be calculated directly from the surface temperature measurements as:

$$h_t = \frac{\varepsilon_r \sigma \left(T_a^4 - T_w^4\right)}{\left[\left(T_w - T_a\right) + \left(Y_w - Y_a\right)/K\right]} \tag{5.7}$$

and

$$h_m = \frac{\varepsilon_r \sigma \left(T_a^4 - T_w^4\right)}{\Delta H_{vap}\left[K\left(T_w - T_a\right) + \left(Y_w - Y_a\right)\right]} \tag{5.8}$$

where Y_w can be calculated from the saturation curve knowing the local temperature.

5.3 Mathematical Model

Four mathematical models were used to calculate the turbulence and two different approaches were followed to solve the wall. The continuity, velocity, energy, moisture content, and additional turbulent transport equations were solved using FLUENT 6.1.18. The used turbulent models are:

1) Standard κ-ε model: it is a semi-empirical method based on the transport equation for turbulent kinetic energy (κ) and its dissipation rate (ε) (Launder and Spalding, 1974). This model was developed assuming that

the flow is fully turbulent and the effects of molecular viscosity are negligible.

2) The RNG- κ-ε model: This model is derived from the Navier-Stokes equations using the "renormalization group" method that results in a model with constants different from those in the standard κ-ε model. It also results in a differential equation for turbulent viscosity that is integrated to obtain an accurate description of how the effective turbulent viscosity varies with the effective Reynolds number, allowing the model to better handle low-Reynolds number and near wall flows. Another advantage of this model is that it calculates the effective inverse Prandtl number (or Smith in the case of mass transfer) as a function of μ_{mol} / μ_{eff}, which is consistent with experimental evidence and allows heat and mass transfer to be calculated in low Reynolds number regions (FLUENT Inc, 2003a).

3) The SST κ-ω model: it is an improved version of the standard κ-ω model based on model transport equations for the turbulence kinetic energy (κ) and the vorticity fluctuation of turbulence (ω). This model was designed to be applied throughout the boundary layer, provided that the near-wall mesh resolution is sufficient.

4) Laminar model: It assumes that the flow is laminar and it does not take into account turbulent effects.

Two wall treatment approaches were used:

1) Enhanced wall treatment (EWT): It is a near-wall modelling method that combines a two-layer model (the viscosity affected near wall region is completely resolved all the way to the viscous sub-layer) together with enhanced wall functions. The near-wall mesh created was fine enough to resolve down to the laminar sub-layer ($y^+ \approx 1$); the created fine mesh started at 0.1 mm on the wall and it was gradually increased. All four turbulent models were solved using the fine near wall mesh.

2) The standard wall functions (SWF): They are the default wall functions in FLUENT and are based on the proposal of Launder and Spalding (1974). This approach is valid for full developed turbulent flow. The near wall mesh was fixed at 2 mm giving a y^+ between 6 and 25. Only the RNG turbulent model was tested with the two different wall approaches.

Additionally, the radiation heat on the cylinder surface was taken in account within the four turbulent models and the two wall approaches combinations. The RNG κ-ε model was also tested under zero radiation conditions. Table 5.1 summarizes all the different models characteristics.

Model	Turb.	Wall mesh	Wall App.	Radiation
A	RNG κ-ε	Fine	EWT	Yes
B	RNG κ-ε	Coarse	SWF	Yes
C	Laminar	Fine	-	Yes
D	Std. κ-ε	Fine	EWT	Yes
E	SST κ-ω	Fine	-	Yes
F	RNG κ-ε	Fine	EWT	No

Table 5.1 Model characteristics.

5.3.1 Boundary conditions

The inlet conditions for the different experimental trials are in table 5.2.

Exp	V (m/s)	Re.	T (K)	Tu.	RH (%)	P (Pa)
1	0.5	2759	293.26	2%	39.5	102103
2	1.5	8275	293.36	2%	39.8	102230
3	3	16531	293.76	2%	40.4	102070

Table 5.2 Inlet conditions

At the tunnel outlet, zero normal gradients are assumed for all variables except pressure. At the walls of the tunnel, zero velocity, heat flux and water flux are assumed.

At the cylinder surface, both heat flux and water concentration change with the position around the cylinder. The mass transfer coefficient changes point by point depending of the development of the boundary layer. Thus, the heat flux, which depends of the evaporation rate, also changes around the surface. On the other hand, the mass flux at the surface depends on the wall water concentration

gradient, which is calculated with the vapour pressure at the wall temperature. Thus, the concentration gradient is function of the wall temperature.

To establish the thermal and mass boundary conditions, conservation of heat and water equilibrium apply. There are two calculation procedures depending of the wall approach used.

5.3.2 Wall enhanced Treatment

In this case the mesh is fine enough and the mass flux in the interface can be calculated with the water concentration at the surface and in the cell next to the surface with the equation:

$$J_w = \rho D_a \frac{(Y_w - Y_c)}{ds} \tag{5.9}$$

The heat flux at the wall is calculated with the heat of vaporization and the radiation given the equation:

$$q = J_w \Delta H_{vap} + \varepsilon_r \sigma \left(T_a^4 - T_w^4\right) \tag{5.10}$$

The water concentration in the interface is calculated point by point around the cylinder as a function of the surface temperature as follows:

1) The vapour pressure at the interface is calculated as a function of temperature using a vapour pressure– temperature equation.

2) The molar concentration is calculated from the relation between the vapour pressure and the total pressure.

3) The mass water concentration is calculated from the molar water concentration and the molar weights.

5.3.3 Standard Wall Function

It was used in model B (modelling the turbulence with the RNG κ-ε model). In this case the surface mass flux was calculated with the equations proposed by Launder and Spalding (1974) assuming that species transport behaves analogously to heat transfer. The mass flux at the wall J_w is calculated from the equation:

$$Y^* = \frac{(Y_w - Y_c)\rho C_\mu^{1/4} k_p^{1/2}}{J_w} = \begin{cases} Scy^* & (y^* < y_c^*) \\ Sc_t\left[\frac{1}{\kappa}\ln(Ey^*) + P_c\right] & (y^* > y_c^*) \end{cases} \tag{5.11}$$

Details of how calculate P_c, y_c^* and Sc_t are given in section 2.9.8.2. The heat flux at the wall and the water concentration at the surface are calculated in the same way as above.

5.3.4 Radiation effects

The heat of radiation was taken in account in modes A to E. Model F neglects the effect of radiation. Thus, equation 5.10 becomes:

$$q = J_w \Delta H_{vap} \tag{5.12}$$

5.3.5 Boundary condition implementation

The thermal and mass boundary conditions are not constant and change point by point around the cylinder. They also depend on the solution of the other transport equation. Heat flux on the wall depends on the mass flux; surface water concentration depends on surface temperature. Therefore, both boundary conditions were established programming a FLUENT UDF (User Defined Function). The Programmed UDF's are:

1) DEFINE_PROFILE(heat_vaporization, t, j): It calculates the heat flux leaving the cylinder interface caused by evaporation and radiation according to the equation 5.10. The calculation is done over each face element around the cylinder. The radiative term is excluded only on model F. The wall mass flux J_w was calculated with the equation 5.9 or 5.11 depending of the wall approach used.
2) DEFINE_PROFILE(water_interface, t, n): It calculates the surface water concentration as a function of temperature following the procedure explained on section 5.3.2. The calculation is also done over each face element around the cylinder.

Because it is necessary to know the concentration profile to start the calculation of the wall heat flux, the modelling is first done with constant boundary conditions. The wall temperature is fixed as the wet bulb temperature and the mass concentration as the calculated in equilibrium at the wet bulb temperature. After getting convergence, the above UDF's were activated and iterations started again until it converges. It was also necessary to use a relaxation factor of 0.1 in the water_interface UDF to get convergence. The full programmed UDFs are in appendix A2.

5.4 Results and analysis

Table 5.3 shows the root mean square percentage error (%RMS) of the different models. The calculation was done comparing the experimental global mass flux $\bar{J}_w$ and temperatures with the corresponded values obtained by modelling. It is clearly seen that the model A gets the better experimental data fit. Figure 5.3 shows the experimental and calculated mass flux $\bar{J}_w$ using the models A to F. The enhanced wall function method gives better results than the standard wall function as can be seen comparing models A and B. This is because the wall functions were developed for full turbulent flows and not for low Reynolds numbers. The RNG κ-ε model gives slightly better results in the calculation of the mass flux than the standard κ-ε model (model D). That can be explained given the improvement in the RNG model that allows calculations at low

Reynolds numbers. Model A gives the best result matching almost perfectly the experimental data.

Model	A	B	C	D	E	F
%RMS	0.51%	2.08%	7.41%	3.90%	7.15%	4.82%

Table 5.3 Global %RMS of the models.

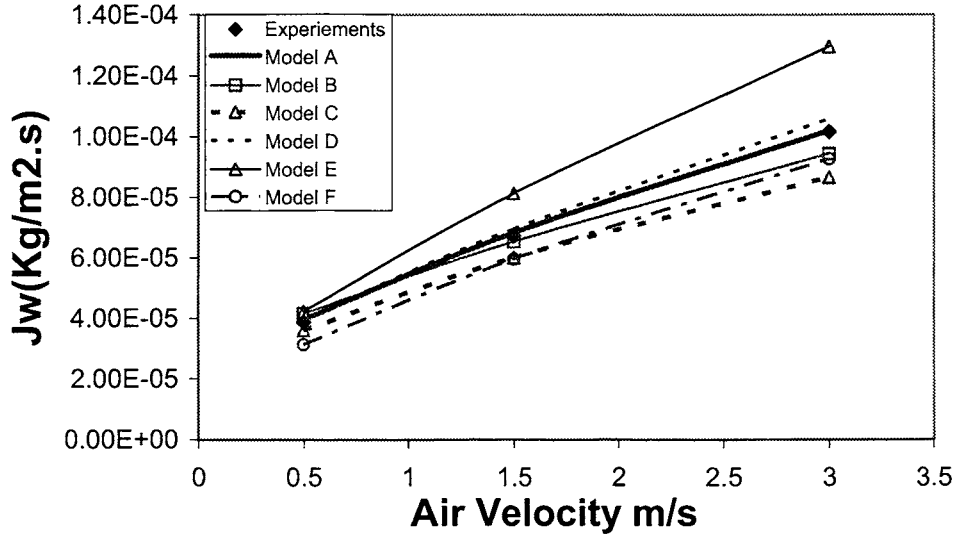


Figure 5.3 Mass flux (J_w) vs. air velocity.

Although the SST κ-ω model (model E) was designed to be applied throughout the boundary layer, it did not fit the data better, as can be seen in Figures 5.3 and tables 5.4 to 5.6. The laminar model (model C) shows the lowest estimate of the

mass flux, $\overline{h}_t$ and $\overline{h}_m$. That was because it did not take in account the effect of turbulence, which is important even though the turbulence intensity was just 2%.

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	286.55	286.16	286.32	
A	286.65	286.32	286.38	0.03%
B	286.68	286.30	286.32	0.03%
C	286.81	286.43	286.45	0.07%
D	286.61	286.23	286.25	0.02%
E	286.57	286.11	286.13	0.04%
F	285.37	285.58	285.88	1.24%

Table 5.4 Temperature (K) vs. air velocity.

Ignoring the effect of radiation (model F) causes underestimation of the mass flux (Figure 5.3) and temperature (Table 5.4). That can be explained with equation 5.10. Radiative heat is in opposite direction to evaporative heat flux and reduces the absolute value of the total wall heat flux. Thus, if radiation is not taken into account, the wall heat flux is higher and the wall temperature is lower. Lower wall temperature means lower water surface concentration (given the dependence of temperature with vapour pressure), causing a lower water concentration gradient (or mass flux) in the wall.

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	0.0104	0.0198	0.0307	
A	0.0107	0.0200	0.0312	1.48%
B	0.0111	0.0192	0.0292	4.51%
C	0.0095	0.0176	0.0261	10.38%
D	0.0108	0.0207	0.0332	5.00%
E	0.0115	0.0247	0.0416	22.38%
F	0.0106	0.0200	0.0313	1.51%

Table 5.5 $\overline{h}_m$ (Kg /m^{2}s) vs. air velocity.

The local prediction of temperature and h_t around the cylinder obtained with model A is good as it can be seen in Figures 5.4 and 5.5 (the angle is taken from the stagnant point).

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	9.22	17.94	28.53	
A	9.71	18.80	28.92	3.67%
B	10.52	17.74	26.15	8.20%
C	8.73	16.52	24.08	9.46%
D	9.68	18.93	29.60	4.18%
E	10.52	22.51	36.69	20.43%
F	9.76	18.81	28.90	3.87%

Table 5.6 $\overline{h}_t$ (W/m^{2}K) vs. air velocity.

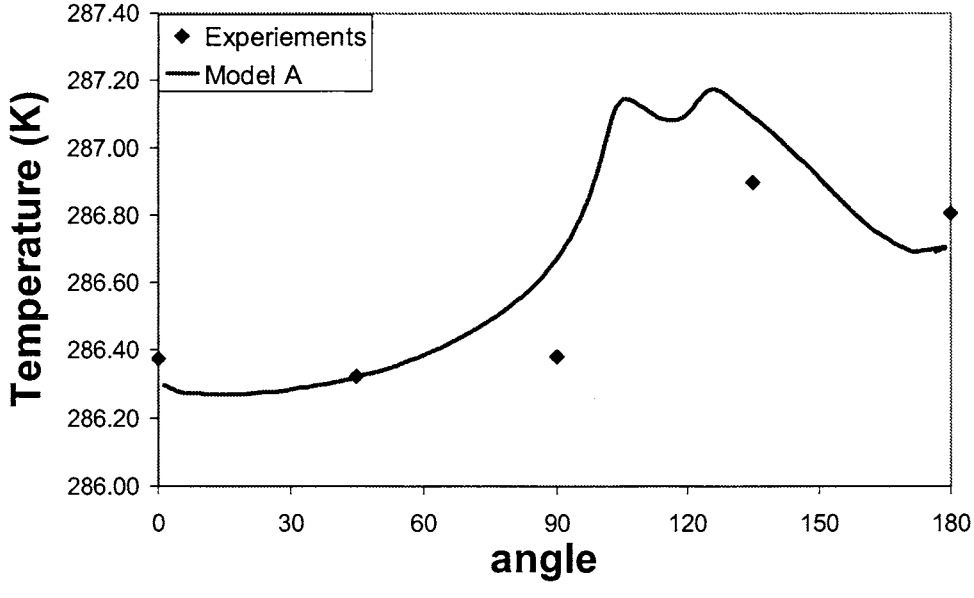


Figure 5.4 Temperature vs. angle at V = 0.5 m/s (model A).

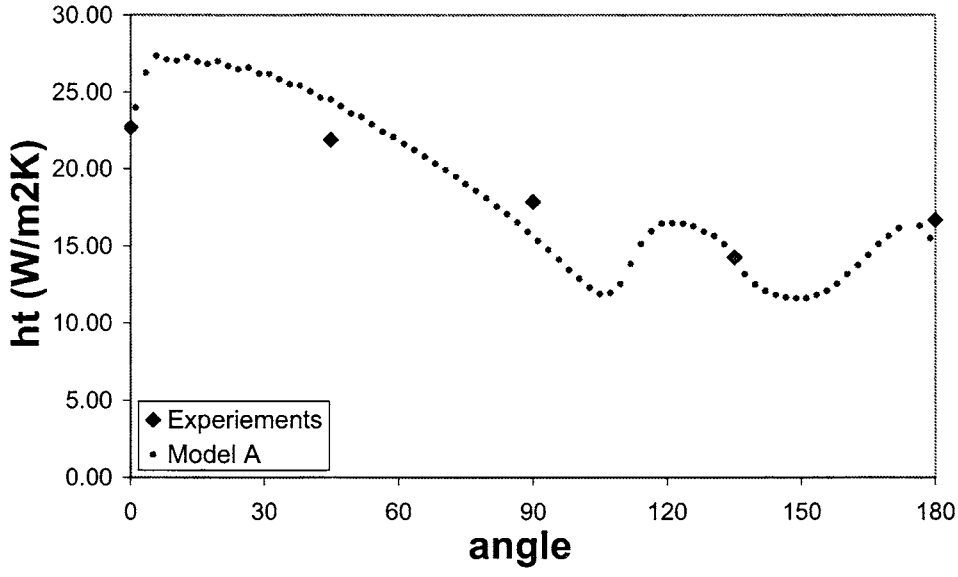


Figure 5.5 Local h_t vs. angle at V = 1.5 m/s (model A).

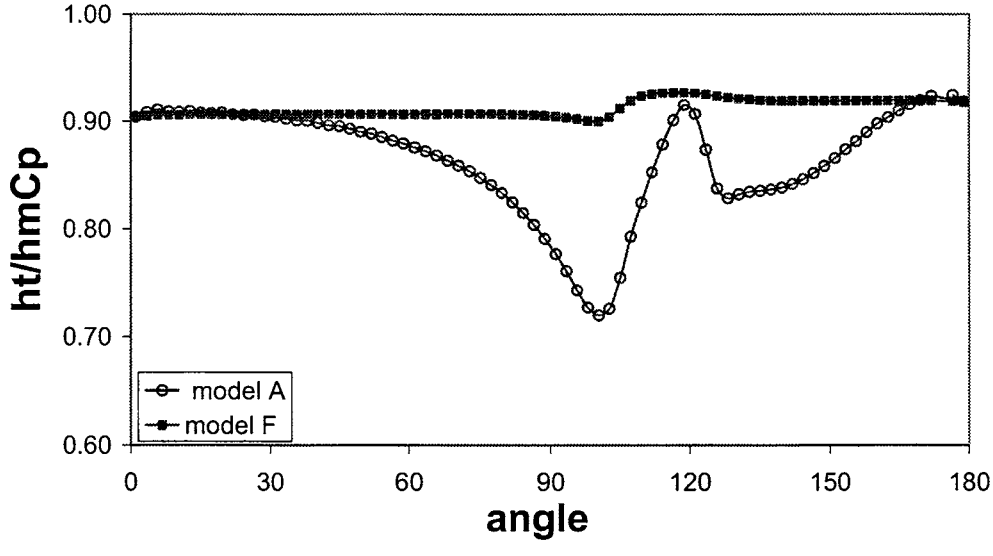


Figure 5.6 Relation $h_t/h_m c_P$ vs. angle at V = 0.5 m/s.

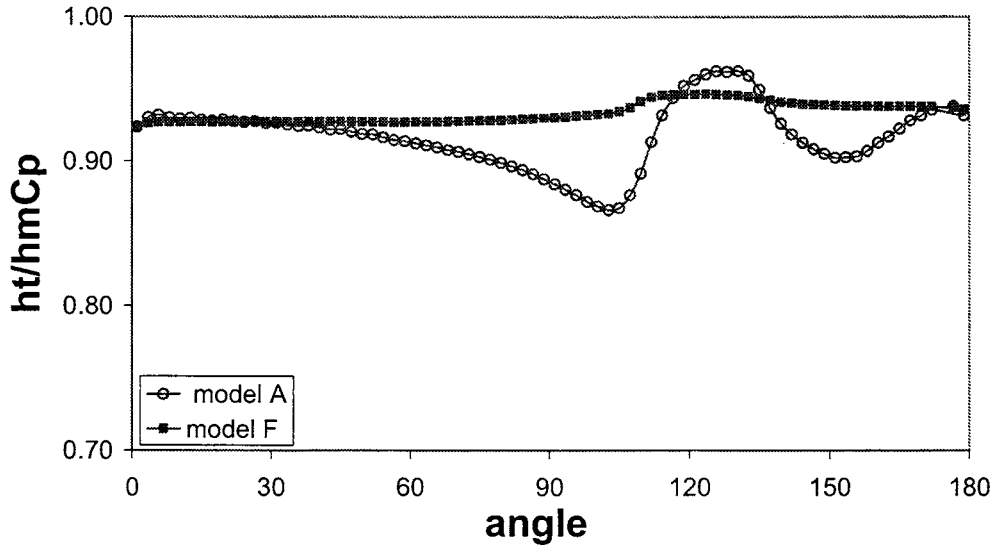


Figure 5.7 Relation $h_t/h_m c_P$ vs. angle at V = 1.5 m/s.

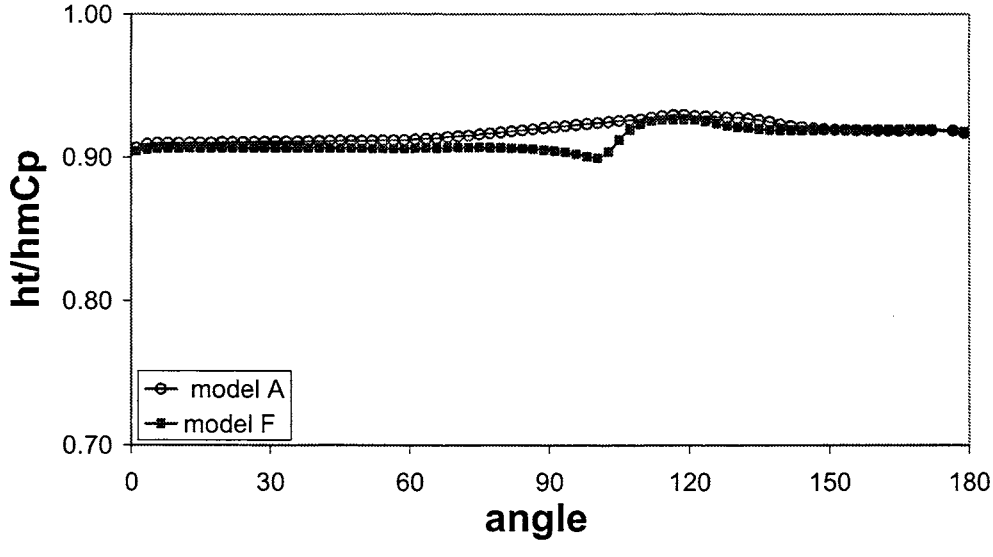


Figure 5.8 Relation $h_t/h_m c_P$ vs. angle at V = 3 m/s.

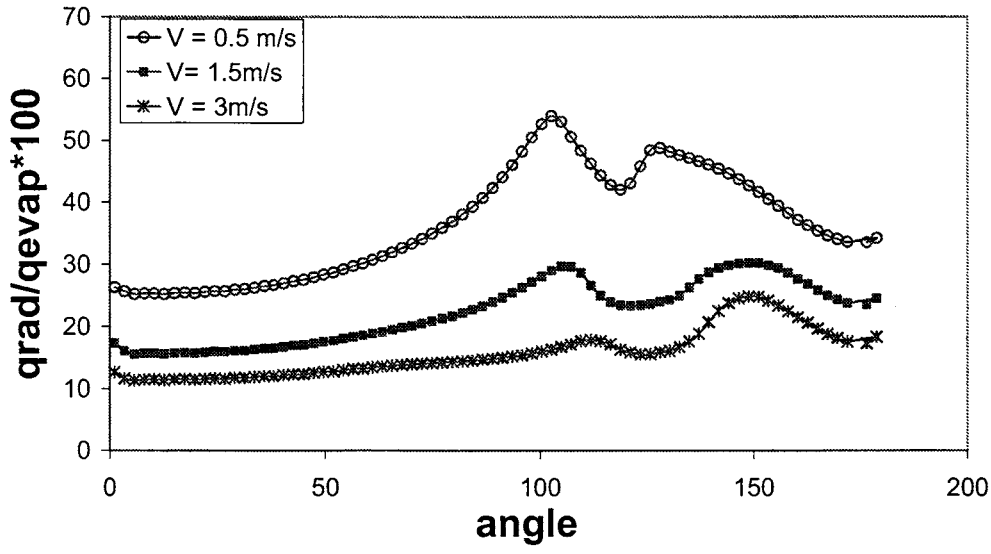


Figure 5.9 Relation q_{rad}/q_{evap} vs. angle.

The relationship between the global heat and mass transfer coefficients ($\overline{h}_t/\overline{h}_m c_P$) is almost constant and close to the value of $Le^{2/3} \approx 0.892$ agreeing with the Chilton-Colburn analogy (Equation 5.5). However, there were differences found in the local relationship ($h_t/h_m c_P$) at low velocities when the radiative heat was taken in account. Figures 5.6, 5.7 and 5.8 show the local relationship between the heat and mass transfer coefficient as a function of the angle using models A and F. Figure 5.6 (made at V = 0.5 m/s) shows that using model A the relationship strongly deviates from a constant value reaching a minimum (19% deviation) at the separation point of the boundary layer. If the radiative heat is not taken in account (model F) the relation is constant and very close to $Le^{2/3}$ as in Equation 5.5. The deviation decreases when the air velocity increases (Figures 5.7 and 5.8), almost getting a constant value at V = 3 m/s. Figure 5.9 shows that the

radiative heat becomes less important compared with the evaporative heat at higher air velocities. The deviation of the Chilton-Colburn equation at low velocities, or low Reynolds numbers, seems to be influenced by the heat of radiation that becomes important at low air velocities. That effect is stronger at the detachment of the boundary layer where the heat transfer coefficient is decreasing while the mass transfer coefficient is increasing, causing the ratio heat to mass transfer coefficients to decrease. At this point, the convection effects decrease and the radiation effect becomes more important causing the surface temperature to rise (Figure 5.4). The normal temperature gradient decreases making the wall heat flux reduce. Therefore, the local heat transfer coefficient decreases. Cess (1962) analytically shows that the local Nusselt number (non-dimensional heat transfer coefficient) in a laminar boundary layer of a fluid along a flat plate decreases by effect of radiation. The effect of radiation on the mass transfer coefficient is opposite to heat transfer coefficient. It makes the mass transfer coefficient to increase. This is because the wall temperature increases, as it was established above, making the vapour pressure and the water composition on the wall to increase. This increment on the wall water composition makes the mass gradient at the wall to rise and therefore, the mass transfer coefficient increases. This work shows that the Chilton-Colburn analogy loses accuracy to relate the local heat and mass transfer coefficients at low air velocities. The accuracy of this analogy (Equation 5.5) to correlate heat and mass transfer coefficients on evaporation and drying processes has been questioned before (Chen *et al.,* 2002).

5.5 Conclusion

- The experiments of water evaporation from a circular cylinder were modelled with different turbulence models and two wall treatment approaches. It was found that the RNG κ-ε model using the enhanced wall treatment and taking in account the effect of radiation fits better the experimental data. This method is recommended on further models.

- CFD modelling shows that at low Reynolds numbers, when the radiative heat is taken in account, the relation $h_t / h_m c_P$ is not constant around the cylinder, reaching a minimum value at the detachment of the boundary layer.

6. CFD MODELING OF HEAT AND MOISTURE TRANSFER ON A TWO- DIMENSIONAL MODEL OF A BEEF LEG

A numerical simulation of simultaneous heat and mass transfer in an ellipse model of a beef leg was carried out using the CFD software FLUENT 6.1.18 Special techniques were used to solve the heat and mass transport equation for both the air and meat phases, due to some limitations in the software. The meat was treated as a sub-region of the fluid with zero momentum transport, and the mass transport in the air and the meat respectively were modelled by different field variables, which are however linked at the interface. In the air, turbulent flow was modelled with the RNG $k-\varepsilon$ model and the boundary layer was fully solved using the FLUENT 6.1.18 enhanced wall treatment. The model predicted local variations in the heat and mass transfer coefficients and temperature and water activity around the ellipse's surface.

6.1 Introduction

During the chilling of beef carcasses after slaughter, cooling and evaporation proceed together and interact with each other to influence surface water activity, microbial growth, weight loss, meat temperature and meat tenderness, all of which are important economically. As the meat cools, heat is conducted through the meat and carried away by the air. The meat surface is warmer and more

humid than the air, resulting in surface evaporation. Water from inside diffuses towards the surface to make up for evaporation. The balance between evaporation and diffusion governs the water activity near the surface, which together with temperature determines the potential for microbial growth.

Both processes, heat and mass transfer, are affected by the flow characteristics and the development of the momentum, heat and mass boundary layers. They are function of the air properties, geometry of the product, chillier room, and the flow characteristics (temperature, humidity, velocity and turbulence). Therefore, local variations in the heat and mass transfer coefficients are expected along the surface (Verboven *et al.*, 1997) producing local differences in temperature and water activity.

Traditional computer models (Davey and Pham, 1997, 2000; Mallikarjunan and Mittal, 1994) have focused on solving the conduction equation in the meat, but the average heat ($\overline{h}_t$) and mass transfer ($\overline{h}_m$) coefficients are calculated with empirical equations. Nguyen and Pham (1999) used CFD to simulate the heat transfer process in a beef carcass chilling, taking in account the conduction inside the meat and the convection in the air phase, but they did not take in account the evaporation, radiation and mass transfer. Hu and Sun (2000) modelled the heat and mass transfer on the air side during the cooling of cylindrical shaped cooked meat. They used the CFD software CFX to calculate the average $\overline{h}_t$ but they did not predict local (h_t) variations. Hu and Sun (2001) modelled the heat and

moisture transfer. The former was modelled in both the solid and air phases using CFD, but the mass transfer was not treated rigorously via CFD modelling. For example, the Lewis relationship was used to calculate mass transfer from heat transfer, and the surface water activity of the meat was assumed to be equal to the relative humidity of the air (on a scale of 0 to 1), which was not necessarily true.

The modelling of mass transfer during cooling is still very approximate. Most models calculate the $\overline{h}_m$ using the Lewis relationship and assume a constant value for water activity (Davey and Pham, 1997, 2000; Hu and Sun, 2000). A source of difficulty is that heat and mass transfer happen on vastly different scales due to the big difference in heat and mass diffusivity. While the whole product is cooled, only the surface layer (a few mm) loses water. This makes it difficult to model the two processes accurately using a homogeneous grid configuration. To solve this problem, Pham and Karuri (1999) used a separate discretization grid for temperature and moisture calculations. However, they solved the equations only for the solid phase and not for the air phase, relying instead on an empirical $\overline{h}_t$ and the Lewis relationship to calculate the $\overline{h}_m$.

No previous work has attempted to solve simultaneously the equations for mass diffusion, fluid flow and heat transfer in both phases together. This is the objective of the present chapter.

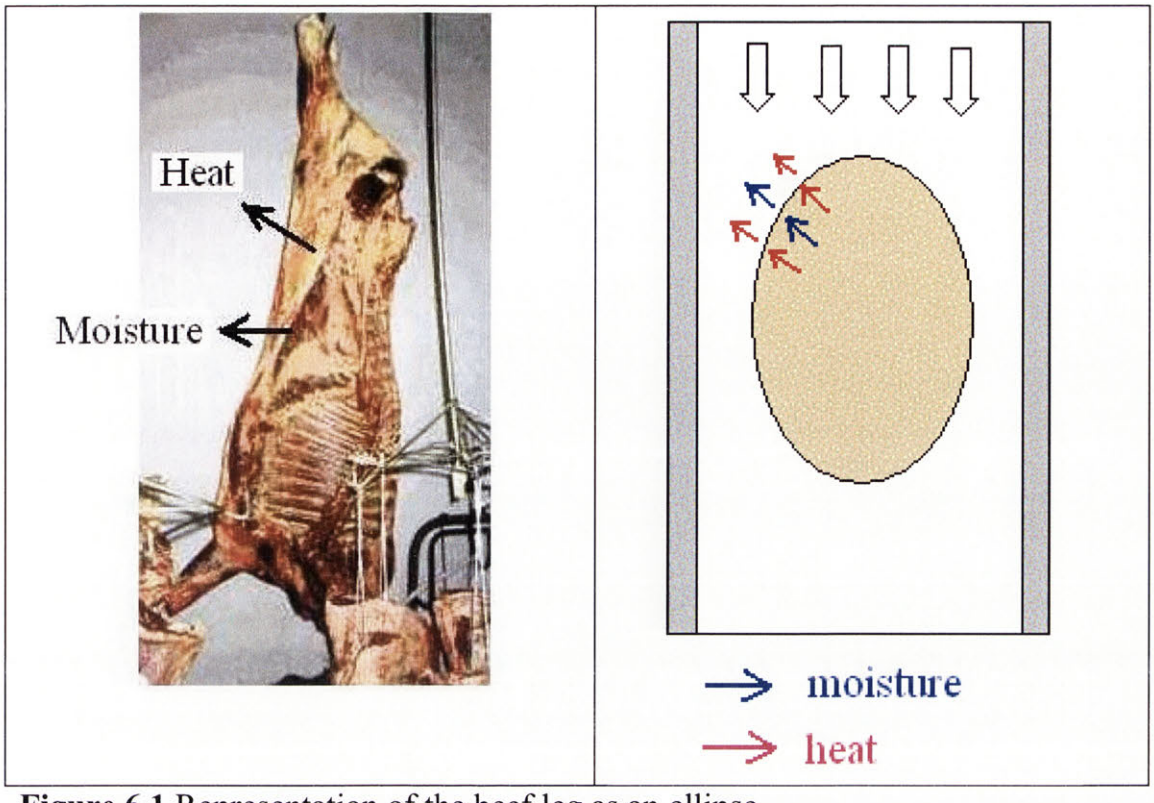


Figure 6.1 Representation of the beef leg as an ellipse.

6.1.1 Problem statement

A beef leg undergoing chilling is modelled as an ellipse (Davey and Pham, 2000), with minor and major diameters of 0.22m and 0.29m respectively, placed inside a wind tunnel 1.5m wide by 2.3m long (figure 6.1). Air enters the tunnel at 277.95°K, 98% relative humidity, atmospheric pressure, 0.54m/s normal to the inlet plane, with a turbulence intensity of 10%. The product is initially at 315.15°K with a moisture content of 75% wet basis. The properties of air were assumed constant except the density that was expressed as function of temperature and Pressure. For the meat, we assume a density of 1111 kgm^{-3}, specific heat of 3407 $Jkg^{-1}\ {}^{o}K^{-1}$, thermal conductivity of 0.397 $Wm^{-1}K^{-1}$ and the water diffusivity is given by the equation of Herbert *et al.* (1978):

$$D_m = 1.10 \times 10^{-6} e^{\left(-\frac{2300}{T}\right)} \tag{6.1}$$

Herbert's equation was used because this work was developed before completing chapter 4. At the meat surface, the water activity (relative humidity on a scale of 0 to 1) is given by a Lewicki-type isotherm equation developed in chapter 3 (Trujillo *et al,* 2003). The moisture content of the air next to the surface is determined from the surface temperature and water activity.

6.2 Mathematical model

6.2.1 Transport equations in the air

In the air, a set of six transport equations are solved:

6.2.1.1 *The continuity equation*

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot \left(\rho \vec{\bar{v}}\right) = 0 \tag{6.2}$$

where ρ is density, t is time and $\vec{\bar{v}}$ is the mean average velocity vector.

6.2.1.2 *The momentum equation*

$$\frac{\partial(\rho \vec{v})}{\partial t} + \nabla \cdot \left(\rho \vec{\bar{v}} \vec{\bar{v}}\right) = -\nabla p + \nabla \cdot \left(\bar{\tau}_{eff}\right) \tag{6.3}$$

where p is pressure and $\bar{\tau}_{eff}$ is the effective stress tensor given by:

$$\bar{\tau}_{eff} = \mu_{eff}\left[\left(\nabla \vec{\bar{v}} + \nabla \vec{\bar{v}}^T\right) - \frac{2}{3}\nabla \cdot \vec{\bar{v}} I\right] \tag{6.4}$$

μ_{eff} is the effective viscosity.

6.2.1.3 *The energy equation*

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot \left(\vec{\bar{v}}(\rho E + p)\right) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \vec{J}_j + \left(\bar{\tau}_{eff} \cdot \vec{\bar{v}}\right)\right) \tag{6.5}$$

where k_{eff} is the effective thermal conductivity, h_j is the enthalpy of the specie j. $\vec{J}_j$ is the diffusion flux of species *j*. E is the specific energy of the fluid defined as:

$$E = h - \frac{p}{\rho} + \frac{v^2}{2} \tag{6.6}$$

where h is the enthalpy of the mixture and $v^2/2$ represents the kinetic energy.

6.2.1.4 *The water transport equation*

$$\frac{\partial(\rho Y_i)}{\partial t} + \nabla \cdot \left(\rho \vec{\bar{v}} Y_i\right) = -\nabla \cdot \vec{J}_i \tag{6.7}$$

where:

$$\vec{J}_i = -\rho D_{i,m,eff} \nabla Y_i \tag{6.8}$$

i represents the water, $D_{i,m,eff}$ is the effective water diffusivity and Y_i is the water weight fraction composition.

6.2.1.5 *The turbulent kinetic energy equation*

$$\frac{\partial}{\partial t}(\rho k) + \sum_j \frac{\partial}{\partial x_j}\left(\rho k \bar{v}_j\right) = \sum_j \frac{\partial}{\partial x_j}\left[\alpha_k \mu_{eff} \frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_M + S_k \tag{6.9}$$

6.2.1.6 *The turbulent dissipation rate equation*

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \sum_j \frac{\partial}{\partial x_j}\left(\rho\varepsilon \bar{v}_j\right) = \sum_j \frac{\partial}{\partial x_j}\left[\alpha_\varepsilon \mu_{eff} \frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}\left(G_k + C_{3\varepsilon} G_b\right) - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} - R_\varepsilon + S_\varepsilon \tag{6.10}$$

The RNG *k- ε* model (equations 6.9 and 6-10) was used to calculate μ_{eff}, k_{eff} and $D_{i,m,eff}$ as it is described in sections 2.9.5 and 2.9.6.The differential equation:

$$d\left(\frac{\rho^2 k}{\sqrt{\varepsilon\mu}}\right) = 1.72\frac{\hat{\upsilon}}{\sqrt{\hat{\upsilon}^3 - 1 + C_\upsilon}}d\hat{\upsilon} \tag{6.11}$$

where:

$\hat{\upsilon} = \mu_{eff}/\mu$

$C_\upsilon \approx 100$

is integrated to obtain the effective viscosity turbulent viscosity μ_{eff}. The effective thermal conductivity is:

$$k_{eff} = \alpha C_p \mu_{eff} \tag{6.12}$$

where α is calculated from

$$\left|\frac{\alpha - 1.3923}{\alpha_0 - 1.3929}\right|^{0.6321}\left|\frac{\alpha - 2.3923}{\alpha_0 - 2.3929}\right|^{0.3679} = \frac{\mu}{\mu_{eff}} \tag{6.13}$$

where $\alpha_0 = 1/\Pr = k/\mu c_p$

The effective diffusivity $D_{i,m,eff}$ is calculated in a manner that is analogous to the method used for the heat transport. The value of α_0 in Equation 6.13 is $\alpha_0 = 1/Sc$, where Sc is the molecular Schmidt number.

Each of the six transport equations in the air phase (equations 6.2, 6.3, 6.5, 6.7, 6.9 and 6.10) has the general form :

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho\phi\bar{v}) = \nabla\cdot(\Gamma\nabla\phi) + S \tag{6.14}$$

where the terms on the left represent accumulation rate and convection, while those on the right represent diffusion and creation/destruction of the field

variableϕ. ϕ can be 1 (in the continuity equation), velocity (in the momentum equation), turbulent kinetic energy, turbulent dissipation rate (in the k and ε transport equations), temperature (in the thermal energy equation), or moisture content (in the moisture transport equation).

6.2.2 Transport equations in the meat

In the meat, only the transport equations for thermal energy and moisture need to be solved. The energy transport equation is:

$$\frac{\partial(\rho_m c_{p,m} T)}{\partial t} = \nabla(k_m \nabla T_m) \tag{6.15}$$

where $\rho_m, c_{p,m}$, k_m and T_m are the density, heat capacity, thermal conductivity and temperature of the meat respectively. The moisture transport equation is:

$$\frac{\partial(\rho_m Y_m)}{\partial t} = \nabla(\rho_m D_m \nabla Y_m) \tag{6.16}$$

where Y_m is the water weight composition of the meat and D_m the water diffusivity.

6.2.3 Boundary conditions

Conditions at the tunnel's inlet are as given under "Problem specifications". At the tunnel outlet, zero normal gradients are assumed for all variables: v, T, Y, ε, κ. At the walls of the tunnel, zero velocity, zero heat flux and zero water flux are assumed.

At the meat surface, thermal, species equilibrium and conservation of heat and mass apply. With regards to conservation of heat and mass, special techniques have to be applied to balance the heat and mass fluxes coming out of the solid with those entering the air phase. After the transport equations in the air have been solved, the water flux that enters the air was calculated cell by cell using the concentration profile in the air control volume next to the solid surface, using the equation:

$$J_w = \rho D_a \frac{(Y_f - Y_c)}{ds} \tag{6.17}$$

The heat flux entering the air was calculated as the sum of convection, evaporative and radiative components:

$$q = q_{conv} + q_{evap} + q_{rad} \tag{6.18}$$

$$q_{evap} = J_w \Delta H_{vap} \tag{6.19}$$

$$q_{rad} = \sigma \varepsilon_r \left(T_{m,s}^4 - T_a^4\right) \tag{6.20}$$

and q_{conv} was calculated using the temperature profile of the air control volume next to the surface:

$$q_{conv} = k_a \frac{(T_f - T_c)}{ds} \tag{6.21}$$

The following equilibrium condition between the air in contact with meat and the meat surface must also hold:

Thermal equilibrium:

$$T_{a,s} = T_{m,s} \tag{6.22}$$

Chemical potential equilibrium:

$$a_{w,m,s} = a_{w,a,s} \tag{6.23}$$

Where $a_{w,m}$ is a function of the water content in the meat surface (a function of meat moisture content as given in Trujillo *et al.*, 2003).

6.2.4 Initial conditions

At zero time, the meat temperature and water composition are constant as per "Problem Statement". It was also assumed that the airflow was fully developed. Thus, a steady state solution for the air phase was done as a preliminary step, with the air temperature near the meat surface kept at 315.15°K and the humidity corresponding to equilibrium with the meat.

6.2.5 Boundary layer treatment

Because the mass and heat fluxes between meat and air are calculated from the temperature and concentration gradient next to that surface, these profiles must be accurately known. Therefore we used FLUENT 6.1.18's *enhanced wall treatment* (section 2.9.8.3, Fluent Inc., 2003a) which is a near-wall modelling method that combines a two-layer model (the viscosity affected near wall region is completely resolved all the way to the viscous sub-layer) with enhanced wall functions. The near-wall mesh must be fine enough to resolve the transport equations down to the laminar sub-layer ($y^+ \approx 1$). Figure 6.2 shows the details of the mesh.

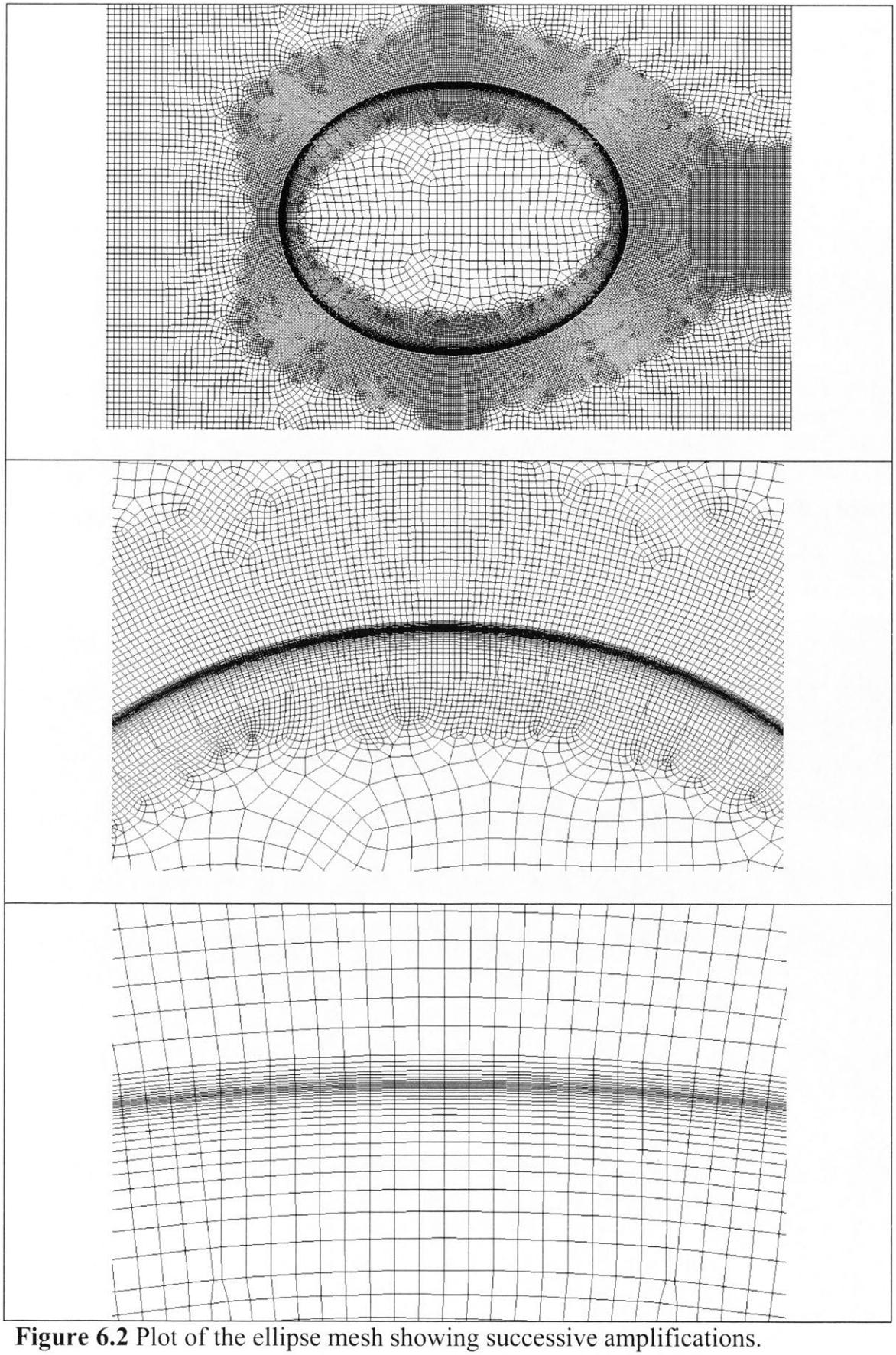


Figure 6.2 Plot of the ellipse mesh showing successive amplifications.

6.2.5.1 *Standard wall functions approach*

The water and convective heat flux at the meat surface, equations 6.17 and 6.21, can also be calculated from the standard wall functions with a gross mesh on the boundary. The water flux at the boundary J_w can be calculated from the equation:

$$Y^* = \frac{(Y_f - Y_c)\rho C_\mu^{1/4} k_p^{1/2}}{J_w} = \begin{cases} Scy^* & (y^* < y_c^*) \\ Sc_t\left[\frac{1}{\kappa}\ln(Ey^*) + P_c\right] & (y^* > y_c^*) \end{cases} \tag{6.24}$$

and the convective heat from the equation:

$$T^* = \frac{(T_f - T_c)\rho c_p C_\mu^{1/4} k_p^{1/2}}{q_{conv}} = \begin{cases} \Pr y^* & (y^* < y_T^*) \\ \Pr_t\left[\frac{1}{\kappa}\ln(Ey^*) + P\right] & (y^* > y_T^*) \end{cases} \tag{6.25}$$

More details about the standard wall functions are given in section 2.9.8.2. This method is not recommended because it is valid for fully turbulent flows. However, this option was programmed and can be activated in the UDF.

6.3 Details of numerical solution

The equations were solved using FLUENT 6.1.18 FLUENT's *segregated method*, where the governing equations are solved sequentially. In that method, each discretized transport equation is linearized implicitly with respect to the equation's dependent variable. Because the equations are non-linear and coupled, iterations must be performed before a converged solution is obtained. A point implicit (Gauss-Seidel) linear equation solver is used by FLUENT in conjunction

with an algebraic multigrid (AMG) method. The pressure – velocity coupling method used was PISO which is recommended for unsteady problems. Time steps of 1 second were used at the beginning, gradually increasing up to 10 minutes at the end of the simulation. Up to 40 iterations were done for each time step.

In the meat, the energy and mass transfer processes could not be solved using FLUENT 6.1.18's existing energy and mass transfer equations because:

- FLUENT cannot solve the mass transfer equation in a solid (the meat). Thus the meat had to be defined as a fluid phase (this is purely a formal definition for FLUENT – since the equations of continuity, momentum, turbulent kinetic energy and turbulent dissipation rate are dropped, the physical behaviour is in effect that of a solid).
- Once the meat was defined as a fluid phase to satisfy the previous requirement, FLUENT 6.1.18 then requires that the physical properties (c_p, μ, D, k, ρ) be the same in all parts of the solution domain (i.e. both air and meat phases are modelled as the same substance). Since these properties were in fact different in the two phases, new transport equations involving new field variables must be defined and solved in the meat.
- FLUENT will only allow the boundary conditions of the mass transport equations to be zero flux or a fixed concentration, while we want to calculate the flux at the meat-air boundary according to equation 6.17.

To overcome the above problems, FLUENT allows the user to define new field variables called UDS (User Defined Scalars). The moisture and temperature inside the meat are considered as new field variables or UDS with their own associated transport properties (diffusivity and density). Defining a UDS simply involves specifying whether there are convective, diffusive and transient terms in the transport equations, and specifying expressions for the transport properties mentioned above.

The FLUENT default menu allows only fixed boundary conditions. Because the boundary conditions at the air-meat interface change with time and are dependent on the values of the field variables, they were modelled using UDFs (User Defined Functions), which are functions programmed by the user in C++ that can be automatically linked with the FLUENT Solver (Fluent Inc., 2003b). With a UDF, we can take the present values of the local field variables (T, Y, etc.) and use them to calculate the boundary conditions at that particular instant. The UDFs are incorporated in the set of equations solved by the segregated solver, and the values predicted by the UDF were updated in each iteration. The UDFs programmed boundary conditions were:

1. DEFINE_PROFILE(mass_flux_meat, tm, j): Calculate the water flux leaving the meat in the interface caused by the mass convection given the development of the mass boundary layer. The calculation is made with the

equation (6.17). It is used as a boundary condition of the UDS-2 that defines the mass transport equation into the meat.

2. DEFINE_PROFILE (heat_flux_meat, tm, j): Calculate the heat flux leaving the meat in the interface caused by convection, radiation and evaporation according to the equation (6.18). It is used as a boundary condition of the UDS-1 that defines the energy equation into the meat.
3. DEFINE_PROFILE(temperature_air_interf, t, n) : makes the temperature of the air on the air meat interface equal to the temperature of the meat surface (Equation 6.22). It is used as a boundary condition of the air phase energy equation.
4. DEFINE_PROFILE(water_air_interf, t, n): Calculate the air water mass concentration on the interface with the following procedure:
   - Read the meat surface temperature (Equation 6.22).
   - Read the water mass concentration in the meat interface and calculate the water content.
   - Calculate the water activity with the Lewicki equation (Trujillo *et al.* 2003) using the Newton-Raphson method.
   - With the water activity and the meat surface temperature, calculate the air water mass concentration using a vapour pressure equation.

   This UDF is used as a boundary condition of the mass equation of the air phase.

On each iteration the solver segregated method does the following:

- Solve the linearized discretized momentum equation in the air phase.
- Solve mass conservation in the air phase and update velocities.
- Solve energy equation in the air phase– this involves calls to UDF 3.
- Solve mass equation in the air phase – this involves calls to UDF 4.
- Solve turbulent kinetic energy in the air phase.
- Solve Eddy dissipation in the air phase.
- Solve energy equation in the meat phase using UDS1. This involves calls to UDF 2.
- Solve mass equation in the meat phase using UDS2. This involves calls to UDF 1.
- Update all properties
- Check for convergence.

That procedure is done each time step until convergence is obtained or 40 iterations have been done. Appendix A3 shows the full collection of programmed UDFs.

6.4 Results and Analysis

Figure 6.3 shows the average surface and centre temperature as a function of time. The surface temperature is in good agreement with the experimental data suggesting that the model is accurate. The measured centre temperature drops faster than the predicted values but this may be caused by the floor slaughter

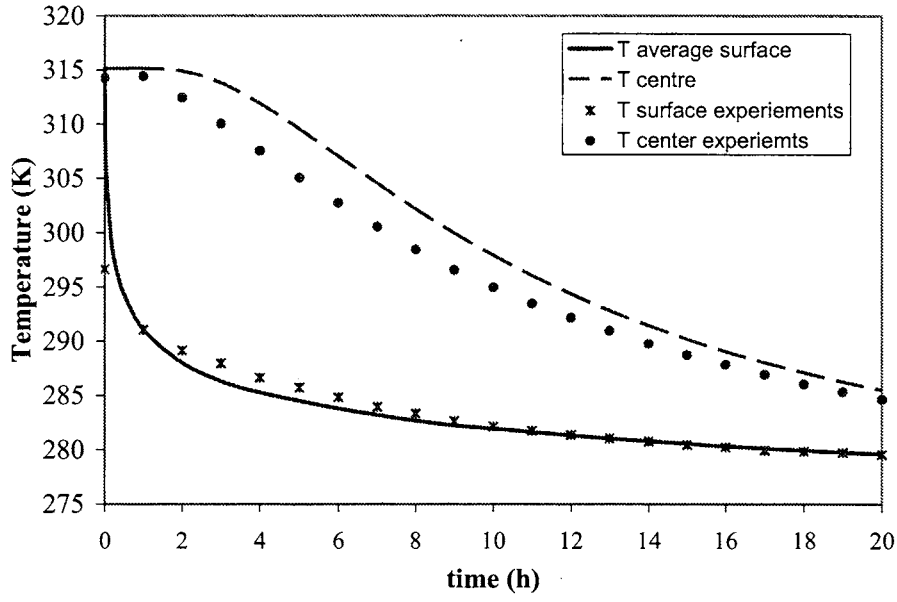


Figure 6.3 Average surface and centre temperature of the meat ellipse.

period that was not taking into account in this model. Figure 6.4 shows the changes of the surface temperature with the position over the sphere (given by the angle) at 1 and 5 hours. Temperature could vary by up to 5C around the ellipse.

Figure 6.5 shows the change of the average surface water activity with time showing that it drops faster during the first 4 hours and then increases, caused by the internal water diffusion that rewets again the meat surface. This is in agreement with the experimental observation of Herbert *et al.* (1978). Figure 6.6 shows the local variations of the water activity with position and Figure 6.7 shows the water concentration profile deep inside the meat at 0° (the impact or

stagnation point) It shows that the mass transfer inside the meat is noticeable

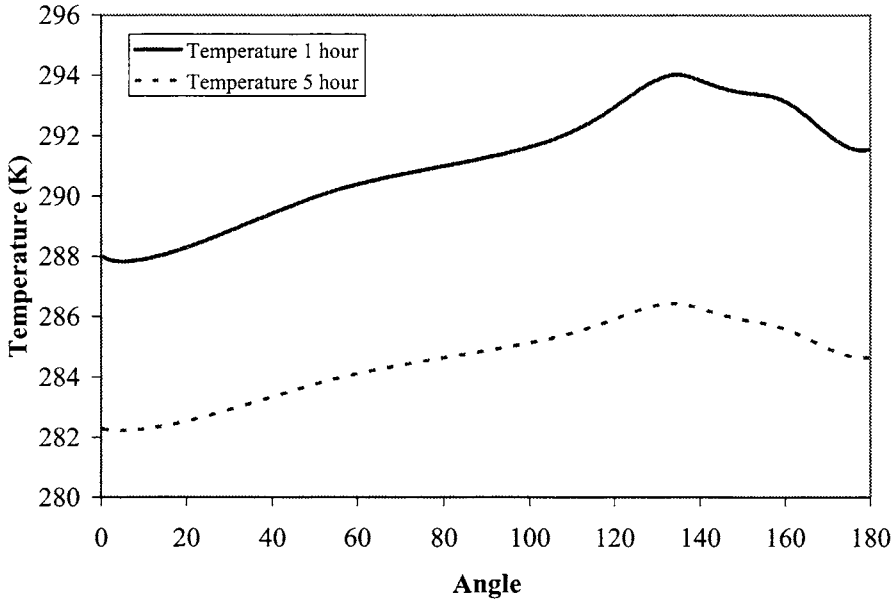


Figure 6.4 Surface temperature profile as a function of the angle at 1 and 5 hours.

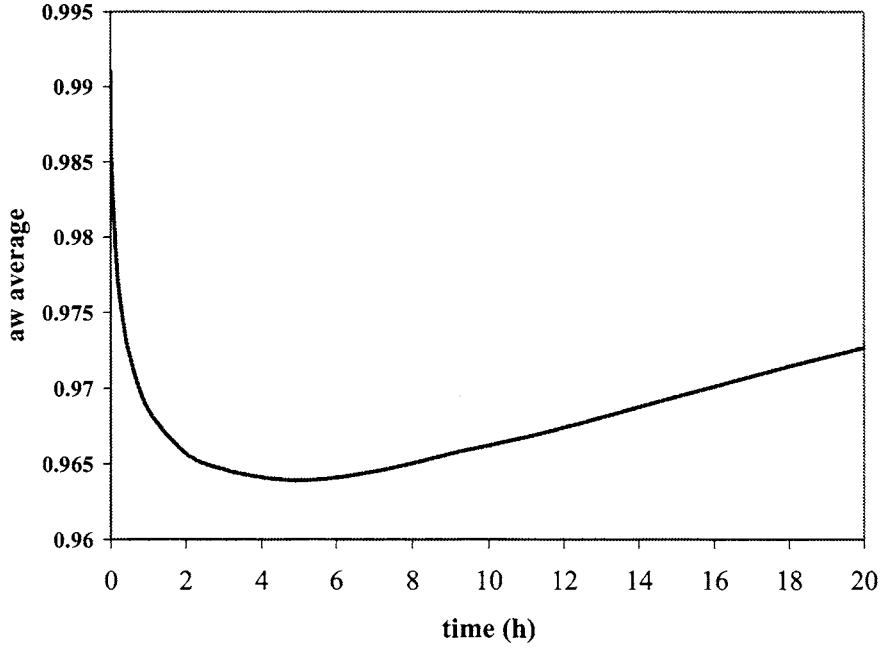


Figure 6.5 Average surface water activity vs. time.

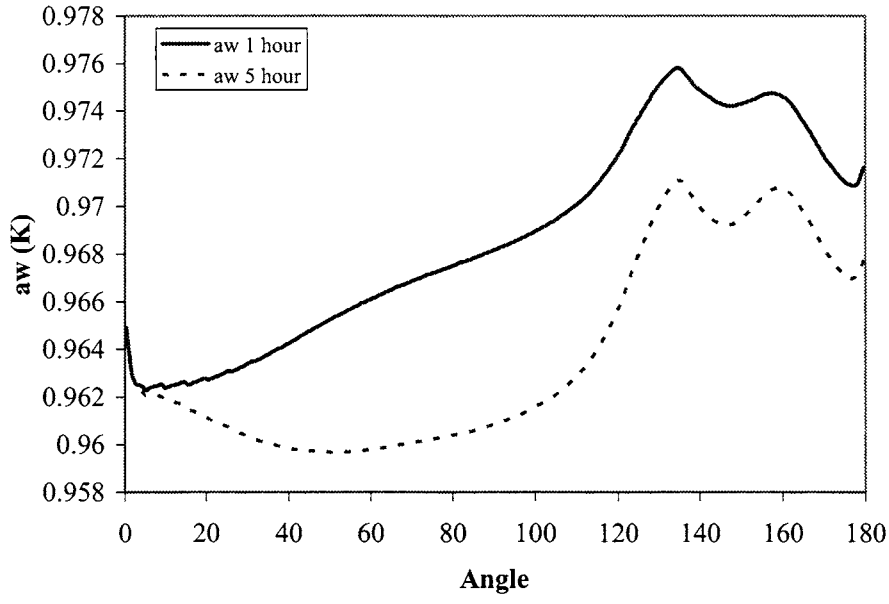


Figure 6.6 Surface water activity as a function of the angle at 1 and 5 hours.

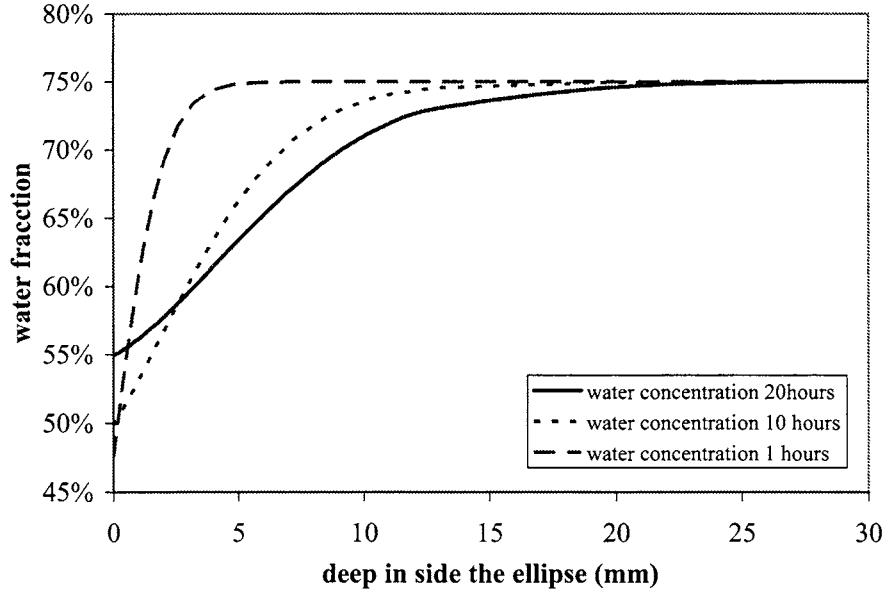


Figure 6.7 Water concentration profile deep inside the meat at 0° (stagnant point) at 1, 10 and 20 hours.

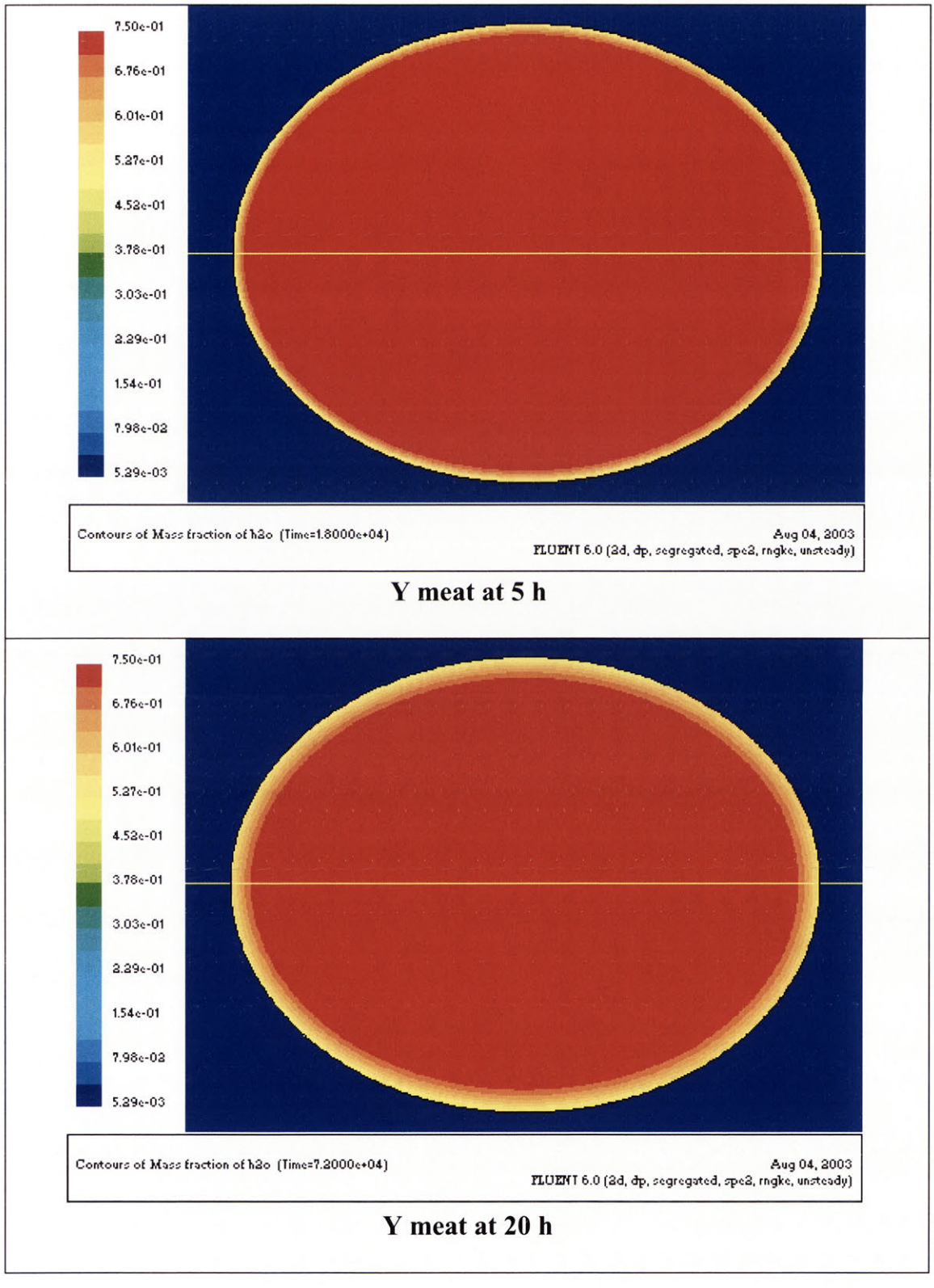
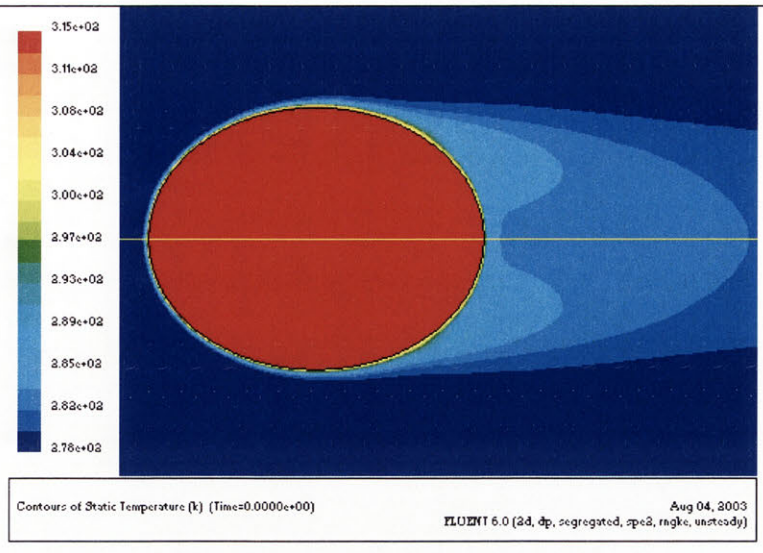
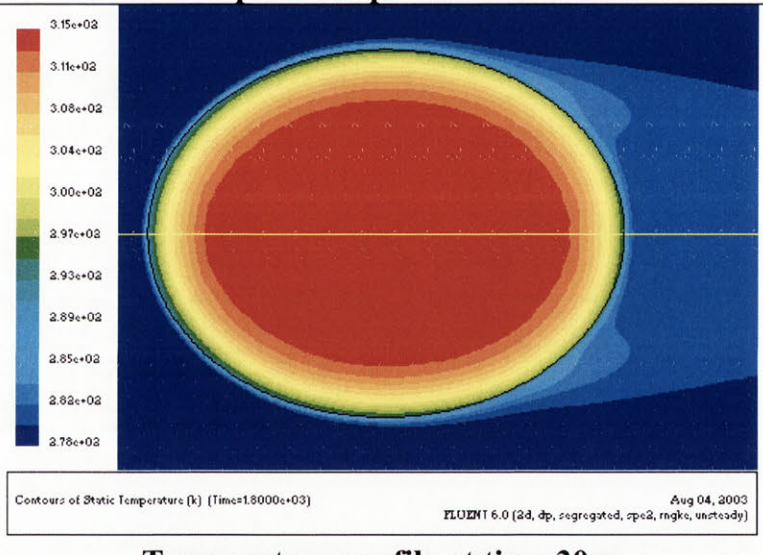


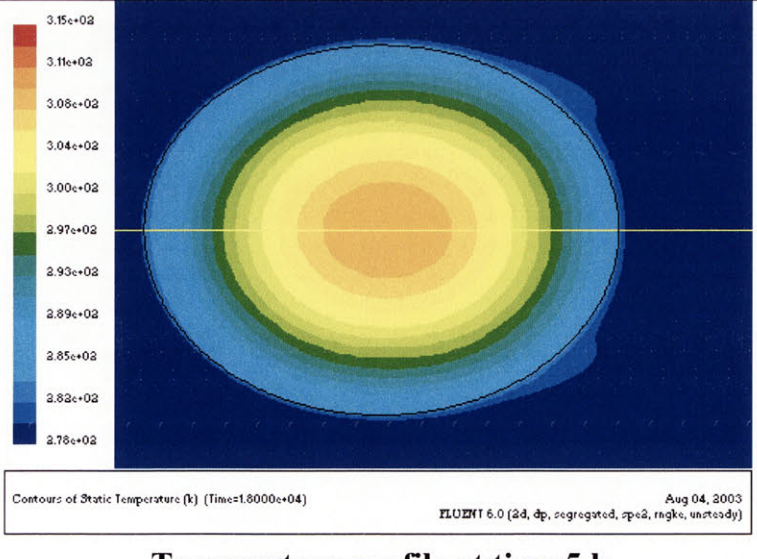
Figure 6.8 Water composition profile at 5h and 20h.



Temperature profile at time 0



Temperature profile at time 30 m



Temperature profile at time 5 h

Figure 6.9 Temperature profile inside the meat time 0, 30m and 5h.

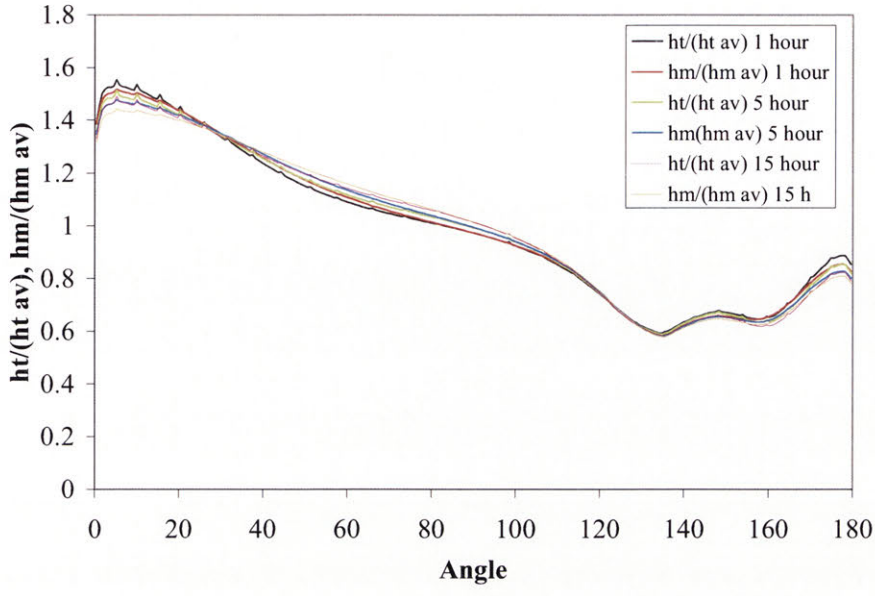


Figure 6.10 Relation between the local and global heat transfer and mass transfer coefficient as a function of the angle at 1 and 5 hours.

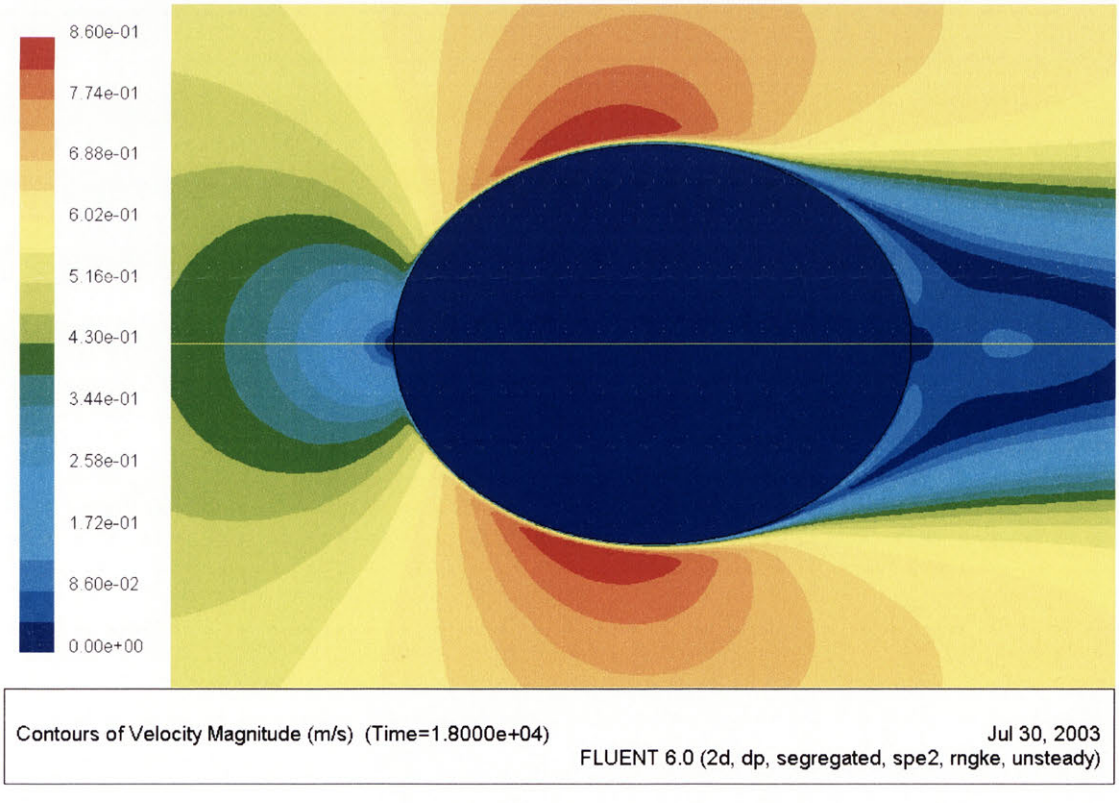


Figure 6.11 Air velocity profile outside the ellipse at 5 hours.

only in a 25 mm surface layer. That can also be seen on figure 6.8 where the water composition profile around the ellipse is shown at 5 h and 20 h. The heat transfer on the other hand, is affecting the entire domain inside the meat, as it is seen in figure 6.9, where it is shown the temperature profile inside the meat at time zero, 30 m and 5 h.

Figure 6.10 shows the ratio between the local and global heat ($h_t / \overline{h}_t$) and mass ($h_m / \overline{h}_m$) transfer coefficients as a function of the angle at 1, 5 and 15 hours. As it is expected, this ratio changes with the angle caused by the development of the thermal and mass boundary layer. However, it is interesting to see that the ratio is almost constant with time and is almost the same between $h_t / \overline{h}_t$ and $h_m / \overline{h}_m$. The lowest value is at about 130° and the maximum at the stagnant point. This result is in agreement with the highest and lowest temperature and water activity zones in figures 6.4 and 6.6. Those local variations can be explained by looking at Figure 6.11 which shows the air velocity profile around the ellipse. It is seen that on the upstream side of the ellipse, between 0° and 90°, the h_t and h_m are higher, while after the de-attachment of the boundary layer, the h_t and h_m are lower, the lowest value being at about 140° where there is a recirculation zone.

6.5 Conclusions

- For the first time, a coupled solution of heat and mass transfer on a 2D beef leg model during meat chilling was developed. The transport

equations on both the air and meat phases were solved simultaneously. Even though FLUENT 6.1.8 is a very powerful CFD software, it has difficulties dealing with simultaneous heat and mass transfer in two different phases, and special techniques had to be used to solve the mass transport equation in the meat. The solid has to be treated as a fluid region of the domain where convection does not occur. The temperature and moisture fields in the solid are represented by new user-defined field variables. At the solid-air interface, the temperature and moisture fields in the fluid and the user-defined fields in the solid are linked by special user-defined equations, representing interfacial equilibria and transfer, to ensure continuity.

- It is also important to take in account the time difficulties involved on CFD simulations. To model 20 hours of chilling took about 6 days in a Pentium 1.5 GHz Computer, making it unpractical for normal industrial calculations.

7. 3-DIMENSIONAL MODEL OF BEEF CARCASSES AND STEADY STATE SIMULATION

In this chapter the complex 3-D geometry of a beef carcass was constructed as a function of weight and fatness. A Matlab program was developed to generate a text file (journal file) with all the instructions to construct the geometry in Gambit (graphic program that creates the geometry and the mesh). The generated mesh contains more than 500.000 cells. With the geometry, a steady state simulation was conducted to analyse the heat (h_t) and mass transfer (h_m) coefficients around the beef surface. Traditional beef chilling models calculate h_t and h_m with empirical equations, which were developed for simple geometrical shapes, but they do not take in account the effects of complex carcass geometry. Heat and mass transfer coefficients are affected by the shape of the object given the development of the momentum, thermal and mass boundary layer.

Additionally, steady state simulations run faster than transient simulations and the heat and mass transfer coefficients obtained can be used in simplified 2D models. To study the influence of free convection, the CFD was run both with and without buoyancy effects.

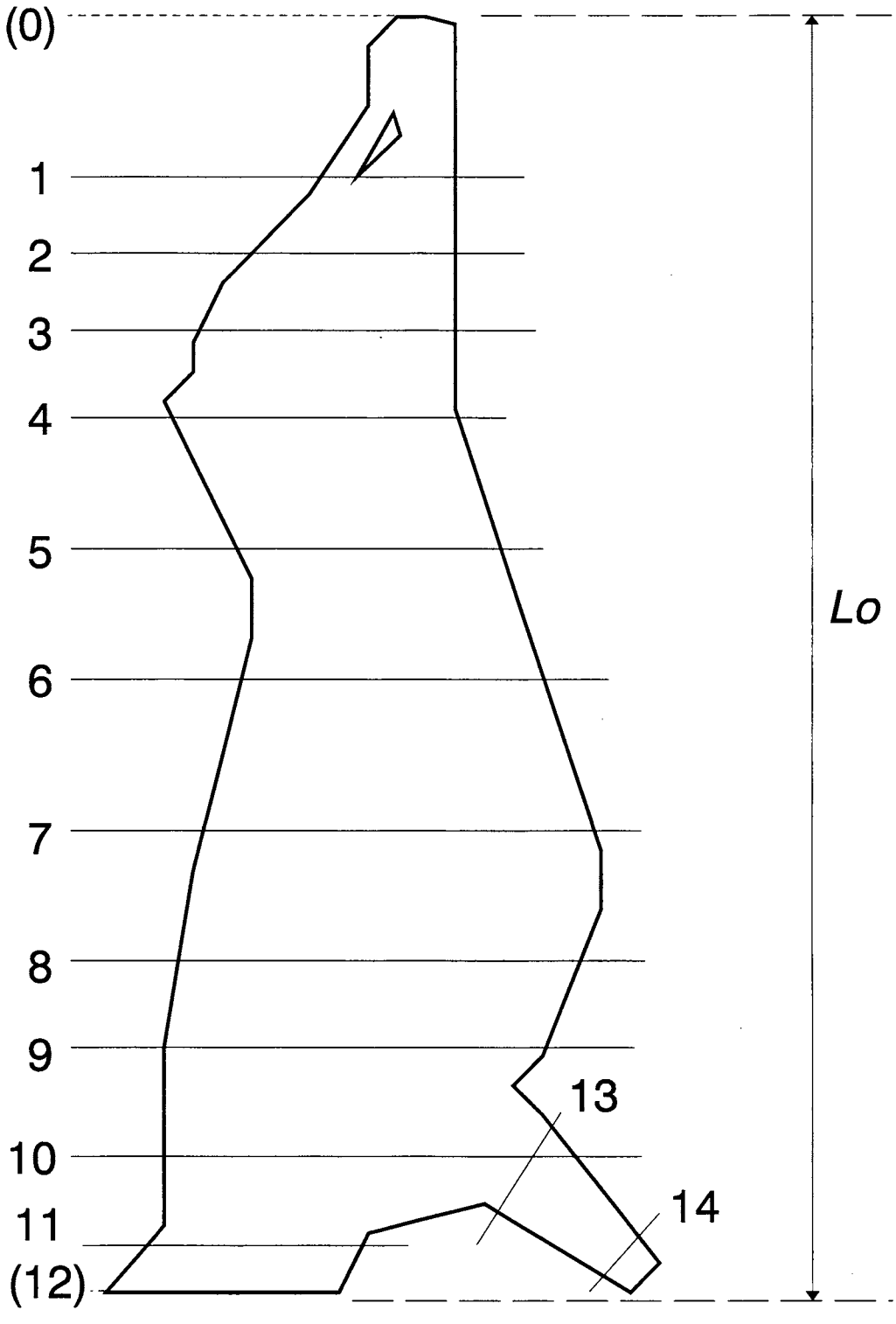


Figure 7.1 Positions of sections for experimental geometry measurements on the beef side (Taken from Davey, 1998).

7.1 Beef side geometry and 3D grid generation

The 3D geometry of the carcass was modelled based on the 2D cross-sectional data from Davey (1998). The geometrical generation procedure is similar to the one used by Nguyen and Pham (1999). The cross-sectional profiles were generated from the weight and fatness of the carcass, using the regression equations developed by Davey (1998), which are based on geometric measurements of 71 beef carcasses. Davey divided the beef side into fourteen cross sections as it is shown in figure 7.1. The cross section of the leg and foreleg (sections 1-3 and 13-14) were treated as ellipses, for where the major and minor axis can be calculated with regression equations. The outer perimeter of sections 4 to 12 were experimentally measured and empirical correlations that express the perimeter as a function of weight and fatness were developed.

The outer section was divided in equal intervals of perimeter ΔP; where the thickness perpendicular to the outer surface was experimentally measured and also correlated with the carcass weight and fatness. The outer cross-sectional profiles of sections 4 to 12 were built using the drawings provided by the Australian Meat and Livestock Corporation (1986). Polynomials were fitted to the outer cross-sectional profiles as shown in these drawings, and then scaled so that length of the polynomials was equal to the regressed experimental perimeter lengths. Inner cross-sectional profiles were then constructed using the regressed beef side thickness. Figures 7.2 to 7.6 show some of the cross-sectional areas.

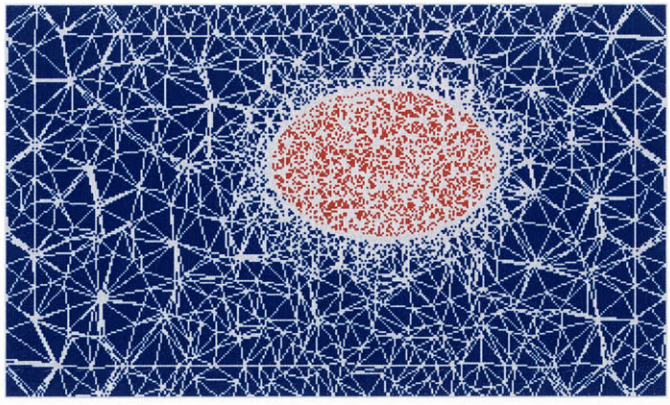


Figure 7.2 Grid section 3.

The overall vertical length, from the top of the leg to the base of the neck was also correlated as a function of the beef weight and fatness. The length between sections was estimated using the mean measured ratio of each length section to the overall length of the side. The surface of the beef side was created by interpolating through all the edges of the 15 cross sectional profiles. The volume of the beef side was then created by connecting the surfaces. The generated geometry of the beef side can be seen in figure 7.7.

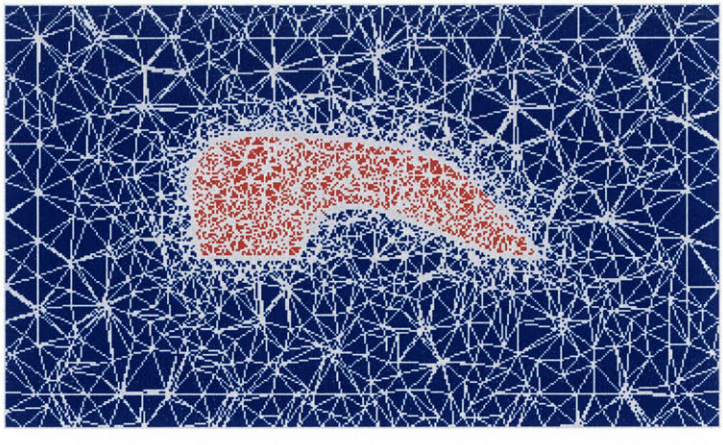


Figure 7.3 Grid section 4.

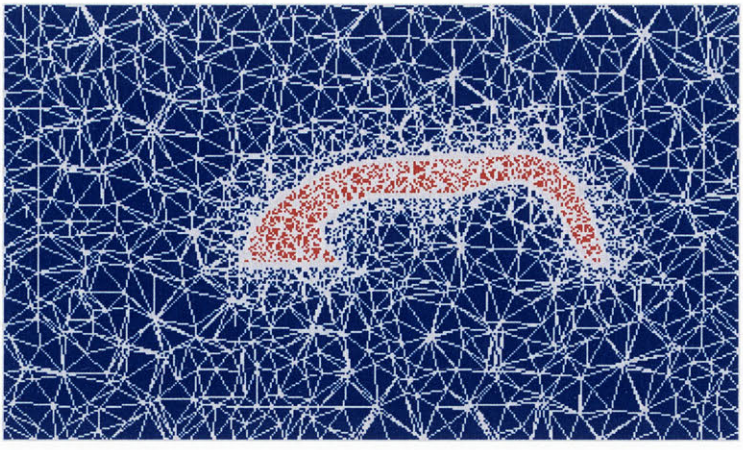


Figure 7.4 Grid section 6.

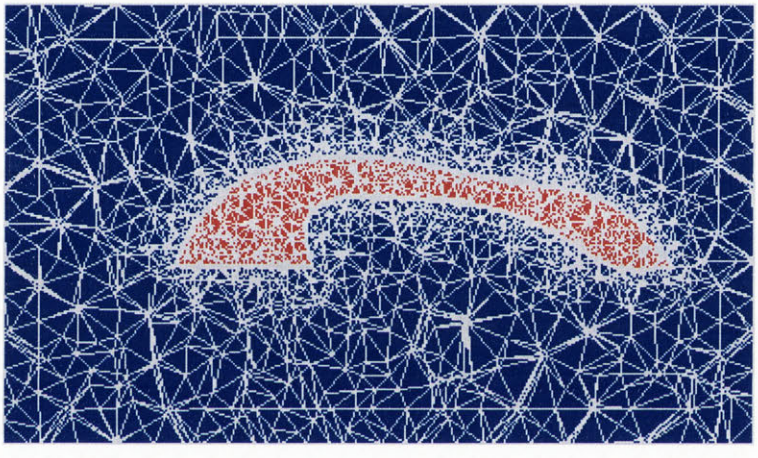


Figure 7.5 Grid section 8.

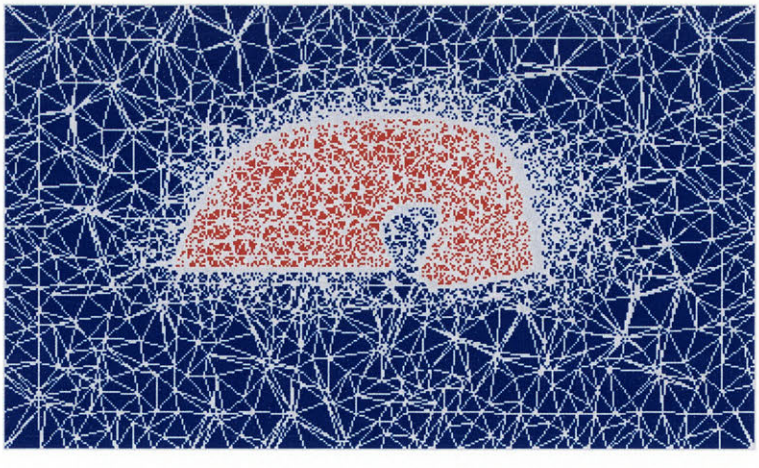


Figure 7.6 Grid section 10.

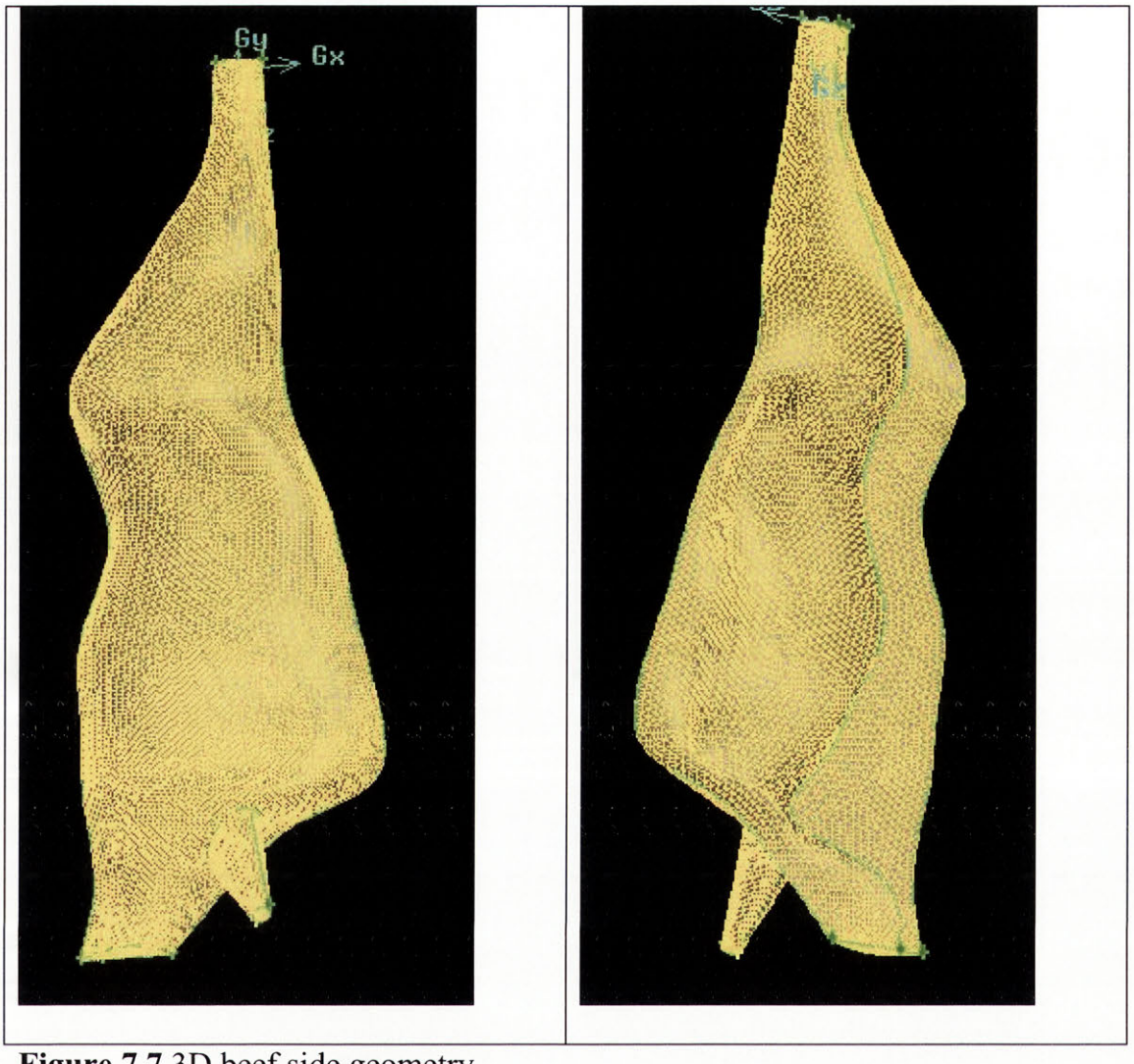


Figure 7.7 3D beef side geometry.

The beef side was inside a wind tunnel of dimensions: 650 mm x 1100 mm x 2775 mm. This apparatus used by Davey was placed inside a real beef chillier room to conduct experimental trials. Both phases, air and beef, were meshed with Gambit using tetrahedral unstructured meshes. Figure 7.8 shows a slide of the unstructured mesh inside the wind tunnel. Figure 7.9 amplifies details of the mesh. The mesh on the air side next to the beef surface (boundary layer on figures 7.9-7.10) was constructed very finely in order to solve the fluid all the

way to the wall using the enhance wall treatment of FLUENT. The mesh in the perpendicular direction of the meat surface must satisfy that $y^+ \approx 1$ (See section 2.9.8.3). The mesh on and parallel to the beef surface does not require being that fine. The velocity and temperature gradients drastically change in the perpendicular direction of the wall but not necessarily in the parallel direction. On the other hand, constructing a very fine mesh in all directions would have required such an enormous amount of cells that the generated information would not have been manageable by even the best personal computer.

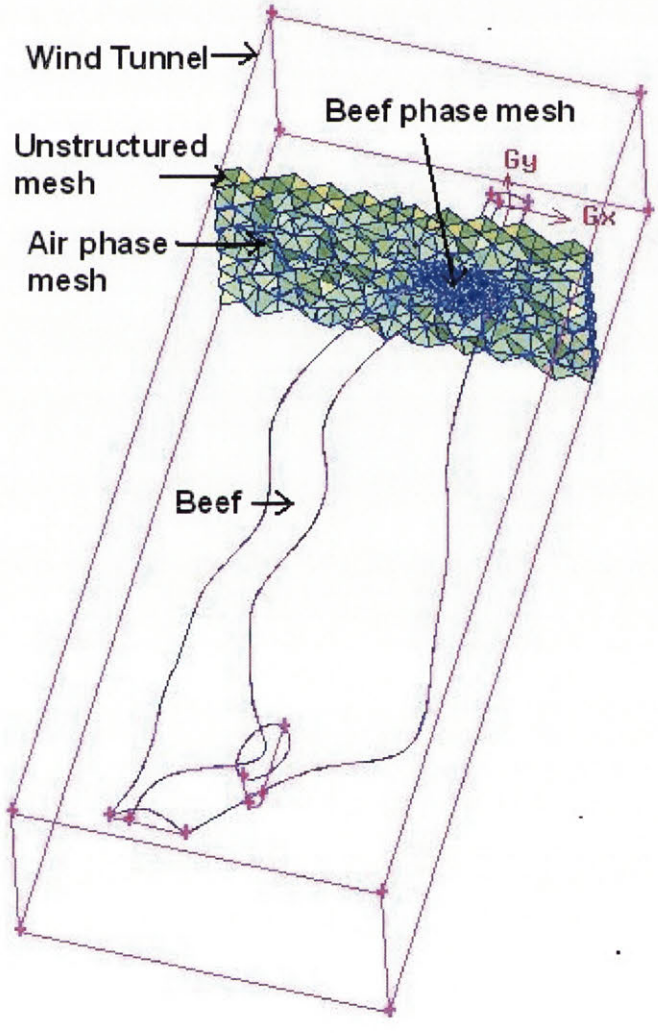


Figure 7.8 Unstructured mesh slide inside the wind tunnel.

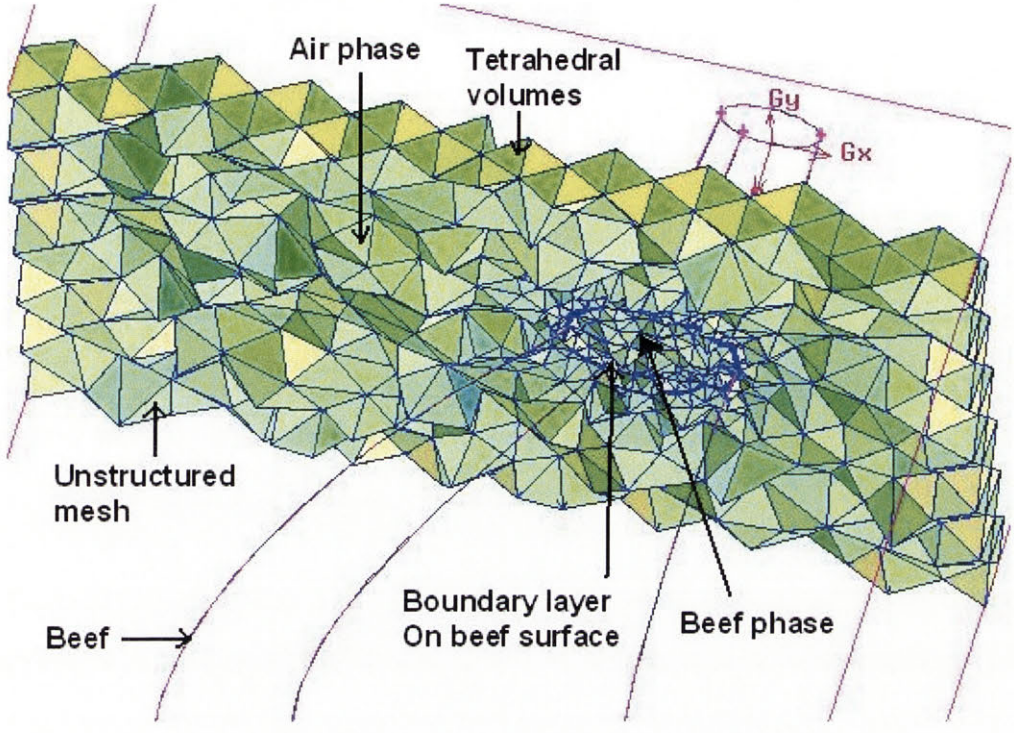


Figure 7.9 Amplification of the unstructured mesh.

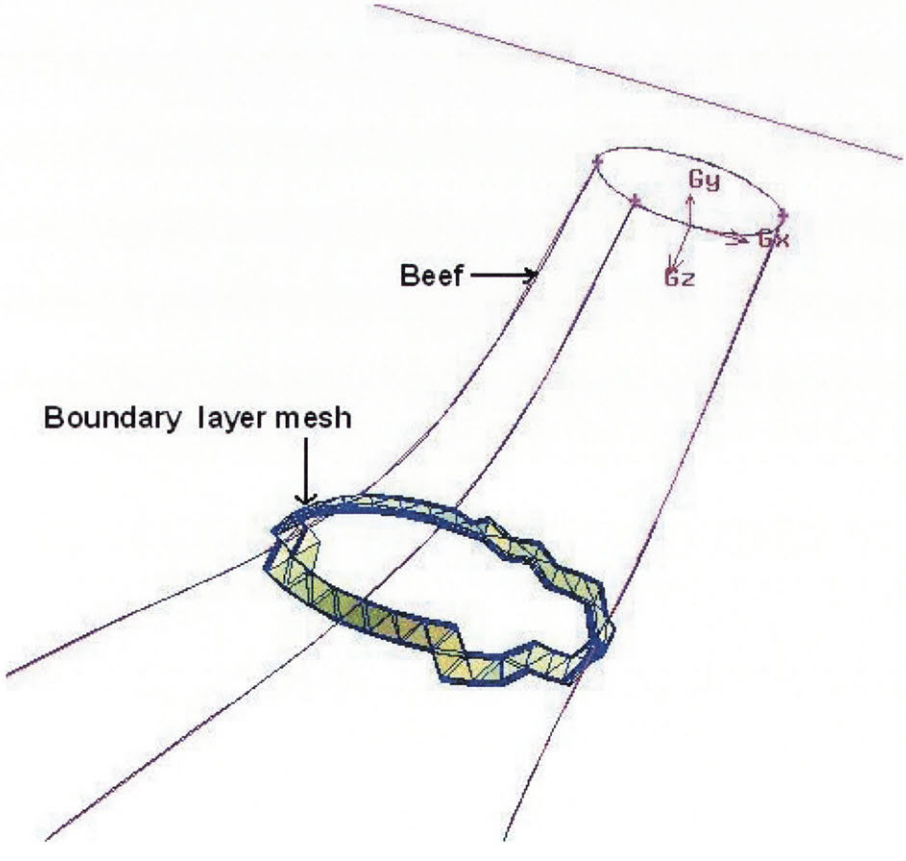


Figure 7.10 Boundary layer mesh around the beef surface.

The desired mesh, fine in the perpendicular direction to the wall but bigger in the parallel direction, was constructed with the "boundary layer option" of Gambit that allows to build structured fine meshes on walls. The thickness of the first row of the mesh on the boundary layer started at 0.1 mm and was steadily increased, moving away from the surface, on the perpendicular direction of the surface. The uniform algorithm was used on Gambit with a growing factor of 1.32 applied to 10 rows. The total distance meshed with the "boundary layer option:" was 4.9 mm. Figure 7.11 shows details of the boundary layer. It is formed of parallelepiped volumes of a bigger triangular base and tiny rectangles perpendicular to the wall.

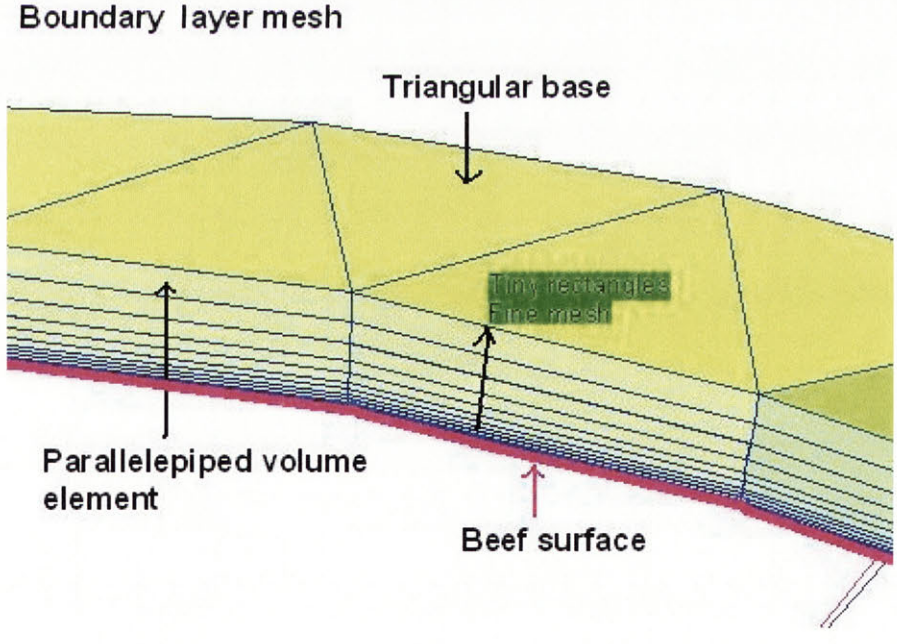


Figure 7.11 Boundary layer mesh details.

The mesh on the meat side, which needs to be fine close to the surface in order to model the water diffusion on the meat, could not be finely meshed with a boundary layer option given constrains on the software. A proper mesh would have had to have been constructed of fine meshing in all directions, but the number of generated cells would have been enormous and unmanageable as it was established earlier. Therefore, the mesh on the beef side was coarse, but the water diffusion equation could not be solved following the procedure established on chapter 6. Thus, an alternative method, where water diffusion is modelled with a separate 1-dimensional mesh, was developed as will be described in chapter 8. The full 3-D mesh was built with a Matlab program that creates a text file with all the instructions to automatically construct the geometry in Gambit. The Matlab program can be seen on the appendix A4.

7.2 Steady State Simulation

A steady state simulation was firstly conducted on the 3D geometrical model of the beef carcass. It was used to obtain the local heat (h_t) and mass transfer (h_m) coefficients around the beef surface. The CFD results were used to calculate h_t and h_m, which are affected by the complex geometry, and which can be used on transient simulations of the solid phase only, thus saving a lot of computation time. They also can be used in simplified 2D models.

Traditional beef chilling models calculate h_t and h_m with empirical equations. Unfortunately, available empirical equations have been usually developed for simple geometrical shapes like cylinders or slabs, but they do not take in account the geometrical effects on real carcasses. Heat and mass transfer coefficients depend on the development of the momentum, thermal and mass boundary layer. Therefore, they are strongly affected by the shape of the object.

In this analysis two cases were considered, forced convection only and the combined effect of forced and natural convection. The RNG k- ε turbulent model and the enhance wall treatment were used as the standard turbulent and wall treatment methods respectively.

7.2.1 Forced convection

The air flow and heat and mass transfer around the beef carcass is described by the conservation equations. The continuity equation is:

$$\nabla \cdot \left(\rho \vec{\bar{v}}\right) = 0 \tag{7.1}$$

where $\vec{\bar{v}}$ is the mean average velocity vector. The momentum equation is:

$$\nabla \cdot \left(\rho \vec{\bar{v}} \vec{\bar{v}}\right) = -\nabla p + \nabla \cdot \left(\bar{\tau}_{eff}\right) \tag{7.2}$$

where:

$$\bar{\tau}_{eff} = \mu_{eff}\left[\left(\nabla \vec{\bar{v}} + \nabla \vec{\bar{v}}^{T}\right) - \frac{2}{3}\nabla \cdot \vec{\bar{v}} I\right] \tag{7.3}$$

The energy equation is:

$$\nabla \cdot \left(\vec{v}\left(\rho E + p\right)\right) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \vec{J}_j + \left(\bar{\tau}_{eff} \cdot \vec{v}\right)\right) \tag{7.4}$$

The water transport equation:

$$\nabla \cdot \left(\rho \vec{v} Y_i\right) = -\nabla \cdot \vec{J}_i \tag{7.5}$$

where:

$$\vec{J}_i = -\rho D_{i,m,eff} \nabla Y_i \tag{7.6}$$

The RNG k- ε model was used to calculate μ_{eff}, k_{eff} and $\rho D_{i,m,eff}$ as described in sections 2.9.5 and 2.9.6.

Boundary conditions are imposed at the inlet, outlet, wind tunnel walls and the beef surface. At the inlet, the defined boundary conditions are velocity, temperature, mass fraction of water, turbulent kinetic energy (k) and turbulent dissipation rate (ε). Given that k and ε are difficult to be determined, FLUENT offers different alternatives to specify the turbulence at the inlet. Among those, the turbulence intensity and the hydraulic diameter were chosen. These two variables are used by FLUENT to estimate k and ε in the inlet.

The outlet boundary condition assumes a zero normal gradient for all flow variables (fully developed flow) except for pressure. At the wind tunnel walls the heat and mass flux was set to zero and the non-slip condition is applied. On the meat surface the temperature and the water concentration were defined. The air humidity next to the beef surface is calculated with the temperature, water

activity of fresh meat and total pressure. No slip condition is also applied. For the CFD solution, the segregated solver, first order upwind and the SIMPLE schemes were used.

In forced convection the local heat transfer coefficient can be expressed as a function of the position, Re and Pr numbers (Incropera and DeWitt, 2002). It is also strongly affected by the turbulence intensity (Kondjoyan and Daudin, 1995). Pr, which is a function of the fluid physical properties, slightly changes in a particular fluid, like air in the normal chilling temperature range. Re on the other hand, strongly changes with the hydrodynamic of the system (air velocity). Thus, the local heat transfer coefficient around beef carcasses is expected to change mainly with position, air velocity and turbulence intensity.

Several CFD runs were conducted to correlated the heat transfer coefficient with Re, Tu and Pr. The mass transfer coefficient is correlated with Re, Tu and Sc. Eleven CFD runs at different air velocities were done to determine the effect of Re on h_t and h_m. The chosen air velocities were: 0.2, 0.4, 0.538, 0.6, 0.8, 1, 1.2, 1.5, 2, 3 and 5 m/s which represent the usual range of air velocities on industrial chillers. The value of 0.538 m/s was chosen because is the air velocity of the run-18 on Davey's work (Davey, 1998). The effect of turbulence was explored by changing the turbulence intensity (5, 10, 20, 40, 60 and 80%). Pr was analysed changing the air temperature on the levels 0°C, 2 °C, 4.9 °C, 10 °C and 15 °C and the beef surface temperature on the levels 10 °C, 20 °C, 30 °C, 42 °C (there is no

activity of fresh meat and total pressure. No slip condition is also applied. For the CFD solution, the segregated solver, first order upwind and the SIMPLE schemes were used.

In forced convection the local heat transfer coefficient can be expressed as a function of the position, Re and Pr numbers (Incropera and DeWitt, 2002). It is also strongly affected by the turbulence intensity (Kondjoyan and Daudin, 1995). Pr, which is a function of the fluid physical properties, slightly changes in a particular fluid, like air in the normal chilling temperature range. Re on the other hand, strongly changes with the hydrodynamic of the system (air velocity). Thus, the local heat transfer coefficient around beef carcasses is expected to change mainly with position, air velocity and turbulence intensity.

Several CFD runs were conducted to correlated the heat transfer coefficient with Re, Tu and Pr. The mass transfer coefficient is correlated with Re, Tu and Sc. Eleven CFD runs at different air velocities were done to determine the effect of Re on h_t and h_m. The chosen air velocities were: 0.2, 0.4, 0.538, 0.6, 0.8, 1, 1.2, 1.5, 2, 3 and 5 m/s which represent the usual range of air velocities on industrial chillers. The value of 0.538 m/s was chosen because is the air velocity of the run-18 on Davey's work (Davey, 1998). The effect of turbulence was explored by changing the turbulence intensity (5, 10, 20, 40, 60 and 80%). Pr was analysed changing the air temperature on the levels 0°C, 2 °C, 4.9 °C, 10 °C and 15 °C and the beef surface temperature on the levels 10 °C, 20 °C, 30 °C, 42 °C (there is no

coefficient can be used to calculate an average or global heat transfer coefficient ($\overline{h}_t$) in some specific areas. Three cross-sections were chosen: leg (cross-section 3), loin (cross-section 7) and shoulder (cross-section 10). Figure 7.13 shows that h_t is higher on the side of the leg that connects with the spinal column. Figure 7.14 shows that air flow is punching on that side of the leg (left on figure 7.14) increasing the heat and mass transfer coefficients. h_t is lower on the rest of the leg. In the case of the shoulder (figure 7.15), h_t is higher on the top centre but lower on the right and left sides.

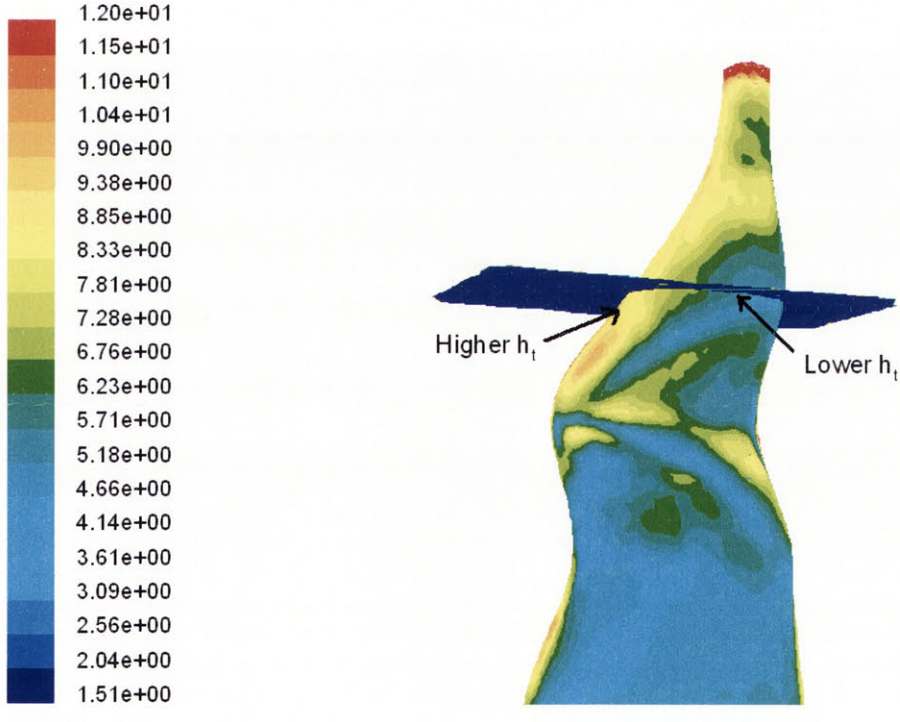


Figure 7.13 Local heat transfer coefficient around cross-section 3 (leg).

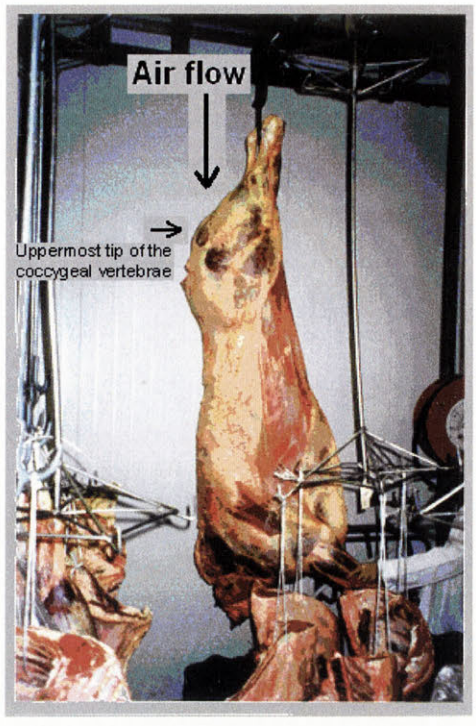


Figure 7.14 Air flow in punching the side of the leg that connects with the spinal column.

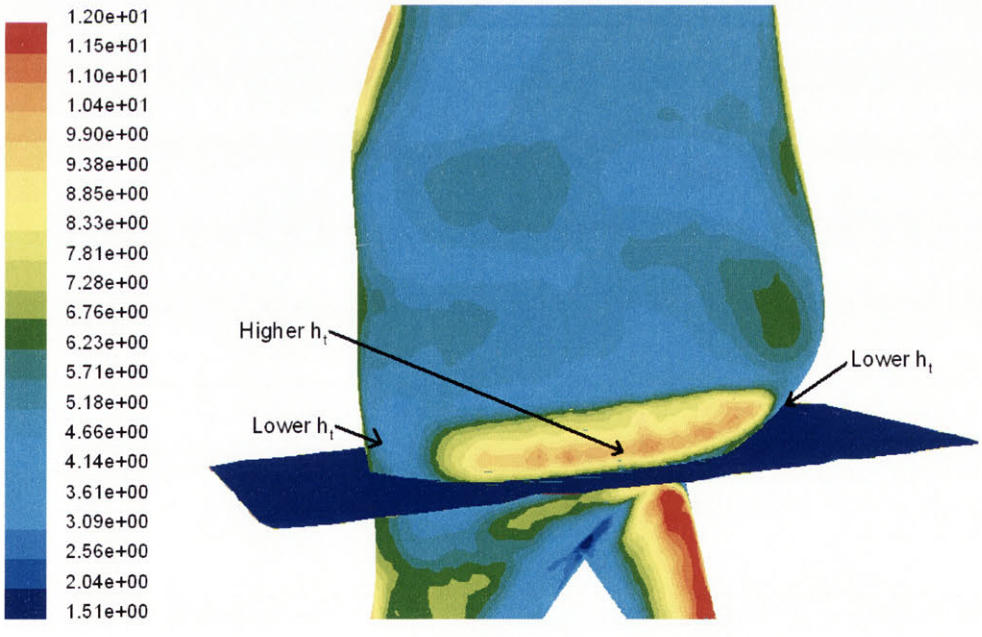


Figure 7.15 Local heat transfer coefficient around cross-section 10 (shoulder).

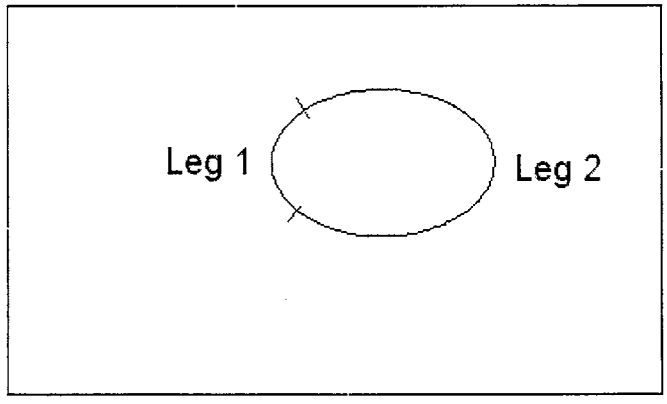


Figure 7.16 sections of the leg.

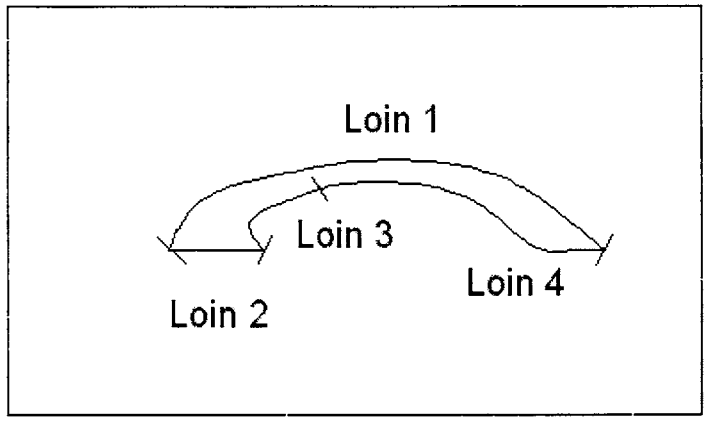


Figure 7.17 sections of the loin.

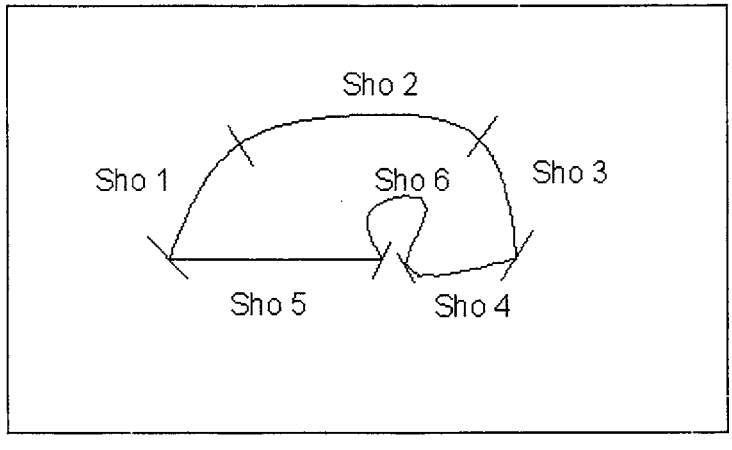


Figure 7.18 sections of the shoulder.

Based on the local heat transfer coefficients figures, the leg was divided in 2 sections, the loin in 4 sections and the shoulder in 6 sections as it is seen in figures 7.16 to 7.18. The area average heat and mass transfer coefficient were calculated in each one of these sections with the equation:

$$\overline{h_t} = \frac{\sum_i A_i h_{t,i}}{A} \tag{7.7}$$

$$\overline{h_m} = \frac{\sum_i A_i h_{m,i}}{A} \tag{7.8}$$

where A_i is the local area where the local heat and mass transfer coefficients are $h_{t,i}$ and $h_{m,i}$ respectively. A is the total area of the section. The average heat and mass transfer coefficients, at constant turbulence intensity, may be represented by equations of the form:

$$\overline{Nu_t} = C\,\mathrm{Re}^m\,\mathrm{Pr}^n \tag{7.9}$$

$$\overline{Sh_m} = D\,\mathrm{Re}^r\,Sc^s \tag{7.10}$$

where $\overline{Nu_t}$ and $\overline{Sh_m}$ are the average Nusselt and Sherwood number respectively. C, D, m, n, r and s are constants that can be fitted using the CFD results. Re number is defined as:

$$\mathrm{Re} = \frac{LV\rho}{\mu} \tag{7.11}$$

where L is the overall vertical length of the carcass, V is the air velocity at the wind tunnel inlet, ρ and μ are the air density and viscosity respectively. Pr and Sc numbers are:

$$\Pr = \frac{\mu c_p}{k_a} \tag{7.12}$$

$$Sc = \frac{\mu}{\rho D_a} \tag{7.13}$$

where k_a, c_p and D_a are the thermal conductivity, heat capacity and water diffusivity respectively. The physical properties ρ μ, k_a, c_p and D_a are calculated at the film temperature:

$$T_{film} = \frac{T_a + T_f}{2} \tag{7.14}$$

where T_a and T_f are the inlet air and beef surface temperatures respectively. It is clearly seen that Pr and Sc can only slightly change with changes in T_a and T_f. Pr merely changed from 0.707 to 0.711 changing the beef surface temperature from 42 C to 10 C. It is observed that Pr is practically constant during the chilling period making difficult to correlate $\overline{Nu_t}$ with Pr and $\overline{Sh_m}$ with Sc. Thus, it was decided to fix the Pr and Sc influence keeping the constants n and s as equal to 1/3, value that is usually reported in most of the empirical equations for $\overline{Nu_t}$ and $\overline{Sh_m}$ on flat plates and cylinders. According to Zdanavichyus *et al.* (1977), the local Nu_t number around a cylinder primary depends upon Reynolds number and turbulence intensity. Kondjoyan and Daudin (1995) indicated that the transfer on both circular and elliptical cylinder is equally affected by the air velocity (Re) and turbulence intensity. Previous investigators did not discuss the role of Pr or Sc in their work.

The constants m and C can be calculated by plotting $\ln\left(\overline{Nu_t}/\mathrm{Pr}^n\right)$ vs. $\ln(\mathrm{Re})$ at constant turbulent intensity (Tu). The data were obtained mainly changing the air velocity while T_a, T_b and Tu were kept constant. The CFD runs changing the beef surface and air inlet temperatures were also included even though the changes on $\ln\left(\overline{Nu_t}/\mathrm{Pr}^n\right)$ with $\ln(\mathrm{Re})$ were minimal given that the air velocity of these runs was constant. These runs were aimed to correlate $\overline{Nu_t}$ with Pr but that was not done given that Pr kept almost constant. Figure 7.19 shows $\ln\left(\overline{Nu_t}/\mathrm{Pr}^n\right)$ vs. $\ln(\mathrm{Re})$ for the sections leg1, loin1 and sho2 (see figures 7.16-7.18). The parameters m and C are found by linear regression.

The turbulence intensity also affects the heat and mass transfer coefficients. Kondjoyan and Daudin (1995) indicated that the transfer on both circular and elliptical cylinder is equally affected by the air velocity (Re) and turbulence intensity. Kondjoyan and Daudin (1993b) reported that the air velocity and turbulence intensity in elliptical cylinders has a greater effect on the heat transfer coefficients than the body shape characteristics (length and ratio).

The effect of turbulence was determined changing the turbulence intensity at constant values of Re and Pr (Figure 7.20). Figure 7.21 shows $\ln\left(\overline{Nu_t}/\overline{Nu_{t,ref}}\right)$ (referred as ln(Nut/Nutturb10) on figure 7.21) vs. $\ln(Tu)$, where $\overline{Nu_{t,ref}}$ is the Nu number at the reference turbulence intensity value (Tu=10%). It is seen that the behaviour is exponential and the data may be fitted by a model like:

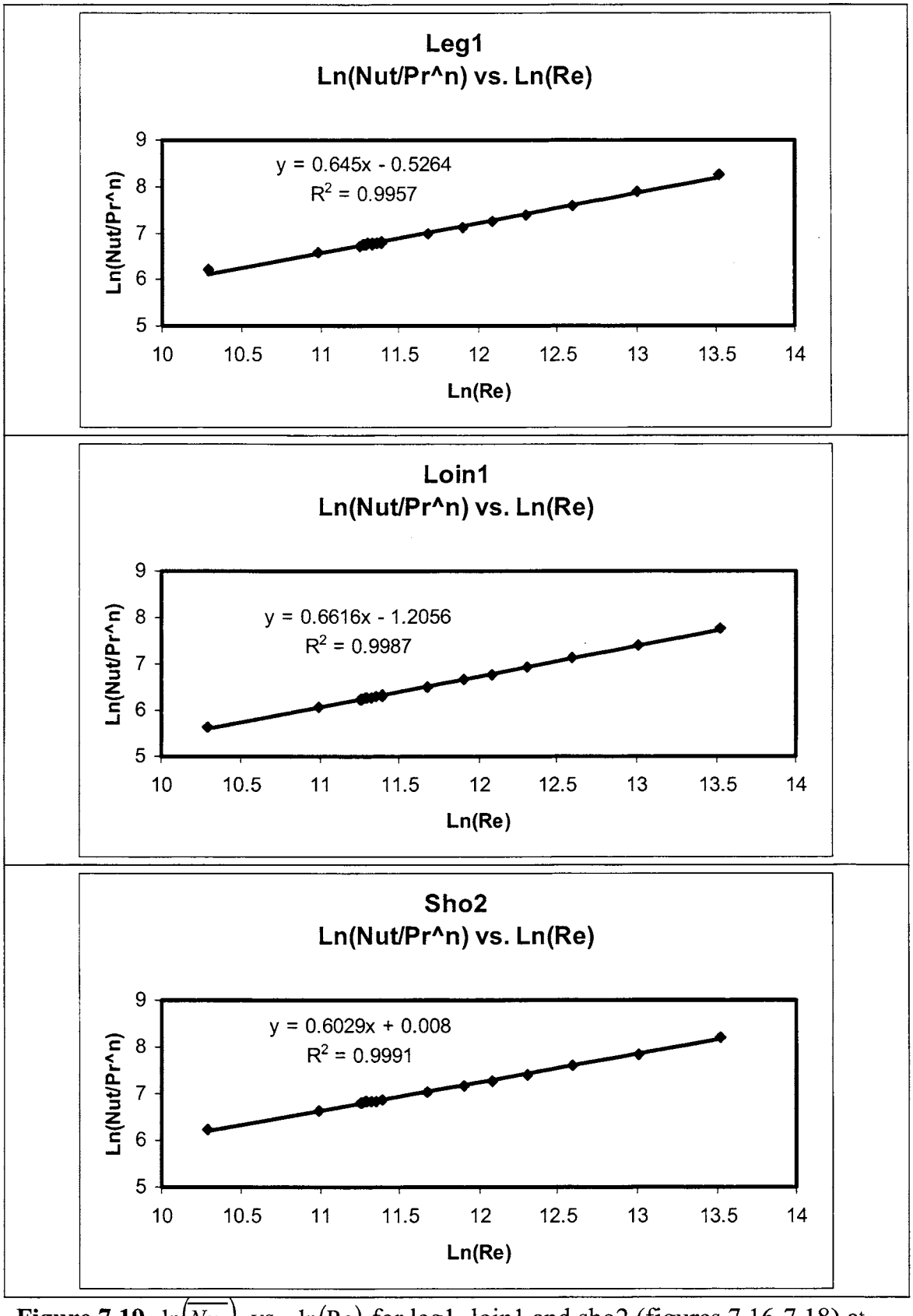


Figure 7.19 $\ln\left(\overline{Nu_t}\right)$ vs. $\ln(\mathrm{Re})$ for leg1, loin1 and sho2 (figures 7.16-7.18) at Tu = 10%.

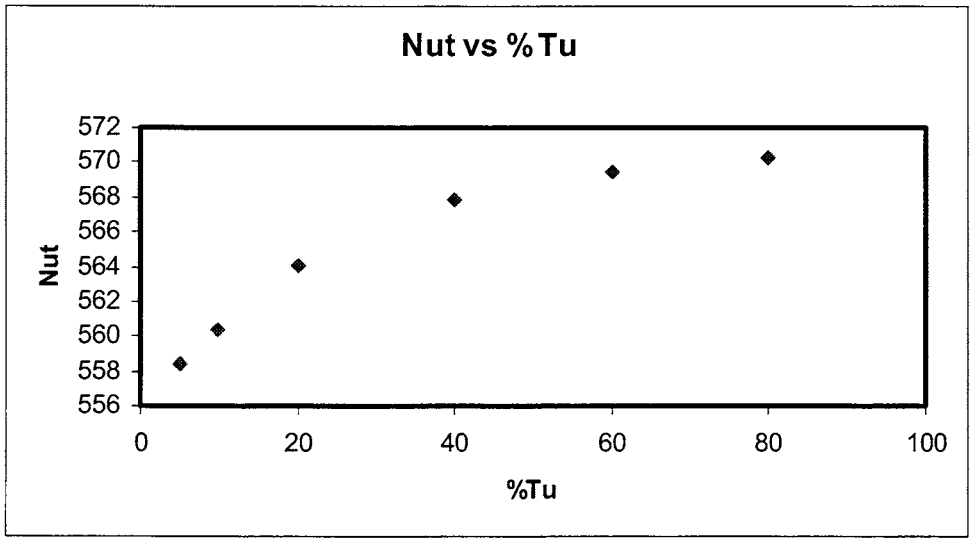


Figure 7.20 Nu_t vs. Tu on sho3 (figure 7.18) at air velocity = 0.538 m/s.

$$\frac{\overline{Nu_t}}{\overline{Nu_{t,\,ref}}} = B \cdot Tu^A \tag{7.15}$$

with equation 7.9:

$$\overline{Nu_t} = C_2 \,\mathrm{Re}^m \,\mathrm{Pr}^n \, Tu^A \tag{7.16}$$

where $C_2 = C \cdot B$.

All the sections follow equation 7.16 very well except loin3. That is seen in Figure 7.22. Figure 7.23 plots $\overline{Nu_t}$ vs. Tu for loin3 showing that at Tu = 20% the $\overline{Nu_t}$ number drops to a lower value, breaking the $\overline{Nu_t}$ increasing trend that exhibits other sections (Figure 7.20). Loin3 is placed in the most twisted cavity of the loin (Figure 7.17), where "abnormal" behaviours, caused by the complex geometry, are expected. According to Kondjoyan and Daudin (1997), the effect of turbulence intensity on the mean transfer coefficient on pork hindquarter is

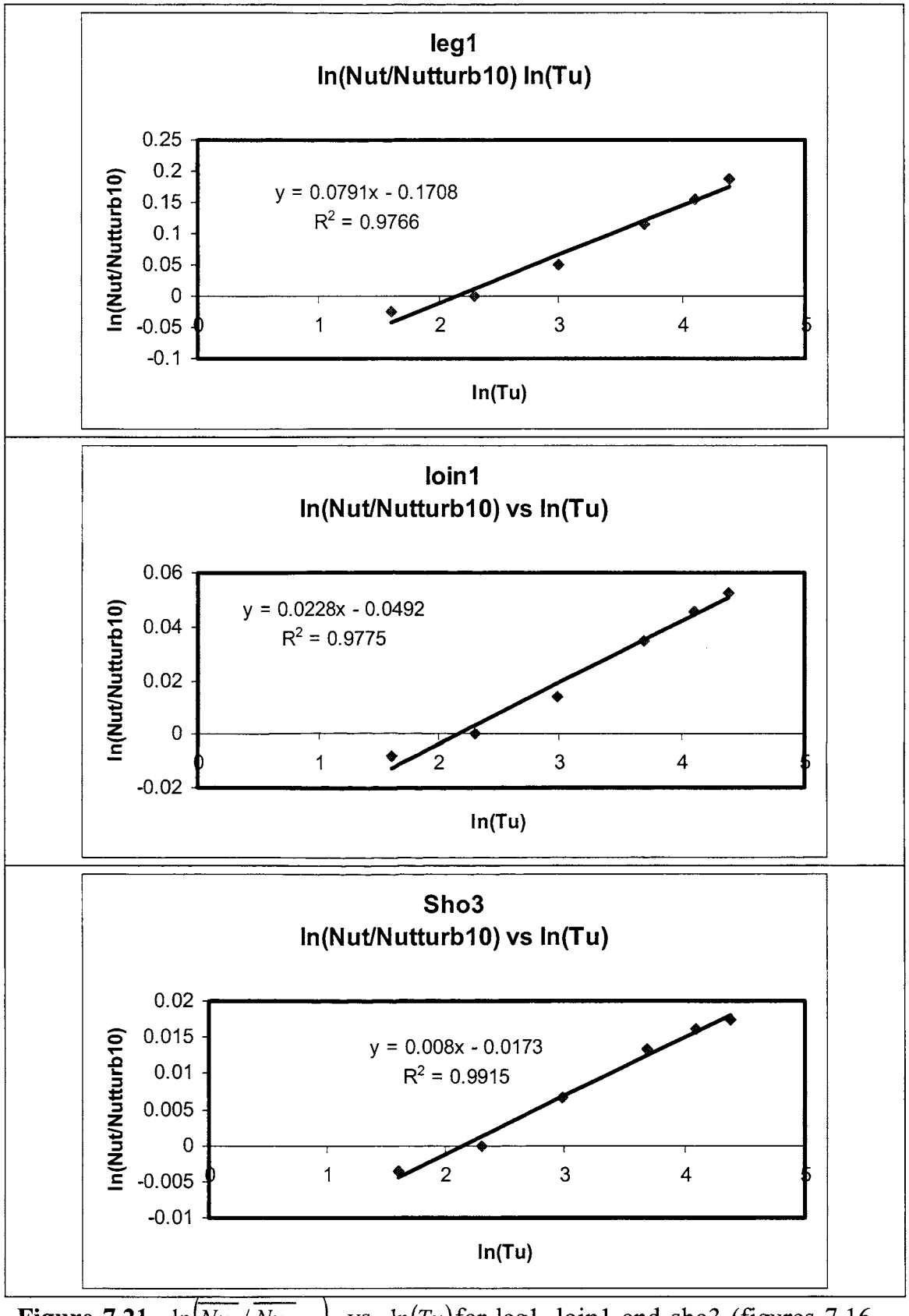


Figure 7.21 $\ln\left(\overline{Nu_t} / \overline{Nu_{t,ref}}\right)$ vs. $\ln(Tu)$ for leg1, loin1 and sho3 (figures 7.16 – 7.18) at air velocity = 0.538 m/s.

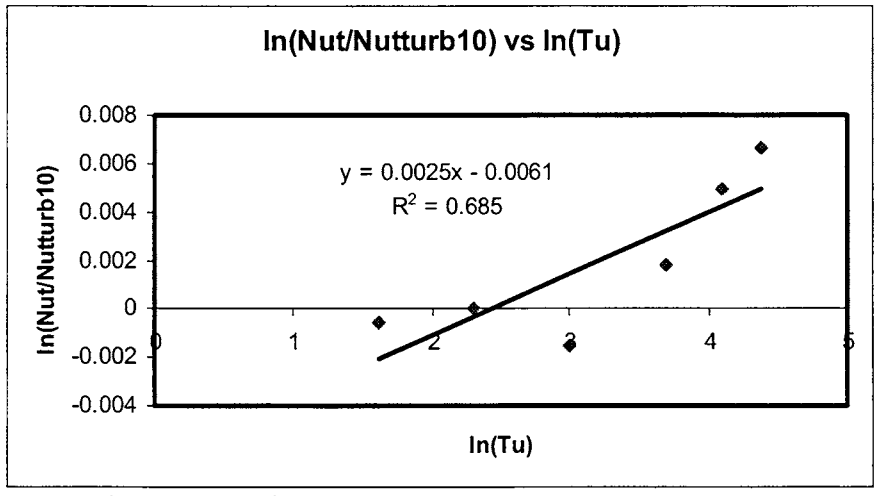


Figure 7.22 $\ln\left(\overline{Nu_t} / \overline{Nu_{t,\,ref}}\right)$ vs. $\ln(Tu)$ for loin3 (figure 7.17) at air velocity = 0.538 m/s.

important but complex from what happens on simple geometrical shapes as cylinders. They reported that an increase in turbulent intensity from 1.3% to 6% decreases the mean transfer coefficients while a further increase in Tu increases those values.

Section sho6, which is the most "hidden" section of the carcass (figure 7.18), also exhibits an "abnormal" behaviour. Even though $\overline{Nu_t}$ is well represented by equation 7.16 (figure 7.24), its value is decreasing when the turbulence intensity is increasing as it is seen in figure 7.25.

The mass transfer coefficients can also be correlated with Re, Pr and Tu with the equation:

$$\overline{Sh} = D_2 \, \mathrm{Re}^r \, Sc^s Tu^E \tag{7.17}$$

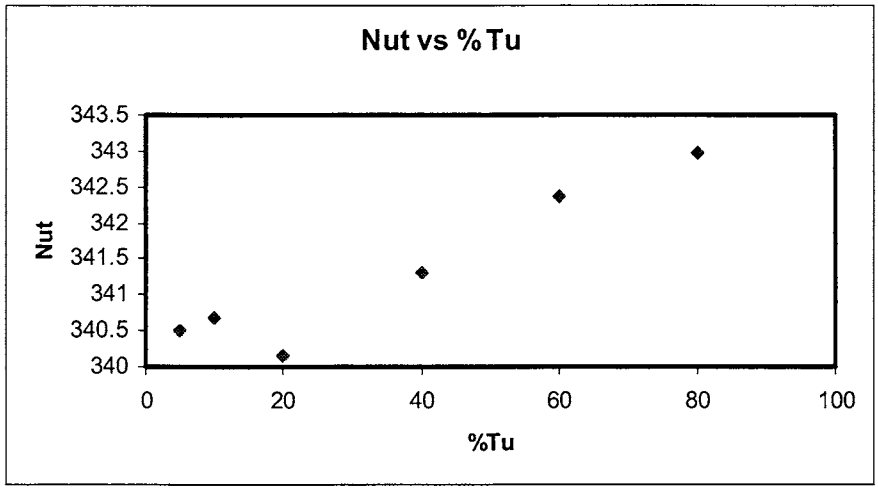


Figure 7.23 Nu_t vs. Tu on loin3 (figure 7.17) at air velocity = 0.538 m/s.

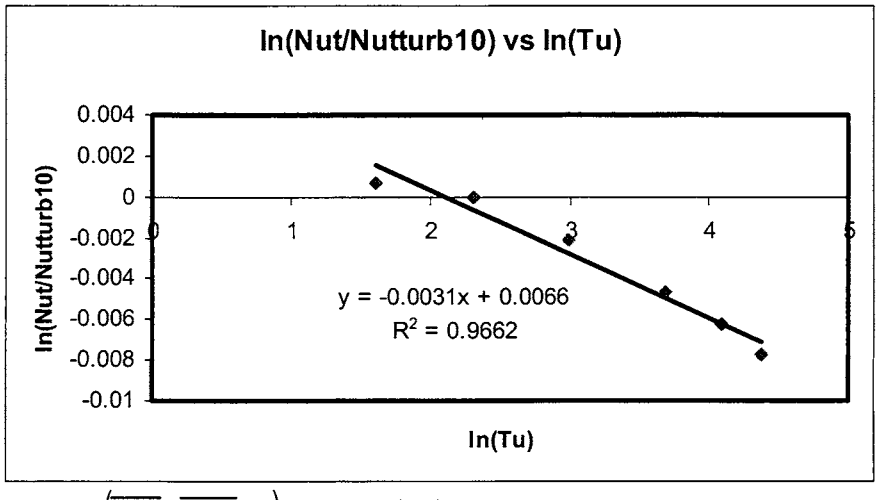


Figure 7.24 $\ln\left(\overline{Nu_t} / \overline{Nu_{t,ref}}\right)$ vs. $\ln(Tu)$ for sho6 (figure 7.18) at air velocity = 0.538 m/s.

Following the same procedure explained above for heat transfer coefficients. Table 7.1 contains the constants for the 12 studied sections. It is seen that the values of the heat transfer constants C_2, m and A are very similar to the equivalent constants of the mass transfer equation D_2, r and E. So it was decided to correlate the heat and mass transfer equation through the Lewis relationship:

$$\frac{\overline{h_t}}{\overline{h_m} c_p} = \left(\frac{Sc}{\Pr}\right)^u = Le^u \tag{7.18}$$

where Le is the Lewis number and u is usually taken as (2/3). The constant u was taken in this work as a fitted parameter. Its values are reported on table 7.1. The root mean squares error (%RMS) of the models 7.17 and 7.18 comparing with the CFD data are very similar showing that equation 7.18 may be used to correlate the heat and mass transfer coefficients as a simpler alternative of equation 7.17.

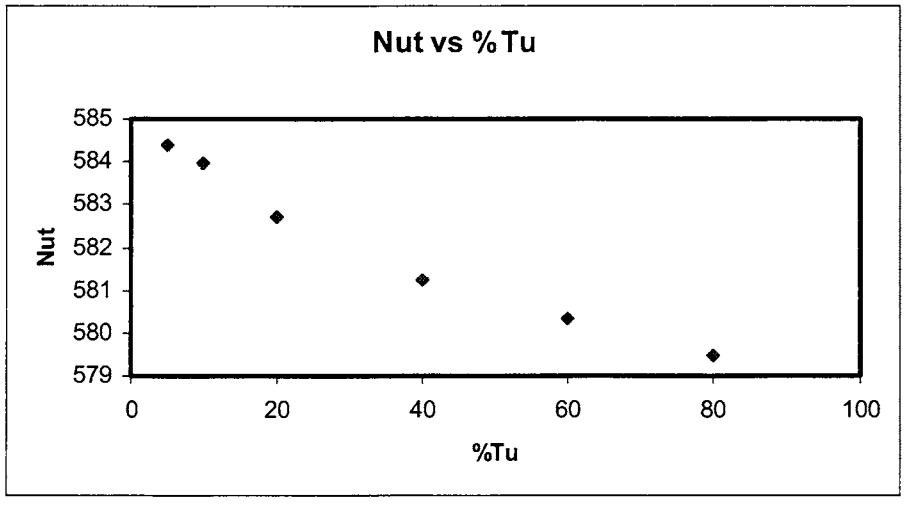


Figure 7.25 Nu_t vs. Tu on sho6 (figure 7.18) at air velocity = 0.538 m/s.

Given the long time required to conduct each CFD run (between 10 -24 hours), the amount of runs were the minimal necessary to fit the empirical equations 7.16 to 7.18. It is recommended to increase the number of CFD runs in further studies to gain a better accuracy.

7.2.2 Combined forced and natural convection

The natural convection affects the heat and mass transfer coefficients when the gravitational force, which acts over a density gradient (caused by a temperature

gradient), imparts a significant momentum contribution to the system. To take this effect into account, equation (7.2) must include the term $\rho\vec{g}$:

$$\nabla \cdot \left(\rho \vec{v}\vec{v}\right) = -\nabla p + \nabla \cdot \left(\bar{\tau}_{eff}\right) + \rho\vec{g} \tag{7.19}$$

Equations 7.1 and 7.2 to 7.6 keep the same. In the previous modelling the fluid properties were calculated as function of temperature. This is a critical aspect, especially the dependence of density with temperature, which must be considered when buoyancy is modelled. The density was calculated with the equation:

$$\rho = \frac{P_T}{\frac{R}{M_w} T} \tag{7.20}$$

where P_T is the total pressure of the system, R is the universal gas constant, T is temperature, and M_w is the molecular weight.

$\overline{Nu_t} = C_2 \mathrm{Re}^m \mathrm{Pr}^n Tu^A$ (7.19), $\overline{Sh} = D_2 \mathrm{Re}^r \mathrm{Pr}^s Tu^E$ (7.20), $\frac{\bar{h}_t}{\bar{h}_m c_p} = Le^u$ (7.21)										
	C_2	D_2	m	r	A	E	u	%RMS (7.19)	%RMS (7.20)	%RMS (7.21)
leg1	0.498	0.503	0.65	0.64	0.0791	0.0786	0.519	0.5133	0.5189	0.5156
leg2	0.155	0.162	0.72	0.71	0.0800	0.0807	0.588	0.5583	0.5878	0.5858
loin1	0.285	0.290	0.66	0.66	0.0228	0.0233	0.305	0.2896	0.3049	0.3040
loin2	0.299	0.307	0.66	0.66	0.0286	0.0294	0.232	0.2187	0.2320	0.2350
loin3	0.116	0.111	0.72	0.72	0.0025	0.0028	0.457	0.3929	0.4568	0.4058
loin4	0.021	0.243	0.85	0.66	0.0218	0.0222	0.458	5.2094	0.4580	5.2185
sho1	0.318	0.331	0.66	0.66	0.0208	0.0213	0.466	0.4498	0.4662	0.4675
sho2	1.003	0.962	0.60	0.60	0.0024	0.0024	0.217	0.2271	0.2166	0.2115
sho3	0.716	0.685	0.60	0.60	0.0080	0.0082	0.245	0.2506	0.2455	0.2440
sho4	0.524	0.504	0.64	0.64	0.0077	0.0078	0.171	0.1727	0.1715	0.1619
sho5	0.216	0.214	0.69	0.69	0.0111	0.0115	0.194	0.1972	0.1937	0.1877
sho6	0.495	0.449	0.64	0.64	-0.0031	-0.0032	0.220	0.2117	0.2201	0.2232

Table 7.1 Constants empirical equations for heat and mass transfer coefficients on the twelve sections on figures 7.16 - 7.18 .

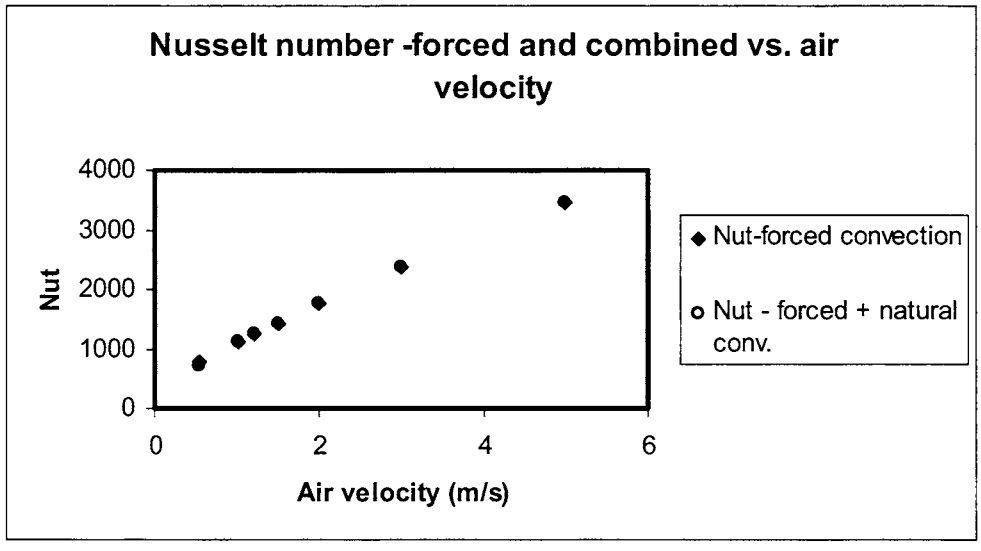


Figure 7.26 Nu_t forced and combined (forced + natural) convection vs. air velocity on section leg1 (figure 7.16) at beef surface temperature of 42 °C.

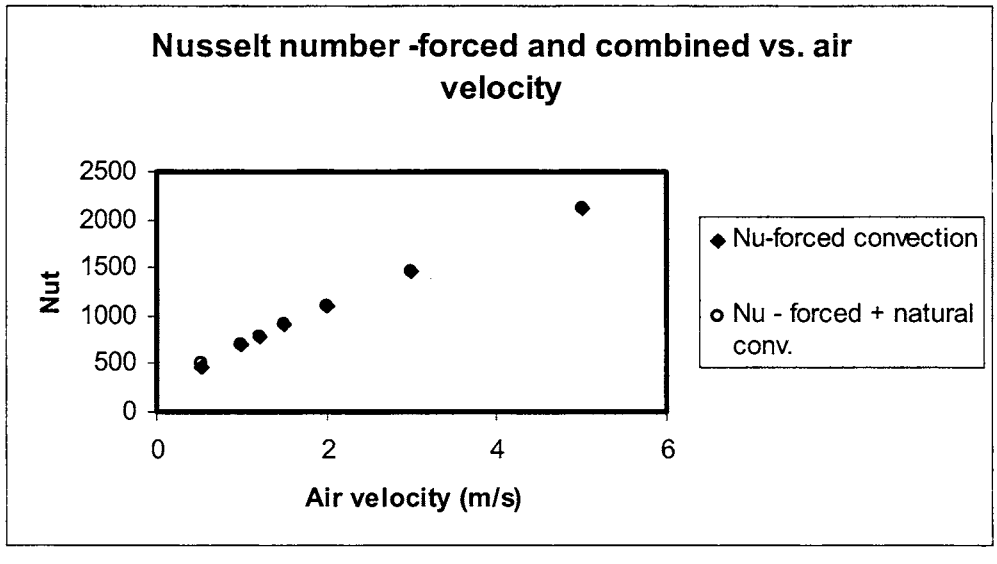


Figure 7.27 Nu_t forced and combined (forced + natural) convection vs. air velocity on section loin1 (figure 7.17) at beef surface temperature of 42 °C.

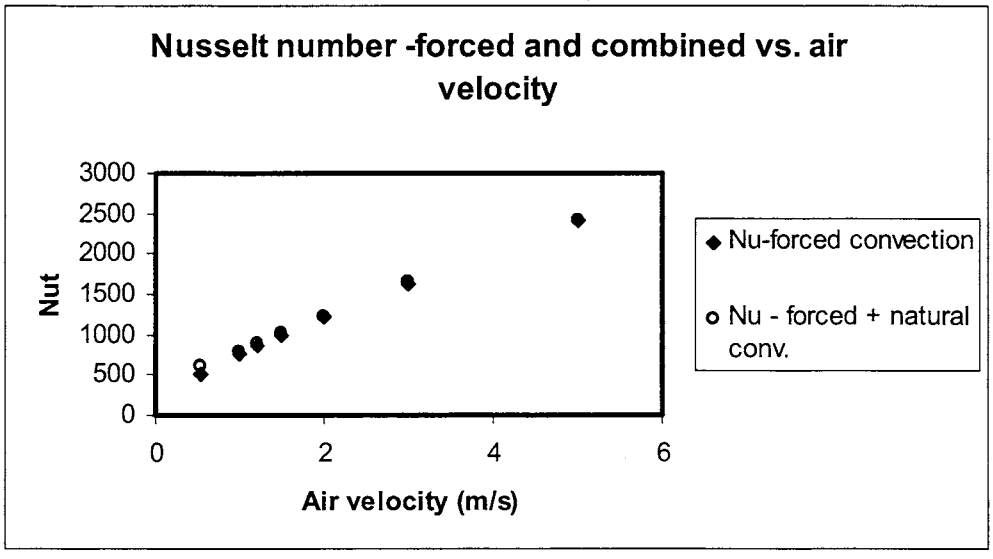


Figure 7.28 Nu_t forced and combined (forced + natural) convection vs. air velocity on section sho1 (figure 7.18) at beef surface temperature of 42 °C.

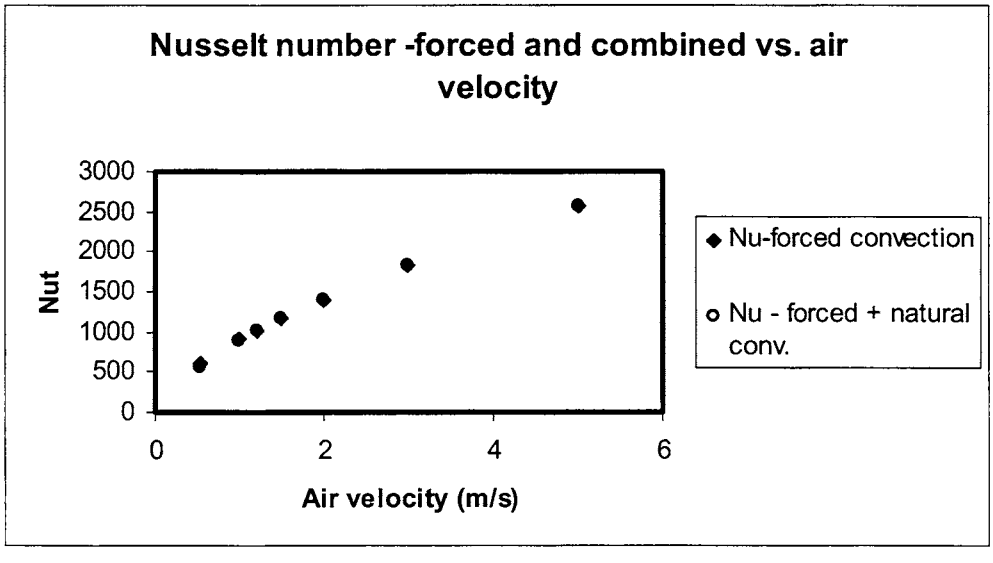


Figure 7.29 Nu_t forced and combined (forced + natural) convection vs. air velocity on section sho4 (figure 7.18) at beef surface temperature of 42 °C.

The CFD runs conducted for forced convection were modified to take the gravitational effects into account. The majority of the runs converged to a stable result on FLUENT except the ones at air velocities, 0.2, 0.4, and 0.8 m/s. Figures 7.26 to 7.29 plot $\overline{Nu_t}$ vs. air velocity on sections leg1, loin1, sho1 and sho4 respectively. $\overline{Nu_t}$ forced and combined are the same in most part of the range except at the lowest air velocity where there is a slight difference. That is expected since buoyancy becomes important at very low air velocities.

The effect of the buoyancy on $\overline{Nu_t}$ is negative in sections leg1 and sho4 but positive on section loin1 and sho1. This can be explained in the following way: the fluid bound to the beef surface tends to ascend by effect of buoyancy if the beef surface temperature is higher than the air temperature. At higher temperature lower air density and vice versa. The gravity makes the heavy fluid, which is farther from the beef surface, to descend and the light fluid (on the beef surface) to ascend. Negative effects appear on sections where the fluid on the beef surface is descending. Positive effects, on the other hand, appear where the fluid bounded to the wall is ascending, or where the buoyancy makes the fluid besides the wall to reverse generating more recirculation. That may happen on very low velocity zones. To see positive and negative zones, a velocity profile plane at constant x-coordinate was obtained with FLUENT (figure 7.30). Figure 7.31 shows the y-z view of the plane. Three positions A, B and C were chosen to analyse the velocity profiles and the effects of buoyancy (positive or negative). Position A is

placed in a high velocity zone on the leg, position B is in a recirculation zone placed in a small depression at the beginning of loin below the leg. Position C is in a low velocity zone on the loin.

Figure 7.32 shows the velocity vector on position A for forced and combined convection. It is seen that buoyancy is reducing the velocity magnitude of the vectors. Thus, it decreases the heat and mass transfer coefficients (negative effect). Figure 7.33 shows the velocity profile on position B for forced convection. There is a recirculation in this position, given that it is placed in a depression. The fluid besides the wall is ascending in this zone. Thus, buoyancy increases the flow velocity in the upstream direction and enhances the recirculation in that zone as it is seen in figure 7.34. The heat and mass transfer coefficients are increased in position B (positive effect).

Figure 7.35 shows the velocity vectors profile on position C for forced convection. The flow is down stream. This position is placed in a low velocity zone. When buoyancy is taken into account (figure 7.36), its effect is strong enough to reverse the flow besides the wall. Recirculation is created and the heat and mass transfer are enhanced (positive effects).

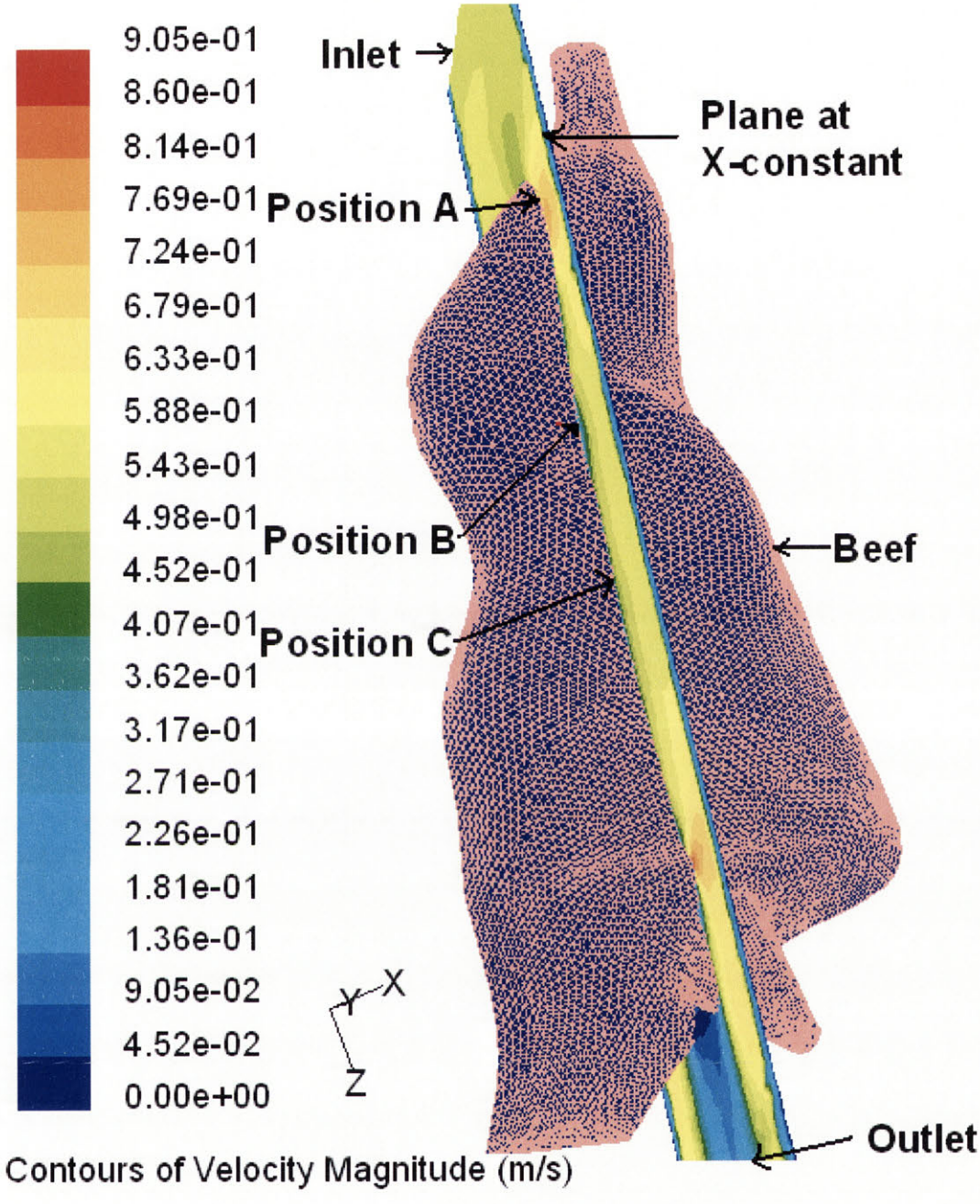


Figure 7.30 Velocity profile plot on a plane at constant x-coordinate.

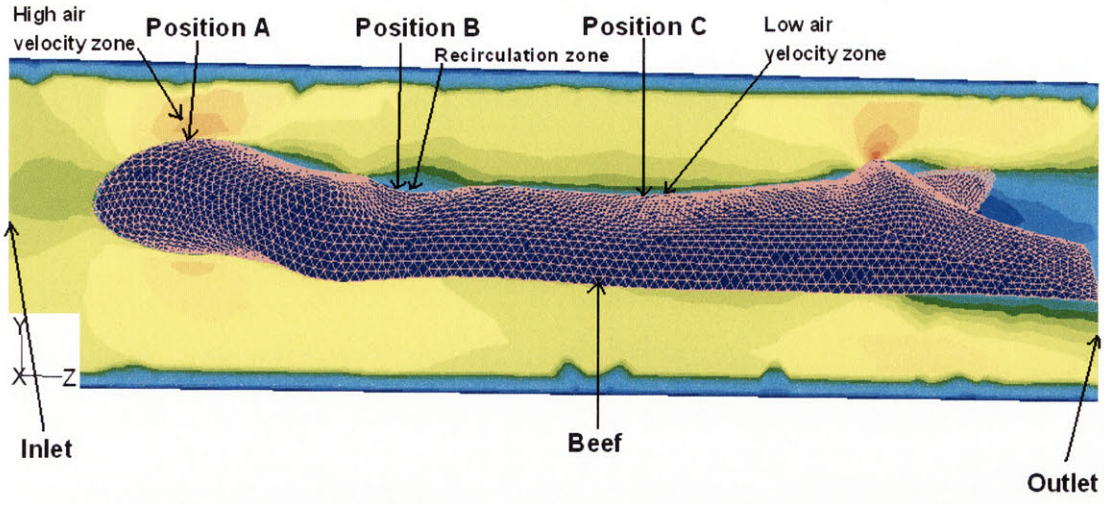


Figure 7.31 Velocity profile plot on a plane at constant x-coordinate (y-z view).

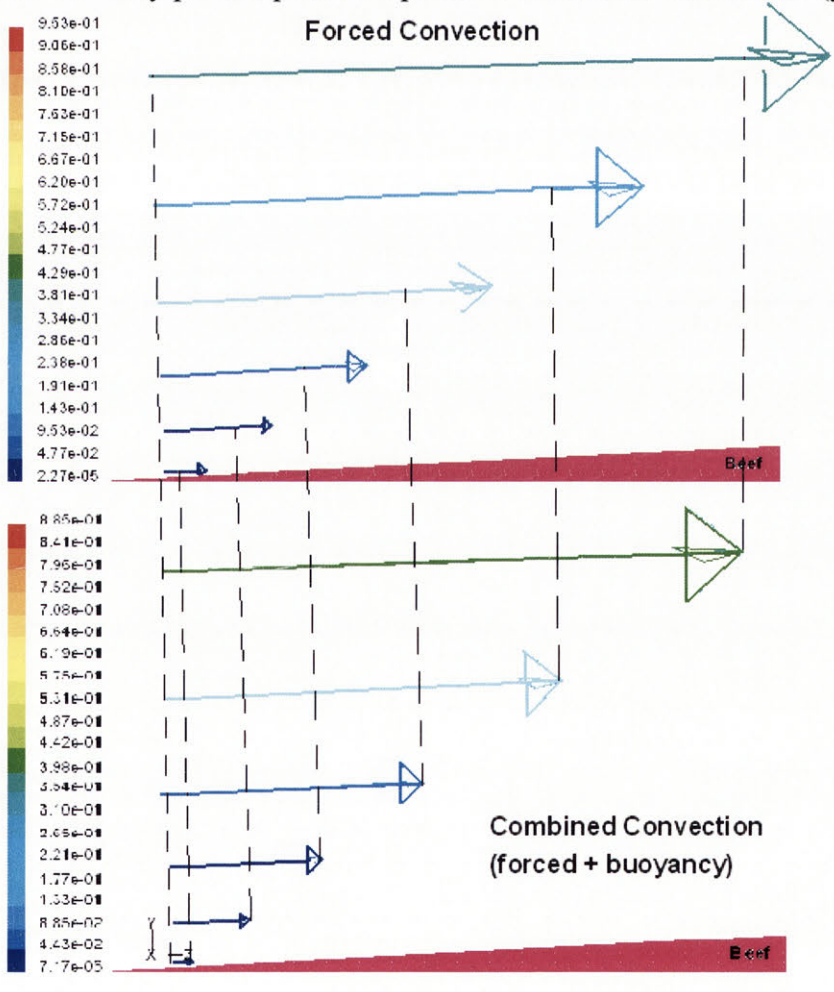


Figure 7.32 Velocity vector profiles on position A (forced and combined convection).

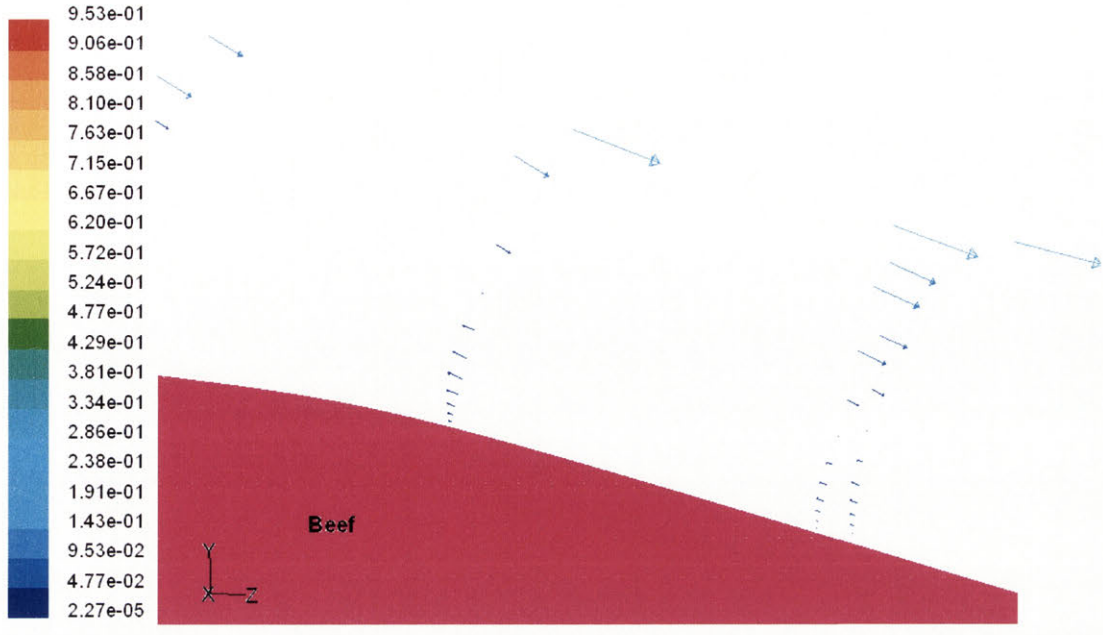


Figure 7.33 Velocity vector profiles on position B (forced convection).

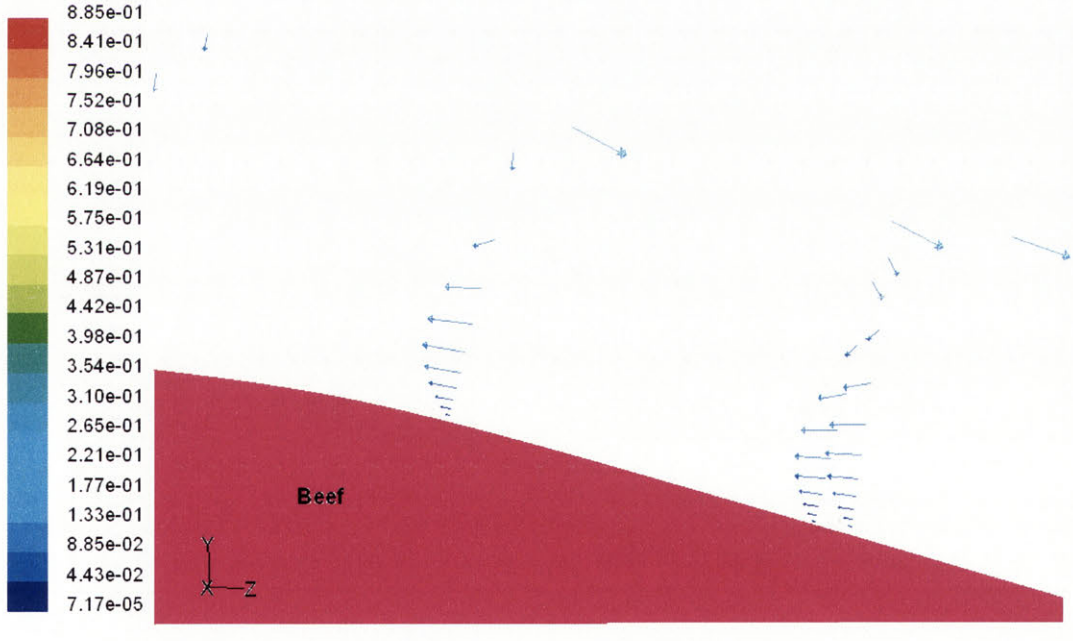


Figure 7.34 Velocity vector profiles on position B (combined convection).

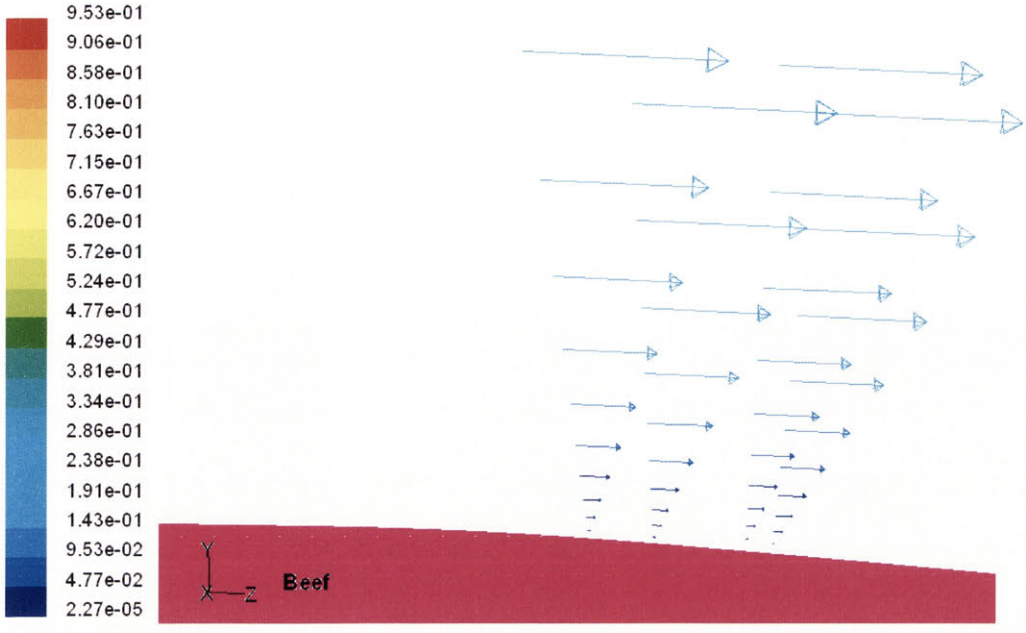


Figure 7.35 Velocity vector profiles on position C (forced convection).

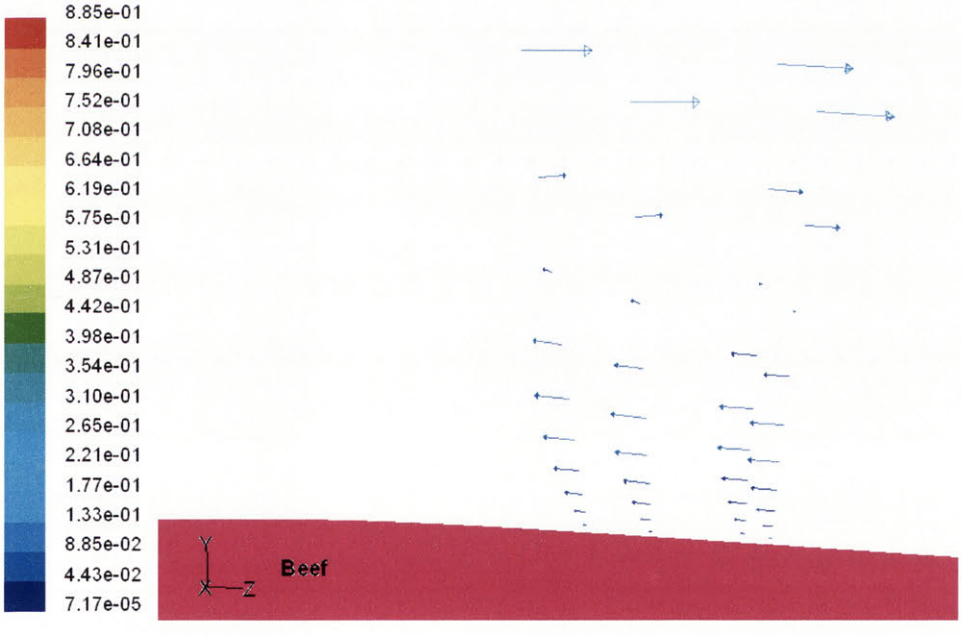


Figure 7.36 Velocity vector profiles on position C (combined convection).

Table 7.2 shows the percentage of difference between the forced and combined Nusselt numbers for sections leg1, loin1 and sho1 as a function of the air velocity. The beef surface temperature keeps constant at 42 °C. The maximum difference, at the lowest air velocity (0.538 m/s), is 14.1%, but it decreases when the air velocity, or Reynolds number, increases.

Section		Leg1			Loin1			Sho1		
V	Re	Nu_f	Nu_c	diff	Nu_f	Nu_c	Diff	Nu_f	Nu_c	diff
0.538	79595	763	672	-6.6	468	501	7.0	519	605	-14.1
1	147946	1107	1046	0.1	694	710	2.3	760	785	-3.2
1.2	177535	1242	1173	0.2	781	794	1.7	855	878	-2.6
1.5	221919	1436	1356	0.2	905	916	1.2	994	1012	-1.7
2	295892	1749	1650	0.1	1099	1106	0.7	1214	1225	-0.9
3	443838	2357	2222	0.1	1466	1469	0.2	1637	1641	-0.3
5	739730	3467	3271	0.0	2122	2121	0.0	2416	2416	0.0

Table 7.2 Effect of buoyancy on Nusselt numbers as a function of air velocity v (m/s) or Reynolds number. Beef surface temperature = 42°C. Nu_f = Nusselt forced. Nu_c = Nusselt combined. diff = percentage of difference.

The Nusselt numbers plot on figures 7.26 to 7.29 and table 7.2 were calculated at a beef surface temperature of 42°C, which offers the highest temperature gradient at the beginning of the chilling process. The buoyancy effects are proportional to the temperature gradient in the wall (beef). Thus, it is expected that at lower beef surface temperatures, buoyancy effects must reduce. Figures 7.37 to 7.39 plot $\overline{Nu_t}$ (forced and combined) vs. air velocity on leg1, loin1 and sho4 as a function of the wall temperature (beef surface). The air velocity was kept at the lowest value (0.538 m/s), where the buoyancy effects are more important. It is seen that buoyancy effects decrease while reducing the beef surface temperature. Table 7.3 shows the percentage of difference between the forced and combined Nusselt

numbers for sections leg1, loin1 and sho1 as a function of the wall temperature. The maximum difference is on sho1 (14.1% at 42°C) but it decreases at 5.3 % when the wall temperature is reduce to 30 °C. Leg1 difference decreases to 1.3% with the same decrease of temperature. Loin1 decreases to 4.6%.

Section	Leg1			Loin1			Sho1		
T. Wall (°C)	Nu_f	Nu_c	diff	Nu_f	Nu_c	Diff	Nu_f	Nu_c	diff
42	763	713	-6.6	468	501	7.0	519	605	-14.1
30	777	767	-1.3	477	499	4.6	528	558	-5.3
20	788	785	-0.5	484	493	1.8	535	550	-2.9
10	799	798	-0.1	490	493	0.6	541	549	-1.5

Table 7.3 Effect of buoyancy on Nusselt numbers as a function of wall temperature (beef surface temperature). Air velocity = 0.538 m/s. Nu_f = Nusselt forced. Nu_c = Nusselt combined. diff = percentage of difference.

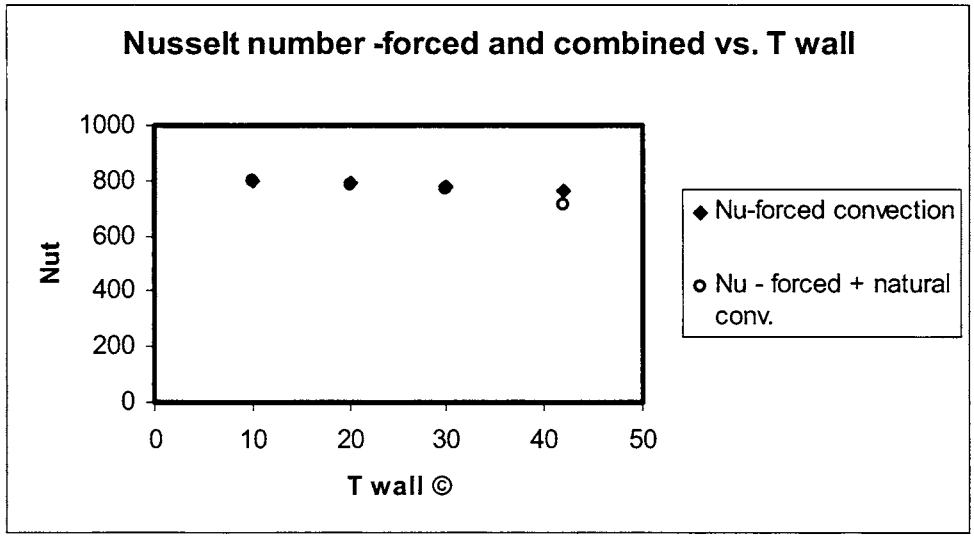


Figure 7.37 Nu_f forced and combined convection vs. air beef surface temperature on section leg1 (figure 7.16) at air velocity = 0.538 m/s, air temperature = 4.9 °C.

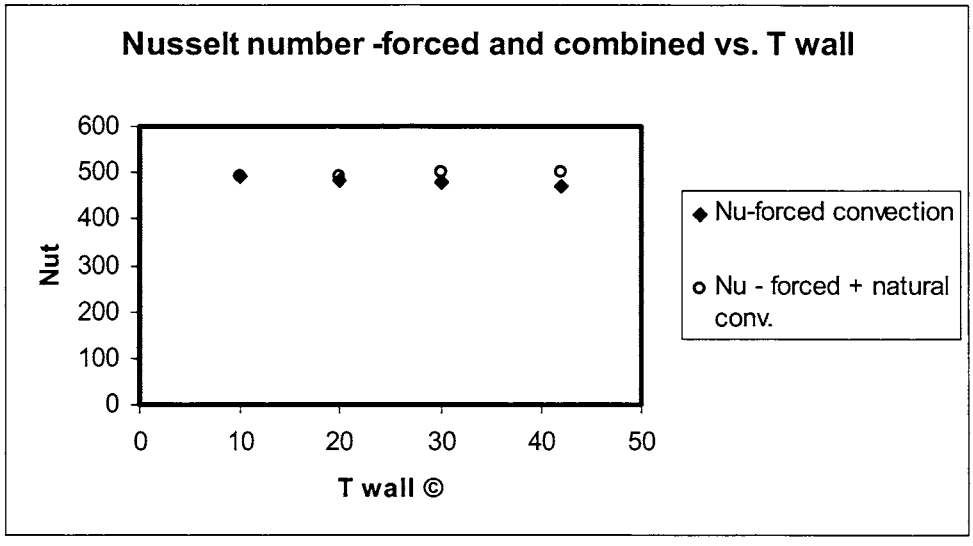


Figure 7.38 Nu_t forced and combined convection vs. air beef surface temperature on section loin1 (figure 7.17) at air velocity = 0.538 m/s, air temperature = 4.9 °C.

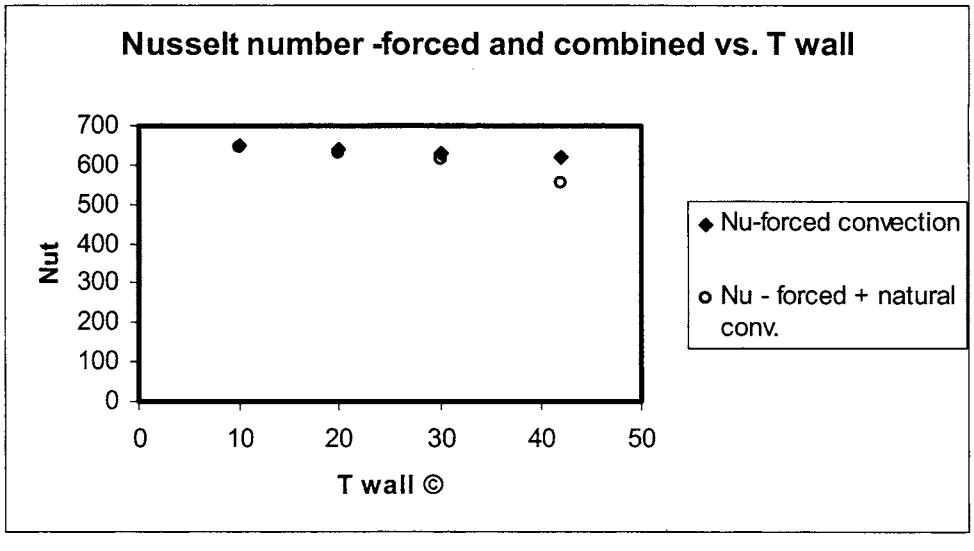


Figure 7.39 Nu_t forced and combined convection vs. air beef surface temperature on section sho4 (figure 7.18) at air velocity = 0.538 m/s, air temperature = 4.9 °C.

The effect of buoyancy on the heat and mass transfer coefficients may be considered unimportant at air velocities higher than 0.538 m/s given that the beef surface temperature when the carcasses enter to the chillier room is usually between 18 °C and 30°C. Buoyancy may be important during the period on the slaughter floor. However, in this first stage there are many unknown variables, like the air velocity and the residence time on the slaughter floor (Davey and Pham, 1997). This period has been usually modelled assuming the air velocity and obtaining the slaughter time by a trial and error method, or assuming the initial meat temperature profile at the beginning of the chilling stage. This stage needs to be studied more deeply in order to be properly modelled. Increased influence of buoyancy is expected when air velocity is decreased to under 0.538 m/s, but this low air velocity range, which may be a research area to explore on the future, was not considered on this work.

7.3 Conclusions

- A Matlab program was developed to automatically construct the 3D geometry of beef carcasses as a function of weight and fatness.
- CFD shows that there are important local variations on the heat and mass transfer coefficients around the beef surface. This is caused by the development of the momentum, heat and mass boundary layer.
- CFD shows that the forced heat and mass transfer coefficients are mainly dependable on Re and Tu.

- The forced heat and mass transfer coefficients around leg, loin and shoulder were correlated as a function of Reynolds and the turbulence intensity.
- The effects of buoyancy at air velocities higher than 0.538 m/s can be ignored. CFD shows that its contribution to the heat and mass transfer coefficient is less than 5.3% at the beginning of the chilling stage.

8. CFD MODELING OF HEAT AND MOISTURE TRANSFER ON A FULL 3- DIMENSIONAL MODEL OF BEEF CARCASSES

The heat and mass transfer processes in beef chilling were simulated using CFD. The 3D geometry of a beef carcass, developed in chapter 7, was used. The numerical method was a modification of the one followed in chapter 6 for the 2D leg model. The water mass transfer inside the meat was conducted in a separate 1-Dimensional mesh and programmed in C++ using a UDF (User defined function) in Fluent. It was possible to simultaneously model the heat and mass transfer process on the meat and on the air phases. However, given the high computational time required on that modelling, a three step method was used to simplify and accelerate the simulation. The steps consist in firstly, conducting a CFD steady simulation only in the air phase, secondly, determining the local heat and mass transfer coefficients, and thirdly, doing the unsteady state simulation only on the meat using the heat and mass transfer coefficients from the previous step. Three runs were completed and the heat load, weight loss, and surface and centre leg, loin and shoulder temperatures were compared with experimental data.

8.1 Unsteady state Simulation

The governing equations for the unsteady state simulations were established in chapter 6. The numerical method applied to solve the set of equations is also similar to the one used in chapter 6. However, given the inability of creating a

fine mesh close to the beef surface on the solid's side, the numerical method was modified. A fine mesh close to the surface is required given that the water diffusion is noticeable only in surface layer approximately 20mm thick. To overcome this problem, the FLUENT macro "DEFINE_ADJUST" was used to program and solve the mass transfer equation on a parallel 1-dimesional mesh. DEFINE_ADJUST is a general-purpose macro that can be used to adjust or modify FLUENT variables that are *not* passed as arguments, modify flow variables (e.g., velocities, pressure) and make different computations. It can also be used to integrate a scalar quantity over a domain and adjust a boundary condition based on the result. The advantage of this macro, and the reason of choosing it, is that a function that is defined using DEFINE_ADJUST executes at every iteration and is called at the beginning of every iteration before transport equations are solved.

The UDF "solve_meat_water" was programmed using the "DEFINE_ADJUST" macro. A separate grid was created as it is seen in figure 8.1. The "normal" grid, that is 3D, is used by Fluent to solve the heat transfer inside the meat (equation 6.15). The "supplementary" grid, which is 1-dimensional, is used by "solve_meat_water" to solve the water diffusion equation. This grid is "attached" to all the face centroids around the beef surface going inside the meat in the perpendicular direction to the face. The mesh goes 24 mm deep inside the meat. 20 elements of different lengths were placed in that distance. Close to the surface, where the concentration gradients are greater, the length of the elements

is smaller. Deeper in the meat, where the mass flux is very small, the length of the elements is bigger. The mesh was defined 1-dimensional because the mass transfer only occurs a few centimetres next to the surface of the meat and hence the mass flux direction is almost perpendicular to the meat surface. A similar procedure was conducted by Karuri and Pham (1999) on a finite difference model.

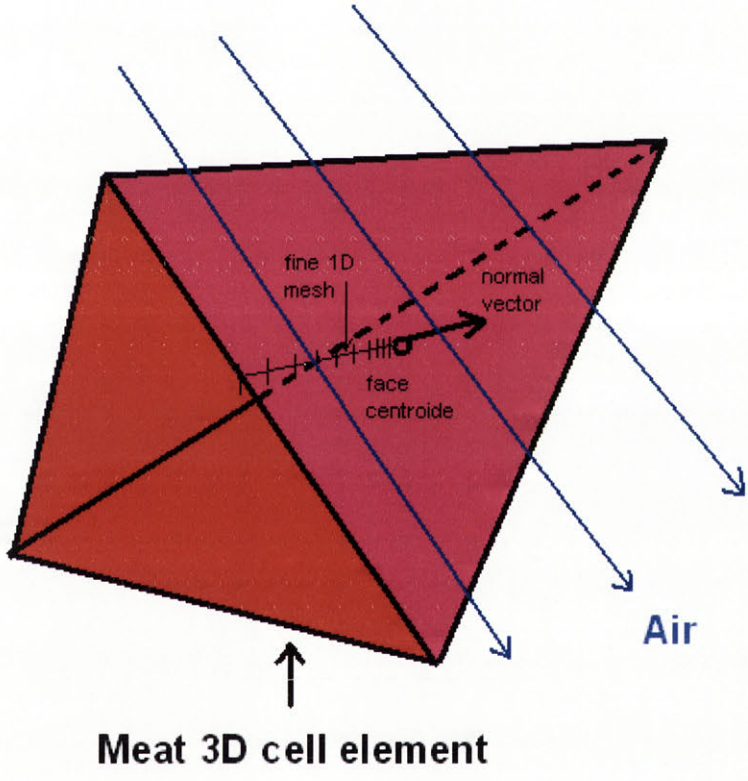


Figure 8.1 Double grid mesh.

At the beginning of the modelling, the water content of each node element is initialized. This information is stored in a UDM (User Defined Memory), which is a FLUENT memory space that must be defined in order to store variables computed by UDFs. The UDF calculates the surface mass flux, reads the time

interval Δt, and solves the water transport equation (6.16) to predict the water profile inside each face cell at the end of the time interval Δt. The procedure is repeated at the beginning of each new time interval. The surface mass flux is calculated from the mass boundary layer on the air interface using 6.17. The finite volume method was used to solve the mass transport equation. Details of the method are explained below.

8.2 Finite Volume method

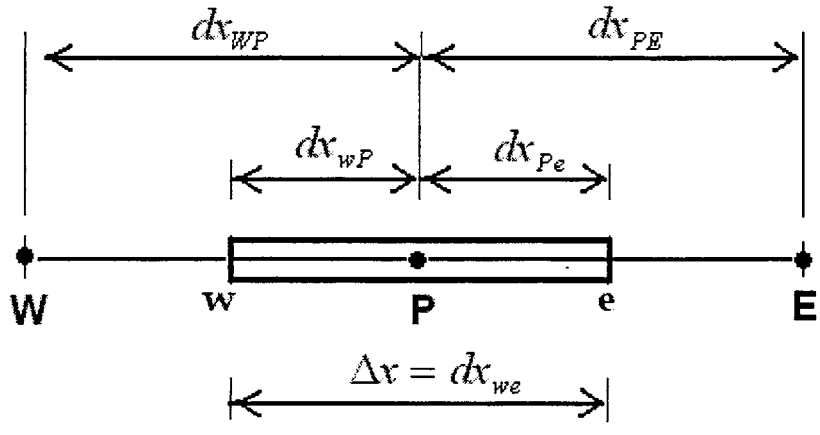
The method is similar to the one developed in chapter 4 but shrinkage was not taking into account. Some shrinkage may occur few millimetres inside the meat, where the water content drops to its lowest value. However, the meat slowly rewets as seen in figure 6.7. Shrinkage was a critical factor to determine the water diffusivity of meat because it affected the full meat samples during the drying period (chapter 4). On the other hand, the water content on the beef surface does not reduce as much as during the drying experiments given that there is constant water migration from the inside of the carcasses. Shrinkage has not been reported on beef carcasses. Therefore, its effect was not taken into account in this model.

Integrating equation 6.16 over a control volume, over a finite time step (Δt) and using the Gauss' divergence theorem, it is transformed in:

$$\int_{CV} \int_{t}^{t+\Delta t} \left(\frac{\partial (Y_m)}{\partial t} dt \right) dV = \int_{t}^{t+\Delta t} \left(\int_{A} n \cdot (D_m \cdot \nabla Y) dA \right) dt \tag{8.1}$$

where Y_m is the water mass fraction and D_m is the water diffusivity (m^2/s). The density (ρ_m), cross-sectional area (A) and the element longitude (Δx) were assumed to be constant (no shrinkage). Considering the one-dimensional control volume in figure 8.2, equation 8.1 can be discretized as:

$$\left(Y_{m,P}^{t+1} - Y_{m,P}^{t}\right)dx = \int_{t}^{t+\Delta t}\left[\left(D_m \cdot \nabla Y_m\right)_e - \left(D_m \cdot \nabla Y_m\right)_w\right]dt \tag{8.2}$$



Nodal point are identified with upper case letter i.e. W, P, E. Boundaries or faces are positioned midway between adjacent nodes and are identified with lower case letters i.e. w, e.

Figure 8.2 Finite volume 1D grid.

For the first volume, on the west face, a convective boundary condition applies:

$$\left(D_m \cdot \rho_m \cdot \nabla Y_m\right)_w = h_m\left(Y_a - Y_s\right) \tag{8.3}$$

For the last volume, on the east face, a constant water composition applies

$$Y_{m,e} = Y_{m,initial} = 0.75 \tag{8.4}$$

Applying the Crank-Nicholson scheme, and approximating the gradients as:

$$\nabla Y_{m,e} = \frac{Y_{m,E} - Y_{m,P}}{dx_{PE}} \tag{8.5}$$

$$\nabla Y_{m,w} = \frac{Y_{m,P} - Y_{m,W}}{dx_{WP}} \tag{8.6}$$

with some algebraic simplifications it is found the flowing set of equations for central volume (i = 2, 3 …, Nv-1), the first volume (i = 1) and the last volume (i = Nv).

8.2.1 Central volume

$$-\frac{1}{2}a_W \cdot Y_{m,W}^{t+1} + a_P \cdot Y_{m,P}^{t+1} - \frac{1}{2}a_E \cdot Y_{m,E}^{t+1} = S_U \tag{8.7}$$

where:

$$a_W = \frac{D_{m,w}^t}{dx_{WP}} \tag{8.8}$$

$$a_E = \frac{D_{m,e}^t}{dx_{PE}} \tag{8.9}$$

$$a_P = a_{Po} + \frac{1}{2}(a_E + a_W) \tag{8.10}$$

$$a_{Po} = \frac{\Delta x}{\Delta t} \tag{8.11}$$

$$S_U = \frac{1}{2}a_E \cdot Y_{m,E}^t + \frac{1}{2}a_W \cdot Y_{m,W}^t + \left[a_{Po} - \frac{1}{2}(a_E + a_W)\right] \cdot Y_{m,P}^t \tag{8.12}$$

8.2.2 First volume

$$a_P \cdot Y_{m,P}^{t+1} - \frac{1}{2}a_E \cdot Y_{m,E}^{t+1} = S_U \tag{8.13}$$

in this case:

$$a_W = 0 \tag{8.14}$$

$$S_U = \frac{1}{2} a_E \cdot Y^t_{m,E} + \left[a_{Po} - \frac{1}{2}(a_E) \right] \cdot Y^t_{m,P} + b \tag{8.15}$$

$$b = \frac{h_m (Y_a - Y_s)}{\rho_m} \tag{8.16}$$

8.2.3 Last volume

$$-\frac{1}{2} a_W \cdot Y^{t+1}_{m,W} + a_P \cdot Y^{t+1}_{m,P} = S_U \tag{8.17}$$

where:

$$S_U = \frac{1}{2} a_W \cdot Y^t_{m,W} + \left[a_{Po} - \frac{1}{2}(a_E + a_W) \right] \cdot Y^t_{m,P} + a_E \cdot Y_{m,initial} \tag{8.18}$$

8.3 Details of numerical solution

The numerical method explained in chapter 6 for the 2D model was modified. The pressure – velocity coupling method used was SIMPLE which gave better results than PISO, even though PISO is recommended for unsteady problems.

UDSs (User Defined Scalar) were defined on the 2D model in order to solve the heat and mass transfer equations in the meat as it is established in section 6.3. The meat was set as a fluid because UDSs can only be defined in a fluid phase. However, using UDSs is unnecessary in the 3D model given that the mass transfer equation was solved using "solve_meat_water" on the parallel grid. The meat can be defined as a solid and the heat transfer equation in the meat (equation 6.15) may be solved by the standard FLUENT procedure.

Boundary conditions at the air-meat interface change with time and are dependent on the values of the field variables. However, as it was established in chapter 6, the FLUENT default menu allows only fixed boundary conditions. Thus, in order to ensure continuity of heat and mass flux on the interface, the boundary conditions were modelled using UDFs (User Defined Functions). Appendix A5 shows the programmed UDFs. The collection of the main UDFs used to solve the 3D problem are:

5. DEFINE_ADJUST(solve_meatwater, domain): Solves the mass transfer equation in the meat using the finite volume method described in section 8.2.
6. DEFINE_PROFILE(mass_flux_meat, tm, j): Calculate the water flux leaving the meat in the interface caused by the mass convection given the development of the mass boundary layer. The calculation is made with the equation 6.17. It is used as a boundary condition of the programmed finite volume method (section 8.2) using the UDF "solve_meat_water".
7. DEFINE_PROFILE (heat_flux_meat, tm, j): Calculate the heat flux leaving the meat in the interface caused by convection, radiation and evaporation according to the equation 6.18. It is used as a boundary condition of the heat transfer equation in the meat.
8. DEFINE_PROFILE(temperature_air_interf, t, n) : makes the air interface temperature equal to the meat interface temperature (Equation 6.8). It is used as a boundary condition of the air phase energy equation.

9. DEFINE_PROFILE(water_air_interf, t, n): Calculate the air water mass concentration on the interface with the same procedure established in section 6.3.

The UDFs "mass_flux_meat" and "heat_flux_meat" can be used with a coarse mesh or a very fine mesh on the interface. In the former case, the standard wall functions approach is applied as it is established in section 6.2.5.1. This is only valid for fully turbulent flows. In the latter case, the beef surface mass and heat fluxes are calculated with equations (6.17) and (6.21) respectively, and the enhanced wall treatment is used. This is the recommended method for wall attached fluids and low Reynolds number. This is the standard method used in the present work. Figure 8.3 compares the heat transfer coefficient calculated using both approaches on the experimental run 18 conducted by Davey (1998). It is seen that the standard wall function approach is on average underestimating the heat transfer coefficients comparing with the enhanced wall function.

This method allows solving the heat and mass transfer equations inside the meat simultaneously with the continuity, momentum, heat, mass transfer, kinetic energy and dissipation energy rate equations in the air phase. Unfortunately, the computation time required is too high. Modelling the first 50 minutes of the slaughter floor period took 5 days. Figure 8.4 shows the temperature profile in the meat and the air phases of run 14 after 50 minutes on the slaughter floor. An alternative method to accelerate the computation is explained in section 8.4.

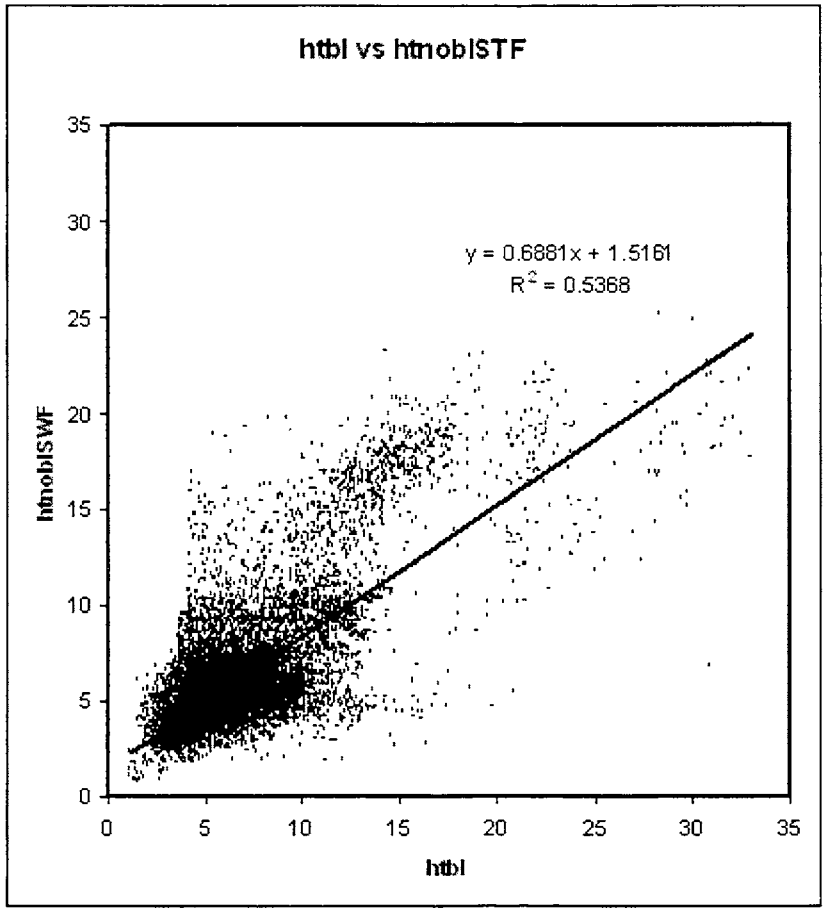


Figure 8.3 Comparison of heat transfer coefficients obtained with the enhanced wall treatment (htbl) vs. heat transfer coefficients using the standard wall functions (htnoblSWF).

8.4 3D CFD simulations using a three step method

In order to accelerate the simulation, the three step method proposed by Hu and Sun (2001) was followed. On the first step, a steady state simulation of the flow field was conducted. On the second step, the local heat and mass transfer coefficients are calculated cell by cell around the beef surface with the information obtained in step 1. Finally, the third step consists of the simultaneous

heat and mass transfer process simulation in the meat carcass only. Appendix A6 contains the UDFs that executes the third step.

This process is valid under the assumption that the heat and mass transfer coefficients are constant during the chilling period. It was found in section 7.2.1 that the forced heat and mass transfer coefficient depends on Reynolds number and the turbulent intensity. If the buoyancy effects are taken into account, the transfer coefficients would also depend on the wall temperature, air temperature and water percentage composition (or humidity). However, the effects of buoyancy on the heat and mass transfer coefficients for air velocities higher than 0.538 m/s may be considered unimportant as it was established on section 7.2.2. Thus, the heat and mass transfer coefficients can be considered constant if the air velocity and the turbulent intensity are constant.

The experimental runs 14, 18 and 32, conducted by Davey (1998) were modelled via CFD with the accelerated technique. In these experiments a wind tunnel apparatus was set up in an experimental chillier room in an industrial beef processing plant. For each trial, the chillier air temperature and the air velocity in the tunnel was set and measured. Each beef side was taken directly off the chain after weighting and washing in hot water, which was usually about 1 hour after slaughter (slaughter floor period). Once inside the chillier, thermocouples were inserted into the leg, loin and shoulder, and the side was moved into the tunnel. Each side was then chilled for 20-25 hours before unloading. Although the air

temperature was set to control at a constant temperature for each run, some variation was apparent during most trials.

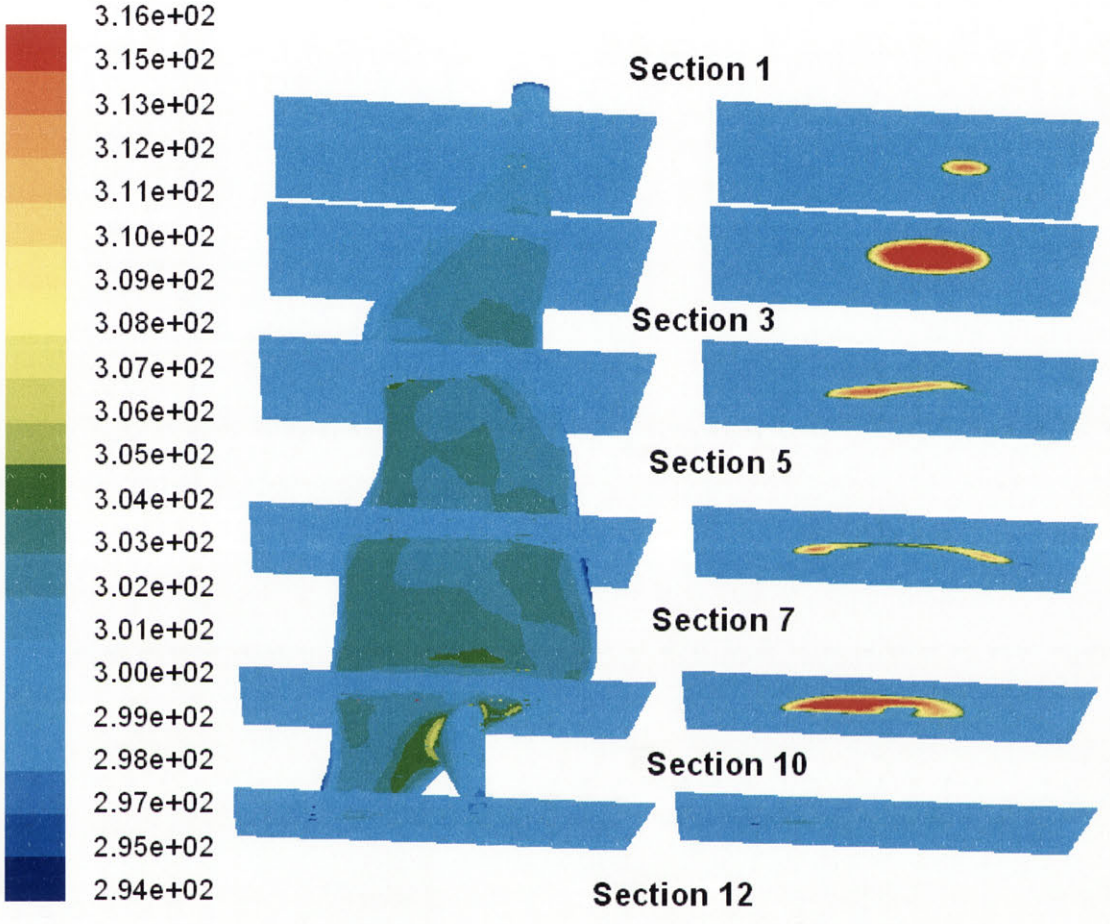


Figure 8.4 Temperature profile around the beef surface and cross-sectional areas on the air meat phases on run 14.

The air flow over the beef side was achieved using an extraction fan, located at the base of the tunnel. A variable speed controller allowed fan speed adjustment. The air flow velocity was measured in the tunnel while there was no product present. An adjusted air velocity was calculated by assuming that the mass flow rate of air throughout the tunnel was unchanged by the beef side (Davey and

Pham, 1997). The air velocity was assumed constant during the chilling period. Air velocities on runs 14, 18 and 32 are 0.99, 0.538 and 0.69 respectively.

The turbulent intensity was not measured in these experiments. Therefore its value must be assumed. Turbulence intensities have been measured by Daudin and Kondjoyan (1991) in an empty chilling room: the value was 38% on average and ranged from 22% to 60% depending on the location on the room. Kondjoyan and Daudin (1997) used a closed circuit wind tunnel to measure the heat and mass transfer coefficients in pork hindquarters. In the clear test wind tunnel chamber the turbulence intensity was less than 1.3%. To promote turbulence, perforated plates were located normal to the flow. A maximum turbulence intensity of 30% was obtained with the plates. Harris *et al.* (2003) used an open wind tunnel to deliver controlled longitudinal air flow conditions over a fibre glass lamb carcass model. Several methods for turbulence generation within the wind tunnel were tried. Only limited success was had generating turbulence achieving a peak value of 8% with a coarse mesh. The turbulence with standard strengtheners was just 1.8%. It seems that turbulent intensity inside typically designed wind tunnels is low.

Davey's wind tunnel was placed in an industrial chillier room where there are usually high turbulent intensities. However, it is expected that inside the tunnel, by effect of the fan, the turbulence intensity should be somewhere between a typical wind tunnel (low turbulence intensity) and an industrial chillier room

(high turbulence intensity). A turbulence intensity of 10% was used for modelling. With this value, good agreement with the experimental heat load was found on runs 14, 18 and 35.

The local heat and mass transfer coefficients in the chilling period were determined via CFD. These transfer coefficients cannot be used to model the slaughter floor period given that the flow pattern is different. Thus, steady state CFD modelling of the slaughter floor was conducted to determine the local heat and mass transfer coefficients on this stage.

Slaughter floor air conditions were measured as 25°C and 63% relative humidity. However, neither the air velocity nor the residence time on the slaughter floor were accurately known as both showed considerable variation (Davey and Pham, 1997). On average the beef sides spent about one hour on the slaughter floor, but hold-ups and chain downtime may extend this to up to two hours for a particular side. Therefore, an air velocity of 0.5m/s was assumed by Davey and Pham (1997). A trial and error technique was used to find the time on the slaughter floor. The total heat lost after slaughter, Q_{Max}, was calculated for each run assuming the beef side would equilibrate fully to the air temperature in the chillier at the end of the chilling period:

$$Q_{Max} = Mc_p \left(T_{Death} - T_a\right) \tag{8.19}$$

The heat lost in the chillier, Q_{Exp}, was then calculated using the FD model (Davey and Pham, 1997) by integrating the experimental heat load versus time curve,

and the heat that had been lost on the slaughter floor found by difference. The length of time on the slaughter floor was completed when the total heat remaining in the finite difference simulation, Q_{FD}, was within 1% of the total experimental heat lost during chilling, Q_{Exp}. This slaughter time was used on the CFD simulations.

The 3 step method permits conducting the simulation of the first 20 hours of chilling plus the slaughter period in 5 days using a Pentium 4 of 2.5 GHz. The method is accurate if the heat and mass transfer coefficients are constant during each of the stages (slaughter floor and chilling). This assumption is basically true if the air velocity and turbulent intensity are constant and if the buoyancy effects may be ignored.

The unsteady state simulation conducted took in account the effects of radiation (equations 6.18 and 6.20). In chapter 5 it was established that at low air velocities, on the detachment point of the boundary layer on a cylinder, the heat transfer coefficient decreases and the mass transfer coefficient increases by effect of radiation. This phenomena has been studied before. Cess (1962) found that the local Nusselt number (non-dimensional heat transfer coefficient) in a laminar boundary layer of a fluid along a flat plate decreases by effect of radiation. The effect of radiation in the heat and mass transfer coefficients can not be quantified using the three steps method. However, this effect may only be important at very low air velocities and high temperature gradients.

8.5 Results

8.5.1 Heat load

Table 8.1 shows the initial and boundary conditions of runs 14, 18 and 32. Figure 8.4 shows the heat load for the three runs and compare them with the Finite difference method of Davey and Pham (1997) and the Finite Element method of Davey and Pham (2000). It can be seen that the predicted value by the three models are in good agreement with the experimental data.

	Run 14	**Run 18**	**Run 32**
Weight (kg)	113.5	108	140
Fatness (mm)	14	6	12
Fatness grade	4	2	3
Air relative humidity (%RH)	98.5	98	99.9
Air Temperature (C)	6.02	4.88	6.57
Air tunnel velocity (m/s)	0.99	0.538	0.69
Slaughter floor time (m)	90	120	85
Initial beef temperature (C)	42.4	42.4	42.4

Table 8.1 Initial and boundary conditions runs 14, 18 and 32.

8.5.2 Temperature

Figure 8.5 shows the leg centre and surface temperature profiles. CFD predictions are in very good agreement with the experimental data. The CFD centre temperature prediction is quite well during the 20 hours of chilling. A slight under-prediction during the peak may be due to the use of an average initial temperature at slaughter, value that can change between particular runs. The surface leg temperature is very well predicted on runs 14 and 32 matching almost perfectly the experimental values. Bigger difference can be seen on run 18 where the surface leg temperature is under predicted by up to 3.5 C during the first 12 hours of chilling. The following 8 hours are quite well predicted.

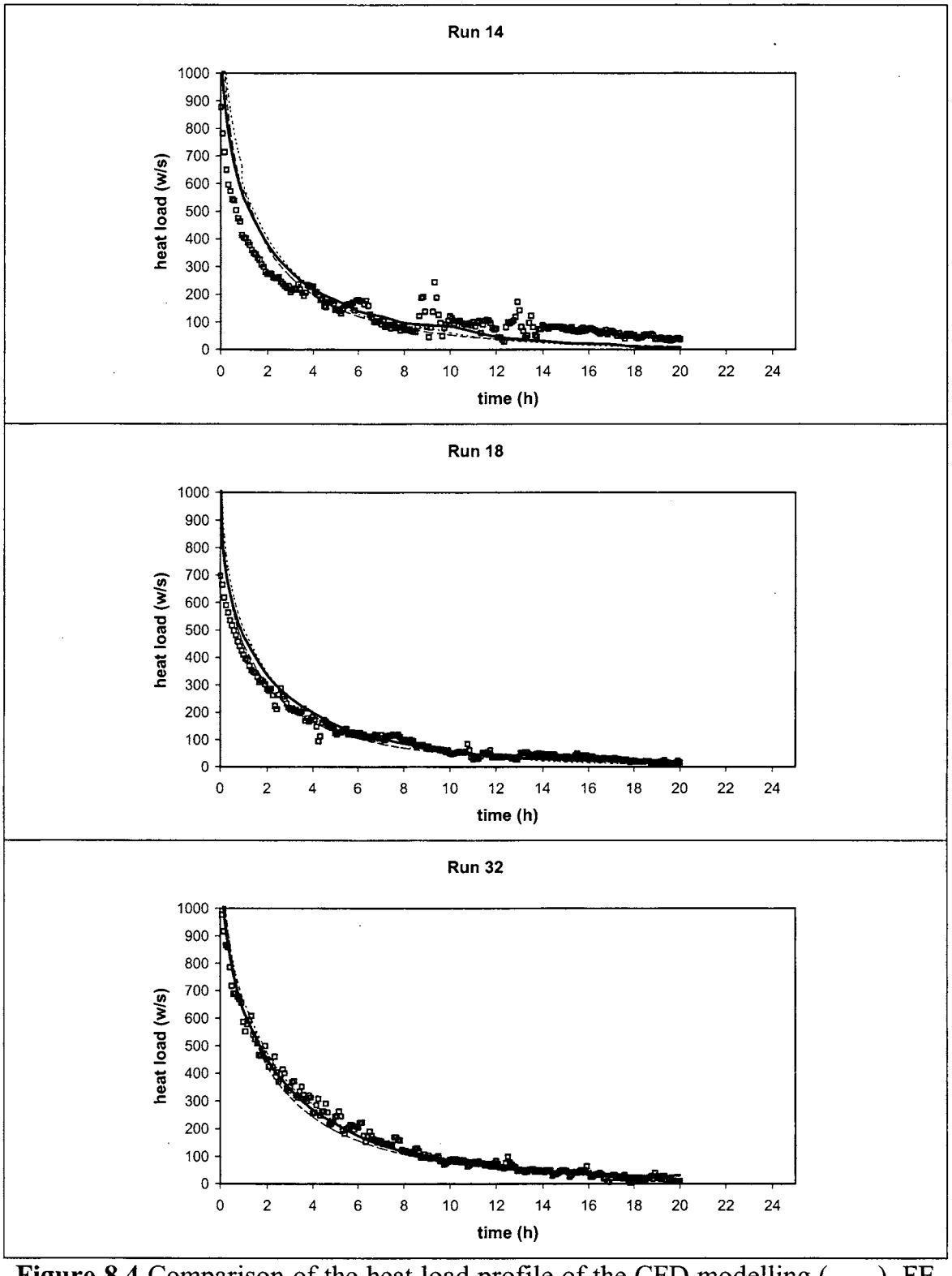


Figure 8.4 Comparison of the heat load profile of the CFD modelling (——), FE model of Davey and Pham (2000) (__ __), FD model of Davey and Pham (1997) (------) and experiments for runs 14, 18 and 32.

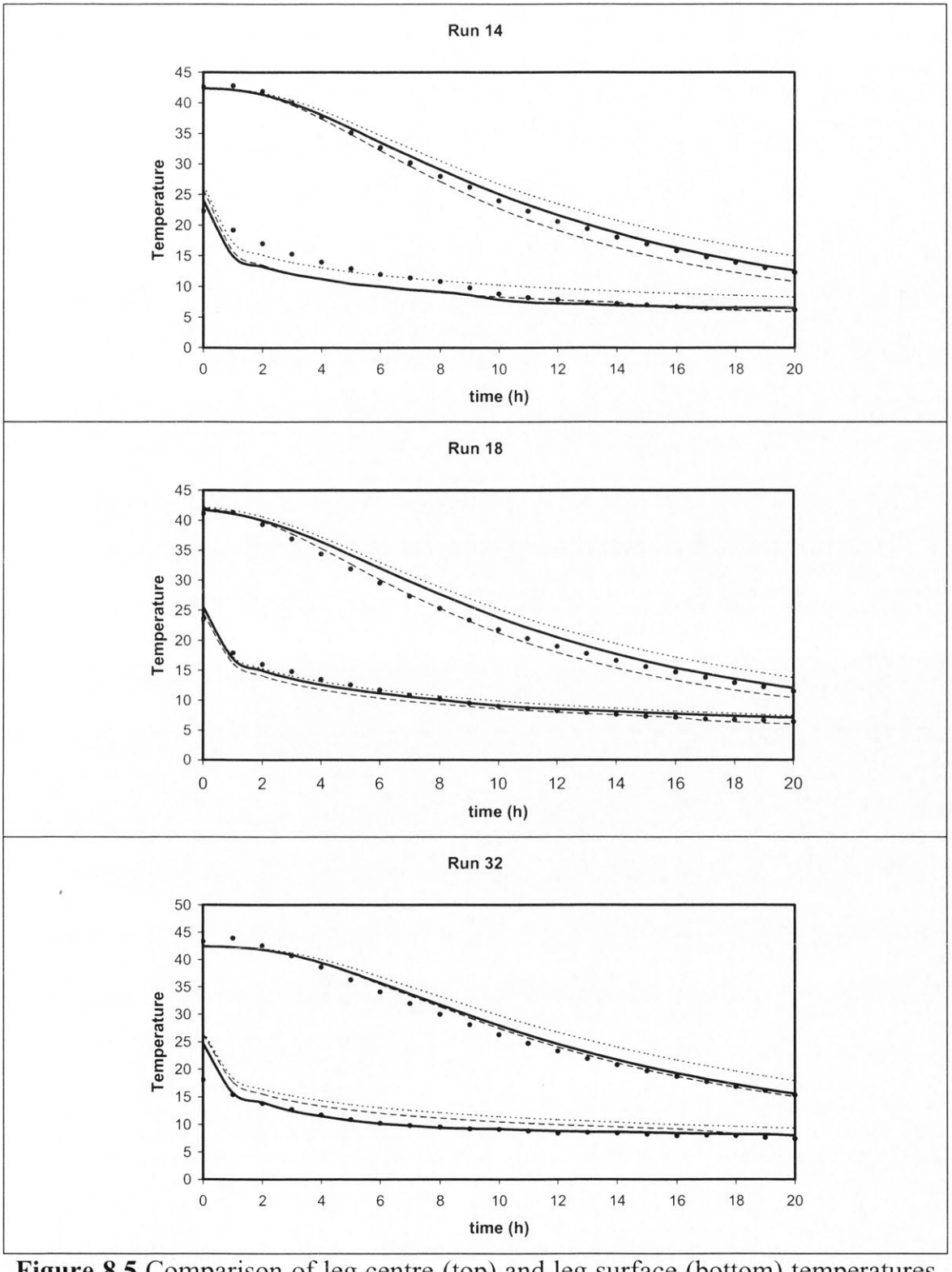


Figure 8.5 Comparison of leg centre (top) and leg surface (bottom) temperatures calculated with the CFD modelling (——), FE model of Davey and Pham (2000) (__ __), FD model of Davey and Pham (1997) (------) and experiments for run 14, 18 and 32.

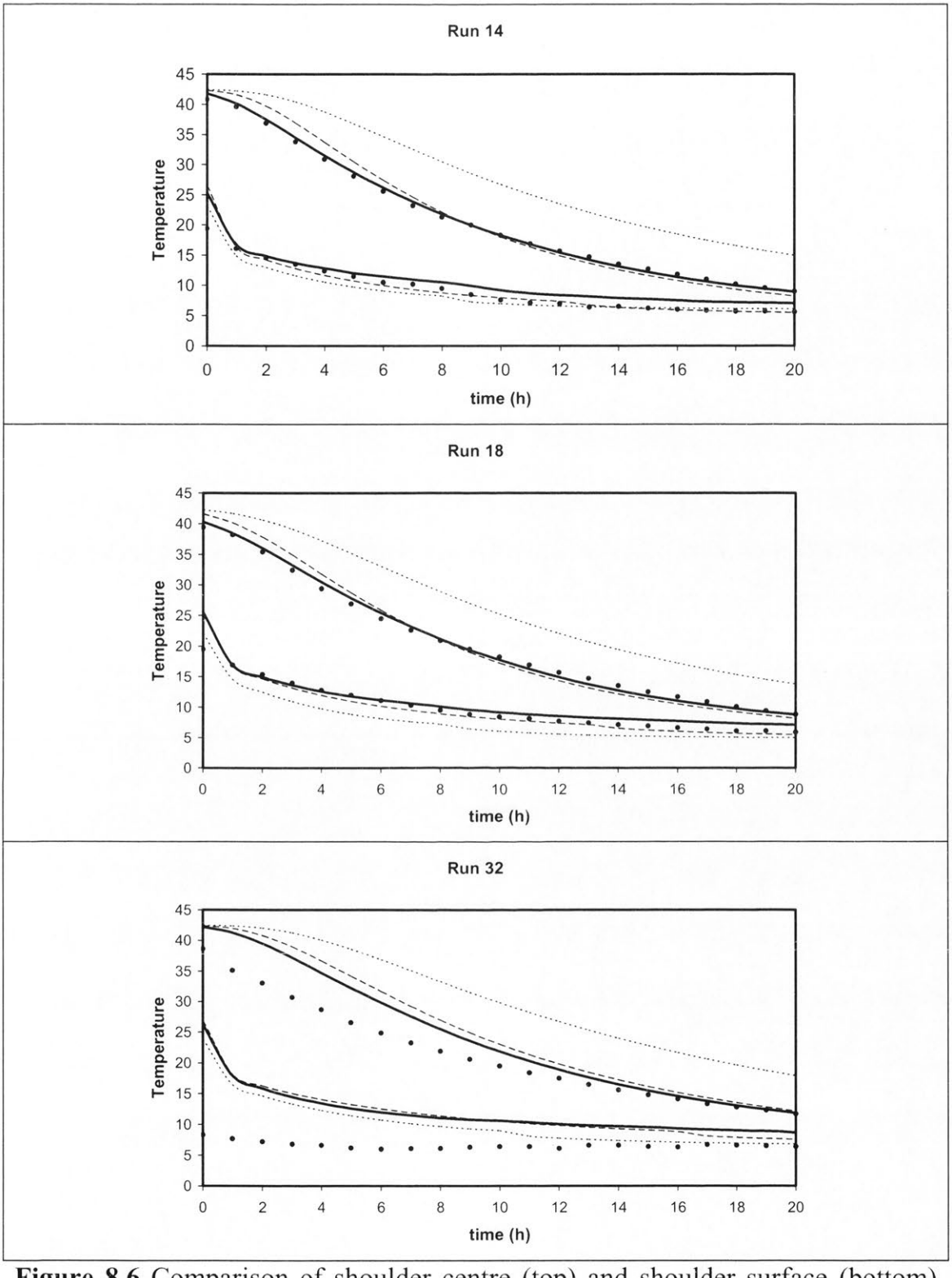


Figure 8.6 Comparison of shoulder centre (top) and shoulder surface (bottom) temperatures calculated with the CFD modelling (——), FE model of Davey and Pham (2000) (__ __), FD model of Davey and Pham (1997) (------) and experiments for run 14, 18 and 32.

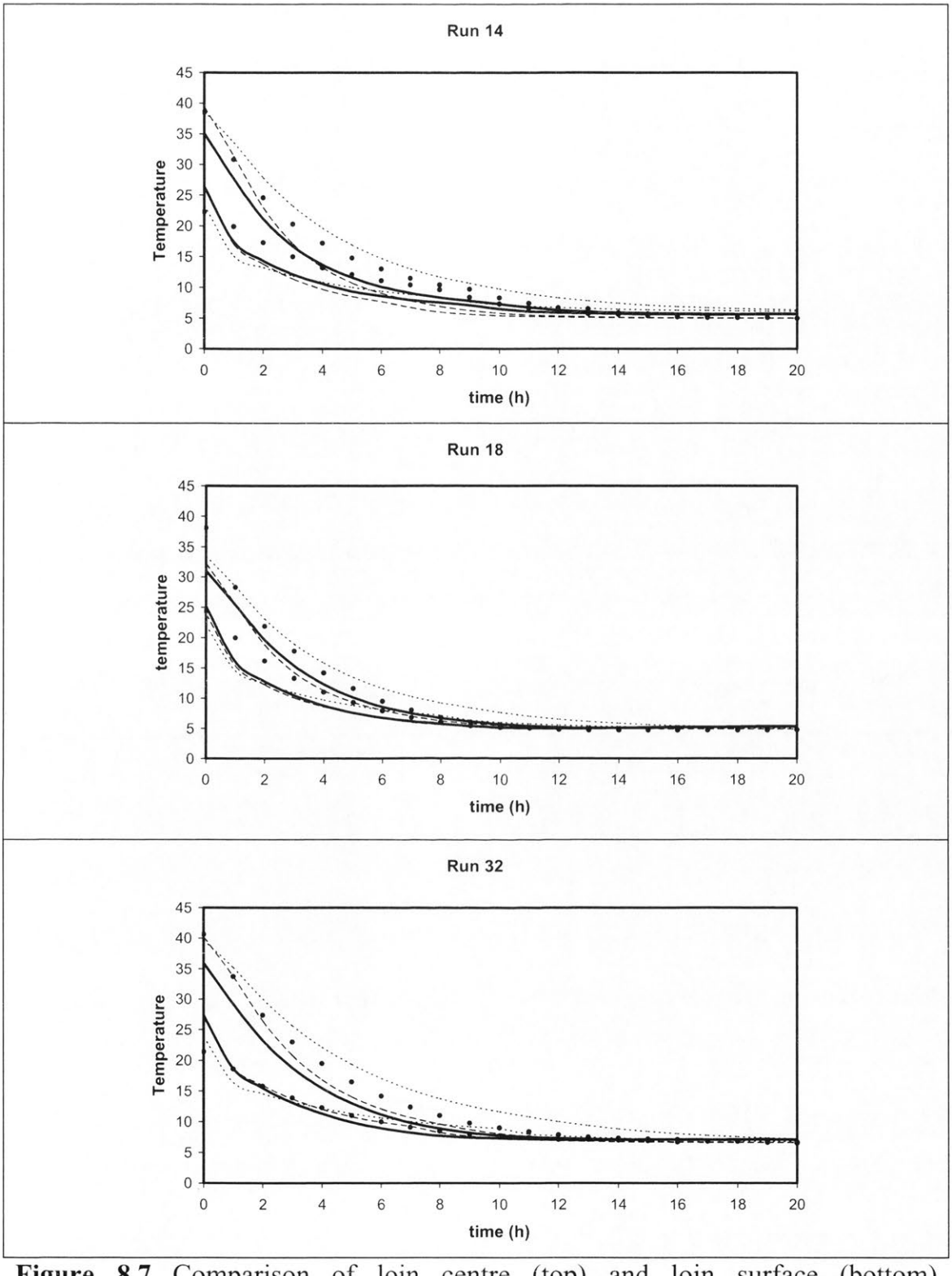


Figure 8.7 Comparison of loin centre (top) and loin surface (bottom) temperatures calculated with the CFD modelling (——), FE model of Davey and Pham (2000) (__ __), FD model of Davey and Pham (1997) (------) and experiments for run 14, 18 and 32.

The FE leg centre temperature prediction is good during the first 10 hours but tends to under-estimate the centre temperature by a few degrees by the end of the chilling period. The surface temperature is under or over predicted in different runs (runs 18 and 32 respectively). FE predictions are not as good as the CFD ones. The FD model exhibits the least agreement. It over estimates the centre leg temperatures in all cases.

Figure 8.6 shows the shoulder centre and surface temperature profiles. CFD predicts the leg centre temperature on runs 14 and 18 better than the other models. The surface temperature is well predicted in run 18 during the first 10 hours but slightly over predicted (no more than 1.5 C) toward the end of the 20 hours of modelling. In run 14, the surface shoulder temperature is slightly over predicted by no more than 1.5 C after 5 hours.

Experimental temperatures in run 32 are lower comparing with all the models prediction, especially the surface temperature that exhibits an abnormal behaviour starting at a very low value (8.3 C). That indicates that there could have been a malfunction in the thermocouples in this particular data set.

FE centre temperature is over predicted at the start of the chilling and under predicted towards the end. The surface temperature is slightly better predicted with the FE than the CFD model. Both centre and surface leg temperatures are

incorrectly predicted by the FD model maybe caused by the incorrect geometrical approach of the shoulder as a slab.

Figure 8.7 shows the centre and surface loin temperatures. The CFD and FE models are under predicting both centre and surface temperatures during the first 10 hours of chilling, but good predictions are obtained after this period. The FD model is over predicting the centre temperature. The FD surface temperature is under predicted during the first 10 hours but well predicted from 10 to 20 hours.

CFD and FE models under predict the centre loin temperature comparing with the FD because the geometry. FD assumes that the loin is a slab while CFD and FE are using the "real" geometrical shape (Figure 8.8), where the loin centre temperature is affected by the heat transfer through three walls.

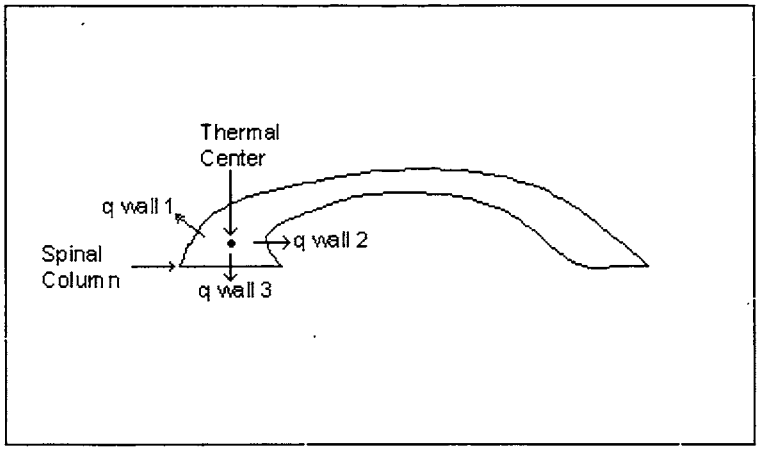


Figure 8.8 Heat transfer process in the loin and its effects in the loin thermal centre.

In general, the CFD model obtains the best temperature predictions followed by the FE model. That can be seen in table 8.2 that shows the average root mean square percentage error (%RMS) in the centre and surface temperature for leg, loin and shoulder using the CFD, FE and FD models. The lowest %RMS is obtained for CFD in all the cases. The FD model temperature predictions are inaccurate.

	%RMS CFD	**%RMS FE**	**%RMS FD**
Leg Centre	0.620	0.726	1.676
Leg Surface	1.259	1.675	2.644
Loin Centre	1.601	1.820	2.615
Loin Surface	1.582	1.868	1.729
Shoulder Centre	1.045	1.548	5.413
Shoulder Surface	5.272	6.886	5.503

Table 8.2 Average root mean square percentage error (%RMS) of the centre and surface temperatures of leg, loin and shoulder.

Figures 8.9 to 8.11 shows the relative value residual (RVR) vs. time and the cumulative residual distribution of the three models (CFD, FE and FD) for the centre leg, loin and shoulder temperatures respectively. The best residual distribution is exhibited by the CFD model (especially for the leg and shoulder). With those figures and table 8.2 It is clearly seen how the temperature predictions improve from the FD to the FE model because the better geometrical representation, and from the FE to the CFD model because of the improvement on the local heat and mass transfer coefficients.

CFD modelling can also predict local variation on the surface temperature. Figure 8.12 plots the temperature profiles on different positions around the leg, loin and shoulder for run 14. It is seen that temperature differences of up to 4.5°C may be found around the shoulder. Figures 8.13 and 8.14 show the chosen positions on the leg, loin and shoulder.

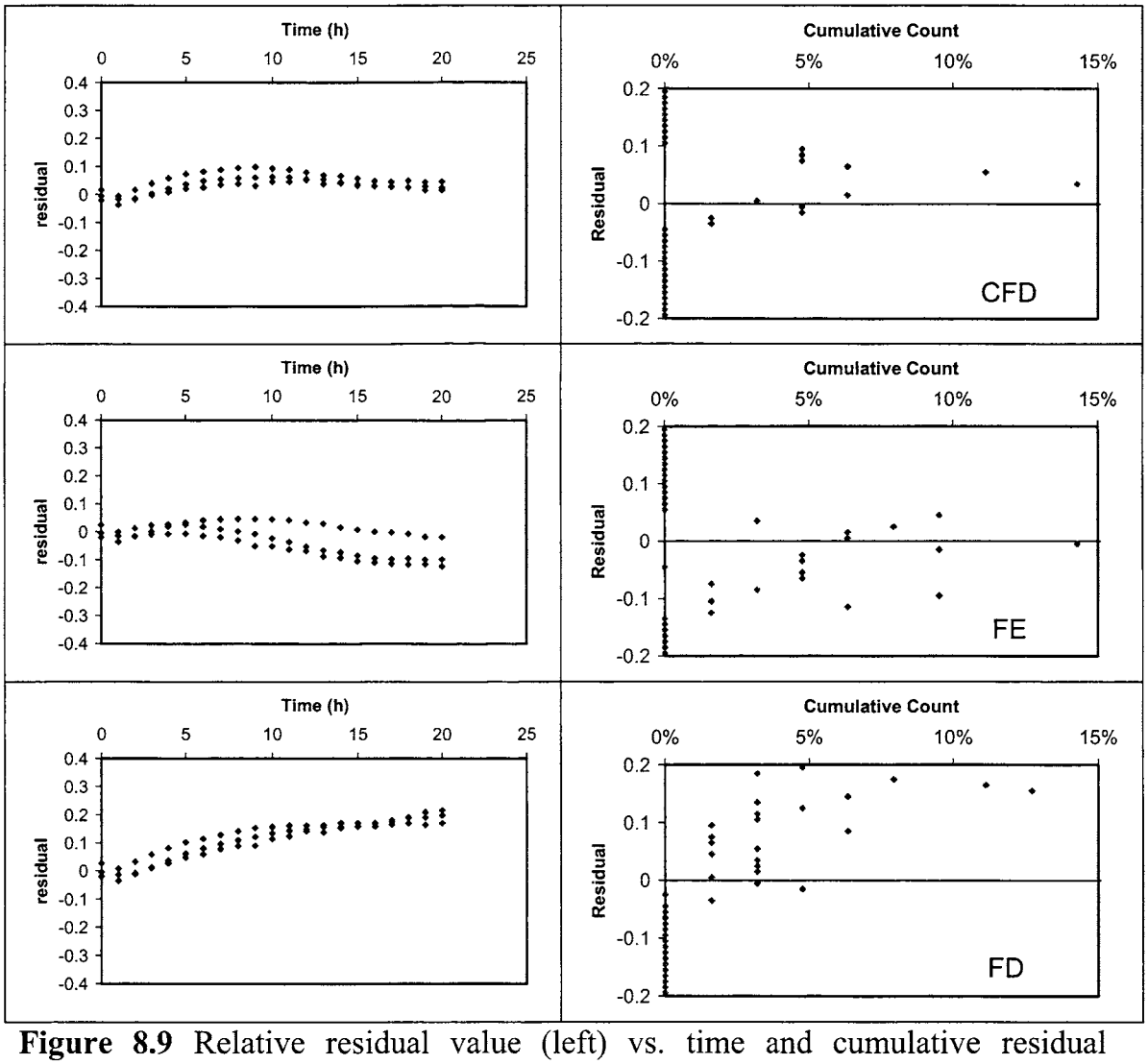


Figure 8.9 Relative residual value (left) vs. time and cumulative residual distribution (right) of the leg centre temperature for CFD, FE and FD models.

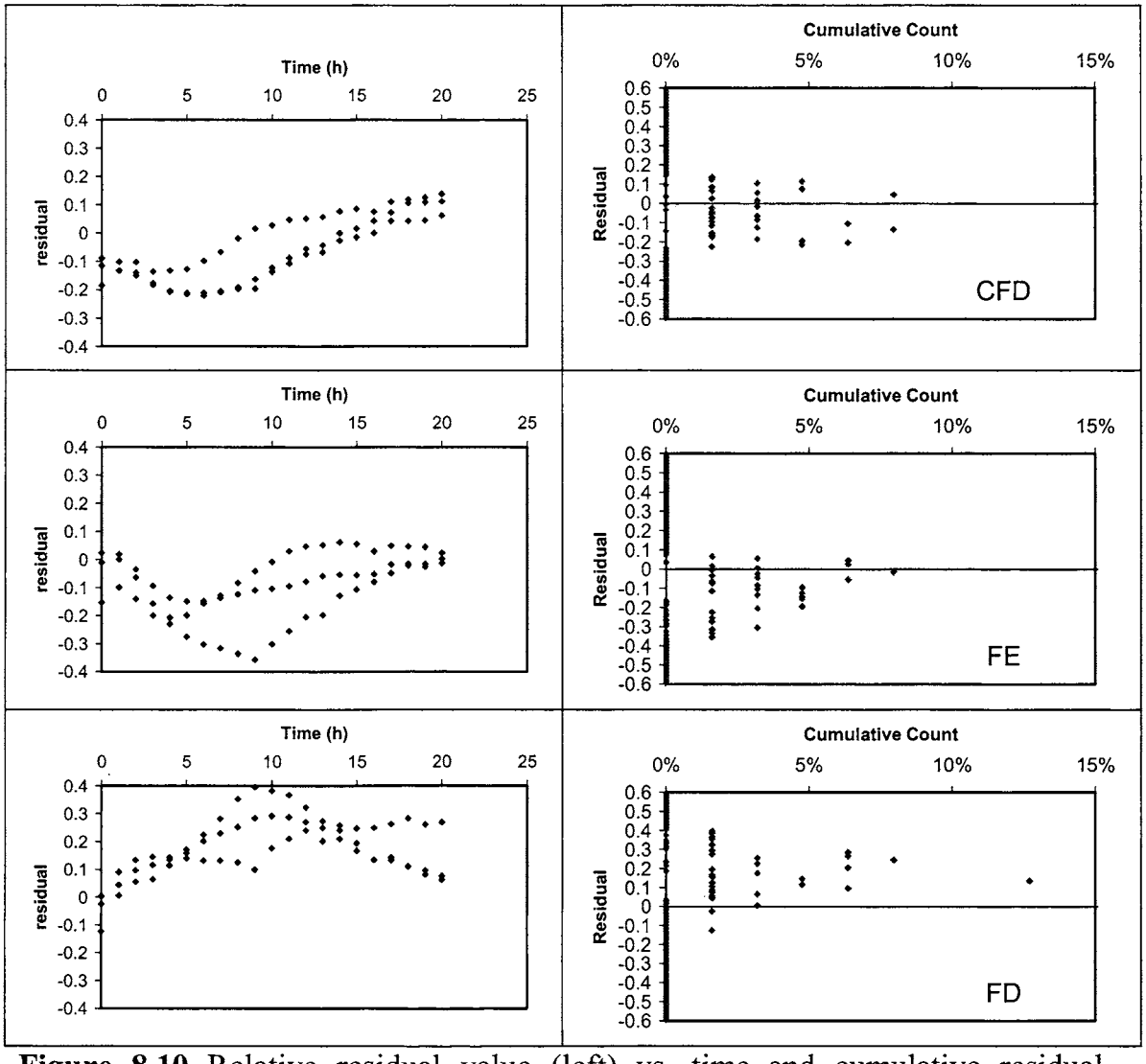


Figure 8.10 Relative residual value (left) vs. time and cumulative residual distribution (right) of the leg loin centre temperature for CFD, FE and FD models.

8.5.3 Weight loss

Figure 8.15 shows the weight loss calculated by the CFD, FE and FD models. CFD is predicting the weight loss of run 18 quite well but over predicting it in runs 14 and 32. The FE and FD models over predict the weight loss in runs 14 and 32 but under predict it in run 18.

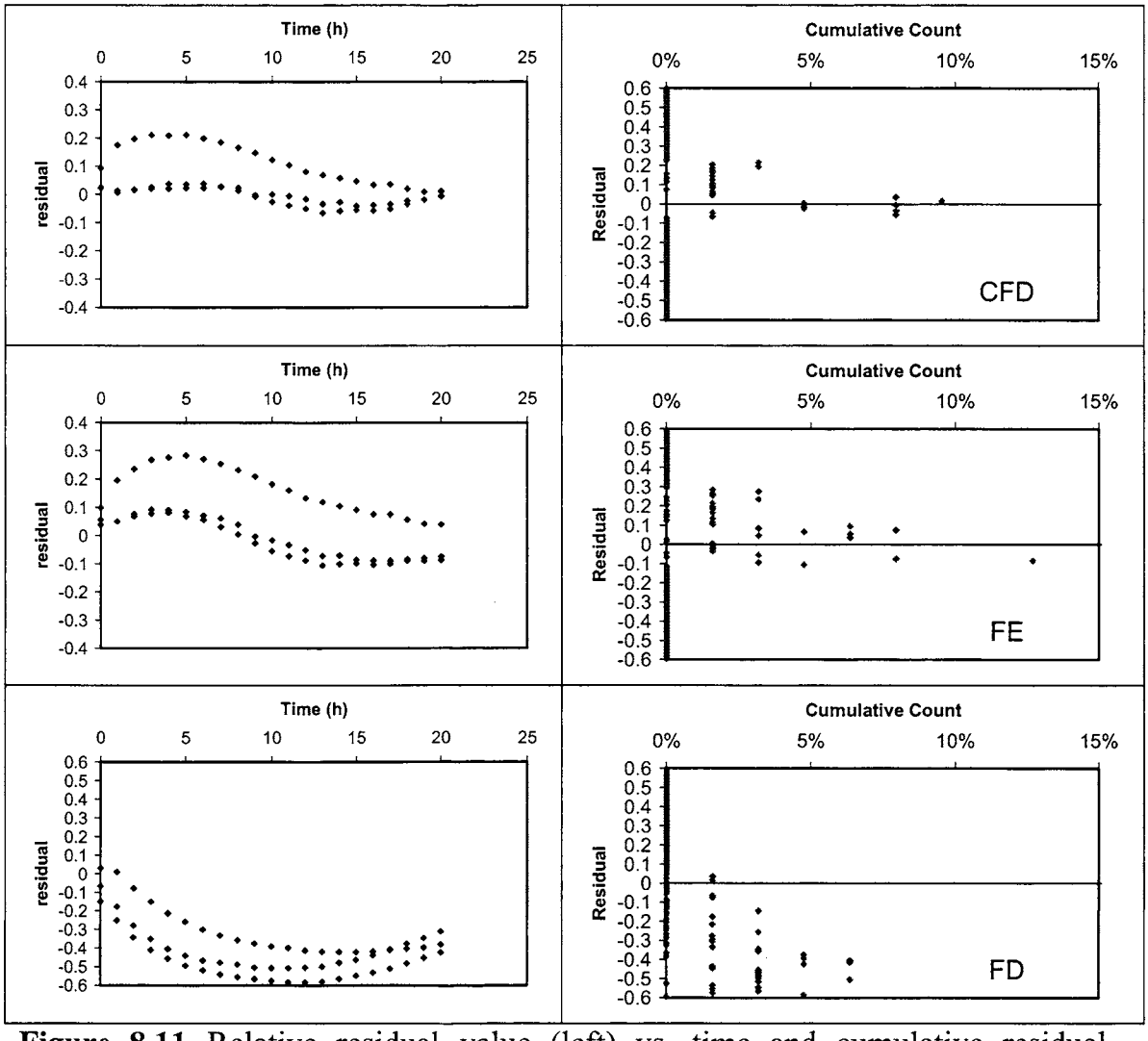


Figure 8.11 Relative residual value (left) vs. time and cumulative residual distribution (right) of the shoulder centre temperature for CFD, FE and FD models.

The CFD over predicting may be caused by the external fat that acts as a resistance to the mass transfer. For run 18 the fat cover is just 6 mm while for runs 14 and 32 it is 14 and 12 mm respectively. However, specific experiments should be design to determine the cause of this behaviour. It would be advisable to conduct evaporation experiments of meat on a simpler geometry like a cylinder, following the weight loss and recording the water activity on the

surface; similar to the experiments on chapter 5 that were designed to measure the local heat and mass transfer coefficients around a cylinder.

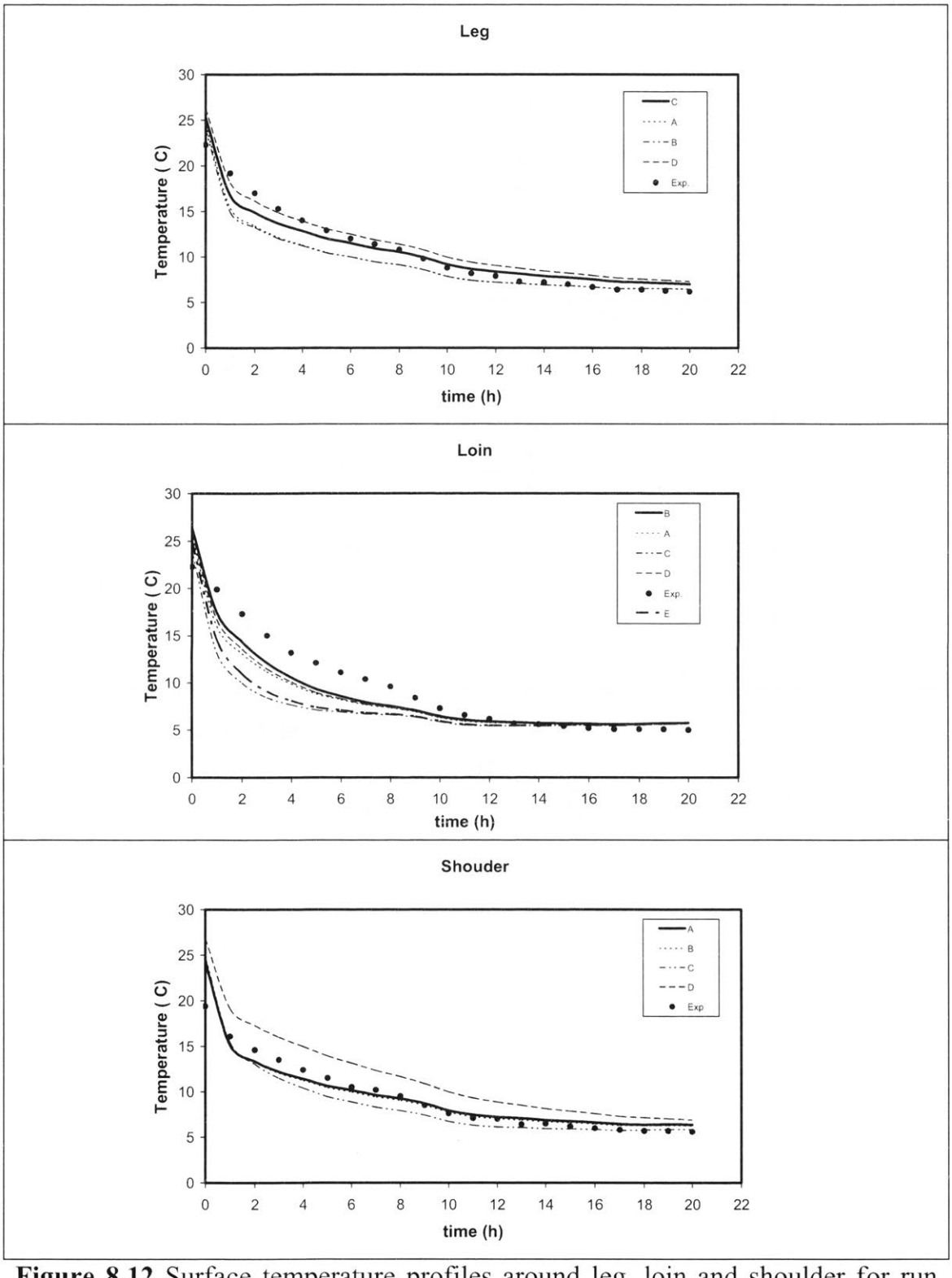


Figure 8.12 Surface temperature profiles around leg, loin and shoulder for run 14.

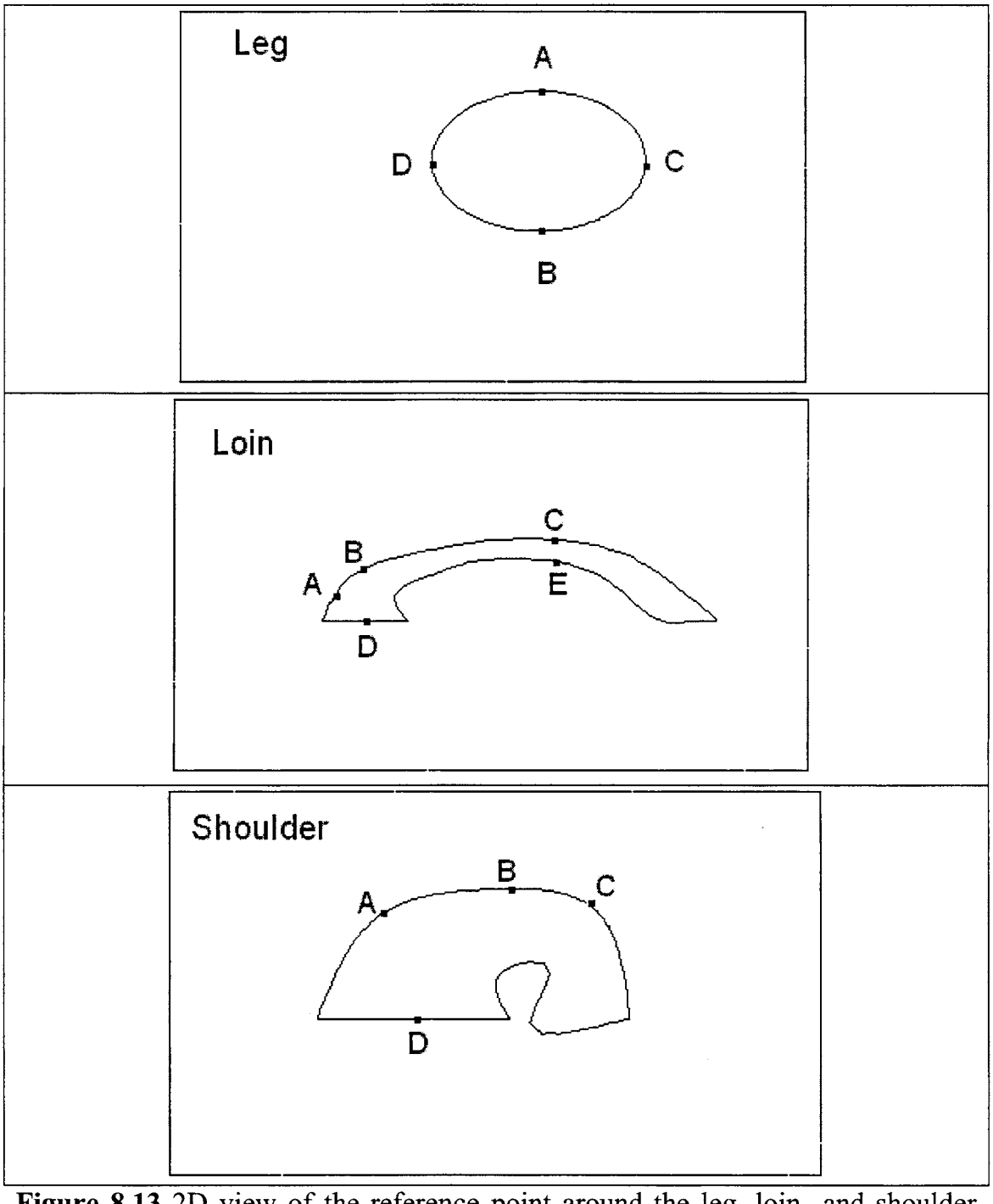


Figure 8.13 2D view of the reference point around the leg, loin and shoulder surfaces.

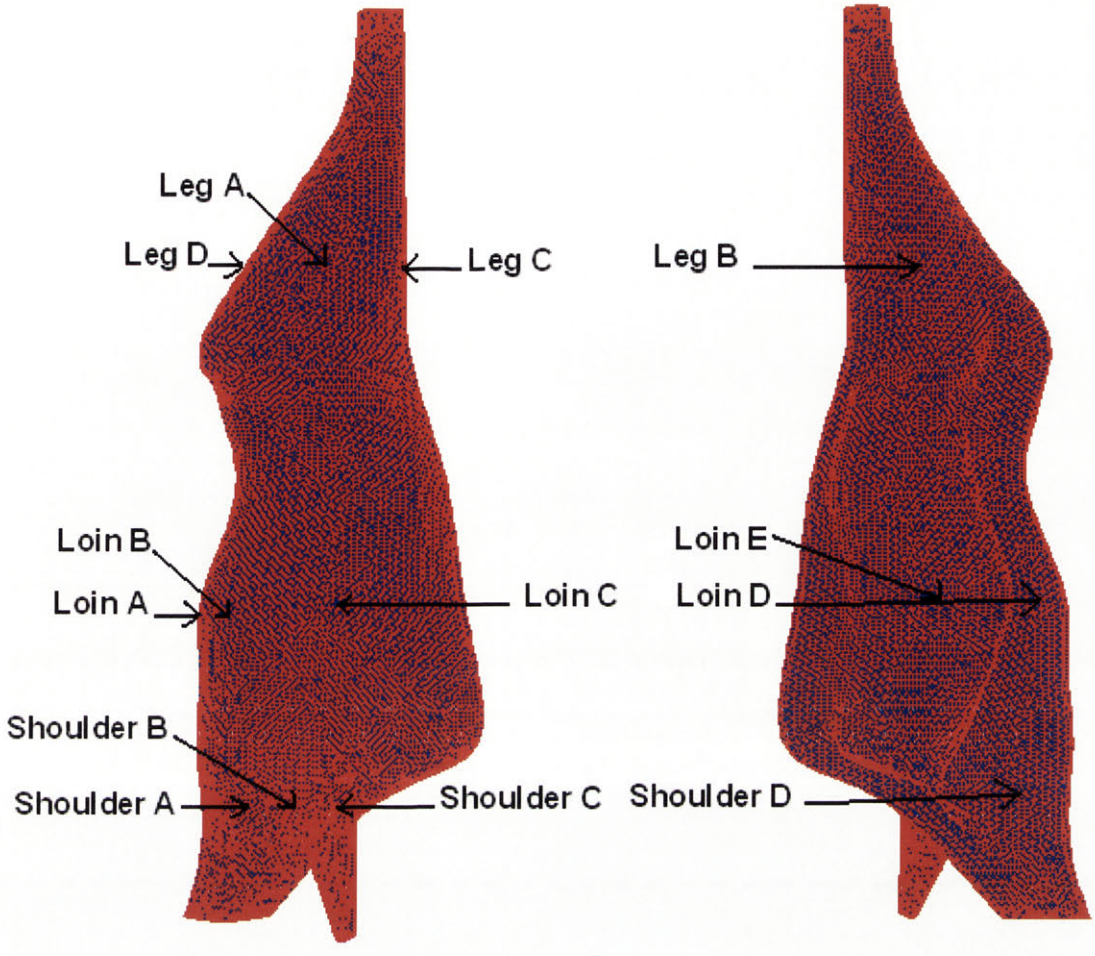


Figure 8.14 3D view of the reference point around the leg, loin and shoulder surfaces.

Figure 8.16 shows the water activity profile (around the positions on figures 8.13- 8.14) for run 14. It shows local variations of the water activity of up to 4% caused by local differences on the heat and mass transfer coefficients. It is also seen that the water activity drops to a minimum value and then increase again, due to rewetting by water diffusing from the inside (Pham and Karuri, 1999; Herbert *et al.* 1978).

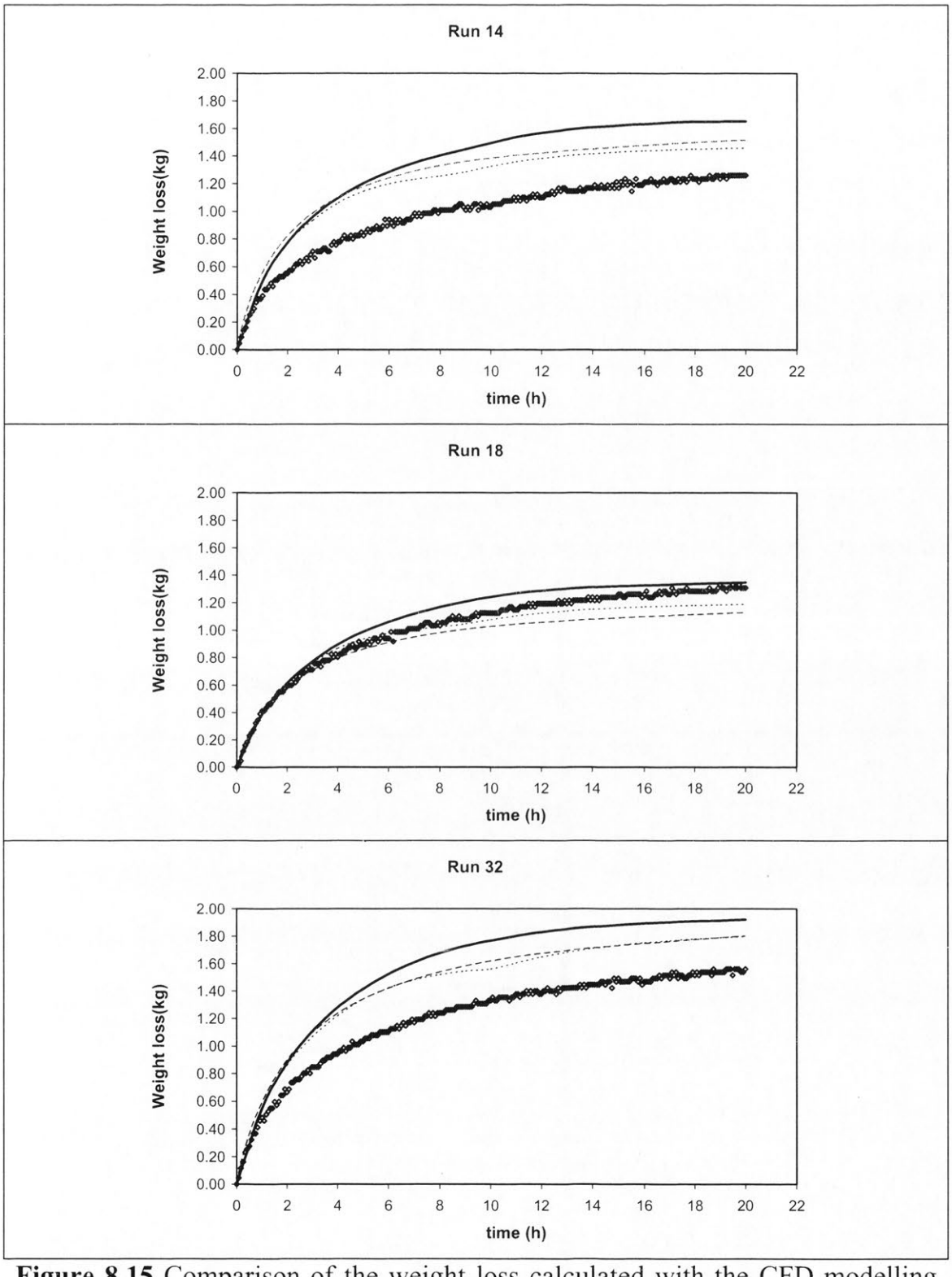


Figure 8.15 Comparison of the weight loss calculated with the CFD modelling (——), FE model of Davey and Pham (2000) (__ __), FD model of Davey and Pham (1997) (------) and experiments for run 14, 18 and 32.

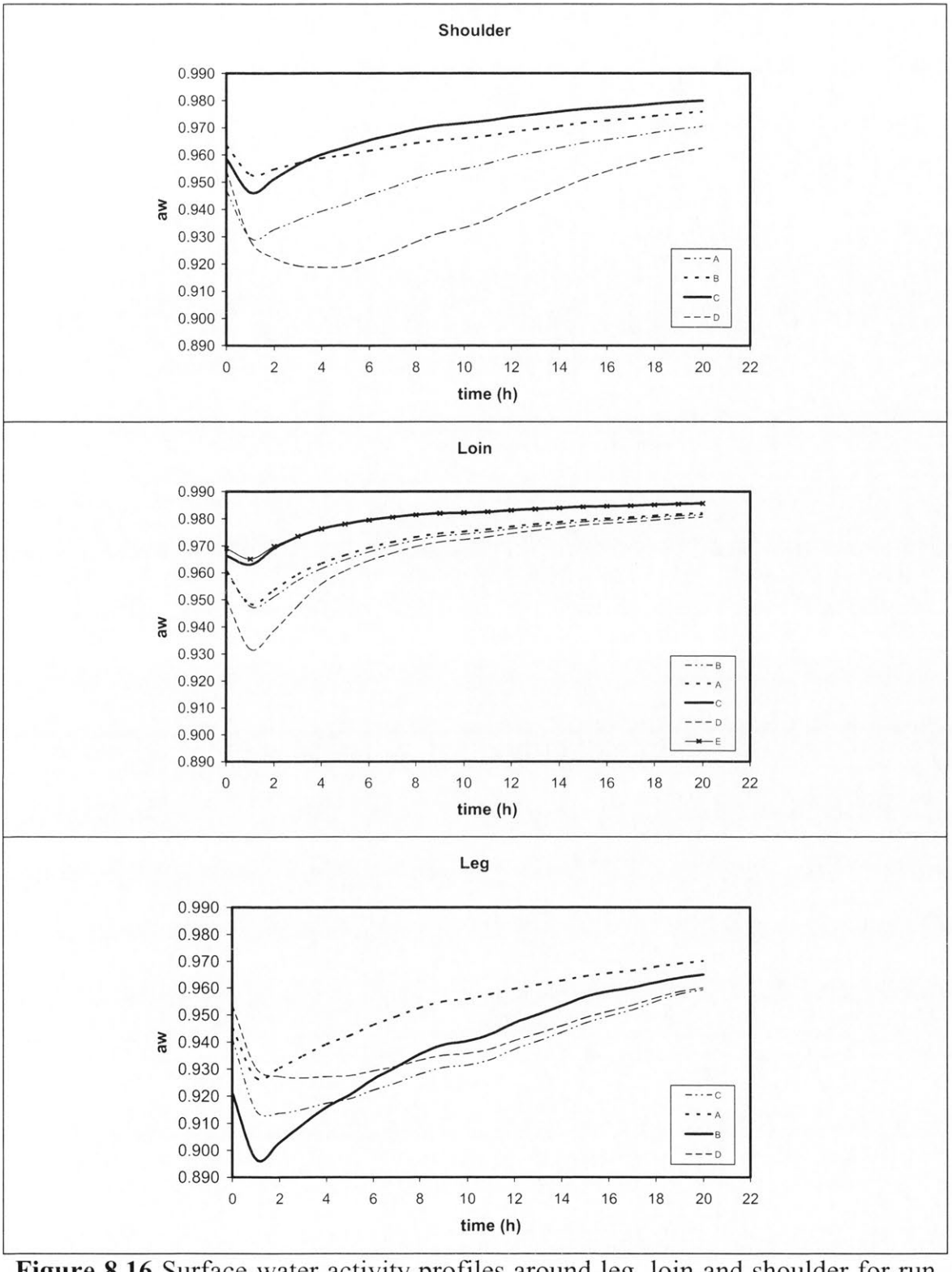


Figure 8.16 Surface water activity profiles around leg, loin and shoulder for run 14.

The minimum water activity reached changes with position depending on the local heat and mass transfer coefficients. It is also expected to change with air velocity, turbulent intensity and air humidity given that the first two affect the local heat and mass transfer coefficients and the last one affects the mass flux on the interface.

The water activity is higher in the loin than in the leg and shoulder. This is because the loin cools faster, reducing the surface vapour pressure and therefore the evaporation. The water activity keeps in the range 0.89-0.98 during the chilling period. Anonymous (1998) reported water activity values between 0.95 and 0.99 during the first 20 hours of chilling. Lovatt and Hill (1998) reported values mostly placed between 0.88 and 0.94. It seems that the approach of water activity equal to 0.85 adopted on the FE and FD models of Davey may be to low.

8.6 Conclusions

- The 3-dimensional unsteady heat and mass transfer process during beef carcass chilling was modelled. A separate 1-dimensional mesh was used to solve the moisture diffusion in a 24 mm surface layer. The finite volume method for modelling the moisture diffusion was implemented and programmed via a UDF.

- At this technological stage it is impractical to do full transient simulations of beef chilling.

- The simultaneous heat and mass transfer process on the air and meat phases was modelled. However, given the long computation time of a full transient simulation, a three step method was used. In the first step, a steady state simulation of the flow field was conducted. In the second step, the local heat and mass transfer coefficients are calculated cell by cell around the beef surface with the information obtained on step 1. Finally, the third step consists of the simultaneous heat and mass transfer process simulation only on the meat carcass.

- Local variations on the heat and mass transfer coefficients were found around the beef carcass surface caused by the development of the momentum, heat and mass boundary layers.

- The CFD heat load predictions are as good as the obtained by the FD or FE models.

- The centre and surface temperatures of the leg and shoulder are very well predicted using CFD. Loin centre and surface temperatures are under predicted during the first ten hours of chilling but very well predicted towards the end of the 20 hours of chilling.

- There is a trend to over predict the weight loss using CFD probably due to neglecting the external fat cover. Better results could be obtained taking into account the heterogeneity inside the meat.

- CFD predicted local temperature variations of up to 4.5°C, and water activity variations of up to 4%, around the beef surface.

- CFD modelling shows the water activity decrease to a minimum value during the first chilling hours followed by a further increase (rewetting). The minimum water activity reached changes with position depending on the local heat and mass transfer coefficients. It is also expected to change with air velocity, turbulent intensity and air humidity.
- It is recommended to conduct evaporation experiments of meat on a simpler geometry like a cylinder, following the weigh loss and recording the water activity on the surface, in order to determine the influence of the fat cover on the weight loss process and to validate the mass transfer CFD modelling.

9. CONCLUSIONS AND RECOMMENDATIONS

9.1 Moisture Sorption Isotherm of fresh lean beef and external beef fat

- The moisture sorption isotherm of fresh lean beef and external beef fat has been successfully measured by using improved techniques to accelerate equilibration and avoid spoilage at high relative humidities.

- The experimental procedure used, based on COST 90 with some geometrical modifications (sorption container geometry, position of the sample and geometry of the sample) was validated with the standard reference material MCC.

- The moisture sorption isotherm (MSI) of the beef at the two highest humidities was obtained by accelerating equilibration with a change of salt. This procedure was reliable for beef samples but not for the fat samples, possible caused by strong hysteresis due to the hydrophobic characteristics of the fat.

- In both cases, lean beef and fat, no significant differences of the MSI with changes in composition was found.

- The MSI of the beef was measured at 5°C, 15 °C, 25 °C and 40 °C. The experimental data was accurately fitted using a model independent of

temperature. The GAB equation fitted the data best, although it predicted a constant value of moisture content at $a_w = 1$. The Lewicki model does not fit well the data in the range (0.60 – 0.90) but gives the best fit at high humidity (0.90 -1.00) and predict that water content approaches infinity when $a_w \rightarrow 1$.

- The MSI of fresh beef surface fat was measure at 5°C, 15°C and 25°C. The experimental water content of the MSI at 5°C is higher than that at 15°C and 25°C. However, no significant difference in the MSI was found between 15°C and 25°C. The experimental data was fitted using the GAB, Lewicki and Peleg models. The best fit was obtained with the Lewicki equation and the parameters were expressed as a function of temperature to make the equation more general, at a slight cost in accuracy.

9.2 Diffusivity of water in meat

- Experimental drying curves obtained at 6.8°C, 19.9°C, 27.9°C and 40.4°C were fitted using three different models: A (constant diffusivity, constant volume, constant temperature and surface moisture), B (convective boundary condition) and C (shrinkage taken into account) to determine the diffusivity of water in beef perpendicular to the fibres.

- The difference between the diffusivity obtained with models A and B decreases when the temperature decreases. This is caused by the slower equilibration of the surface at lower temperatures.

- The model that fits the experimental data best and exhibits the closer normal shape population of the cumulative relative residual value is Model C, which takes shrinkage into account, because it is a better representation of the actual physical phenomena.

- The relation diffusivity-temperature can be accurately expressed with an Arrhenius type equation. The obtained constants using model B are D_o = 1.73E-05 m^2 /s and E_a = -26495.24 J/ mol .K. and those using model C are: D_o = 1.09E-06 m^2 /s and E_a = -20847.5 J/ mol .K.

9.3 Experiments of water evaporation from a circular cylinder

- The experiments of water evaporation from a circular cylinder were modelled with different turbulence models and two wall treatment approaches. It was found that the RNG κ-ε model using the enhanced wall treatment and taking in account the effect of radiation fits better the experimental data. This method is recommended on further models.

- CFD modelling shows that at low Reynolds numbers, when the radiative heat is taken in account, the relation $h_t / h_m c_P$ is not constant around the cylinder, reaching a minimum value at the detachment of the boundary layer.

9.4 CFD modelling of heat and moisture transfer on a two-dimensional model of a beef leg

- A coupled solution of heat and mass transfer on a 2D beef leg model during meat chilling was developed. The transport equations on both the air and meat phases were solved simultaneously. Even though FLUENT 6.1.8 is a very powerful CFD software, it has difficulties dealing with simultaneous heat and mass transfer in two different phases, and special techniques had to be used to solve the mass transport equation in the meat. The solid has to be treated as a fluid region of the domain where convection does not occur. The temperature and moisture fields in the solid are represented by new user-defined field variables. At the solid-air interface, the temperature and moisture fields in the fluid and the user-defined fields in the solid are linked by special user-defined equations, representing interfacial equilibria and transfer, to ensure continuity.

9.5 3D model of beef carcasses and steady state simulation

- A Matlab program was developed to automatically construct the 3D geometry of beef carcasses as a function of weight and fatness.

- CFD shows that there are important local variations on the heat and mass transfer coefficients around the beef surface. This is caused by the development of the momentum, heat and mass boundary layer.

- CFD shows that the forced heat and mass transfer coefficients are mainly dependable on Re and Tu.

- The forced heat and mass transfer coefficients around leg, loin and shoulder were correlated as a function of Reynolds and the turbulence intensity.

- The effects of buoyancy at air velocities higher than 0.538 m/s can be ignored. CFD shows that its contribution to the heat and mass transfer coefficient is less than 5.3% at the beginning of the chilling stage.

9.6 CFD modelling of heat and moisture transfer on a full 3D model of beef carcasses

- The 3-dimensional unsteady heat and mass transfer process during beef carcass chilling was modelled. A separate 1-dimensional mesh was used to solve the moisture diffusion in a 24 mm surface layer. The finite volume method for modelling the moisture diffusion was implemented and programmed via a UDF.

- At this technological stage it is impractical to do full transient simulations of beef chilling.

- The simultaneous heat and mass transfer process on the air and meat phases was modelled. However, given the long computation time of a full transient simulation, a three step method was used. In the first step, a steady state simulation of the flow field was conducted. In the second step, the local heat and mass transfer coefficients are calculated cell by cell around the beef surface with the information obtained on step 1. Finally, the third step consists of the simultaneous heat and mass transfer process simulation only on the meat carcass.

- Local variations on the heat and mass transfer coefficients were found around the beef carcass surface caused by the development of the momentum, heat and mass boundary layers.

- The CFD heat load predictions are as good as those obtained by the previous FD or FE models of Davey.

- The centre and surface temperatures of the leg and shoulder are very well predicted using CFD. Loin centre and surface temperatures are under predicted during the first ten hours of chilling but very well predicted towards the end of the 20 hours of chilling.

- There is a trend to over predict the weight loss using CFD probably due to neglecting the external fat cover. Better results could be obtained taking into account the heterogeneity inside the meat.

- CFD predicted local temperature variations of up to 4.5°C, and water activity variations of up to 4%, around the beef surface.

- CFD modelling shows the water activity decrease to a minimum value during the first chilling hours followed by a further increase (rewetting). The minimum water activity reached changes with position depending on

the local heat and mass transfer coefficients. It is also expected to change with air velocity, turbulent intensity and air humidity.

- It is recommended to conduct evaporation experiments of meat on a simpler geometry like a cylinder, following the weigh loss and recording the water activity on the surface, in order to determine the influence of the fat cover on the weight loss process and to validate the mass transfer CFD modelling.

REFERENCES

1. **Anonymous.** (1998). Safe Beef Carcass Chilling Procedures. Internal Report to Meat and Livestock Australia.

2. **Australian Meat and Livestock Corporation.** (1986). Technical Manual of Australian Meat. Australian Meat and Livestock Corporation, Sydney.

3. **Baucour P. and Daudin J. D.** (2000). Development of a new method for fast measurement of water sorption isotherms in t he high humidity range validation gelatine gel. Journal of Food Engineering. **44**, 97 -107.

4. **Berg C. and Bruin S.** (1981). Water activity and its estimation in food systems: theoretical aspects. In Water Activity: Influences on Food Quality. Edited by L. B. Rockland and G. F. Steward. Academic Press. New York.

5. **Bird R. B., Steward W. E. and Lightfood E. N.** (1960). Transport Phenomena. New York, N.Y. Wiley.

6. **Carslaw H. S. and Jaeger J. C.** (1959). Conduction of Heat in Solids. Oxford University Press, London.

7. **Cess R. D.** (1962). The effect of radiation upon forced-convection heat transfer. Appl. Sci. Res., Section A, **10**, 430-438,.

8. **Chen X. D., Lin S. X. Q. and Chen G.** (2002). On the ratio of heat to mass transfer coefficient for water evaporation and its impact upon drying modelling. Int. J. Heat and Mass Transfer, **45**, 4369-4372.

9. **Chuntranuluck S., Wells C. M. and Cleland A. C.** (1998a). Prediction of chilling times of foods in situations where evaporative cooling is significant – Part 1. Methods development. Journal of Food Engineering. **37**, 112-125.

10. **Chuntranuluck S., Wells C. M. and Cleland A. C.** (1998b). Prediction of chilling times of foods in situations where evaporative cooling is significant – Part 2. Experimental testing. Journal of food engineering. **37**, 127-141.

11. **Cleland A. C. and Earle R. L.** (1982). A simple method for prediction of heating and cooling rates in solids of various shapes. International Journal of Refrigeration. **5**, 98-106.

12. **Crank J.** (1975). The mathematics of diffusion. 2nd Ed. Oxford University Press, Oxford.

13. **Daudin J. D. & Kondjoyan A.** (1991). Influence de l'indice de turbulence de l'ecoulement sur les procedes de traitement thermique de solides par l'air chaud. These de Docteur-Ingenieur de l'ENSIA,Massy, France.

14. **Daudin J. D. and Kuitche A.** (1996). Modelling of temperature and weight loss kinetics during meat chilling for time variable conditions using an analytical based method – III. Calculations versus measurements on pork carcasses hindquarters. Journal of Food Engineering. **29**, 39-62.

15. **Daudin J. D. and Swain M. V. L.** (1990). Heat and mass transfer in chilling and storage of meat. Journal of Food Engineering. **12**, 95- 115.

16. **Davey L. M.** (1998). Measurement and prediction of product heat load and weight loss during beef chilling, PhD Thesis, University of New South Wales, Sydney.

17. **Davey L. M. and Pham Q. T.** (1997). Predicting the dynamic product head load and weight loss during beef chilling using a multi-region finite difference approach. International Journal of Refrigeration. **20**, No. 7, 470-482..

18. **Davey L. M. and Pham Q. T.** (2000). A multi – Layered two-dimensional finite element model to calculate dynamic product heat load and weight loss during beef chilling. International Journal of Refrigeration. **23**, 444-456.

19. **Delgado A. E. and Sun D. W.** (2002a). Desorption isotherms for cooked and cured beef and pork. Journal of food Engineering. **51**, 163 – 170.

20. **Delgado A. E. and Sun D. W.** (2002b). Desorption isotherms and glass transition temperature for chicken meat. Journal of Food Engineering. **55**, 1-8.

21. **FLUENT Inc.** (2003a). FLUENT 6.1.18 user's guide, Vols. 1-5, FLUENT Inc., Lebanon NH USA.

22. **FLUENT Inc.** (2003b). FLUENT 6.1.18 UDF manual, FLUENT Inc., Lebanon NH USA.

23. **Greenfield H., Arcot J. Emerson E., Hutchison G. L. and Wills R. B. H.** (1998). Laboratory Instruction for Food Composition Studies. Department of Food Science and Technology, University of New South Wales.

24. **Greenspan L.** (1976). Humidity fixed points of binary saturated aqueous solutions. Journal of Research of the National Bureau of Standards. **81A**, No. 1, 89 – 96.

25. **Gou P., Comaposada J. & Arnau J.** (2002). Meat pH and meat fibre direction effects on moisture diffusivity in salted ham muscles dried at 5 °C. Meat Science. **61**, 25-31.

26. **Gou P., Comaposada J. & Arnau J.** (2003). NaCl content and temperature effects on moisture diffusivity in the gluteus medius muscle of pork ham. Meat Science. **63**, 29 - 34.

27. **Harris M. B., Carson J. K., Willix J. & Lovatt S.J.** (2003) Local surface heat transfer coefficients on a model lamb carcass. Journal of Food Engineering. **61**, 421-429.

28. **Harris M. B. and Willix J.** (2000). Measurement of mass transfer coefficients on carcasses shaped objects. Internal Memorandum IM 494. MINIRZ, Hamilton, New Zealand.

29. **Herbert L. S., Lovett, D. A. and Radford, R. D.** (1978). Evaporative Weight Loss during meat chilling, Food Technology in Australia. April, **30**, 145 – 148.

30. **Hernandez J. A., Pavon G. & Garcia M. A.** (2000). Analytical solution of mass transfer equation considering shrinkage for modelling food-drying kinetics. Journal of Food Engineering. **45**, 1-10.

31. **Hossain Md. M., Clealand D. J., and Cleland A. C.** (1992). Prediction of freezing and thawing times for foods of regular multi-dimensional shape by using an analytically derived geometric factor. International Journal of Refrigeration. **15**, No. 4, 227-234.

32. **Hu, Z., Sun, D. W.** (2000). CFD simulation of heat and moisture transfer for predicting cooling rate and weight loss of cocked meats. Journal of Food Engineering. **46**, 189-197.

33. **Hu, Z., Sun, D. W.** (2001). Predicting local surface heat transfer coefficients by different turbulent $\kappa-\varepsilon$ models to simulate heat and moisture transfer during air-blast chilling. International Journal of Refrigeration. **24**, No. 7, 702 -717.

34. **Iglesias H. A. and Chirife J.** (1982). Handbook of Food Isotherms. Academic Press Inc. (New York).

35. **Incropera F. P. and DeWitt D. P.** (2002). Heat and Mass Transfer. 5th edition. John Wiley & Sons.

36. **Kestin J. and Wood R. T.** (1971). The influence of turbulence on mass transfer from cylinders. Journal of Heat Transfer. **93**, 321—327.

37. **Kondjoyan A. and Daudin J. D.** (1993a). "Determination of transfer coefficients by psychrometry". International Journal Heat and Mass Transfer. **36**, No. 7, 1807-1818.

38. **Kondjoyan A. and Daudin J. D.** (1993b). Heat and mass transfer coefficients at the surface of elliptical cylinders placed in a turbulent air flow. Journal of Food Engineering. **20**, 339-367.

39. **Kondjoyan A. and Daudin J. D.** (1995). Effects of free stream turbulence intensity on heat and mass transfer at the surface of a circular and an elliptical cylinder, axis ratio 4. International Journal of Heat and Mass Transfer. **38**, 1735-1749.

40. **Kondjoyan A. and Daudin J. D.** (1997). Heat and mass transfer coefficients at the surface of a pork hindquarter. Journal of Food Engineering. **32**, 225-240.

41. **Kuitche A., Daudin J. D. and G. Letang** (1996a). Modelling of temperature and weight loss kinetics during meat chilling for time variable conditions using an analytical-based method – I. The model and its sensitivity to certain parameters. Journal of Food Engineering. **28**, 55-84.

42. **Kuitche A., Letang G. and Daudin J. D.** (1996b). Modelling of temperature and weight loss kinetics during meat chilling for time variable conditions using an analytical based method – II. Calculations versus measurements on wet plaster cylinders and cast. Journal of Food Engineering. **28**, 85-107.

43. **Labuza, T. P, Hyman, C. R.** (1998). Moisture migration and control in multidomain foods. Trends in Food Science & Technology. Elsevier Science Ltd. Cambridge U.K. **9**, 47-55.

44. **Lang K. W., McCune T. D. and Steinberg M. P.** (1981). A proximity equilibration cell for rapid determination of sorption isotherms. Journal of Food Science. **46**, 936 – 938.

45. **Launder B.E. and Spalding D.B.** (1974). The numerical computation of turbulent flows. Computer Methods in Applied Mechanics and Engineering. **3**, 269-289.

46. **Lewicki P. P.** (1998). A 3 parameter equation for food moisture sorption isotherms. Journal of Food Process Engineering. **21**, 127-144.

47. **Lewis J. S.** (1971). Heat transfer predictions from mass transfer measurements around a single cylinder in cross-flow. International Journal Heat and Mass Transfer, **14**, 325-329.

48. **Lomauro C. J., Bakshi A. S., & Labuza T. P.** (1990). Moisture transfer properties of dry and semi moist foods. Journal of Food Science, **55**, 218-223.

49. **Lovatt S. J and Hill H. K.** (1998). Surface water activity during meat cooling. IIR Proceedings Series "Refrigeration Science and Technology". Sofia, Bulgaria, 465-472.

50. **Lovatt S. J., Pham Q. T., Cleland A. C. and Loeffen M. P. F.** (1993). A new method of predicting the time-variability of product heat load during food cooling- Part 1: Theoretical considerations. Journal of Food Engineering. **18**, 13-36.

51. **Lin Z., Cleland A.C., Sellarach G.F. and Cleland D.J.** (1996a). A simple method for prediction of chilling times for objects of two-

dimensional irregular shape. International Journal of Refrigeration. **19**, 95-106.

52. **Lin Z., Cleland A.C., Sellarach G.F. and Cleland D.J.** (1996b). A simple method for prediction of chilling times: Extension to 3-D irregular shapes. International Journal of Refrigeration. **9**, 107-114.

53. **Mallikarjunan P. and Mittal G. S.** (1994). Heat and mass transfer during beef carcass chilling – Modelling and simulation. Journal of Food Engineering. **23**, 277-292.

54. **Marshall S. A. and James R. W.** (1975). Dynamic analysis of an industrial refrigeration system to investigate capacity control. Proc. Inst. Mech. Engrs. **189**, 437 - 445.

55. **May B. K & Perre P.** (2002). The importance of considering exchange surface area reduction to exhibit a constant drying flux period in foodstuffs. Journal of Food Engineering. **54**, 271-283.

56. **Mayor L. & Sereno A. M.** (2004). Modelling shrinkage during convective drying of food materials: a review. Journal of Food Engineering. **61**, 373-386.

57. **Merts I., Lovatt S. J. & Lawson C. R.** (1998). Diffusivity of moisture in whole muscle meat measured by a drying curve method. IIR Proceedings Series "Refrigeration Science and Technology". Sofia, Bulgaria, 473-479.

58. **Mirade P. S., Kondjoyan A. and Daudin J. D.** (2002). Three-dimensional CFD calculations for designing large food chillers. Computers and Electronics in Agriculture. **34**, 67-88.

59. **Motarjemi Y.** (1998). A study of some physical properties of water in foodstuffs. Ph.D. Thesis, Department of Food Engineering, Lund University, Lund, Sweden.

60. **Mulet A.** (1994). Drying modelling and water diffusivity in carrots and potatoes. Journal of Food Engineering, **22**, 329-348.

61. **Nguyen A. V. and Pham Q. T.** (1999). A computational fluid dynamic model of beef chilling. 20th International Congress of Refrigeration, IIR/IIF, Sydney.

62. **Palnitkar M. P and Heldman D. R.** (1971). Equilibrium moisture characteristics of freeze-dried beef components. Journal of Food Science. **36**, 1015 – 1018.

63. **Pham Q. T.** (1989). Prediction of thermal conductivity of meats and other animal products from composition data. 5th International Congress on Engineering and Foods (ICEF 5), Cologne.

64. **Pham, Q. T.** (1994). Competitive evolution: A natural approach to operator selection. AI 94 Evolutionary Computational Workshop, 21 November, Armidale, Australia.

65. **Pham Q. T.** (1996). Prediction of calorimetric properties and freezing time of foods from composition data. Journal of Food Engineering. **30**, 95-107.

66. **Pham Q. T.** (2001). Prediction Of Cooling/Freezing/Thawing Time And Heat Load. In: Advances in Food Refrigeration, ed. Da-Wen Sun. at Leatherhead Food RA, Leatherhead, 110-152.

67. **Pham Q. T.** (2001). Thermal Processing Operations: Cooling and Freezing. In: Food Process Modelling, ed. L.M.M. Tijskens, M.L.A.T.M. Hertog and B.M. Nicolai. Woodhead Publishing, Cambridge, England, Chapter **15**, 312-339.

68. **Pham Q. T. and Karuri N. W.** (1999). A computationally efficient technique for calculating simultaneous heat and mass transfer during food chilling. 20th International Congress of Refrigeration, IIR/IIF, Sydney.

69. **Pham Q. T. and Nguyen A. V.** (2000). Mean convective heat transfer coefficient of beef carcasses computed by a computational fluid dynamic model. FOODSIM 2000: International Conference on Simulation in Food and BioIndustries, Nantes, France, June 26-27.

70. **Radford R. D.** (1976). Water transport in meat. International Institute of refrigeration. Australian National Committee. Joint meeting of commissions C2, D1, D2, D3 and E1. Melbourne-Australia. Report 62, 1-8.

71. **Radford R. D., Herbert C. & Lovett D. A.** (1976). Chilling of meat- A mathematical model for heat and mass transfer. Proc. Comm. C2 Meeting, IIR Bull. Annex 1976-1, 323-330.

72. **Peleg, M.** (1992). Assessment of a semi-empirical 4 parameter general model for sigmoid moisture sorption isotherms. Journal of Food Process Engineering. **16**, 21-37.

73. **Rahmna M. S., Perera C. O. and Thebaud C.** (1998). Desorption isotherm and heat pump drying kinetics of peas. Food Research International. **30**, 485 – 491.

74. **Ross T.** (1999). Predictive Microbiology model in the Meat Industry. Meat and Livestock Australia, Sydney.

75. **Saravacos G. D. and Stinchfield R. M.** (1965). Effect of temperature and pressure on the sorption of water vapour by freeze–dried food materials. Journal of Food Science. **30**, 779-786.

76. **Sing R. R. B., Rao K. H., Anjaneyulu A. S. R. and Patil G. R.** (2001). Moisture sorption properties of smoked chicken sausages from spent hen meat. Food Research International. **34**, 143 -148.

77. **Spiess W. E. L. and Wolf W.** (1987). Critical evaluation of methods to determine Moisture sorption isotherms. In Water activity: Theory and Applications in food. Academic Press. New York. 215-233.

78. **Sun D. W. and Hu Z.** (2002). CFD predicting the effects of various parameters on core temperature and weight loss profiles of cooked meat during vacuum cooling. Computers and Electronics in Agriculture. **34**, 111-127.

79. **Sun D. W. and Hu Z.** (2003). CFD simulation of coupled heat and mass transfer through porous foods during vacuum cooling. International Journal of Refrigeration. **26**, 19-27

80. **Scott G. and Richardson P.** (1997). The application of computational fluid dynamics in the food industry. Trends in Food Science & Technology. **8**, 119- 124.

81. **Taylor A. A.** (1961). Determination of Moisture equilibria in dehydrated foods. Food Technology. **15**, 536 – 540.

82. **Timmerman E. O. and Chirife J.** (1991). The physical state of water sorberd at high activities in starch in terms of the GAB sorption equation. Journal of Food Engineering. **13**, 171 – 179.

83. **Timmerman E. O, Chirife J. and Iglesias H. A.** (2001). Water sorption of foods and foodstuffs: BET or GAB parameter?. Journal of Food Engineering. **48**, 19-31.

84. **Tocci A. M. and Masheroni R. H.** (1995). Heat and mass transfer coefficients during the refrigeration, freezing and storage of meats, meat products and analogues. Journal of Food Engineering. **26**, 147-160.

85. **Trujillo F. J., Yeow P. C. & Pham Q. T.** (2003). Moisture sorption isotherm of fresh lean beef and external fat. Journal of Food Engineering. **60**, 357-366.

86. **Veerstage H. K and Malalasekera W.** (1995). An introduction to Computational Fluid Dynamics – The finite volume method. Prentice – Hall.

87. **Verboven P., Nicolai B. M., Scheerlinck N. and Baerdemaeker J. D.** (1997). The local surface heat transfer coefficient in thermal food process calculations: A CFD approach. Journal of Food Engineering. **33**, 15-35.

88. **Weisser H.** (1985). Influence of temperature on sorption equilibria. In Properties of water in food: in relation to quality and stability. Edited by D. Simatos and J. L. Multon. Martinus Nijhoff Publishers. Dordrecht, Boston.

89. **Wiangkaew C.** (2003). Measurement of the effective water diffusivity in meat. Master Thesis, University of New South Wales. Sydney, NSW, Australia.

90. **Wolf W., Spiess W. E. L. and Jung G.** (1985). Standardization of isotherm measurements (Cost-Project 90 and 90 bis). In Properties of Water in Food: In Relation to Quality and stability. Edited by D. Simatos and J. L. Multon. Martinus Nijhoff Publishers. Dordrecht, Boston.

91. **Wolfstein M.** (1969). The Velocity and Temperature Distribution of One-Dimensional Flow with Turbulence Augmentation and Pressure Gradient. International Journal of Heat Mass Transfer. **12**, 301-318.

92. **Yakhot V and Orszag S. A.** (1986). Renormalization Group Analysis of Turbulence. I Basic Theory. Journal of scientific computing. **1**, 3-51.

93. **Yeow P. C.** (2001). Measurement of food moisture isotherms, Bachelor Thesis in Chemical Engineering, University of New South Wales, Sydney Australia

94. **Zdanavichyus G. B, Survila V. Yu. and Zhukauskas A. A.** (1977). Influence of the degree of turbulence of an inflowing air stream on local heat transfer on a circular cylinder in the critical flow regime. International Chemical Engineering. **17**, 115-121.

95. **Zogzas N. P & Maroulis Z. B.** (1996). Effective moisture diffusivity estimation from drying data. A comparison between different methods of analysis. Drying Technology. **14**, No.7-8, 1543-2573.

96. **Zogzas N. P, Maroulis Z. B. & Marinos-Kouris D.** (1994). Moisture diffusivity methods of experimental determination – A review. Drying Technology, **12**, No. 3, 483-515.

A1. FINITE VOLUME VBA SUB-ROUTINE OF WATER DIFFUSION TAKING SHRINKAGE INTO ACCOUNT

```
Sub OneD_diff2(x2 As Double, TimeStep As Double, _
        PrintStep As Double, N As Integer, _
        ExitT As Double, _
        Time As Double, Eqv_HeatBalanceError As Double, _
         Xratio() As Double)

  Dim i As Integer, iTime As Long, PrintFreq As Integer
  Dim dX As Double
  Dim ExitNow As Boolean
  Dim SurfAw As Double, SampleY As Double
  Dim Xavg As Double ' average X
  Dim mass_flux As Double, De As Double, Dw As Double, Xe As Double, Xw As Double, dxep As
        Double
  Dim dxwp As Double, FAe As Double, FAw As Double, FAp As Double, RhoSe As Double, RhoSw
        As Double
  Dim RhoSp As Double, apo As Double, aw As Double, ae As Double, ap As Double, awall As Double
  Dim bcons As Double, Yeto As Double, Ywto As Double, Ypto As Double, su As Double
  Dim DD(22) As Double, CC(22) As Double, EE(22) As Double, BB(22) As Double
  Dim A As Double, B As Double
  'Geometric and other factors
  N = 22

  'Initiate
  iTime = 0
  kTime = 1 'output counter
  Time = 0
  For i = 0 To N
    Xratio(i) = Xinit
  Next i

  SurfAw = fWaterAct(Xinit)          ' calculation water act. surface
  If SurfAw > 1 Then
    SurfAw = 1
  End If
  WB_Unsat AirTemperature, Yair, SurfAw, SampleT, SampleY     'calculation of air humidity and init.
        sampleT

  PrintFreq = PrintStep / TimeStep
  Row = 18
  With Sheets("FD_diff")
    .Cells(Row, 1) = Time / 3600
    .Cells(Row, 4) = Avg_X(Xratio, N)
    .Cells(Row, 5) = Xratio(0)
    For i = 1 To N
      .Cells(Row, 7 + i) = Xratio(i)
    Next i
  End With

  X_calc(kTime) = Avg_X(Xratio, N)
  ExitNow = False
```

```
Do 'Start iteration.........................
  iTime = iTime + 1
  Time = iTime * TimeStep
  mass_flux = ky * (Yair - SampleY)
  bcons = mass_flux / RhS_calc(Xratio(1)) / Area_calc(Xratio(1))
  For i = 1 To N
    ' calculation of the water content in the east and west
    If i = 1 Then
      Xw = Xratio(0)
      Xe = 0.5 * (Xratio(1) + Xratio(2))
    ElseIf i = N Then
      Xw = 0.5 * (Xratio(N) + Xratio(N - 1))
      Xe = Xratio(N)
    Else
      Xw = 0.5 * (Xratio(i) + Xratio(i - 1))
      Xe = 0.5 * (Xratio(i) + Xratio(i + 1))
    End If

    ' calculation of dxep, dxwp and dX
    dxep = dXsh_calc(Xe, dXo(i))
    dxwp = dXsh_calc(Xw, dXo(i - 1))
    If i = 1 Then
      dX = dxwp + 0.5 * dxep
    ElseIf i = N Then
      dX = 0.5 * dxwp + dxep
    Else
      dX = 0.5 * (dxep + dxwp)
    End If


    De = fDiffusivity(SampleT, Xe)
    Dw = fDiffusivity(SampleT, Xw)
    FAe = Area_calc(Xe)
    FAw = Area_calc(Xw)
    FAp = 0.5 * (FAe + FAw)
    RhoSe = RhS_calc(Xe)
    RhoSw = RhS_calc(Xw)
    RhoSp = RhS_calc(Xratio(i))

    apo = dX / TimeStep
    aw = FAw * RhoSw * Dw / dxwp / FAp / RhoSp
    ae = FAe * RhoSe * De / dxep / FAp / RhoSp
    Ypto = Xratio(i)

    Select Case i
    Case 1
      Yeto = Xratio(i + 1)
      su = ae * Yeto / 2 + (apo - ae / 2) * Ypto + bcons
      awall = aw
      aw = 0
    Case N
      Ywto = Xratio(N - 1)
      ae = 0
      su = ae * Yeto / 2 + aw * Ywto / 2 + (apo - aw / 2 - ae / 2) * Ypto
    Case Else
      Yeto = Xratio(i + 1)
      Ywto = Xratio(i - 1)
      su = ae * Yeto / 2 + aw * Ywto / 2 + (apo - aw / 2 - ae / 2) * Ypto
    End Select
```

```
        ap = 0.5 * (aw + ae) + apo
        DD(i) = ap
        EE(i) = -ae / 2
        CC(i) = -aw / 2
        BB(i) = su

      Next i
      tridag 1, N, CC, DD, EE, BB, Xratio
      Xratio(0) = Xratio(1) + bcons / awall
      SurfAw = fWaterAct(Xratio(0))
      WB_Unsat AirTemperature, Yair, SurfAw, SampleT, SampleY
      'Outputs
      If iTime Mod PrintFreq = 0 Then
       kTime = kTime + 1
       Xavg = Avg_X(Xratio, N)
       X_calc(kTime) = Xavg
       Row = Row + 1
       With Sheets("FD_diff")
         .Cells(Row, 1) = Time / 3600
         .Cells(Row, 3) = RhS_calc(Xavg)
         .Cells(Row, 4) = Xavg
         .Cells(Row, 5) = Xratio(0)
         .Cells(Row, 6) = SurfAw
         .Cells(Row, 7) = SampleT
         For i = 1 To N
           .Cells(Row, 7 + i) = Xratio(i)
         Next i
       End With
      End If
      'Test for exit condition
      ExitNow = (Time >= ExitT)

   Loop Until ExitNow

End Sub

Function RhS_calc(Xratio As Double) As Double
   If nX = 1 Then
      RhS_calc = RhoS * (1 + Beta * Xinit) / (1 + Beta * Xratio)
   ElseIf nX = 2 Then
      RhS_calc = RhoS * (1 + TrialG.X(2) * Xinit) / (1 + TrialG.X(2) * Xratio)
   Else
      RhS_calc = RhoS * (1 + Beta * Xinit) / (1 + Beta * Xratio)
   End If
   If Sh_cont = False Then
      RhS_calc = RhoS
   End If
End Function


Function Area_calc(Xratio As Double) As Double
   If nX = 1 Then
      Area_calc = ((1 + Beta * Xratio) / (1 + Beta * Xinit)) ^ (2 / 3)
   ElseIf nX = 2 Then
      Area_calc = ((1 + TrialG.X(2) * Xratio) / (1 + TrialG.X(2) * Xinit)) ^ (2 / 3)
   Else
      Area_calc = ((1 + Beta * Xratio) / (1 + Beta * Xinit)) ^ (2 / 3)
   End If
```

```
  If Sh_cont = False Then
    Area_calc = 1
  End If
End Function

Function dXsh_calc(Xratio As Double, dX As Double) As Double
  If nX = 1 Then
    dXsh_calc = dX * ((1 + Beta * Xratio) / (1 + Beta * Xinit)) ^ (1 / 3)
  ElseIf nX = 2 Then
    dXsh_calc = dX * ((1 + TrialG.X(2) * Xratio) / (1 + TrialG.X(2) * Xinit)) ^ (1 / 3)
  Else
    dXsh_calc = dX * ((1 + Beta * Xratio) / (1 + Beta * Xinit)) ^ (1 / 3)
  End If
  If Sh_cont = False Then
    dXsh_calc = dX
  End If
End Function
```

A2. UDFS PROGRAMMED IN C++ TO SOLVE TO MODEL THE EVAPORATION PROCESS AROUND A CIRCULAR CYLINDER

```
# include "udf.h"

/******************************** variables that must be defined
*********************************************************/

real Df = 0.00002388;                /* define the effective diffusivity */
real K = 0.0256;                     /* thermal conductivity */
real Hvap = 2459180;                 /* heat of vaporization of water in J / Kg */
real To = 293.36;                    /* Inlet temperature */
real yo = 0.005857;                  /* water mass composition at the inlet */
real E = 0.91;                       /* Plaster emisivity */
real Pt = 102230;                    /* cell total presure */
real sigma = 5.67e-8;                /* sigma constant */
int Radiation_Control = 1;           /* 1 when take in account the radiation , 0, when do not */
real alfa = 0.1;                     /* is the relaxation factor in the UDF that calculate the water
                                     on the interface */
int zone_ID_wp = 6;                  /* ID zone of the wall in contact with the air */

/*****************************************************************************************
*************************************/



DEFINE_PROFILE(heat_vaporization, t, j)
{

        int i = 0;                   /* define the first component =water */
        real T;                      /* Face Temperature K */
        real yf;                     /* define the wat. comp. on the face */
        real yc;                     /* define the wat. comp. on the cell */
        real density;                /* density average of face and cell */
        real mass_flux;              /* define the mass fflux on the
                                     wall (face) on Kg Water / s m2 */
        real A[ND_ND];               /* vector Area of the face */
        real Area;                   /* Magnitud of the vector Area */
        real Cface[ND_ND];           /* face centroid vector */
        real Ccell[ND_ND];           /* cell centroid vector */
        face_t f;                    /* face of the face loop */
        cell_t co;                   /* cell asociated to the face */
        Thread *to;                  /* cell thread for co */
        real es[ND_ND];              /* unit vector directin from the
                                     cell centroid to the face cent.*/
        real dro[ND_ND];             /* vector distance connecting the cell and face centroid */
        real ds;                     /* distance between the cell centroide and face centroide */

        real A_by_es;
        real Zero[ND_ND];
```

```
        Zero[0] = 0;
        Zero[1] = 0;



        begin_f_loop(f, t)
         {

                T = F_T(f,t);                /* Face Temperature  K */
                co = F_C0(f,t);              /* cell asociated to the face */
                to = THREAD_T0(t);           /* cell thread for co */
                yf = F_YI(f, t, i);          /* define the wat. comp. on the face */
                yc = C_YI(co, to, i);        /* define the wat. comp. on the cell */
                density = C_R(co, to);       /* density calcualte on the cell*/
                F_AREA(A, f, t);
                Area = NV_MAG(A);                    /* Area */
                F_CENTROID(Cface, f, t);

                C_CENTROID(Ccell, co, to);

                NV_VV(dro, =, Cface, -, Ccell);/* vector distance between the face and cell centr. */
                ds = NV_MAG(dro);                    /* distance between the face and cell centroids */
                NV_V_VS(es, = ,Zero, + , dro, *, (1/ds));/* unit vector directing from thecell centroid
                                                          to the face centr.*/
                A_by_es = NV_DOT(A, A) / NV_DOT(A, es);



                mass_flux = Df * (yf - yc) / ds * A_by_es /Area * Density;



                F_PROFILE(f, t, j) = -Hvap* mass_flux + Radiation_Control   * E * sigma *
                (pow(To,4) - pow(T,4));


         }
        end_f_loop(f,t)




}



DEFINE_PROFILE(water_interface, t, n)
{
        real T;                      /* Face Temperature */
        real Pw;                     /* face vapor presure */
        int i = 0;                   /* define the first component =water */
        real x;                      /* molec. compos. water /air */
        real y;                      /* weight comp. water / air */
        real yf;                     /* define the wat. comp. on the face */
        real dy;                     /* define the dy */
        face_t f;
        real c1 = -5800.2206;        /* cte of vapor presure */
```

```
real c2 = 1.3914993;
real c3 = -0.048640239;
real c4 = 0.000041764768;
real c5 = -0.000000014452093;
real c6 = 6.5459673;

begin_f_loop(f, t)
 {
        T = F_T(f,t);       /* face temperature */
        Pw = exp( c1 / T + c2 + c3 * T + c4 * pow(T , 2) + c5 *pow(T, 3) + c6 * log(T) );
                            /* vapor presure */
        x = Pw / Pt;        /* molec. compos. water /air */
        yf = F_YI(f, t, i);/* define the wat. comp. on the face */
        y = x * 18.016 / (x * 18.016 + (1 - x) * 28.96) ;/* weight comp. water / air */
        dy = y - yf;

        F_PROFILE(f, t, n) = yf + alfa * dy;



 }
end_f_loop(f,t)


}
```

A3. UDFS PROGRAMMED IN C++ TO SOLVE THE SIMULTANEOUS HEAT AND MOISTURE TRANSFER ON THE 2D LEG MODEL

```
# include "udf.h"


/********************************* variables that must be defined**********
*****************************************/
/****** constants ******/
real Cpmeat = 3407;          /* Cp meat */
real kmeat = 0.397;          /* k meat */
real denmeat = 1111;         /* meat density */
real Dtmeat = 1.0488e-7;     /* thermal diffusivity */
real Hvap = 2452240;         /* heat of vaporization of water in J / Kg */
real E = 0.92;               /* Plaster emisivity */
real kar = 0.4187;           /* vor karman constant */
real Ew = 9.793;             /* wall function constant */
real sigma = 5.67e-8;        /* sigma constant */
real Cmu = 0.0845;           /* constant of the RNG model */
real Mwater = 18.016;        /* warter molecular weight*/
real Mair =  28.96;          /* air molecular weight*/
real R = 8314.47;            /* universal constant in pa m3 / Kgmol/ K;*/
/*******************************************************************************
*************************************/



/***************************** Initial and operation conditions
*************************************************************/
/* it must be the same imformation in the boundary condtions panel */
real Pt = 101325;            /* cell total presure */
real To = 277.95;            /* Inlet air temperature */
real yo = 0.005295;          /* water mass composition at the inlet */
real RH = 0.98;              /* Humidity as a fracction */
real Yomeat = 0.75;          /* initial water composition of meat */
real Tomeat = 315.15;        /* initial meat temperature */
/*******************************************************************************
*************************************/


/********************** Time variable and vector to store and solve the water diffusion in meat
************/
real last_ts = -1;           /*  Global variable.  Time s never <0 used in solve_meatwater*/
real last_ts2 = -1;          /*  Global variable.  Time s never <0 used in heat_flux_meat */
real last_ts3 = -1;          /*  Global variable.  Time s never <0 used in mass_flux_meat */
int nfaces = 234;            /* faces elements on the surface */
int nnodes = 20;             /* number on nodes per face to calculate the  concentration inside the meat */
int niter;                   /* n iterations */
int lref = 116;              /* face reference number to print values */
int nitermax = 9;            /* niter to calculate fluxes */

/*******************************************************************************
**************/
```

```
/************************************** program controls
*****************************************************************/
real aweps = 0.0005;        /* coenvergence criterium to find the aw using newthon method */
int Radiation_Control = 1;/* 1 when take in account the radiation , 0, when do not */
real alfa = 1;              /* is the relaxation factor in the UDF that calculate the water on the interf */
real alfasc = 1;            /* is the relaxation factor to calculate the inverse Sc or ycstar */
real yeps = 0.01;           /* tolerance calculating ycstar */
int conwall = 1;            /* 1 define solving the wall, 2 wall approach */
int surf_meat_ID = 4;       /* ID zone of the wall in contact with the meat */
int surf_air_ID = 12;       /* ID zone of the wall in contact with the air */
int meat_ID = 2;            /* ID zone of the meat */
int air_ID = 3;             /* ID zone of the air */
/*******************************************************************************
*************************************/

/*******************************************************************************
*************************************/



real water_activity(real aw, real xm)
{

        real xmg = 0.0565;          /* GAB constant */
        real cg = 4.2396;           /* GAB constant */
        real k = 0.9692;            /* GAB constant */
        real f = 0.0488;            /* Lewicki constant */
        real g = 0.8761;            /* Lewicki constant */
        real h = -34.7794;          /* Lewicki constant */

        real fun = 10;              /* function equal to zero to calculate aw */
        real funder;                /* derivative of the function */


        do
        {
                if ( aw >= 0.958093)
                {
                        fun = xm - f / pow((1-aw),g) + f/(1+pow(aw,h));
                        funder = -f * g / pow((1-aw),(g+1)) -f * h *pow(aw,(h-1))
                        /pow((1+pow(aw,h)),2);
                }
                else
                {
                        fun = xm - xmg * cg * k * aw/((1-k*aw)*(1+  (cg-1)*k*aw));
                        funder = xmg * cg * k/( (1- k * aw) * ( 1 + (cg-1) *k * aw))*(-1 + aw * (cg-1)
                        * k / (1+ (cg-1)* k * aw)- k * aw /(1- k * aw));
                }


                aw = aw - fun/funder;

        } while (fabs(fun) >= aweps );


        return aw;
```

```
}

real InvSceff( real alfaso, real viseff, real vis)
{
        real alfas;

        real fun;       /* function on newton method to find alfa */
        real derfun;    /* derivate function on newton method to find alfa */

        alfas = alfaso;

        do
        {
                fun = pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), 0.6321) * pow(( (alfas +
                2.3929)/(alfaso + 2.3929)  ), 0.3679) - vis / viseff;
                derfun = pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), 0.6321) * 0.3679 * pow((
                (alfas + 2.3929)/(alfaso +  2.3929) ), -0.6321) + pow(( (alfas + 2.3929)/(alfaso
                + 2.3929) ), 0.3679) * 0.6321 * pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), -
                0.3679);
                alfas = alfas - alfasc*fun/derfun;

        } while (fabs(fun) > yeps );

        return alfas;

}

real ycstar_calc( real Sc, real Sceff, real ycstar, real Pc)
{

/* ycstart= nondimensional mass or thermal sublayer thickness */
        real fun;       /* function on newton method to find ycstar */
        real derfun;    /* derivate function on newton method to find ycstar */

        fun = Sc * ycstar - Sceff * (1 / kar * log(Ew * ycstar ) + Pc );

        do
        {

                fun = Sc * ycstar - Sceff * (1 / kar * log(Ew * ycstar ) + Pc );
                derfun =  Sc - Sceff / kar / ycstar;
                ycstar = ycstar - alfasc*fun / derfun;
```

```
        } while (fabs(fun) > yeps );


        return ycstar;



}



real massflux(face_t f, Thread *t)
{

        int i = 0;          /* define the first component =water */
        real Df;            /* define the effective diffusivity of water in air*/
        real Tf;            /* Face temperature */
        real yf;            /* define the wat. comp. on the face */
        real yc;            /* define the wat. comp. on the cell */
        real Ystar;         /* nondimensional composition */
        real yp;            /* yp = distance between the point p and the wall */
        real ystar;         /* ystart = nondimensional longitud from node to wall */
        real ycstar;        /* nondimensional mass sublayer thickness */
        real kp;            /* turb. kinetic energy dissipation rate at the cell point */
        real density;       /* density average of face and cell */
        real viseff;        /* effective viscosity */
        real vis;           /* viscosity */
        real Sc;            /* laminar schmidt mumber */
        real Sceff;         /* effective turbulent schmidt mumber */
        /*real Deff;        /* effective diffusivity of water */
        real Pc;            /* term in the wall aprx */
        real mass_flux;     /* define the mass flux on the wall (face) on Kg Water / s m2 */
        real A[ND_ND];      /* vector Area of the face */
        real Aunit[ND_ND];/* unit vector perpendicular to the area, same direction of the area vector */
        real Area;          /* Magnitud of the vector Area */
        real Cface[ND_ND];/* face centroid vector */
        real Ccell[ND_ND];/* cell centroid vector */
        real Diff[ND_ND];/* vector distance between the cell and face centroids */
        real es[ND_ND]; /* unit vector directin from the cell centroid to the face centroid */
        real dro[ND_ND];/* vector distance connecting the cell and face centroid */
        real ds;            /* distance between the cell centroide and face centroide */
        real A_by_es;
        real Zero[ND_ND];

        Domain *d;
        Thread *to;
        Thread *ta;
        cell_t co;


        Zero[0] = 0;
        Zero[1] = 0;
```

```
d = Get_Domain(1);

co = F_C0(f,t);                /* cell asociated to the face */
to = THREAD_T0(t);             /* cell thread for co */
ta = Lookup_Thread(d, surf_air_ID);/* Thread of the wall in contact with air*/
yf = F_YI(f, t, i);            /* define the wat. comp. on the face */
yc = C_YI(co, to, i);          /* define the wat. comp. on the cell */
Tf = F_T(f, t);                /* Face temperature */
Df = -2.01366e-5+1.53234e-7*Tf;  /* Diffusivity at face temp. */
density = C_R(co, to);         /* density calculate on the cell */
F_AREA(A, f, t);
Area = NV_MAG(A);              /* Area */
F_CENTROID(Cface, f, t);
C_CENTROID(Ccell, co, to);
NV_VV(dro, =, Cface, -, Ccell);  /* vector distance between the face and cell centroids */
ds = NV_MAG(dro);              /* distance between the face and cell centroids */
NV_V_VS(es, = ,Zero, + , dro, *, (1/ds));   /* unit vector directing from the cell centroid to the
                                               face centroid */
A_by_es = NV_DOT(A, A) / NV_DOT(A, es);


if ( conwall = 1)
{
        mass_flux = Df * (yf - yc) / ds * A_by_es / Area *density;
}

else
{
        NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));/* unit vector perpendicular to the
                                                       area, same direction of the area vector */

        /* calculation of Sc and Sceff */
        vis = C_MU_L(co,to);/* viscosity */
        kp = C_K(co, to);/* turb. kinetic energy dissipation rate at the cell point */
        yp = NV_DOT(dro, Aunit);/* yp = distance between the point  p and the wall */
        ystar = density * pow(Cmu , 0.25) * pow(kp , 0.5) * yp / vis;
        Sc = vis / density / Df;          /* laminar schmidt mumber */
        viseff = C_MU_EFF(co, to);        /* effective viscosity */
        Sceff = 1 / InvSceff(F_UDMI(f, ta, 2), viseff, vis);   /* effective turbulent schmidt
                                                                  number */
        F_UDMI(f, ta, 2) = 1/Sceff;/* effective turbulent schmidt number stored */

        /* calculation of Ycstar */
        Pc = 9.24* ( pow((Sc / Sceff), 0.75 ) - 1 ) * (1 + 0.28 * exp(-0.007* Sc / Sceff ) );
        ycstar = ycstar_calc( Sc, Sceff, F_UDMI(f,ta, 4), Pc);
        F_UDMI(f,ta, 4) = ycstar;

        if (ystar <= ycstar)
        {
                Ystar = Sc * ystar;
        }
        else
        {
                Ystar =  Sceff * (1 / kar * log(Ew * ystar ) + Pc );
        }
```

```
                mass_flux = (yf - yc ) / Ystar * density * pow(Cmu, 0.25) * pow (kp, 0.5);

        }

        return mass_flux;


}


real heatconvection(face_t f, Thread *t)
{

        real Tf;            /* define the face temperature */
        real Tc;            /* define the cell temperature */
        real Tstar;         /* nondimensional Temperature */
        real yp;            /* yp = distance between the point p and the  wall */
        real ystar;         /* ystart = nondimensional longitud from node to wall */
        real ycstar;        /* nondimensional mass sublayer thickness */
        real kp;            /* turb. kinetic energy dissipation rate at the cell point */
        real density;       /* density average of face and cell */
        real viseff;        /* effective viscosity */
        real vis;           /* viscosity */
        real Kair;          /* thermal conductivity air mix */
        real Cpair;         /* Cp mix of air */
        real Pr;            /* laminar Pr mumber */
        real Preff;         /* effective turbulent Pr mumber */
        real Pc;            /* term in the wall aprx */
        real heat_flux_conv;        /* define the heat flux by convection */
        real A[ND_ND];              /* vector Area of the face */
        real Aunit[ND_ND];          /* unit vector perpendicular to the area, same direction of the area
                                    vector */
        real Area;                  /* Magnitud of the vector Area */
        real Cface[ND_ND];          /* face centroid vector */
        real Ccell[ND_ND];          /* cell centroid vector */
        real Diff[ND_ND];           /* vector distance between the cell and face   centroids */
        real es[ND_ND];             /* unit vector directin from the cell centroid  to the face centroid */
        real dro[ND_ND];            /* vector distance connecting the cell and face centroid */
        real ds;                    /* distance between the cell centroide and face centroide */
        real A_by_es;
        real Zero[ND_ND];

        Domain *d;
        Thread *to;
        Thread *ta;
        cell_t co;


        Zero[0] = 0;
        Zero[1] = 0;


        d = Get_Domain(1);
```

```
co = F_C0(f,t);                    /* cell asociated to the face */
to = THREAD_T0(t);                 /* cell thread for co */
ta = Lookup_Thread(d, surf_air_ID);/* Thread of the wall in contact with air*/
Tf = F_T(f, t);                    /* define the face temperature */
Tc = C_T(co, to);                  /* define the cell temperature */
density = C_R(co, to);
Kair = C_K_L(co,to);               /* themal conductivity */
Cpair = C_CP(co,to);               /* Cp */
F_AREA(A, f, t);
Area = NV_MAG(A);                  /* Area */
F_CENTROID(Cface, f, t);
C_CENTROID(Ccell, co, to);
NV_VV(dro, =, Cface, -, Ccell);    /* vector distance between the face and cell centroids */
ds = NV_MAG(dro);                  /* distance between the face and cell cent.*/
NV_V_VS(es, = ,Zero, + , dro, *, (1/ds));/* unit vector directing from the cell centroid to the
                                   face centroid */
A_by_es = NV_DOT(A, A) / NV_DOT(A, es);


if (conwall =1)
{
        heat_flux_conv = Kair * (Tf - Tc) / ds * A_by_es / Area;        }

else
{
        NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));
        /* unit vector perpendicular to the area, same direction  of the area vector */

        /* calculation of Pr and Preff */
        vis = C_MU_L(co,to);       /* viscosity */
        kp = C_K(co, to);          /* turb. kinetic energy dissipation rate at the cell point */
        yp = NV_DOT(dro, Aunit);/* yp = distance between the point  p and the wall */
        ystar = density * pow(Cmu , 0.25) * pow(kp , 0.5) * yp / vis;
        Pr = vis * Cpair / Kair;                /* laminar Pr mumber */
        viseff = C_MU_EFF(co, to);              /* effective viscosity */
        Preff =  1 / InvSceff(F_UDMI(f, ta, 3), viseff, vis);    /* effective turbulent schmidt
                                                number */
        F_UDMI(f, ta, 3) = 1/ Preff;/           *effective turbulent Schmidt number  stored*/

        /* calculation of ycstar */
        Pc = 9.24* ( pow((Pr / Preff), 0.75 ) - 1 ) * (1 + 0.28 *exp(-0.007* Pr / Preff ) );
        ycstar = ycstar_calc( Pr, Preff, F_UDMI(f, ta, 5), Pc);
        F_UDMI(f, ta, 5) = ycstar;

        if (ystar <= ycstar)
        {
                Tstar = Pr * ystar;
        }
        else
        {
                Tstar =  Preff * (1 / kar * log(Ew * ystar ) + Pc );
        }



        heat_flux_conv = (Tf - Tc ) / Tstar * density * Cpair * pow(Cmu, 0.25) * pow (kp, 0.5);

}
```

```
        return heat_flux_conv;



}


real heattotal(face_t f, Thread *t)
{

        int l = 0;
        real T;                     /* Face Temperature  K */
        real mass_flux;             /* define the mass flux on the wall (face) on  Kg Water / s m2 */
        real heat_flux_total;       /* define the total heat flux on the wall (face) on W /  m2 */
        real heat_flux_conv;        /* define the convective heat flux on the wall (face) on W /  m2 */
        real heat_flux_rad;         /* define the radiative heat flux on the wall   (face) on W /  m2 */


        T = F_T(f,t);       /* Face Temperature  K */
        mass_flux = massflux(f,t);
        heat_flux_conv = heatconvection(f,t);
        heat_flux_rad =  Radiation_Control * E * sigma * (pow(To,4) -pow(T,4));
        heat_flux_total = - Hvap * mass_flux - heat_flux_conv + heat_flux_rad;

        return heat_flux_total;
}



DEFINE_ON_DEMAND(demand_initialization)
{
        Domain *d;
        int i = 0;          /* define the first component  = water */
        int j = 0;          /* define the first UDS for heat transport in   meat */
        int k = 1;          /* define the second UDS for mass transport in meat */
        Thread *tmeat;      /* Thread of meat zone */
        Thread *t;
        cell_t c;

        d = Get_Domain(1);
        tmeat = Lookup_Thread(d, meat_ID);          /* Thread of meat */

        begin_c_loop(c, tmeat)
        {
                C_YI(c, tmeat, i) = Yomeat;         /* water composition */
                C_UDSI(c, tmeat, j) = Tomeat;       /* T*Cpmeat  */
                C_T(c, tmeat) = Tomeat;             /* T  */
                C_UDSI(c, tmeat, k) = Yomeat;       /* water composition */
                C_K(c, tmeat) = 0;                  /* turb. knetic energy */
                C_U(c, tmeat) = 0;                  /* x velocity */
                C_V(c, tmeat) = 0;                  /* y velocity */


        }
        end_c_loop(c, tmeat)


        t = Lookup_Thread(d, air_ID);       /* Thread of air */
```

```
        begin_c_loop(c, t)
        {
                C_UDSI(c, t, j) = To;                   /* T  */
                C_UDSI(c, t, k) = yo;                   /* water composition */

        }
        end_c_loop(c, t)


}




DEFINE_ON_DEMAND(dem_init_fluxes)
{
        Domain *d;
        int m;              /* integer that define the number of nodes */

        Thread *tm;
        Thread *ta;
        face_t f;

        d = Get_Domain(1);



        tm = Lookup_Thread(d, surf_meat_ID);        /* Thread of the wall in contact with meat*/
        ta = Lookup_Thread(d, surf_air_ID);         /* Thread of the wall in contact with air*/

        begin_f_loop(f, tm)
         {

                /* extra declarations to store used values to acelerate  processing */
                /* position cero is ocupaid by heat flux total  */
                F_UDMI(f, ta, 0) = heattotal(f, ta)/ kmeat * Dtmeat *  denmeat;
                F_UDMI(f, ta, 1) = 0.9999;          /* water activity */
                F_UDMI(f, ta, 2) = 1/0.7;           /* alfasoc inverse of  effective scmith */
                F_UDMI(f, ta, 3) = 1/0.7;           /* inverse of effective prandatl */
                F_UDMI(f, ta, 4) = 11.2;            /* ycstarm mass boundary  layer thickness */
                F_UDMI(f, ta, 5) = 11.2;            /* ycstart thermal boundary layer thickness */
                F_UDMI(f, ta, 6) = massflux(f, ta);


         }
        end_f_loop(f,tm)

}



DEFINE_PROFILE(heat_flux_meat, tm, j)
{

        int l;
        real T;                     /* Face Temperature  K */
```

```
        real curr_ts;                   /* current time */
        real heat_flux_total;           /* define the total heat flux on the wall (face) on W /  m2 */
        face_t  f;                      /* face of the face loop , has the same number in the air or meat
                                        face*/
        Domain *domain;
        Thread *t;                      /* Thread of the surface in contact with air*/


        domain = Get_Domain(1);
        t = Lookup_Thread(domain, surf_air_ID);    /* Thread of the wall in contact with air*/


        curr_ts = CURRENT_TIME;
        if (last_ts2 != curr_ts)
        {
                last_ts2 = curr_ts;

                begin_f_loop(f, tm)
                {

                                F_PROFILE(f, tm, j) = F_UDMI(f,t,0);


                }
                end_f_loop(f,tm)

        }
        else
        {
                if (niter == nitermax)
                {
                        begin_f_loop(f, t)
                        {
                                F_UDMI(f,t,0) = heattotal(f,t)/ kmeat * Dtmeat * denmeat;

                                F_UDMI(f,t,6) = massflux(f, t);


                        }
                        end_f_loop(f,t)
                }

        }



}




DEFINE_PROFILE(mass_flux_meat, tm, j)
{


        int l;
        real curr_ts;       /* current time */
        real mass_flux;     /* define the mass flux on the wall (face) on  Kg Water / s m2 */
```

```
        face_t  f;          /* face of the face loop , has the same number in the air or meat face*/
        Domain *domain;
        Thread *t;                  /* Thread of the surface in contact with air*/

        domain = Get_Domain(1);
        t = Lookup_Thread(domain, surf_air_ID);    /* Thread of the wall in contact with air*/


        curr_ts = CURRENT_TIME;
        if (last_ts3 != curr_ts)
        {
                last_ts3 = curr_ts;

                begin_f_loop(f, tm)
                {

                                F_PROFILE(f, tm, j) = -F_UDMI(f,t,6);


                }
                end_f_loop(f,tm)

        }



}



DEFINE_PROFILE(water_air_interf, t, n)
{
        real T;             /* Face Temperature */
        real Pw;            /* face vapor presure */
        int i = 0;          /* define the first component  = water */
        int k = 1;          /* define the masstransfer meat UDS */
        int l;              /* number that identifie the face number*/
        real x;             /* molec. compos. water /air */
        real y;             /* weight comp. water / air */
        real yf;            /* define the wat. comp. on the air face */
        real ym;            /* define the wat. weight composition on the meat face */
        real xm;            /* water content of the meat on the interf.*/
        real aw;            /* water activity */
        real dy;            /* define the dy */
        face_t f;
        real c1 = -5800.2206;       /* cte of vapor presure */
        real c2 = 1.3914993;
        real c3 = -0.048640239;
        real c4 = 0.000041764768;
        real c5 = -0.000000014452093;
        real c6 = 6.5459673;
```

```
        Domain *domain;
        Thread *tm;                 /* Thread of the surface in contact with meat*/
        face_t fm;                  /* face of the surface in contact with meat */


        domain = Get_Domain(1);
        tm = Lookup_Thread(domain, surf_meat_ID);          /* Thread of the wall in contact with
                                                              meat*/
        l = 0;


        begin_f_loop(f, t)
         {

                T = F_T(f,t);       /* face temperature */
                Pw = exp( c1 / T + c2 + c3 * T + c4 * pow(T , 2) + c5 *pow(T, 3) + c6 * log(T) );
                                    /* vapor presure */
                ym = F_UDSI(f, tm, k);    /* define the wat. weight    composition on the meat face */
                xm = ym/(1-ym); /* water content of the meat on the  interface */
                aw = F_UDMI(f,t,1);
                aw = water_activity(aw, xm);          /* water activity */
                F_UDMI(f,t,1) = aw;
                x = aw * Pw / Pt;           /* molec. compos. water /air */
                yf = F_YI(f, t, i);         /* define the actual wat. weight  comp. on the air face */
                y = x * Mwater / (x * Mwater + (1 - x) * Mair);          /* define the new wat. weight
                                                                          comp. water / air */
                dy = y - yf;

                F_PROFILE(f, t, n) = yf + alfa * dy;



         }
        end_f_loop(f,t)


}



DEFINE_PROFILE(temperature_air_interf, t, n)
{

        int j = 0;                  /* first UDS - temperature */
        Domain *domain;
        Thread *tm;                 /* Thread of the surface in contact with meat */
        real T;                     /* Temperature */
        real Tnew;
        real Told;
        real dT;
        face_t f;

        domain = Get_Domain(1);
        tm = Lookup_Thread(domain, surf_meat_ID);          /* Thread of the wall in contact with
                                                              meat*/


        begin_f_loop(f, t)
```

```
        {
                Told = F_T(f, t);           /* air temperature  */
                Tnew =  F_UDSI(f,tm,j);  /* air temperature equal to meat temperature */
                dT = Tnew - Told;
                T = Told + alfa * dT;

                F_PROFILE(f, t, n) = T;   /* air temperature equal to meat temperature */

        }
        end_f_loop(f,t)



}




DEFINE_UDS_UNSTEADY(uds_time_heat, c, t, i, apu, su)
{

        real vol = C_VOLUME(c,t);/* volume of the cell*/
        real time = RP_Get_Real("physical-time-step");/* time interval */
        real phi_old = C_STORAGE_R(c, t, SV_UDSI_M1(i));/* uds in the previous time step  */

        *apu = -denmeat *vol/time;/* implicit part of the unsteady term */
        *su =     denmeat * vol / time * phi_old;/* explicit part of the unsteady term */



}



DEFINE_ADJUST(adjust_cylinder, d)
{
        int j = 0;          /* UDS of heat transfer equation on meat */
        int k = 1;          /* UDS of mass transfer equation on meat */
        real T;             /* cell temperature */
        real D;             /* water diffusivity on meat */
        real curr_ts;       /* current time */
        cell_t c;
        Thread *tm;         /* Thread of meat zone */
        tm = Lookup_Thread(d, meat_ID);/* Thread of the meat zone */

        begin_c_loop(c, tm)
        {
                T = C_UDSI(c, tm, j);                /* cell temperature*/
                D = 1.10e-6*exp(-2300/T)*denmeat;/* diffusivity in m2/s * density */
                C_UDSI_DIFF(c, tm, k) = D;/        * water diffusivity on meat */
        }
        end_c_loop(c, tm)


        curr_ts = CURRENT_TIME;
        if (last_ts != curr_ts)
        {
```

```
            last_ts = curr_ts;
            niter = 1;
        }
        else
        {
            niter = niter +1;
        }

}
```

A4. MATLAB PROGRAM THAT GENERATES THE JOURNAL FILE TO CONSTRUCT THE GEOMETRY IN GAMBIT

```
clear
weight= 108;        % in Kg
fg = 2;             % fatness grade
fatness = 6;        %' in mm

lenght = (0.0975 +0.4965*log(weight)-0.03567*fg)*1000;          % in mm
seglenght = segments(lenght);
Lean = 0.565;                                                   % constants to calculate the density
FatLnMt = 0.227;
Bone = 0.20764;
density = 1070 * Lean + 920 * FatLnMt + (1330 + 1410 + 1440 + 1560) / 4 * Bone;  %kg/m3
volume = weight/density

facvol = 0.9585;%0.9401; %0.9585;          %correction given the volume of the beef mesh in fluent

for section = 1:14;
    if section == 1
          cons = cons1(weight, fatness);
          [area(section), ellipse(section, :), fac] = area1(cons, facvol);
               elseif section == 2
          cons = cons2(weight, fatness);
          [area(section), ellipse(section, :), fac] = area1(cons, facvol);
               elseif section == 3
               cons = cons3(weight, fatness);
          [area(section), ellipse(section, :), fac] = area1(cons, facvol);
          elseif section == 4
          cons = cons4(weight, fatness);
        poly = polynom4;
        fcor = facvol*1.3;
        [area(section), fac, gext4, gint4 ] = area2(cons, poly, section, facvol, fcor);
      elseif section == 5
          cons = cons5(weight, fatness);
        poly = polynom5;
        fcor = facvol*1.0;
        [area(section), fac, gext5, gint5] = area2(cons, poly, section, facvol, fcor);
      elseif section == 6
          cons = cons6(weight, fatness);
        poly = polynom6;
        fcor = facvol*1.2;
        [area(section), fac, gext6, gint6] = area2(cons, poly, section, facvol, fcor);
      elseif section == 7
          cons = cons7(weight, fatness);
        poly = polynom7;
        fcor = facvol*1.0;
        [area(section), fac, gext7, gint7] = area2(cons, poly, section, facvol, fcor);
      elseif section == 8
          cons = cons8(weight, fatness);
        poly = polynom8;
        fcor = facvol*1.3;
        [area(section), fac, gext8, gint8] = area2(cons, poly, section, facvol, fcor);
      elseif section == 9
          cons = cons9(weight, fatness);
        poly = polynom9;
```

```
        fcor = facvol*1.4;
        [area(section), fac, gext9, gint9] = area2(cons, poly, section, facvol, fcor);
      elseif section == 10
          cons = cons10(weight, fatness);
        poly = polynom10;
        fcor = facvol*1.0;
        [area(section), fac, gext10, gint10] = area2(cons, poly, section, facvol, fcor);
      elseif section == 11
          cons = cons11(weight, fatness);
        poly = polynom11;
        fcor = facvol*1.4;
        [area(section), fac, gext11, gint11] = area2(cons, poly, section, facvol, fcor);
      elseif section == 12
          cons = cons11(weight, fatness);
        poly = polynom11;
        fcor = facvol*0.4;
        [area(section), fac, gext12, gint12] = area2(cons, poly, section, facvol, fcor);
      elseif section == 13
        cons = cons13(weight, fatness);
        [area(section), ellipse(section, :), fac] = area1(cons, facvol);
          elseif section == 14
          cons = cons14(weight, fatness);
          [area(section), ellipse(section, :), fac] = area1(cons, facvol);
          else
    end
end

areaavg(1) = (area(1) + 0.5*area(1))/2;
areaavg(2:12) = (area(2:12) + area(1:11))/2;
areaavg(13) = ( area(13) + area(14))/2;
seglenght = seglenght * facvol^(1/3);
volumecal = sum(areaavg.* seglenght )/(1000^3);      % in m3
volume = weight/density;                                  % in m3



% some graphic definitions
z1 = seglenght(1) + seglenght(2) + seglenght(3) + seglenght(4);
tetarot = 0;


npoint = 0;                          % account the total number of calculate points

jour = fopen('jour1.txt','w');

for section = 1:14
  if section == 1
    % creation of the ellipse. it needs: a, b, and the center, it create three points
    % in the first case it create two ellipses and the first one it is supose that has
    % a half area

        % minor and mayor axes of the ellipse
    a = ellipse(section,1);
    b = ellipse(section,2);

    ao = (2^-0.5)*a;
    bo = (2^-0.5)*b;

        % some graphic definitions to use in sdection 5, 6, 7
```

```
        slope1 = (seglenght(5) + seglenght(6) + seglenght(7))/(3*bo);


        % 1 center cordinates of the ellipse
        xo = 0;
        yo = ao;
        zo  = 0;
        center = [ xo yo zo ];
  npoint = pellipse(ao, bo, center, npoint, jour);

  xo = bo - b;
  yo = a;
  zo = zo + seglenght(section);
  zbig(section) = zo;
  center = [ xo yo zo ];
  centerbig(section, :) = center;
  npoint = pellipse(a, b, center, npoint, jour);

elseif (section == 2) | (section == 3)
   % minor and mayor axes of the ellipse
        a = ellipse(section,1);
  b = ellipse(section,2);

  % 1 center cordinates of the ellipse
        yo = a;
  xo = bo - b;
  zo  = zo + seglenght(section);
  zbig(section) = zo;
  center = [ xo yo zo ];
  centerbig(section, :) = center;
  npoint = pellipse(a, b, center, npoint, jour);

elseif section == 4
  % geometry 4
  zo  = zo + seglenght(section);
  zbig(section) = zo;
  % position on  line
  next = size(gext4,1);
  nint = size(gint4,1);
  xc = gext4(next,1)-bo ;
  gext4(:,1) = gext4(:,1)-xc;
  gint4(:,1) = gint4(:,1)-xc;
  yc = gext4(1,2);
  gext4(:,2) = gext4(:,2)-yc;
  gint4(:,2) = gint4(:,2)-yc;
  npoint = censection(gext4, gint4, npoint, zo, jour);
elseif section == 5
  % geometry 5
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext5,1);
        nint = size(gint5,1);
  xc = gext5(next,1)-bo - (zo - z1)/slope1;
  gext5(:,1) = gext5(:,1)-xc;
  gint5(:,1) = gint5(:,1)-xc;
  yc = gext5(1,2);
  gext5(:,2) = gext5(:,2)-yc;
  gint5(:,2) = gint5(:,2)-yc;
```

```
  npoint = censection(gext5, gint5, npoint, zo, jour);
   elseif section == 6
  % geometry 6
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext6,1);
        nint = size(gint6,1);
  xc = gext6(next,1)-bo - (zo - z1)/slope1;
  gext6(:,1) = gext6(:,1)-xc;
  gint6(:,1) = gint6(:,1)-xc;
  yc = gext6(1,2);
  gext6(:,2) = gext6(:,2)-yc;
  gint6(:,2) = gint6(:,2)-yc;
        b6 = gext6(1,1);                    % to use in the geometry 7, 8, 9 , 10
        npoint = censection(gext6, gint6, npoint, zo, jour);
   elseif section == 7
  % geometry 7
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext7,1);
        nint = size(gint7,1);
  xc = gext7(1,1)+bo + abs(b6) +0.8*bo;
  gext7(:,1) = gext7(:,1)-xc;
  gint7(:,1) = gint7(:,1)-xc;
  yc = gext7(1,2);
  gext7(:,2) = gext7(:,2)-yc;
  gint7(:,2) = gint7(:,2)-yc;
   npoint = censection(gext7, gint7, npoint, zo, jour);
   elseif section == 8
  % geometry 8
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext8,1);
        nint = size(gint8,1);
  xc = gext8(1,1)+bo + abs(b6) +0.87*bo;
  gext8(:,1) = gext8(:,1)-xc;
  gint8(:,1) = gint8(:,1)-xc;
  yc = gext8(1,2);
  gext8(:,2) = gext8(:,2)-yc;
  gint8(:,2) = gint8(:,2)-yc;
  npoint = censection(gext8, gint8, npoint, zo, jour);
   elseif section == 9
  % geometry 9
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext9,1);
        nint = size(gint9,1);
  xc = gext9(1,1)+bo + abs(b6) + 1.07*bo;
  gext9(:,1) = gext9(:,1)-xc;
  gint9(:,1) = gint9(:,1)-xc;
  yc = gext9(1,2);
  gext9(:,2) = gext9(:,2)-yc;
  gint9(:,2) = gint9(:,2)-yc;
  npoint = censection(gext9, gint9, npoint, zo, jour);
elseif section == 10
```

```
  % geometry 10
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        z10 = zo;                          % to be use on the next secction
  % position on  line
  next = size(gext10,1);
        nint = size(gint10,1);
  xc = gext10(1,1)+bo + abs(b6) + 0.93*bo;
  gext10(:,1) = gext10(:,1)-xc;
  gint10(:,1) = gint10(:,1)-xc;
  yc = gext10(1,2);
  gext10(:,2) = gext10(:,2)-yc;
  gint10(:,2) = gint10(:,2)-yc;
  npoint = censection(gext10, gint10, npoint, zo, jour);
  % information for the foreleg
  ref10 = [gext10(3,1) gext10(3,2) zo];
  ref10b = [gext10(4,1) gext10(4,2) zo];
elseif section == 11
  % geometry 11
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext11,1);
        nint = size(gint11,1);
  xc = gext11(1,1)+bo + abs(b6) + 0.93*bo;                % same position x of the latter seccion
  gext11(:,1) = gext11(:,1)-xc;
  gint11(:,1) = gint11(:,1)-xc;
  yc = gext11(1,2);
  gext11(:,2) = gext11(:,2)-yc;
  gint11(:,2) = gint11(:,2)-yc;
  npoint = censection(gext11, gint11, npoint, zo, jour);
  zref11 = zo;
elseif section == 12
  % geometry 12
  zo  = zo + seglenght(section);
  zbig(section) = zo;
        % position on  line
  next = size(gext12,1);
        nint = size(gint12,1);
  xc = gext12(1,1)+bo + abs(b6) + 1.8*bo;             % same position x of the latter seccion
  gext12(:,1) = gext12(:,1)-xc;
  gint12(:,1) = gint12(:,1)-xc;
  yc = gext12(1,2);
  gext12(:,2) = gext12(:,2)-yc;
  gint12(:,2) = gint12(:,2)-yc;
  npoint = censection(gext12, gint12, npoint, zo, jour);

elseif section == 13
   % minor and mayor axes of the ellipse
        ae = ellipse(section,1);
  be = ellipse(section,2);

   movx = 80;          %it is the movementn of x drawing the ellipse

  % the process to made the ellipse is to choose three points (1 ,2 ,3)
  % then to create a plane whit the points
  % to place the center of the ellipse in the half way between points 2 an 3
  % the mayor ax of the ellipse is in the direction  2 - 3
  % then to determine the cordiantes of the the focus
```

```
% determine the points of the ellipse whit the equation of the ellipse and the equation of the plane


%1) Determination of the point 1 which comes from the section 10
      x1 = gext10(4,1);
y1 = gext10(4,2);
z1  = z10;

% defining the point 2 as the last point on the section 11
x2 = gext11(3,1);
y2 = gext11(3,2);
z2 = zref11;

%3) Determination of the point 3 which comes from the section 10
      x3 = gext10(5,1);
y3 = gext10(5,2) + 2.5*ae;
z3  = z10;

      %4) creation of the plane:
% fistly it was created the normal vector of the plane which is the cross product between
% the vector difference between the points 3 - 1 and 3 -2
% diff1 = (x3 - x1)i + (y3 - y1)j + (z3 - z1)k
% diff2 = (x3 - x2)i + (y3 - y2)j + (z3 - z2)k
% the perpendicular vector is the cross product of diff1 and diff2 and is:
% per = ai + bj + ck where:
% a = (y3-y1)*(z3-z2) - (y3-y2)*(z3-z1)
% b = (x3-x1)*(z3-z2) - (x3 - x2)*(z3-z1)
% c = (x3 - x1)*(y3-y2) - (x3-x2)*(y3-y1)

a = (y3 - y1)*(z3 - z2) - (y3 - y2)*(z3 - z1);
b = -( (x3 - x1)*(z3 - z2) - (x3 - x2)*(z3 - z1));
c = (x3 - x1)*(y3 - y2) - (x3 - x2)*(y3 - y1);

% The ecuation of the plane is given by the equation :
% [ (x - x3)i + (y - y3)j + (z - z3)k ] . (ai + bj +ck) = 0
% thus, the plane is :
% ax + by + cz = ax3 + by3 + cz3

% 5) the center of the ellipse is given by:
dist12 = ((x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2)^0.5;% distance between points 1 and 2
x0 = x1 - be*(x1 - x2)/dist12;
      y0 = y1 - be*(y1 - y2)/dist12;
z0 = z1 - be*(z1 - z2)/dist12;
center = [(x0-movx) y0 z0];


% c is given by:
ce = (a^2 + b^2)^0.5;          %it can be use to find the focus

% the distance between 2 and 3 is:
no = ( (x3 - x2)^2 + (y3 - y2)^2 + (z3 - z2)^2 )^0.5;

%6) the two points on the mayor axes are
%6) the point on the mayor axes is going to be the point 1

xp1 = x0 - be*(x3 - x2)/no;
yp1 = y0 - be*(y3 - y2)/no;
zp1 = z0 - be*(z3 - z2)/no;
```

```
xp2 = x0 + be*(x3 - x2)/no;
yp2 = y0 + be*(y3 - y2)/no;
zp2 = z0 + be*(z3 - z2)/no;

point1 = [(x1-movx) y1 z1];

% 7) The cross product between diff2 = (x3 - x2)i + (y3 - y2)j + (z3 - z2)k
%and the perpendicular vector per = ai + bj + ck  gives another
% perpendicular vector which can be use to calculate the 2 points in the minor axes
% the vector is given by vecax2 = di + ej + fk  where:

d = c * (y3 - y2) - b*(z3 - z2);
e = -( c * (x3 - x2) - a*(z3 - z2));
f = b*(x3 - x2) - a*( y3 - y2);

% the norm of this vector is:

nor2 = (d^2 + e^2 + f^2)^0.5;

% the two points on the minor axes are
xp3 = x0 + ae*d/nor2;
yp3 = y0 + ae*e/nor2;
zp3 = z0 + ae*f/nor2;

xp4 = x0 - be*d/nor2;
yp4 = y0 - be*e/nor2;
zp4 = z0 - be*f/nor2;

      point2 = [(xp3-movx) yp3 zp3];

      % the points are going to be allocated in a matrix
xp = [xp1; xp3; xp2; xp4];
yp = [yp1; yp3; yp2; yp4];
zp = [zp1; zp3; zp2; zp4];

gext13 = [xp yp zp];

      %npoint = censection2(gext13, npoint, jour)
      npoint = pellipse2(center,point1, point2, npoint, jour);

elseif section == 14
      % minor and mayor axes of the ellipse
      ae = ellipse(section,1);
be = ellipse(section,2);

      % the section 14 is place at the distance h, which is the lenght
      % of the of the foreleg, from the center of the previous ellipse on the perpendicualr direction
% of plane of the ellipse

h = seglenght(13);      % height of the foreleg

% the middle point is the center of the previous ellipse:

p4 = center;

% the perpendicular vector to that plane is givne by per = ai + bj + ck
%5 the norm of that vector is:
norper = (a^2 + b^2 + c^2)^0.5;
```

```
% and the unit perpendicular vector is:
a = a/norper;
b = b/norper;
      c = c/norper;

      % the center of the new ellpse is prjecting the previous center (p4) in the direction of the
% perpendicular vector a distance equal to h. it can be done solving:
% (ha)i + (hb)j + (hc)k = (x5 -x4)i + (y5-y4)j +(z5-z4)k.. where p5 = center = (x5, y5, z5) and
% p4 = (x4, y4, z4), thus,
x5 = h*a +p4(1);
y5 = h*b +p4(2);
      z5 = h*c +p4(3);

center = [x5 y5 z5];

% the point on the mayor axe is calculate with the direction between point (xp1, yp1, zp1) and
% (xp2, yp2,zp2). the direction unit vector is :
a2 = (xp2 - xp1);
b2 = (yp2 - yp1);
      c2 = (zp2 - zp1);

% the norm is
nor3 = (a2^2 +b2^2 + c2^2)^0.5;

% and the unit vector is:
a2 = a2 / nor3;
b2 = b2 / nor3;
c2 = c2 / nor3;

% half mayor diameter is be/2 . the equation to solve is:
% (be/2*a2)i + (be/2*b2)j + (be/2*c2)k = (x6 -x5)i + (y6-y5)j +(z6-z5)k.. where p5 = center = (x5,
y5, z5) and
% p6 = (x6, y6, z6) is the point in the mayor axe
      x6 = be/2*a2+x5;
 y6 = be/2*b2+y5;
z6 = be/2*c2+z5;

p6 = [x6, y6, z6];

      % the perpendicular vector in the direction of the minor axe is
% vecax2 = di + ej + fk  and the norm is nor2, thus, the unit vector in that dirrection is:
d = d/nor2;
e = e/nor2;
f = f/nor2;

      % half mayor diameter is ae/2 . the equation to solve is:
% (ae/2*d)i + (ae/2*e)j + (ae/2*f)k = (x7 -x5)i + (y7-y5)j +(z7-z5)k.. where p5 = center = (x5, y5, z5)
and
% p7 = (x7, y7, z7) is the point in the minor axe
      x7 = ae/2*d +x5;
 y7 = ae/2*e +y5;
z7 = ae/2*f +z5;

p7 = [x7, y7, z7];


      npoint = pellipse2(center, p6, p7, npoint, jour);
```

```
  else

  end

end

% selection of the solver
a1 = 'solver select "FLUENT 5/6"';
fprintf(jour,a1);
fprintf(jour,'\n',");


% now the fist 4 ellipses are going to be slpit out
a1 = 'edge split "edge.1" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.34" parameter 0.3 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.2" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.36" parameter 0.3 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.3" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.38" parameter 0.3 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.4" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.40" parameter 0.3 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");

% Split of the two ellipses of the foreleg
a1 = 'edge split "edge.32" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.33" parameter 0.5 connected';
fprintf(jour,a1);
fprintf(jour,'\n',");



% Creation of the 5 conector edges
a1 = 'edge create nurbs "vertex.129" "vertex.123" "vertex.114" "vertex.99" "vertex.80" "vertex.59"
"vertex.38" "vertex.22" "vertex.13" "vertex.147" "vertex.145" "vertex.143" "vertex.141" interpolate';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge create nurbs "vertex.132" "vertex.126" "vertex.119" "vertex.108" "vertex.91" "vertex.70"
"vertex.49" "vertex.31" "vertex.18" "vertex.148" "vertex.146" "vertex.144" "vertex.142" interpolate';
```

```
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge create nurbs "vertex.131" "vertex.125" "vertex.118" "vertex.107" "vertex.90" "vertex.69"
"vertex.48" "vertex.30" "vertex.17" "vertex.11" "vertex.8" "vertex.5" "vertex.2" interpolate';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge create straight "vertex.136" "vertex.139"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge create straight "vertex.149" "vertex.150"';
fprintf(jour,a1);
fprintf(jour,'\n',");


% creation of the seven surfaces
a1 = 'face create uedges "edge.44" "edge.45" udirections 0 0 vedges "edge.31" "edge.28" "edge.25"
"edge.22" "edge.19" "edge.16" "edge.13" "edge.10" "edge.7" "edge.40" "edge.38" "edge.36" "edge.34"
vdirections 0 0 0 0 0 0 0 0 0 0 0 0 0 net';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create uedges "edge.45" "edge.46" udirections 0 0 vedges "edge.30" "edge.27" "edge.24"
"edge.21" "edge.18" "edge.15" "edge.12" "edge.9" "edge.6" "edge.41" "edge.39" "edge.37" "edge.35"
vdirections 0 0 0 0 0 0 0 0 0 0 0 0 0 net';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create uedges "edge.44" "edge.46" udirections 0 0 vedges "edge.29" "edge.26" "edge.23"
"edge.20" "edge.17" "edge.14" "edge.11" "edge.8" "edge.5" "edge.4" "edge.3" "edge.2" "edge.1"
vdirections 0 0 0 0 0 0 0 0 0 0 1 1 1 1 net';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create wireframe "edge.29" "edge.30" "edge.31" real';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create wireframe "edge.1" "edge.34" "edge.35" real';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create wireframe "edge.42" "edge.32" real';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create wireframe "edge.43" "edge.33" real';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create skin "edge.42" "edge.43" directions 0 0';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'face create skin "edge.32" "edge.33" directions 0 0';
fprintf(jour,a1);
fprintf(jour,'\n',");


%creation of the two volumes
a1 = 'volume create stitch "face.1" "face.2" "face.3" "face.4" "face.5" real';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'volume create stitch "face.6" "face.7" "face.8" "face.9" real';
fprintf(jour,a1);
fprintf(jour,'\n',");

% union of the volumes 1 and 2
```

```
a1 = 'volume unite volumes "volume.1" "volume.2"';
fprintf(jour,a1);
fprintf(jour,'\n',");


% creation of the wind tunnel
a1 = 'volume create width 1100 depth 650 height 2775 brick';
fprintf(jour,a1);
fprintf(jour,'\n',");

% now the volume is going to be centered
longx = gext10(5,1) - gext5(1,1);
xcent = (1100 - longx)/2;
xmove = (gext10(5,1) + xcent) - 1100/2;
longy = ellipse(3,1);
ycent = (650 - longy)/2;
ymove = 650/2 - ycent;
longz = lenght;
zcent = (2775 - longz)/2;
zmove = 2775/2 - zcent;
a2 = 'volume move "volume.2" offset ';
carx = int2str(xmove);
cary = int2str(ymove);
carz = int2str(zmove);

a1 = [a2 blanks(1) carx blanks(1) cary blanks(1) carz];
fprintf(jour,a1);
fprintf(jour,'\n',");


% substract of volume
a1 = 'volume subtract "volume.2" volumes "volume.1" ';
fprintf(jour,a1);
fprintf(jour,'\n',");

%re-construction of the beef volume
a1 = 'volume create stitch "face.1" "face.2" "face.3" "face.4" "face.5" "face.7" "face.8" "face.10" real';
fprintf(jour,a1);
fprintf(jour,'\n',");


% delete of extra vertex
a1 = 'vertex delete "vertex.3" "vertex.1" "vertex.6" "vertex.4"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.9" "vertex.7" "vertex.12" "vertex.10"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.14" "vertex.15" "vertex.16"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.19" "vertex.20" "vertex.21"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.23" "vertex.24" "vertex.25" "vertex.26" "vertex.27" "vertex.28" "vertex.29"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.32" "vertex.33" "vertex.34" "vertex.35" "vertex.36" "vertex.37"';
fprintf(jour,a1);
```

```
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.39" "vertex.40" "vertex.41" "vertex.42" "vertex.43" "vertex.44" "vertex.45"
"vertex.46" "vertex.47"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.50" "vertex.51" "vertex.52" "vertex.53" "vertex.54" "vertex.55" "vertex.56"
"vertex.57" "vertex.58"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.60" "vertex.61" "vertex.62" "vertex.63" "vertex.64" "vertex.65" "vertex.66"
"vertex.67" "vertex.68"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.71" "vertex.72" "vertex.73" "vertex.74" "vertex.75" "vertex.76" "vertex.77"
"vertex.78" "vertex.79"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.81" "vertex.82" "vertex.83" "vertex.84" "vertex.85" "vertex.86" "vertex.87"
"vertex.88" "vertex.89"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.92" "vertex.93" "vertex.94" "vertex.95" "vertex.96" "vertex.97" "vertex.98"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.100" "vertex.101" "vertex.102" "vertex.103" "vertex.104" "vertex.105"
"vertex.106"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.109" "vertex.110" "vertex.111" "vertex.112" "vertex.113"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.115" "vertex.116" "vertex.117"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.135" "vertex.120" "vertex.121" "vertex.122"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.137" "vertex.138" "vertex.140"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.124" "vertex.127" "vertex.128"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'vertex delete "vertex.130" "vertex.133" "vertex.134"';
fprintf(jour,a1);
fprintf(jour,'\n',");


% delete of the extra parte of the foreleb edge
a1 = 'edge split "edge.47" vertex "vertex.151" connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge delete "edge.47" lowertopology';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge split "edge.48" vertex "vertex.152" connected';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge delete "edge.48" lowertopology';
```

```
fprintf(jour,a1);
fprintf(jour,'\n',");

%delete of the other edges
a1 = 'edge delete "edge.2" "edge.36" "edge.37" "edge.3" "edge.38" "edge.39" "edge.4" "edge.40"
"edge.41" lowertopology';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge delete "edge.5" "edge.7" "edge.6" "edge.8" "edge.10" "edge.9" "edge.11" "edge.13" "edge.12"
lowertopology';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge delete "edge.14" "edge.16" "edge.15" "edge.17" "edge.19" "edge.18" "edge.20" "edge.22"
"edge.21" lowertopology';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge delete "edge.23" "edge.25" "edge.24" "edge.26" "edge.28" "edge.27" lowertopology';
fprintf(jour,a1);
fprintf(jour,'\n',");


% boundary layer in the air and meat
a1 = 'blayer create first 0.1 growth 1.329 total 4.920659 rows 10 transition 1 trows 0 continuous wedge';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'blayer attach "b_layer.1" volume "volume.2" "volume.2" "volume.2" "volume.2" "volume.2"
"volume.2" "volume.2" "volume.2" face "face.1" "face.2" "face.3" "face.4" "face.5" "face.7" "face.8"
"face.10"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'blayer create first 1 growth 1 total 1 rows 1 transition 1 trows 0 continuous wedge';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'blayer attach "b_layer.2" volume "volume.3" "volume.3" "volume.3" "volume.3" "volume.3"
"volume.3" "volume.3" "volume.3" face "face.1" "face.2" "face.3" "face.4" "face.5" "face.7" "face.8"
"face.10"';
fprintf(jour,a1);
fprintf(jour,'\n',");

% mesh of the  smallest edges
a1 = 'edge mesh "edge.43" "edge.33" successive ratio1 1 size 5';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge mesh "edge.35" "edge.34" "edge.1" successive ratio1 1 size 5';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge mesh "edge.54" "edge.53" successive ratio1 1 size 10';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge mesh "edge.56" "edge.55" successive ratio1 1 size 10';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'edge mesh "edge.30" "edge.29" "edge.31" successive ratio1 1 size 10';
fprintf(jour,a1);
fprintf(jour,'\n',");


% meashing faces of the beef
a1 ='face mesh "face.7" triangle size 5';
```

```
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.5" triangle size 5';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.8" triangle size 10';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.10" triangle size 10';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.4" triangle size 10';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.1" triangle size 20';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.2" triangle size 20';
fprintf(jour.a1);
fprintf(jour,'\n','');
a1 ='face mesh "face.3" triangle size 20';
fprintf(jour,a1);
fprintf(jour,'\n','');


% meshing of the volume 2 air
a1 = 'face mesh "face.15" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'face mesh "face.16" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'face mesh "face.12" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'face mesh "face.11" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'face mesh "face.13" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'face mesh "face.14" triangle size 60';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = 'volume mesh "volume.2" tetrahedral size 5';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = '';
fprintf(jour,a1);
fprintf(jour,'\n','');
a1 = '';
fprintf(jour,a1);
fprintf(jour,'\n','');

% meshing of the volumes 3 meat
a1 = 'volume mesh "volume.3" tetrahedral size 20';
fprintf(jour,a1);
fprintf(jour,'\n','');
```

```
a1 = 'physics create "wall" btype "WALL" face "face.11" "face.12" "face.13" "face.14"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'physics create "inflow" btype "INLET_VENT" face "face.15"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'physics create "outflow" btype "OUTFLOW" face "face.16"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'physics create "Beef" ctype "SOLID" volume "volume.3"';
fprintf(jour,a1);
fprintf(jour,'\n',");
a1 = 'physics create "Air" ctype "FLUID" volume "volume.2"';
fprintf(jour,a1);
fprintf(jour,'\n',");

fclose(jour);

function [area, ellipse, fac] = area1(cons, facvol)

  a = ( cons(1)*facvol^(1/3) )/2;
  b = ( cons(2)*facvol^(1/3) )/2;
  area = pi*a*b;
 fac = facvol;
ellipse = [a b];

function [area, fac, gext, gint] = area2(cons, poly, section, facvol, fcor)


  if section == 4 | section == 9
    colums = size(cons,2);          % number of elements on the vector cons
    colmsp = size(poly,2);          % number of elements on the vector poly
     p = cons(colums-1);                % perimeter
    alfa = cons(colums);                % alfa angle
    xmax = poly(colmsp);                % this is the maxmimun value of x which is gonna be
                                        found
    xmin = poly(colmsp-1);              % minimun value of x
    pol = poly(1:(colmsp-2));           % polinom
    ymin = polyval(pol,xmin);
    der = polyder(pol);                 % derivate of the polynom
    ndiv = 100;                         % number of divison to calculate the integral

    % this section is a bisection routine to find the xmax that match the end of the polynom whit the
     projection line
    % at the angle alfa of the begining of the polynom.
    % alfa is the angle respect the perpendicular, so beta is alfa respect the fix cordinate system
    % the slop of the tangent line at the begining of the polynom is:
    slope = polyval(der,xmin);
    derpol = polyval(der,xmin);
    if derpol > 0
      xvec = 1 / ( ( 1 + (1/derpol)^2) ^0.5 );
      yvec = -1 /(derpol*(1 + (1/derpol)^2)^0.5);
    else
      xvec = -1 / ( ( 1 + (1/derpol)^2) ^0.5 );
      yvec = 1 /(derpol*(1 + (1/derpol)^2)^0.5);
    end
    beta = atan2(yvec, xvec);
```

```
    gama = beta - (alfa - 90) * ( 2 * pi / 360);
    ymax = polyval(pol,xmax);
    func = atan((ymax-ymin)/(xmax-xmin))-gama;

    %initialization of the method
    xleft = xmax;
    yleft = func;
    xright = 310;
    ymaxr = polyval(pol,xright);
    yright = atan((ymaxr-ymin)/(xright-xmin))-gama;

          % bisection method
          count = 0;
    while (xright - xleft)> 0.1 & count < 100
      count = count +1;
      xmiddle = (xright+xleft)/2;
      ymaxm = polyval(pol,xmiddle);
      ymiddle = atan((ymaxm-ymin)/(xmiddle-xmin))-gama;
      if (ymiddle*yleft) >0
        xleft = xmiddle;
        yleft = ymiddle;
      else
        xright = xmiddle;
        yright = ymiddle;
      end
    end
    xmax = (xright+xleft)/2;
    poly(colmsp)= xmax;
    fac = scalelenght(poly, ndiv, p); % correction factor to scale the polynom
    fac = fac*(fcor)^(1/3);                % new fac correction for the poly
    poly = polyscaled(poly, fac);              % scale the polymer


    [gext, gint, area] = pointspol3(cons, poly, ndiv,section, facvol, 0);

     else
     colums = size(cons,2);                          % number of elements on the vector
       p = cons(colums-1);                           % perimeter
     alfa = cons(colums);                            % alfa angle
     ndiv = 100;                                         % number of divison to calculate the integral
     fac = scalelenght(poly, ndiv, p);      % correction factor to scale the polynom
     fac = fac*(fcor)^(1/3);                    % new fac correction for the poly
     poly = polyscaled(poly, fac);              % scale the polymer

    [gext, gint, area] = pointspol2(cons, poly, ndiv,section, facvol, 0);
  end

function npoint = censection2(gext, npoint, jour)

next= size(gext,1);
% this marks the initial and final points of the external and internal curve
npinext = npoint+1;
npfinext = npoint+next;
txnpinext = num2str(npinext,'%1.0f');
txnpfinext = num2str(npfinext,'%1.0f');


texver = '';              % first value of the string with the vertex information
```

```
for i = 1:next
  npoint = npoint +1;
  xext = gext(i,1);
  yext = gext(i,2);
  zext = gext(i,3);
  printpoint(xext, yext, zext, npoint, jour);
  pointext = num2str(npoint,'%1.0f');
  texver = strcat(texver, ' "vertex.',pointext,'"');

end

% in this part texver is join to the first point
pointext = num2str((npoint-next+1),'%1.0f');
texver = strcat(texver, ' "vertex.',pointext,'"');

cartx = strcat('edge create nurbs', texver, ' interpolate');
fprintf(jour,cartx);
fprintf(jour,'\n',");

function npoint = censection(gext, gint, npoint, zo, jour)

% the first ndat points are fron the external side and the others
% are from the internal

next= size(gext,1);
nint = size(gint,1);
% this marks the initial and final points of the external and internal curve
npinext = npoint+1;
npfinext = npoint+next;
npinint = npfinext +1;
npfinint = npfinext + nint;
txnpinext = num2str(npinext,'%1.0f');
txnpfinext = num2str(npfinext,'%1.0f');
txnpinint = num2str(npinint,'%1.0f');
txnpfinint = num2str(npfinint,'%1.0f');



texver = '';                % first value of the string with the vertex information

for i = 1:next
  npoint = npoint +1;
  xext = gext(i,1);
  yext = gext(i,2);
  zext = zo;
  printpoint(xext, yext, zext, npoint, jour);
  pointext = num2str(npoint,'%1.0f');
  texver = strcat(texver, ' "vertex.',pointext,'"');

end
nendpext = npoint;
cartx = strcat('edge create nurbs', texver, ' interpolate');
fprintf(jour,cartx);
fprintf(jour,'\n',");


texver = '';

function cons = cons1(weight, fatness)
```

```
    a = 82.264 + 0.24674 * weight;%minor axis
    b = 129.81 + 0.30763 * weight;  %major axis
  p = 330.99 + 0.81783*weight;          %outer perimeter

  cons = [a, b, p];
function cons = cons2(weight, fatness)

    a = 166.201 + 0.39938 * weight;    %minor axis
    b = 221.8 + 0.40322 * weight;  %major axis
  p = 632.98 + 0.9456 * weight - 0.8058 * fatness;          %outer perimeter

  cons = [a, b, p];


function cons = cons3(weight, fatness)

    a = 160.748 + 0.74349 * weight + 0.3928 * fatness;   %minor axis
    b = 292.08  + 0.87735 * weight - 0.5401 * fatness;  %major axis
  p = 770.06 + 2.228 * weight - 0.933 * fatness;        %outer perimeter

  cons = [a, b, p];
function cons = cons4(weight, fatness)

    cons(1) = 201.89 + 0.6372 * weight + 0.0024 * fatness;
  cons(2) = 135.65 + 0.4241 *weight + 1.2627 * fatness;
  cons(3) = 36.85 + 0.5737 * weight + 0.717 * fatness;
  cons(4) = 32.2 + 0.544 * weight + 0.8358 * fatness;
  cons(5) = 34.088;
  p = 539.55 + 1.7516 *weight;          %outer perimeter
  alfa = 60;
  cons(6) = p;
  cons(7) = alfa;
function cons = cons5(weight, fatness)

    cons(1) = 109.24 + 0.5203 * weight + 0.2425 * fatness;
  cons(2) = 108.18 + 0.4266 * weight + 1.0421 * fatness;
  cons(3) = 89.47 + 0.2633 * weight + 0.8848 * fatness;
  cons(4) = 37.71 + 0.352 * weight + 0.6213 * fatness;
  cons(5) = 8.89 + 0.383 * weight + 1.7852 * fatness;
  cons(6) = - 18.08 + 0.4591 * weight + 2.0941 * fatness;
  cons(7) = 19.197 + 0.19119 * weight + 0.24916 * fatness;
  cons(8) = 11.296 + 0.06397 * weight + 0.4121 * fatness;
    cons(9) = 25.5147;
  p = 466.83 + 1.9834 * weight ;          %outer perimeter
  alfa = 97;
  cons(10) = p;
  cons(11) = alfa;

function cons = cons6(weight, fatness)

    cons(1) = 102.934 + 0.3872 * weight - 0.0611 * fatness;
  cons(2) = 76.71 + 0.2272 * weight + 0.942 * fatness;
  cons(3) = 33.13 + 0.1965 * weight + 0.7674 * fatness;
  cons(4) = 12.31 + 0.1614 * weight + 1.6602 * fatness;
  cons(5) = -7.019 + 0.2573 * weight + 2.0101 * fatness;
  cons(6) = 2.718 + 0.2501 * weight + 1.7633 * fatness;
  cons(7) = 15.849 + 0.1543 * weight + 0.8935 * fatness;
  cons(8) = 16.351 + 0.1542 * weight + 0.6634 * fatness;
```

```
  cons(9) = 22.01 + 0.09561 * weight + 0.9348 * fatness;
  cons(10) = 35.417 - 0.0523 * weight + 0.553 * fatness;
  cons(11) = 19.0882;
  p = 474.81 + 1.8568 * weight ;          %outer perimeter
  alfa = 114;
  cons(12) = p;
  cons(13) = alfa;
function cons = cons7(weight, fatness)

    cons(1) = 115 + 0.368 * weight - 0.322 * fatness;
  cons(2) = 64.4 + 0.288 * weight + 0.619 * fatness;
  cons(3) = 29.47 + 0.183 * weight + 1.28 * fatness;
  cons(4) = 16.3 + 0.0886 * weight + 1.24 * fatness;
  cons(5) = -8.34 + 0.264 * weight + 0.869 * fatness;
  cons(6) = 6.97 + 0.135 * weight + 0.948 * fatness;
  cons(7) = 5.5 + 0.167 * weight + 1.06 * fatness;
  cons(8) = -1.17 + 0.335 * weight + 1 * fatness;
  cons(9) = 9.31 + 0.395 * weight + 0.821 * fatness;
  cons(10) = 27.7 + 0.317 * weight + 0.077 * fatness;
  cons(11) = 44.314;
  p = 583 + 1.92 * weight ;          %outer perimeter
  alfa = 110;
  cons(12) = p;
  cons(13) = alfa;
function cons = cons8(weight, fatness)

    cons(1) = 161.65 + 0.3676 * weight - 0.3473 * fatness;
  cons(2) = 73.142 + 0.5718 * weight + 0.392 * fatness;
  cons(3) = 75.653 + 0.8402 * weight + 0.11 * fatness;
  cons(4) = 22.22 + 0.2829 * weight + 1.3523 * fatness;
  cons(5) = 21.928 + 0.1787 * weight + 0.9571 * fatness;
  cons(6) = 19.599 + 0.2127 * weight + 0.4607 * fatness;
  cons(7) = -3.46 + 0.2891 * weight + 0.8064 * fatness;
  cons(8) = -17.025 + 0.4008 * weight + 1.4446 * fatness;
  cons(9) = 15.855 + 0.3848 * weight + 0.9739 * fatness;
  cons(10) = 46.273 + 0.2984 * weight + 0.4704 * fatness;
  cons(11) = 80.882;
  p = 558.67 + 2.0147 * weight ;          %outer perimeter
  alfa = 75;
  cons(12) = p;
  cons(13) = alfa;
function cons = cons9(weight, fatness)

    cons(1) = 224.06 + 0.7054 * weight - 1.2058 * fatness;
  cons(2) = 76.43 + 0.499 * weight - 0.24 * fatness;
  cons(3) = 80.23 + 0.6832 * weight + 0.1262 * fatness;
  cons(4) = 89.99 + 0.8821 * weight + 0.3942 * fatness;
  cons(5) = 37.35 + 0.4616 * weight + 0.9846 * fatness;
  cons(6) = 13.54 + 0.4428 * weight + 0.7786 * fatness;
  cons(7) = 0.45 + 0.5463 * weight + 0.8327 * fatness;
  cons(8) = 37.463 + 0.4795 * weight - 0.0112 * fatness;
  cons(9) = 132.176;
  p = 571.64 + 2.1411 * weight - 1.129 * fatness ;          %outer perimeter
  alfa = 75;
  cons(10) = p;
  cons(11) = alfa;
function cons = cons10(weight, fatness)

    cons(1) = 240.43 + 0.9654 * weight - 1.7274 * fatness;
```

```
  cons(2) = 149.01 + 0.4027 * weight - 0.7572 * fatness;
  cons(3) = 64.73 + 0.6152 * weight + 1.1346 * fatness;
  cons(4) = 25.53 + 0.9312 * weight - 0.3002 * fatness;
  cons(5) = 137.07 + 0.4063 * weight - 1.001 * fatness;
  p = 573.06 + 2.1616 * weight - 1.197 * fatness ;          %outer perimeter
  alfa = 78;
  cons(6) = p;
  cons(7) = alfa;
function cons = cons11(weight, fatness)

    cons(1) = 155.75 + 1.1856 * weight - 1.71 * fatness;
  cons(2) = 46.93 + 0.5634 * weight - 0.0673 * fatness;
  cons(3) = 47.87 + 0.3104 * weight;
  p = 219.59 + 1.8734 * weight ;          %outer perimeter
  alfa = 78;
  cons(4) = p;
  cons(5) = alfa;
function cons = cons13(weight, fatness)

    a = 111.21 + 0.24588 * weight - 0.2246 * fatness;      %minor axis
    b = 158.637 + 0.57254 * weight - 0.5396 * fatness;  %major axis
  p = 452.48 + 1.2721 *weight - 0.7076 * fatness;          %outer perimeter

  cons = [a, b, p];
function cons = cons14(weight, fatness)

    a = 64.169 + 0.24372 * weight - 0.1935 * fatness;      %minor axis
    b = 85.028 + 0.23227 * weight - 0.5107 * fatness;  %major axis
  p = 240.3 + 0.68295 *weight - 1.0771 * fatness;          %outer perimeter

  cons = [a, b, p];
function f = funline(der,x)

f = ( ( polyval(der,x) )^2 + 1 )^0.5;

function int = integpol(poly)

% this function made the integration of the polynom to find the area under the curve
n = size(poly,2);
powerpol = (n-2) - 1;          % power of the polynom
powercof = 1;
for i = (powerpol+2):-1:1
  if i == (powerpol + 2)
    pol(i) = 0;
  else
    pol(i) = poly(i)/ powercof;  % integrated polynom
     powercof = powercof + 1;
  end

end
xmin = poly(n-1);
xmax = poly(n);
int = polyval(pol, xmax) - polyval(pol, xmin);

function l = lenghtline(poly, ndiv, fac)

% this function calculate the lenght of the polynom
% ndiv is number of division . It should be even
poly = polyscaled(poly, fac);          % the polynom is scaled
```

```
n = size(poly,2);
pol = poly(1:(n-2));                    % polynom
xmin = poly(n-1);
xmax = poly(n);
der = polyder(pol);                     % derivate of the polynom
dx = (xmax - xmin) /ndiv;

% calculation of the integration to find the lenght of the line or perimeter.
% it use the simpon method
sum = funline(der,xmin) + 4* funline(der, (xmax-dx)) + funline(der, xmax);
x = xmin;
for i = 1:2:( (ndiv+1)-3)
  x = x + dx;
  sum = sum + 4 * funline(der, x);
  x = x + dx;
  sum = sum + 2 * funline(der, x);
end
l = sum * dx / 3;

function npoint = pellipse2(center,point1, point2, npoint, jour)

xo = center(1);
yo = center(2);
zo = center(3);
npoint = npoint +1;
printpoint(xo, yo, zo, npoint, jour);
carpointo = num2str(npoint,'%1.0f');


%  this is the point of the mayor axes
npoint = npoint +1;
x = point1(1);
y = point1(2);
z  = point1(3);
printpoint(x, y, z, npoint, jour);
carpointb = num2str(npoint,'%1.0f');


%  this is the point of the minor axes
npoint = npoint +1;
x = point2(1);
y = point2(2);
z  = point2(3);
printpoint(x, y, z, npoint, jour);
carpointa = num2str(npoint,'%1.0f');


cartx = strcat('edge create center "vertex.',carpointo,'" major "vertex.',carpointb,'" onedge
"vertex.',carpointa,'" start 0 end 360 ellipse');
fprintf(jour,cartx);
fprintf(jour,'\n',");

function npoint = pellipse(a, b, center, npoint, jour)

xo = center(1);
yo = center(2);
zo = center(3);
npoint = npoint +1;
printpoint(xo, yo, zo, npoint, jour);
```

```
carpointo = num2str(npoint,'%1.0f');


% this is the point of the mayor axes
npoint = npoint +1;
x = xo + b;
y = yo;
z = zo;
printpoint(x, y, z, npoint, jour);
carpointb = num2str(npoint,'%1.0f');


% this is the point of the minor axes
npoint = npoint +1;
x = xo;
y = yo + a;
z = zo;
printpoint(x, y, z, npoint, jour);
carpointa = num2str(npoint,'%1.0f');


cartx = strcat('edge create center "vertex.',carpointo,'" major "vertex.',carpointb,'" onedge
"vertex.',carpointa,'" start 0 end 360 ellipse');
fprintf(jour,cartx);
fprintf(jour,'\n',"");

function [ gext, gint, area] = pointspol2(cons, poly, ndiv, section, facvol, cplot)

    colums = size(cons,2);                    % number of elements on the cons vector
  colums2 = size(poly,2);                 % number of elements on the poly vector
  p = cons(colums-1);                     % perimeter
  alfa = cons(colums);                    % alfa angle
  int1 = integpol(poly);                  % area under the polynom

  % this part of the program calculate the points on the poly
  n = colums - 2;                         % number of diameters
  m = colums2 - 2;                            % polymonial coeficents on the vecor poly
  l = lenghtline(poly, ndiv, 1);          % lenght line after scaling the polynom
  dp = l/(n-1);
  % that gives the longitud of the delts of the perimeter
  % it is assumed that the  the l is equal to the perimeter p
  % The polymer should be scaled before to call this program. Otherwise,
  % The calculation is made withe the lenght of the polymer

  pol = poly(1:m);                            % polynom
  xmin = poly(colums2 - 1);               % this is the x min of the polymer
  xmax = poly (colums2);                  % this is the x min of the polymer
  ymin = polyval(pol,xmin);
  ymax = polyval(pol,xmax);
  tetarot = atan2((ymax-ymin),(xmax-xmin));


  der = polyder(pol);                     % derivate of the polynom
  dx = (xmax - xmin) /ndiv;
  tmin = xmin;
  sum = 0;
  j = 1;
  x(j) = xmin;
  y(j) = polyval(pol, xmin);
```

```
sumbefore = 0;
for i = 1:(ndiv+1)
  tmax = tmin + dx;
  dt =(tmax - tmin)/2;
  % calculation of the integration to find the lenght of the line or perimeter.
     % it use the simpon method
     sum = sum + dt/3*( funline(der,tmin) + 4* funline(der, (tmin+dt)) + funline(der, tmax));
  if sum > (j*dp)
    x(j+1) = tmin + (tmax - tmin)/(sum - sumbefore)*(j*dp - sumbefore);
    y(j+1) = polyval(pol, x(j+1));
    j = j+1;
  else
    sumbefore = sum;
  end
  tmin = tmax;
end
x(n) = xmax;
y(n) = polyval(pol, xmax);

for i = 1:n
  derpol = polyval(der,x(i));
  if derpol > 0
    xvec = 1 / ( ( 1 + (1/derpol)^2) ^0.5 );
    yvec = -1 /(derpol*(1 + (1/derpol)^2)^0.5);
  else
    xvec = -1 / ( ( 1 + (1/derpol)^2) ^0.5 );
    yvec = 1 /(derpol*(1 + (1/derpol)^2)^0.5);
  end
  if i == 1
    beta = atan2(yvec, xvec);
    gama = beta - (alfa - 90) * ( 2 * pi / 360);
    xd(i) = x(i) + cons(i)*cos(gama)*facvol^(1/3);
    yd(i) = y(i) + cons(i)*sin(gama)*facvol^(1/3);
  else
    xd(i) = x(i) + cons(i) * xvec*facvol^(1/3);
    yd(i) = y(i) + cons(i) * yvec*facvol^(1/3);
  end
end
mat = [x' y' xd' yd'];            % this are all the original points

% the points are going to be rotate
gext = mat(:,1:2);
gint = mat(:,3:4);
yoe = gext(1,2);
xoe = gext(1,1);
option = 1;                       % 1 for the external and 2 for the internal
% on these sections the structure is rotate to match at z = 0 the begining and the end
gext = rotation2(gext, yoe, xoe, option, tetarot);
gint = rotation2(gint, yoe, xoe, 2, tetarot);
mat = [gext gint];

x = gext(:,1);
y = gext(:,2);
xd = gint(:,1);
yd = gint(:,2);

% this estructure makes the segments streight
yd(1) = y(1);
xd(1) = x(1)+ cons(1);
```

```
yd(n) = y(n);
xd(n) = x(n)- cons(n);


if section == 4;
  x2 = [x(1); x(3:n)];
  y2 = [y(1); y(3:n)];
        xd2 = [xd(1); xd(3:n)];
  yd2 = [yd(1); yd(3:n)];
elseif section == 5;
  x2 = [x(1); x(4:n)];
  y2 = [y(1); y(4:n)];
        xd2 = [xd(1); xd(4:n)];
  yd2 = [yd(1); yd(4:n)];
elseif section == 6 | section == 7;
  x2 = [x(1); x(3:n)];
  y2 = [y(1); y(3:n)];
        xd2 = [xd(1); xd(3:n)];
  yd2 = [yd(1); yd(3:n)];
elseif section == 8;
  x2 = [x(1); x(4:n-2); x(n)];
  y2 = [y(1); y(4:n-2); y(n)];
        xd2 = [xd(1); xd(4:n-2); xd(n)];
  yd2 = [yd(1); yd(4:n-2); yd(n)];
elseif section == 9;
  x2 = [x(1); x(5:n)];
  y2 = [y(1); y(5:n)];
        xd2 = [xd(1); xd(5:n)];
  yd2 = [yd(1); yd(5:n)];
elseif section == 10;
  x2 = [x(1); x(3:n)];
  y2 = [y(1); y(3:n)];
        xd2 = [xd(1); xd(3:n)];
  yd2 = [yd(1); yd(3:n)];
elseif (section == 11) | (section == 12)

  % this is modify leaving the inner line as a straight line
  x2 = x;
  y2 = y;
  m = ( y2(3) - y(1) ) / ( x2(3) - x2(1) );
  xd2(1) = 0.25*(x2(3) - x2(1)) + x2(1);
  xd2(2) = 0.5*(x2(3) - x2(1)) + x2(1);
  xd2(3) = 0.75*(x2(3) - x2(1)) + x2(1);
  yd2 = m*(xd2 - x2(1)) + y2(1);
  xd2 = xd2';
  yd2 = yd2';
else
  x2 = x;
  y2 = y;
  xd2 = xd;
  yd2 = yd;
end

geom = [x2 y2 xd2 yd2];
gint = geom(:,3:4);

% the calculations of area have mistakes on section 6, 7, 8, 9
if section == 6 | section == 7;
```

```
    m = size(geom,1);                % it gives the number of points on geom
    xd2p = [xd2(1); xd2(3:m)];                    % the integral is done begining on the first point but is
ont taken the second one
    yd2p = [yd2(1); yd2(3:m)];          % so, this is an aproximation
    m = m-1;
    xd3 = [xmin xd2p' xmax];
     yd3 = [polyval(pol, xmin) yd2p' polyval(pol, xmax) ];
     m3 = m+2;                                % number of points on the vector xd3
     int2 = trapezoidal(m3, xd3, yd3);
    area = int1 - int2;

  elseif section == 10;
    yp = yd2(2);
    xp = xd2(3);
    int2 = 0.5*( (xd2(4) - xd(1)) + (xp - xd2(1)) ) * yp + (xp - xd2(1))*(yd2(3)-yp) / 2;
    area = int1 - int2;

  else
    m = size(geom,1);                % it gives the number of points on geom
    xd3 = [xmin xd2' xmax];
     yd3 = [polyval(pol, xmin) yd2' polyval(pol, xmax) ];
     m3 = m+2;                                % number of points on the vector xd3
     int2 = trapezoidal(m3, xd3, yd3);
    area = int1 - int2;
  end


     % cplot = 0, doesnt print, = 1, print all the points, = 2 print the selected points
     if cplot == 1
    xx = (xmin:dx:xmax);                          % xx and yy give the first polinom
    yy = polyval(pol, xx);
    g = [xx' yy'];
    g = rotation2(g, yoe, xoe, option, tetarot);
    xx = g(:,1)';
    yy = g(:,2)';
    %xp2 = (xd(1):dx:xd(n));                      % xp2 and yp2 are the spline interpolation of the points
of the second polinom
    %yp2 = spline(xd,yd,xp2);
    %plot(xx, yy,'g-',mat(:,3), mat(:,4),'ko',mat(:,1), mat(:,2),'ro',xp2,yp2,'b-')
    plot(xx, yy,'g-',mat(:,3), mat(:,4),'ko',mat(:,1), mat(:,2),'ro')
  elseif cplot == 2
    xx = (xmin:dx:xmax);                          % xx and yy give the first polinom
    yy = polyval(pol, xx);
    g = [xx' yy'];
    g = rotation2(g, yoe, xoe, option, tetarot);
    xx = g(:,1)';
    yy = g(:,2)';
    plot(xx, yy,'g-',geom(:,3), geom(:,4),'ko',geom(:,1), geom(:,2),'ro')

  end

  mat = geom;


function [ gext, gint, area] = pointspol3(cons, poly, ndiv, section, facvol, cplot)

     colums = size(cons,2);                       % number of elements on the cons vector
  colums2 = size(poly,2);                     % number of elements on the poly vector
  p = cons(colums-1);                         % perimeter
```

```
alfa = cons(colums);                    % alfa angle
int1 = integpol(poly);                  % area under the polynom


% this part of the program calculate the points on the poly
n = colums - 2;                         % number of diameters .
m = colums2 - 2;                            % polymonial coeficents on the vecor poly
l = lenghtline(poly, ndiv, 1);          % lenght line after scaling the polynom
dp = l/(n-1);
% that gives the longitud of the delts of the perimeter
% it is assumed that the  the l is equal to the perimeter p
% The polymer should be scaled before to call this program. Otherwise,
% The calculation is made withe the lenght of the polymer

pol = poly(1:m);                            % polynom
xmin = poly(colums2 - 1);               % this is the x min of the polymer
xmax = poly (colums2);                  % this is the x min of the polymer
der = polyder(pol);                     % derivate of the polynom
dx = (xmax - xmin) /ndiv;
tmin = xmin;
sum = 0;
j = 1;
x(j) = xmin;
y(j) = polyval(pol, xmin);
sumbefore = 0;
for i = 1:(ndiv+1)
  tmax = tmin + dx;
  dt =(tmax - tmin)/2;
  % calculation of the integration to find the lenght of the line or perimeter.
     % it use the simpon method
     sum = sum + dt/3*( funline(der,tmin) + 4* funline(der, (tmin+dt)) + funline(der, tmax));
  if sum > (j*dp)
    x(j+1) = tmin + (tmax - tmin)/(sum - sumbefore)*(j*dp - sumbefore);
    y(j+1) = polyval(pol, x(j+1));
    j = j+1;
  else
    sumbefore = sum;
  end
  tmin = tmax;
end
x(n) = xmax;
y(n) = polyval(pol, xmax);

for i = 1:n
  derpol = polyval(der,x(i));
  if derpol > 0
    xvec = 1 / ( ( 1 + (1/derpol)^2) ^0.5 );
    yvec = -1 /(derpol*(1 + (1/derpol)^2)^0.5);
  else
    xvec = -1 / ( ( 1 + (1/derpol)^2) ^0.5 );
    yvec = 1 /(derpol*(1 + (1/derpol)^2)^0.5);
  end
  if i == 1
    beta = atan2(yvec, xvec);
    gama = beta - (alfa - 90) * ( 2 * pi / 360);
    xd(i) = x(i) + cons(i)*cos(gama)*facvol^(1/3);
    yd(i) = y(i) + cons(i)*sin(gama)*facvol^(1/3);
  else
    xd(i) = x(i) + cons(i) * xvec*facvol^(1/3);
```

```
    yd(i) = y(i) + cons(i) * yvec*facvol^(1/3);
  end
end
mat = [x' y' xd' yd'];            % this are all the original points;

% rotation of the surface keeping the real angle
tetarot = 0;
[mat(:,1:2), mat(:,3:4)] = rotation(mat(:,1:2), mat(:,3:4), tetarot);
% this modification is to reduce the lenght of the section 4
if section ==4
  mat(1,1) = mat(1,1) - (mat(1,1) - mat(2,1))*1;
end


gext = mat(:,1:2);

x = mat(:,1)';
y = mat(:,2)';
xd = mat(:,3)';
yd = mat(:,4)';


% this estructure makes the last segment streight
yd(n) = y(n);
xd(n) = x(n)- cons(n);

if section == 4;
  x2 = [x(1) x(3:n)];
  y2 = [y(1) y(3:n)];
       xd2 = [xd(1) xd(3:n)];
  yd2 = [yd(1) yd(3:n)];
elseif section == 8;
  x2 = [x(1) x(5:n)];
  y2 = [y(1) y(5:n)];
       xd2 = [xd(1) xd(5:n)];
  yd2 = [xd(1) yd(5:n)];
elseif section == 9;
  x2 = [x(1) x(5:n)];
  y2 = [y(1) y(5:n)];
       xd2 = [xd(1) xd(5:n)];
  yd2 = [yd(1) yd(5:n)];
elseif section == 10;
  x2 = [x(1) x(3:n)];
  y2 = [y(1) y(3:n)];
       xd2 = [xd(1) xd(3:n)];
  yd2 = [yd(1) yd(3:n)];
elseif (section == 11) | (section == 12)

  % this is modify leaving the inner line as a straight line
  x2 = x;
  y2 = y;
  m = ( y2(3) - y(1) ) / ( x2(3) - x2(1) );
  xd2(1) = 0.25*(x2(3) - x2(1)) + x2(1);
  xd2(2) = 0.5*(x2(3) - x2(1)) + x2(1);
  xd2(3) = 0.75*(x2(3) - x2(1)) + x2(1);
  yd2 = m*(xd2 - x2(1)) + y2(1);

else
  x2 = x;
```

```
    y2 = y;
    xd2 = xd;
    yd2 = yd;
  end
  geom = [x2' y2' xd2' yd2'];  % this are the same points used by lucy
  gint = geom(:,3:4);
  m = size(geom,1);            % it gives the number of points on geom
  xd3 = [xmin xd2 xmax];
  yd3 = [polyval(pol, xmin) yd2 polyval(pol, xmax) ];
  m3 = m+2;                            % number of points on the vector xd3
  int2 = trapezoidal(m3, xd3, yd3);
  area = int1 - int2;

     % cplot = 0, doesnt print, = 1, print all the points, = 2 print the selected points
     if cplot == 1
    xx = (xmin:dx:xmax);                        % xx and yy give the first polinom
    yy = polyval(pol, xx);
    %xp2 = (xd(1):dx:xd(n));                    % xp2 and yp2 are the spline interpolation of the points
of the second polinom
    %yp2 = spline(xd,yd,xp2);
    %plot(xx, yy,'g-',mat(:,3), mat(:,4),'ko',mat(:,1), mat(:,2),'ro',xp2,yp2,'b-')
    plot(xx, yy,'g-',mat(:,3), mat(:,4),'ko',mat(:,1), mat(:,2),'ro')
  elseif cplot == 2
    xx = (xmin:dx:xmax);                        % xx and yy give the first polinom
    yy = polyval(pol, xx);
    plot(xx, yy,'g-',geom(:,3), geom(:,4),'ko',geom(:,1), geom(:,2),'ro')

  end

  mat = geom;


function poly = polynom4

poly(1) = 1.218730000E-11;
poly(2) = -1.050580000E-08;
poly(3) = 3.280160000E-06;
poly(4) = -4.371160000E-04;
poly(5) = 1.669565600E-02;
poly(6) = 1.208266369E+00;
poly(7) = 6.404528799E+01;
poly(8) = 11.2431;              %xmin
poly(9) = 265.562;              %xmax

function poly = polynom5

poly(1) = -1.3180000000E-13;
poly(2) = 1.1475300000E-10;
poly(3) = -3.9838600000E-08;
poly(4) = 6.8503300000E-06;
poly(5) = -5.6925000000E-04;
poly(6) = 1.3979724E-02;
poly(7) = 9.15379115E-01;
poly(8) = 1.0813036840E+02;
poly(9) = 13.4860;              %xmin
poly(10)  = 274.1480;           %xmax
```

```
function poly = polynom6

poly(1) = -1.32735E-15;
poly(2) = 1.53093E-12;
poly(3) = -7.34351E-10;
poly(4) = 1.89342E-07;
poly(5) = -2.83355E-05;
poly(6) = 0.002494418;
poly(7) = -0.128935294;
poly(8) = 4.113140419;
poly(9) = 87.9878179;
poly(10) = 5.26323;          %xmin
poly(11)  = 297.408;         %xmax


function poly = polynom7

poly(1) = 4.90445E-14;
poly(2) = -5.32977E-11;
poly(3) = 2.35953E-08;
poly(4) = -5.51299E-06;
poly(5) = 0.000728262;
poly(6) = -0.055005741;
poly(7) = 2.357776103;
poly(8) = 120.9301015;
poly(9) = 1.658;          %xmin
poly(10)  = 293.8;           %xmax


function poly = polynom8

poly(1) = 3.44518E-14;
poly(2) = -4.35084E-11;
poly(3) = 2.15048E-08;
poly(4) = -5.44912E-06;
poly(5) = 0.000780945;
poly(6) = -0.067454156;
poly(7) = 3.528328616;
poly(8) = 64.3750684;
poly(9) = 9.8553;         %xmin
poly(10)  = 291.642;         %xmax


function poly = polynom9

poly(1) = 3.44518E-14;
poly(2) = -4.35084E-11;
poly(3) = 2.15048E-08;
poly(4) = -5.44912E-06;
poly(5) = 0.000780945;
poly(6) = -0.067454156;
```

```
poly(7) = 3.528328616;
poly(8) = 64.3750684;
poly(9) = 9.8553;            %xmin
poly(10)  = 291.642;            %xmax
```

```
function poly = polynom10

poly(1) = -9.12522E-15;
poly(2) = 1.10758E-11;
poly(3) = -5.599E-9;
poly(4) = 1.52759E-06;
poly(5) = -0.000243759;
poly(6) = 0.023075048;
poly(7) = -1.262542335;
poly(8) = 37.8443233;
poly(9) = -424.9879651;
poly(10) = 32.7232;            %xmin
poly(11)  = 289.707;            %xmax
```

```
function poly = polynom11

poly(1) = 1.25359E-11;
poly(2) = -1.2422E-08;
poly(3) = 4.63229E-06;
poly(4) = -0.000811868;
poly(5) = 0.062044255;
poly(6) = -0.64638617;
poly(7) = 24.83161746;
poly(8) = 28.3062;            %xmin
poly(9) = 285.054;            %xmax
```

```
function  pol = polyscaled(poly, fac)

% this function scaled the polynomial
n = size(poly,2);
pol = poly(1:(n-2));                % polynom
powerpol = (n-2) - 1;            % power of the polynom
powercof = 0;
for i = (powerpol+1):-1:1
  pol(i) = pol(i)/(fac^(powercof-1));
  powercof = powercof + 1;
end
xmin = poly(n-1)*fac;
xmax = poly(n)*fac;
pol = [pol xmin xmax];
```

```
function p = printpoint(x, y, z, npoint, jour)

carx = num2str(x,'%10.4e');
cary = num2str(y,'%1.4e');
carz = num2str(z,'%1.4e');
```

```
carpoint = num2str(npoint,'%1.0f');
xjo = strcat('$x',carpoint,' = ',carx);
yjo = strcat('$y',carpoint,' = ',cary);
zjo = strcat('$z',carpoint,' = ',carz);
fprintf(jour,xjo);
fprintf(jour,'\n',");
fprintf(jour,yjo);
fprintf(jour,'\n',");
fprintf(jour,zjo);
fprintf(jour,'\n',");
txjo = strcat(' $x',carpoint,' $y',carpoint,' $z',carpoint);
cartx = 'vertex create coordinates ';
w = strcat(cartx, txjo);
fprintf(jour,w);
fprintf(jour,'\n',");
p = 1;

function gext = rotation2(gext, yoe, xoe, option, tetarot)

next = size(gext,1);
yext = gext(:,2) - yoe;
xext = gext(:,1) - xoe;

lenghtext = ( yext.^2 + xext.^2 ).^0.5;

if option == 1
    tetaext1(1) = 0;
  tetaext1(2:next) = ( atan(yext(2:next)./xext(2:next)) );
  tetaext1 = tetaext1';
else
  tetaext1 = ( atan(yext./xext) );
end


tetaext2 = tetaext1 - tetarot;

gext(:,1) = lenghtext.*cos(tetaext2) + xoe;
gext(:,2) = lenghtext.*sin(tetaext2) + yoe;


function [gext, gint] = rotation(gext, gint, tetarot)

next = size(gext,1);
nint = size(gint,1);
yoe = gext(1,2);
xoe = gext(1,1);
yext = gext(:,2) - yoe;
xext = gext(:,1) - xoe;
yint = gint(:,2) - yoe;
xint = gint(:,1) - xoe;

lenghtext = ( yext.^2 + xext.^2 ).^0.5;
lenghtint = ( yint.^2 + xint.^2 ).^0.5;

tetaext1(1) = 0;
tetaext1(2:next) = ( atan(yext(2:next)./xext(2:next)) );
tetaint1(1:nint) = ( atan(yint(1:nint)./xint(1:nint)) );

tetaext1 = tetaext1';
```

```
tetaint1 = tetaint1';

tetaext2 = tetaext1 + tetarot - tetaint1(1);
tetaint2 = tetaint1 + tetarot - tetaint1(1);

gext(:,1) = lenghtext.*cos(tetaext2) + xoe;
gext(:,2) = lenghtext.*sin(tetaext2) + yoe;

gint(:,1) = lenghtint.*cos(tetaint2) + xoe;
gint(:,2) = lenghtint.*sin(tetaint2) + yoe;



function fac = scalelenght(poly, ndiv, p)

% this func. calculate the scale factor to equal the polinom lenght and
% the perimeter
fac = 1;
l = 10 * p;
cont = abs(l-p)/p;
while cont > 0.0001
     l = lenghtline(poly, ndiv, fac);
     fit = l/p;
  fac = fac/fit;
  cont = abs(l-p)/p;
end


function seg = segments(lenght)

llo = [ 0.11385 0.0730 0.0730 0.1231 0.0563 0.0744 0.1387 0.0775 0.0845 0.0423 0.0635 0.0803
0.1473];
l = sum (llo(1:12));
%llo(1:12) = llo(1:12)./l;
seg = llo*lenght;


function int = simpsonpol(poly, ndiv,fac)

n = size(poly,2);
pol = poly(1:(n-2))*fac;          % polynom. Factor is the scale factor
xmin = poly(n-1);
xmax = poly(n);
dx = (xmax - xmin) /ndiv;
sum = polyval(pol, xmin) + 4* polyval(pol, (xmax - dx)) + polyval(pol, xmax);
x = xmin;
for i = 1:2:( (ndiv+1)-3)
  x = x + dx;
  sum = sum + 4 * polyval(pol,x);
  x = x + dx;
  sum = sum + 2 * polyval(pol, x);
end
int = sum * dx / 3;

function int = trapezoidal(m, x, y)

% m is the number of points
int = 0;
for i = 1:(m-1)
```

```
  int = int + 0.5*( y(i)+y(i+1) )*(x(i+1)-x(i));
end
```

A5. UDFS PROGRAMMED IN C++ TO SOLVE TO HEAT AND MOISTURE TRANSFER ON THE 3D BEEF CARCASS MODEL SIMULTANEOUSLY SOLVING THE PROCESS INSIDE THE MEAT AND IN THE AIR PHASE

```
# include "udf.h"

/********************************* variables that must be defined
*********************************************************/
/****** constants ******/
real Cpmeat = 3407;                    /* Cp meat */
real kmeat = 0.397;                    /* k meat */
real denmeat = 1111;                   /* meat density */
real Dtmeat = 1.0488e-7;               /* thermal diffusivity */
real Hvap = 2452240;                   /* heat of vaporization of water in J / Kg */
real E = 0.92;                         /* Plaster emisivity */
real kar = 0.4187;                     /* vor karman constant */
real Ew = 9.793;                       /* wall function constant */
real sigma = 5.67e-8;                  /* sigma constant */
real Cmu = 0.0845;                     /* constant of the RNG model */
real Mwater = 18.016;                  /* warter molecular weight*/
real Mair =  28.96;                    /* air molecular weight*/
real R = 8314.47;                      /* universal constant in pa m3 / Kgmol/ K;
/*****************************************************************************
*************************************/

/***************************** Initial and operation conditions
*********************************************************/
/* it must be the same imformation in the boundary condtions panel */
real tsl = 5400;                       /* slaughter time sec */
real Pt = 101325;                      /* cell total presure */
real To = 279.17;                      /* Inlet air temperature */
real Tofloor = 298.15;                 /* floor air temperature */
real yo = 0.005730549;                 /* water mass composition at the inlet */
real yofloor = 0.012452342;            /* water mass composition at the inlet floor conditions */
real RH = 0.985;                       /* Humidity as a fracction */
real Yomeat = 0.75;                    /* initial water composition of meat */
real Tomeat = 315.55;                  /* initial meat temperature */
real Voair = 0.989;                    /* air velocity on the inlet */
/*****************************************************************************
*************************************/

/********************** Time variable and vector to store and solve the water diffusion in meat
************/
real last_ts = -1;                 /*  Global variable.  Time s never <0 used in solve_meatwater */
real last_ts2 = -1;                /*  Global variable.  Time s never <0 used in heat_flux_meat */
int nnodes = 20;                   /* number on nodes per face to calculate the concentration inside the
                                   meat */
int niter;                         /* n iterations */
int nitermax = 19;                 /* niter to calculate fluxes */
int lref = 116;                    /* face reference number to print values */
real Xpos[] = {0, 0.000005, 0.000015, 0.00005, 0.0001, 0.00025, 0.0005, 0.00075, 0.001, 0.0015, 0.002,
0.003, 0.004, 0.005, 0.006, 0.008, 0.01, 0.012, 0.014, 0.016, 0.020, 0.024};
```

```
real Tairtab[] = {7.2, 7.4, 7.1, 6.9, 6.9, 6.5, 6.7, 6.4, 6.5, 6, 5.3, 5.2, 5.3, 5.5, 5.4, 5.5, 5.5, 5.4, 5.6, 5.7,
5.7};
real timetab[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
real watercont[] = {0.006287615,0.006318742, 0.006186619, 0.006119355, 0.006077779, 0.005917065,
0.005980425, 0.005884068, 0.005900769, 0.005762241, 0.005430462, 0.00537669, 0.005471543,
0.005465084, 0.005500349, 0.005551778, 0.005513497, 0.005484137, 0.005583786, 0.005630782,
0.005619935};


/*****************************************************************************************
**************/

/************************************** program controls
*****************************************************************/
real aweps = 0.0005;            /* coenvergence criterium to find the aw using newthon method */
int Radiation_Control = 1;      /* 1 when take in account the radiation , 0, when do not */
real alfa = 1;                  /* is the relaxation factor in the UDF that calculate the water on the
                                interface */
real alfasc = 1;                /* is the relaxation factor to calculate the inverse Sc or ycstar */
real yeps = 0.01;               /* tolerance calculating ycstar */
int conwall = 1;                /* 1 define solving the wall, 2 wall approach */
int surf_meat_ID = 4;           /* ID zone of the wall in contact with the meat */
int surf_air_ID = 10;           /* ID zone of the wall in contact with the air */
int meat_ID = 3;                /* ID zone of the meat */
int air_ID = 2;                 /* ID zone of the air */
/*****************************************************************************************
**************************************/
/*****************************************************************************************
**************************************/


real water_activity(real aw, real xm)
{

        real xmg = 0.0565;      /* GAB constant */
        real cg = 4.2396;       /* GAB constant */
        real k = 0.9692;        /* GAB constant */

        real f = 0.0488;        /* Lewicki constant */
        real g = 0.8761;        /* Lewicki constant */
        real h = -34.7794;      /* Lewicki constant */

        real fun = 10;          /* function equal to zero to calculate aw */
        real funder;            /* derivative of the function  */

        do
        {
                if ( aw >= 0.958093)
                {
                        fun = xm - f / pow((1-aw),g) + f/(1+pow(aw,h));
                        funder = -f * g / pow((1-aw),(g+1)) -f * h * pow(aw,(h-1))
                        /pow((1+pow(aw,h)),2);
                }
                else
                {
                        fun = xm - xmg * cg * k * aw/((1-k*aw)*(1+  (cg-1)*k*aw));
                        funder = xmg * cg * k/( (1- k * aw) * ( 1 + (cg-1) * k * aw))*(-1 + aw * (cg-1)
                        * k / (1+ (cg-1)* k * aw) - k * aw /(1- k * aw));
                }
```

```
                aw = aw - fun/funder;

        } while (fabs(fun) >= aweps );


        return aw;

}

real yair(real time)
{
        real yoair;
        int i;
        int k;


        if (time <= tsl)
        {
                yoair = yofloor;
        }
        else
        {
                time = (time - tsl)/3600;
                if (time <= 20)
                {
                        for (i = 0; i <= 19; i++)
                        {
                                if ( timetab[i+1] < time )
                                {
                                k = i+1;

                                }
                        }
                        yoair = watercont[k] + (watercont[k+1] - watercont[k]) / (timetab[k+1] -
                        timetab[k]) * (time - timetab[k]);

                }
                else
                {
                        yoair = watercont[20];

                }

        }

        return yoair;

}



real tempair(real time)
{
        real tair;
        int i;
        int k;
```

```
        if (time <= tsl)
        {
                tair = Tofloor;
        }
        else
        {
                time = (time - tsl)/3600;
                if (time <= 20)
                {
                        for (i = 0; i <= 19; i++)
                        {
                                if ( timetab[i+1] < time )
                                {
                                k = i+1;

                                }
                        }
                        tair = Tairtab[k] + (Tairtab[k+1] - Tairtab[k]) / (timetab[k+1] - timetab[k]) *
(time - timetab[k]);

                }
                else
                {
                        tair = Tairtab[20];

                }
                tair = tair +273.15;


        }

        return tair;

}



real InvSceff( real alfaso, real viseff, real vis)
{

                real alfas;

                real fun;               /* function on newton method to find alfa */
                real derfun;            /* derivate function on newton method to find alfa */


                alfas = alfaso;

                do
                {
                        fun = pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), 0.6321) * pow(( (alfas +
                        2.3929)/(alfaso + 2.3929) ), 0.3679) - vis / viseff;
                        derfun = pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), 0.6321) * 0.3679 * pow((
                        (alfas + 2.3929)/(alfaso + 2.3929) ), -0.6321) + pow(( (alfas + 2.3929)/(alfaso
                        + 2.3929) ), 0.3679) * 0.6321 * pow(( (alfas - 1.3929)/(alfaso - 1.3929) ), -
                        0.3679);
                        alfas = alfas - alfasc*fun/derfun;
```

```
        } while (fabs(fun) > yeps );

        return alfas;

}

real ycstar_calc( real Sc, real Sceff, real ycstar, real Pc)
{
        /* ycstart= nondimensional mass or thermal sublayer thickness */
        real fun;                       /* function on newton method to find ycstar */
        real derfun;                    /* derivate function on newton method to find ycstar */

        fun = Sc * ycstar - Sceff * (1 / kar * log(Ew * ycstar ) + Pc );

        do
        {

                fun = Sc * ycstar - Sceff * (1 / kar * log(Ew * ycstar ) + Pc );
                derfun =  Sc - Sceff / kar / ycstar;
                ycstar = ycstar - alfasc*fun / derfun;

        } while (fabs(fun) > yeps );

        return ycstar;

}

real massflux(face_t f, Thread *t)
{

        int i = 0;                      /* define the first component  = water */
        real Df;                        /* define the effective diffusivity of water in air*/
        real Tf;                        /* Face temperature */
        real yf;                        /* define the wat. comp. on the face */
        real yc;                        /* define the wat. comp. on the cell */
        real Ystar;                     /* nondimensional composition */
        real yp;                        /* yp = distance between the point p and the wall */
        real ystar;                     /* ystart = nondimensional longitud from node to wall */
        real ycstar;                    /* nondimensional mass sublayer thickness */
        real kp;                        /* turb. kinetic energy dissipation rate at the cell point */
        real density;                   /* density average of face and cell */
        real viseff;                    /* effective viscosity */
        real vis;                       /* viscosity */
        real Sc;                        /* laminar schmidt mumber */
        real Sceff;                     /* effective turbulent schmidt mumber */
        real Pc;                        /* term in the wall aprx */
        real mass_flux;                 /* define the mass flux on  wall (face)  (Kg Water/s.m2) */
```

```
real A[ND_ND];                          /* vector Area of the face */
real Aunit[ND_ND];                      /* unit vector perpendicular to the area, same
                                        direction of the area vector */
real Area;                              /* Magnitud of the vector Area */
real Cface[ND_ND];                      /* face centroid vector */
real Ccell[ND_ND];                      /* cell centroid vector */
real es[ND_ND];                         /* unit vector directin from the cell centroid to the
                                        face centroid */
real dro[ND_ND];                        /* vector distance connecting the cell and face
                                        centroid */
real ds;                        /* distance between the cell centroide and face centroide */

real A_by_es;
real Zero[ND_ND];

Domain *d;
Thread *to;
Thread *ta;
cell_t co;

Zero[0] = 0;
Zero[1] = 0;
Zero[2] = 0;


d = Get_Domain(1);

co = F_C0(f,t);                                 /* cell asociated to the face */
to = THREAD_T0(t);                              /* cell thread for co */
ta = Lookup_Thread(d, surf_air_ID);             /* Thread of the wall in contact with air*/

yf = F_YI(f, t, i);                     /* define the wat. comp. on the face */
yc = C_YI(co, to, i);                   /* define the wat. comp. on the cell */
Tf = F_T(f, t);                         /* Face temperature */
Df = -2.01366e-5+1.53234e-7*Tf;         /* Diffusivity at face temperature */
density = C_R(co, to);                  /* density calculate on the cell */

F_AREA(A, f, t);
Area = NV_MAG(A);                       /* Area */
F_CENTROID(Cface, f, t);
C_CENTROID(Ccell, co, to);
NV_VV(dro, =, Cface, -, Ccell);         /* vector distance between the face and cell
                                        centroids */
ds = NV_MAG(dro);                       /* distance between the face and cell centroids */
NV_V_VS(es, = ,Zero, + , dro, *, (1/ds));  /* unit vector directing from the cell centroid to the
                                        face centroid */
A_by_es = NV_DOT(A, A) / NV_DOT(A, es);



if ( conwall == 1)
{
        mass_flux = Df * (yf - yc) / ds * A_by_es / Area * density;
}

else
{
        NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));        /* unit vector perpendicular to the
                                                area, same direction of the area vector */
```

```
            /* calculation of Sc and Sceff */
            vis = C_MU_L(co,to);                          /* viscosity */
            kp = C_K(co, to);                             /* turb. kinetic energy dissipation
                                                          rate at the cell point */
            yp = NV_DOT(dro, Aunit);                      /* yp = distance between the
                                                          point p and the wall */
            ystar = density * pow(Cmu , 0.25) * pow(kp , 0.5) * yp / vis;
            Sc = vis / density / Df;                      /* laminar schmidt mumber */
            viseff = C_MU_EFF(co, to);                    /* effective viscosity */
            Sceff = 1 / InvSceff(F_UDMI(f, ta, 2), viseff, vis);  /* effective turbulent schmidt
                                                          number */
            F_UDMI(f, ta, 2) = 1/Sceff;                   /* effective turbulent schmidt
                                                          number stored */

            /* calculation of Ycstar */
            Pc = 9.24* ( pow((Sc / Sceff), 0.75 ) - 1 ) * (1 + 0.28 * exp(-0.007* Sc / Sceff ) );
            ycstar = ycstar_calc( Sc, Sceff, F_UDMI(f,ta, 4), Pc);
            F_UDMI(f,ta, 4) = ycstar;

            if (ystar <= ycstar)
            {
                    Ystar = Sc * ystar;
            }
            else
            {
                    Ystar =  Sceff * (1 / kar * log(Ew * ystar ) + Pc );
            }


            mass_flux = (yf - yc ) / Ystar * density * pow(Cmu, 0.25) * pow (kp, 0.5);

        }

        return mass_flux;

}

real heatconvection(face_t f, Thread *t)
{

        real Tf;                        /* define the face temperature */
        real Tc;                        /* define the cell temperature */
        real Tstar;                     /* nondimensional Temperature */
        real yp;                        /* yp = distance between the point p and the wall */
        real ystar;                     /* ystart = nondimensional longitud from node to wall */
        real ycstar;                    /* nondimensional mass sublayer thickness */
        real kp;                        /* turb. kinetic energy dissipation rate at the cell point */
        real density;                   /* density average of face and cell */
        real viseff;                    /* effective viscosity */
        real vis;                       /* viscosity */
        real Kair;                      /* thermal conductivity air mix */
        real Cpair;                     /* Cp mix of air */
        real Pr;                        /* laminar Pr mumber */
        real Preff;                     /* effective turbulent Pr mumber */
        real Pc;                        /* term in the wall aprx */
```

```
real heat_flux_conv;                    /* define the heat flux by convection */
real A[ND_ND];                          /* vector Area of the face */
real Aunit[ND_ND];                      /* unit vector perpendicular to the area, same
                                        direction of the area vector */
real Area;                              /* Magnitud of the vector Area */
real Cface[ND_ND];                      /* face centroid vector */
real Ccell[ND_ND];                      /* cell centroid vector */
real es[ND_ND];                         /* unit vector directin from the cell centroid to the
                                        face centroid */
real dro[ND_ND];                        /* vector distance connecting the cell and face
                                        centroid */
real ds;                                /* distance between the cell centroide and face
                                        centroide */
real A_by_es;
real Zero[ND_ND];

Domain *d;
Thread *to;
Thread *ta;
cell_t co;


Zero[0] = 0;
Zero[1] = 0;
Zero[2] = 0;

d = Get_Domain(1);


co = F_C0(f,t);                         /* cell asociated to the face */
to = THREAD_T0(t);                      /* cell thread for co */
ta = Lookup_Thread(d, surf_air_ID);     /* Thread of the wall in contact with air*/

Tf = F_T(f, t);                         /* define the face temperature */
Tc = C_T(co, to);                       /* define the cell temperature */
density = C_R(co, to);
Kair = C_K_L(co,to);                    /* themal conductivity */
Cpair = C_CP(co,to);                    /* Cp */
F_AREA(A, f, t);
Area = NV_MAG(A);                       /* Area */
F_CENTROID(Cface, f, t);
C_CENTROID(Ccell, co, to);
NV_VV(dro, =, Cface, -, Ccell);         /* vector distance between the face and
                                        cell centroids */
ds = NV_MAG(dro);                       /* distance between the face and cell
                                        centroids */
NV_V_VS(es, = ,Zero, + , dro, *, (1/ds)); /* unit vector directing from the cell
                                        centroid to the face centroid */
A_by_es = NV_DOT(A, A) / NV_DOT(A, es);


if (conwall ==1)
{
        heat_flux_conv = Kair * (Tf - Tc) / ds * A_by_es / Area;
}

else
{
```

```
            NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));        /* unit vector perpendicular to the
                                                                area, same direction of the area vector */

            /* calculation of Pr and Preff */
            vis = C_MU_L(co,to);                                /* viscosity */
            kp = C_K(co, to);                                   /* turb. kinetic energy dissipation rate at
                                                                the cell point */
            yp = NV_DOT(dro, Aunit);                            /* yp = distance between the point p and
                                                                the wall */
            ystar = density * pow(Cmu , 0.25) * pow(kp , 0.5) * yp / vis;
            Pr = vis * Cpair / Kair;                            /* laminar Pr mumber */
            viseff = C_MU_EFF(co, to);                          /* effective viscosity */
            Preff =  1 / InvSceff(F_UDMI(f, ta, 3), viseff, vis);/* effective turbulent schmidt
                                                                number */
            F_UDMI(f, ta, 3) = 1/ Preff;                        /* effective turbulent schmidt number
                                                                stored*/

            /* calculation of ycstar */
            Pc = 9.24* ( pow((Pr / Preff), 0.75 ) - 1 ) * (1 + 0.28 * exp(-0.007* Pr / Preff ) );
            ycstar = ycstar_calc( Pr, Preff, F_UDMI(f, ta, 5), Pc);
            F_UDMI(f, ta, 5) = ycstar;

            if (ystar <= ycstar)
            {
                    Tstar = Pr * ystar;
            }
            else
            {
                    Tstar =  Preff * (1 / kar * log(Ew * ystar ) + Pc );
            }


            heat_flux_conv = (Tf - Tc ) / Tstar * density * Cpair * pow(Cmu, 0.25) * pow (kp, 0.5);

        }

        return heat_flux_conv;


}


real heattotal(face_t f, Thread *t)
{
        real T;                     /* Face Temperature  K */
        real mass_flux;             /* define the mass flux on the wall (face) on Kg Water / s m2 */
        real heat_flux_total;       /* define the total heat flux on the wall (face) on W /  m2 */
        real heat_flux_conv;        /* define the convective heat flux on the wall (face) on W /  m2 */
        real heat_flux_rad;         /* define the radiative heat flux on the wall (face) on W /  m2 */
        real dhvap;
        real Toair;
        real flow_time;

        flow_time = RP_Get_Real("flow-time");
        Toair = tempair(flow_time);
        T = F_T(f,t);                                           /* Face Temperature  K */
        mass_flux = massflux(f,t);
        heat_flux_conv = heatconvection(f,t);
```

```
        heat_flux_rad =  Radiation_Control * E * sigma * (pow(Toair,4) - pow(T,4));
        dhvap = 2.5e6 -2.5e3*(T - 273.15);
        heat_flux_total = - dhvap * mass_flux - heat_flux_conv + heat_flux_rad;

        return heat_flux_total;
}


DEFINE_ON_DEMAND(dem_init_watercontent)
{
        Domain *d;
        int m;                                  /* integer that define the number of nodes */

        Thread *tm;
        Thread *ta;
        face_t f;
        Thread *tmeat;                          /* Thread of meat zone */
        cell_t c;

        d = Get_Domain(1);


        tm = Lookup_Thread(d, surf_meat_ID);    /* Thread of the wall in contact with meat*/
        ta = Lookup_Thread(d, surf_air_ID);     /* Thread of the wall in contact with air*/

        begin_f_loop(f, tm)
         {

                for (m = 0; m <= nnodes; m++)
                {
                        F_UDMI(f, tm, m) = Yomeat;
                }

                /* extra declarations to store used values to acelerate processing */
                /* position cero is ocupaid by heat flux total */
                F_UDMI(f, ta, 0) = heattotal(f, ta);
                F_UDMI(f, ta, 1) = 0.9999;      /* water activity */
                F_UDMI(f, ta, 2) = 1/0.7;       /* alfasoc inverse of effective scmith */
                F_UDMI(f, ta, 3) = 1/0.7;       /* inverse of effective prandatl */
                F_UDMI(f, ta, 4) = 11.2;        /* ycstarm mass boundary layer thickness */
                F_UDMI(f, ta, 5) = 11.2;        /* ycstart thermal boundary layer thickness */
                F_UDMI(f, ta, 6) = massflux(f, ta);


         }
        end_f_loop(f,tm)
        F_UDMI(1,ta,7) = 0;     /* acumulative weight lost 1*/

        tmeat = Lookup_Thread(d, meat_ID);      /* Thread of meat */

        begin_c_loop(c, tmeat)
        {
                C_T(c, tmeat) = Tomeat;         /* Tomeat  */


        }
        end_c_loop(c, tmeat)
}
```

```
DEFINE_PROFILE(tair_inflow, t, n)
{

        real flow_time;
        real Toair;
        face_t f;

        flow_time = RP_Get_Real("flow-time");
        Toair = tempair(flow_time);

        begin_f_loop(f, t)
          {


                F_PROFILE(f, t, n) = Toair;

          }
        end_f_loop(f,t)


}


DEFINE_PROFILE(yair_inflow, t, n)
{

        real flow_time;
        real yoair;
        face_t f;

        flow_time = RP_Get_Real("flow-time");
        yoair = yair(flow_time);

        begin_f_loop(f, t)
          {


                F_PROFILE(f, t, n) = yoair;

          }
        end_f_loop(f,t)


}

DEFINE_PROFILE(vair_inflow, t, n)
{

        real flow_time;
        real vair;
        face_t f;

        flow_time = RP_Get_Real("flow-time");
        if (flow_time < tsl)
        {
                vair = 0.5;
        }
        else
```

```
        {
                vair = Voair;
        }
        begin_f_loop(f, t)
         {


                F_PROFILE(f, t, n) = vair;

         }
        end_f_loop(f,t)


}

DEFINE_PROFILE(heat_flux_meat, tm, j)
{

        real curr_ts;                   /* current time */

        face_t  f;                      /* face of the face loop , has the same number in the air or
meat face*/
        Domain *domain;
        Thread *t;                      /* Thread of the surface in contact with air*/


        domain = Get_Domain(1);
        t = Lookup_Thread(domain, surf_air_ID);    /* Thread of the wall in contact with air*/


        curr_ts = CURRENT_TIME;
        if (last_ts2 != curr_ts)
        {
                last_ts2 = curr_ts;

                begin_f_loop(f, tm)
                {

                                F_PROFILE(f, tm, j) = F_UDMI(f,t,0);


                }
                end_f_loop(f,tm)

        }
        else
        {
                if (niter == nitermax)
                {

                        begin_f_loop(f, t)
                        {
                                F_UDMI(f,t,0) = heattotal(f,t);
                                F_UDMI(f,t,6) = massflux(f, t);


                        }
                        end_f_loop(f,t)
                }
```

```
        }

}


DEFINE_PROFILE(water_air_interf, t, n)
{
        real T;                         /* Face Temperature */
        real Pw;                        /* face vapor presure */
        int i = 0;                      /* define the first component  = water */
        real x;                         /* molec. compos. water /air */
        real y;                         /* weight comp. water / air */
        real yf;                        /* define the wat. comp. on the air face */
        real ym;                        /* define the wat. weight composition on the meat face */
        real xm;                        /* water content of the meat on the interface */
        real aw;                        /* water activity */
        real dy;                        /* define the dy */
        face_t f;
        real c1 = -5800.2206;           /* cte of vapor presure */
        real c2 = 1.3914993;
        real c3 = -0.048640239;
        real c4 = 0.000041764768;
        real c5 = -0.000000014452093;
        real c6 = 6.5459673;

        Domain *domain;
        Thread *tm;                     /* Thread of the surface in contact with meat */

        domain = Get_Domain(1);
        tm = Lookup_Thread(domain, surf_meat_ID);/* Thread of the wall in contact with meat*/


        begin_f_loop(f, t)
         {

                T = F_T(f,t);           /* face temperature */
                Pw = exp( c1 / T + c2 + c3 * T + c4 * pow(T , 2) + c5 * pow(T, 3) + c6 * log(T) );
                                        /* vapor presure */
                ym = F_UDMI(f, tm, 0);  /* define the wat. weight composition on the meat face */
                xm = ym/(1-ym);         /* water content of the meat on the interface */
                aw = F_UDMI(f,t,1);
                aw = water_activity(aw, xm);
                /* water activity */
                F_UDMI(f,t,1) = aw;
                x = aw * Pw / Pt;       /* molec. compos. water /air */
                yf = F_YI(f, t, i);     /* define the actual wat. weight  comp. on the air face */
                y = x * Mwater / (x * Mwater + (1 - x) * Mair);/* define the new wat. weight comp.
                                        water / air */

                dy = y - yf;

                F_PROFILE(f, t, n) = yf + alfa * dy;

         }
        end_f_loop(f,t)
```

```
}

DEFINE_PROFILE(temperature_air_interf, t, n)
{

        Domain *domain;
        Thread *tm;                                     /* Thread of the surface in contact with
meat */
        real T;                                         /* Temperature */
        real Tnew;
        real Told;
        real dT;
        face_t f;

        domain = Get_Domain(1);
        tm = Lookup_Thread(domain, surf_meat_ID);       /* Thread of the wall in contact with
meat*/


        begin_f_loop(f, t)
         {
                Told = F_T(f, t);               /* air temperature  */
                Tnew =  F_T(f,tm);              /* air temperature equal to meat temperature */

                dT = Tnew - Told;
                T = Told + alfa * dT;

                F_PROFILE(f, t, n) = T;         /* air temperature equal to meat temperature */


         }
        end_f_loop(f,t)


}



DEFINE_ADJUST(solve_meatwater, domain)
{
        int l = 0;                      /* integer that identify the face */
        int k;                          /* integer tgat identify the node to calculate the meat water
diffusion */
        real Tf;                        /* Face Temperature  K */
        real Tc;                        /* cell temperature */
        real yp;                        /* yp = distance between the point p and the wall */
        real De;                        /* Difusivity on the east */
        real Dw;                        /* Diffusivity on the west */
        real Ypto;                      /* water concentration in node p at time old t*/
        real Yeto;                      /* water concentration in node east at time old t*/
        real Ywto;                      /* water concentration in node west at time old t*/
        real ae;                        /* ae*/
        real ap;                        /* ap*/
        real aw;                        /* aw*/
        real awall;                     /* aw in the wall */
        real apo;                       /* apo */
        real su;                        /* su term */
```

```
real dx;                        /* longitud of the element dx */
real dt;                        /* time interval */
real dxep;                      /* dx east */
real dxwp;                      /* dx west */
real Xe;                        * X position of the east boundary */
real Xe1;                       /* X position of the east boundary of the first element*/
real Xw;                        /* X position of the weast bondary */
real b;                         /* source term given the mass diffusion */
real curr_ts;                   /* current time */
real Factor;                    /* factor use to solve the TDM */
real Dface;                     /* Difusivity at the face */
real Dcell;                     /* Difusivity at the cell node or yp distance from the wall */
real mass_flux;                 /* define the mass flux on the wall (face) on Kg Water / s m2
*/
real A[ND_ND];                  /* vector Area of the face */
real Area;                      /* Magnitud of the vector Area */
real Aunit[ND_ND];              /* unit vector perpendicular to the area, same direction of the
                                area vector */
real Cface[ND_ND];              /* face centroid vector */
real Ccell[ND_ND];              /* cell centroid vector */
real dro[ND_ND];                /* vector distance connecting the cell and face centroid */
real ds;                        /* distance between the cell centroide and face centroide */

real DD[20];                    /* principal diagonal */
real EE[20];                    /* upper diagonal */
real CC[20];                    /* lower diagonal */
real BB[20];                    /*  vector */
real XX[20];                    /* vector solution */
real flow_time;                 /* flow time */
real Wlossdt1;                  /* weight loss in dt with the mass flux average*/
real Zero[ND_ND];


Thread *tm;                     /* Thread of the wall in contact with meat*/
Thread *ta;                     /* Thread of the wall in contact with air*/
face_t f;
Thread *to;
cell_t co;

Zero[0] = 0;
Zero[1] = 0;
Zero[2] = 0;

dt = RP_Get_Real("physical-time-step");     /* time interval */
flow_time = RP_Get_Real("flow-time");
tm = Lookup_Thread(domain, surf_meat_ID);/* Thread of the wall in contact with meat*/
ta = Lookup_Thread(domain, surf_air_ID);  /* Thread of the wall in contact with air*/
Wlossdt1 = 0;

curr_ts = CURRENT_TIME;
if (last_ts != curr_ts)
{
        last_ts = curr_ts;
        niter = 1;

        begin_f_loop(f, tm)
          {

                co = F_C0(f,tm);                        /* cell asociated to the face */
```

```
to = THREAD_T0(tm);                    /* cell thread for co */
Tf = F_T(f,tm);                        /* Face Temperature  K */
Tc = C_T(co, to);                      /* cell temperature */
Dface = 1.09e-6*exp(-2507.52/Tf);      /* Difusivity at the face */
Dcell = 1.09e-6*exp(-2507.52/Tc);      /* Difusivity at the cell node or
                                          distance from the wall */
mass_flux = F_UDMI(f, ta, 6);
b =  -mass_flux/denmeat;

/********** calculation of the distance between the cell node and cell face =
yp  ********/
F_AREA(A, f, tm);
Area = NV_MAG(A);                                  /* Area */
F_CENTROID(Cface, f, tm);

C_CENTROID(Ccell, co, to);

NV_VV(dro, =, Cface, -, Ccell);    /* vector distance          between the
                                      face and cell centroids */
NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));/* unit vector perpendicular to
                                 the area, same direction of the area vector */
ds = NV_MAG(dro);        /* distance between the face and cell centroids */
yp = NV_DOT(dro, Aunit);/* yp = distance between the point p and  wall */

/*****************************************************************************
**************/

/*************calculation of the weight loss ****************************/
Wlossdt1 = Wlossdt1 + mass_flux * Area * dt;

/*********************************************************************/

for (k = 1; k <= nnodes; k++)
{
        /***** positions dx, dxep, dxwp ***/
        switch(k)
        {
                case 1:
                        Xe = (Xpos[k+1] + Xpos[k])/2;
                        Xe1 = Xe;
                        Xw = 0;
                        break;
                case 20:
                        Xe = Xpos[k+1];
                        Xw = (Xpos[k] + Xpos[k-1])/2;
                        break;
                default:
                        Xe = (Xpos[k+1] + Xpos[k])/2;
                        Xw = (Xpos[k] + Xpos[k-1])/2;
        }
        dx = Xe - Xw;
        dxep = Xpos[k+1] - Xpos[k];
        dxwp = Xpos[k] - Xpos[k-1];
        /*******************************/

        /******************* calculation of De and Dw *********/
        if (Xe <=  yp)
        {
                De = Dface + (Dcell - Dface)/(yp - 0)*(Xe - 0);
```

```
                }
                else
                {
                        De = Dcell;
                }
                if (Xw <= yp)
                {
                        Dw = Dface + (Dcell - Dface)/(yp - 0)*(Xw - 0);
                }
                else
                {
                        De = Dcell;
                }

/*********************************************************/

                apo = dx/dt;
                aw = Dw/dxwp;
                ae = De/dxep;
                Ypto = F_UDMI(f,tm, k);
                switch(k)
                {
                        case 1:
                                Yeto = F_UDMI(f,tm, k+1);
                                su = ae * Yeto/2 + (apo - ae/2)*Ypto + b;
                                awall = aw;
                                aw = 0;
                                break;

                        case 20:
                                Ywto = F_UDMI(f,tm, k-1);
                                su = aw * Ywto/2 + (apo -ae/2 - aw/2)*Ypto +
                                ae*Yomeat;
                                break;
                        default:
                                Yeto = F_UDMI(f,tm, k+1);
                                Ywto = F_UDMI(f,tm, k-1);
                                su = ae * Yeto/2 + aw * Ywto/2 + (apo - aw/2 -
                                ae/2)*Ypto;


                }
                ap = 0.5*(aw +ae) + apo;
                DD[k] = ap;
                EE[k] = -ae/2;
                CC[k] = -aw/2;
                BB[k] = su;
        }
        /******************TDM matrix solution
        ****************************/
        /*********** Forward Elimination ****************/
        for (k = 2; k <= nnodes; k++)
        {
        Factor = CC[k] / DD[k- 1];
        DD[k] = DD[k] - EE[k - 1] * Factor;
        BB[k] = BB[k] - BB[k - 1] * Factor;
        }

        /************** Back substitution ********************/
```

```
                XX[nnodes] = BB[nnodes] / DD[nnodes];
                F_UDMI(f,tm, nnodes) = XX[nnodes];

                for (k = (nnodes-1); k >= 1; k--)
                {
                XX[k] = (BB[k] - EE[k] * XX[k + 1]) / DD[k];
                        F_UDMI(f,tm, k) = XX[k];

                }

/*****************************************************************/
                dxwp = Xpos[1] - Xpos[0];
                F_UDMI(f, tm, 0) = F_UDMI(f, tm, 1) + b/awall;

                l = l+1;


            }
        end_f_loop(f,tm)

        /***************** total accumulative weight loss ************/
        F_UDMI(1,ta,7) = F_UDMI(1,ta,7) + Wlossdt1;        /* acumulative weight lost at
        begining*/
        /*************************************************************/

    }
    else
    {
        niter = niter +1;

    }

}
```

A6. UDFS PROGRAMMED IN C++ TO SOLVE TO HEAT AND MOISTURE TRANSFER ON THE 3D BEEF CARCASS MODEL SOLVING ONLY THE PROCESS INSIDE THE MEAT

```
# include "udf.h"
/* beef was defined as solid and it was only solved the beef*/
/********************************** variables that must be defined
*********************************************************/
/****** constants ******/
real Cpmeat = 3407;                    /* Cp meat */
real kmeat = 0.397;                    /* k meat */
real denmeat = 1111;                   /* meat density */
real Dtmeat = 1.0488e-7;               /* thermal diffusivity */
real Hvap = 2452240;                   /* heat of vaporization of water in J / Kg */
real E = 0.92;                         /* Plaster emisivity */
real kar = 0.4187;                     /* vor karman constant */
real Ew = 9.793;                       /* wall function constant */
real sigma = 5.67e-8;                  /* sigma constant */
real Cmu = 0.0845;                     /* constant of the RNG model */
real Mwater = 18.016;                  /* warter molecular weight*/
real Mair =  28.96;                    /* air molecular weight*/
real R = 8314.47;                      /* universal constant in pa m3 / Kgmol/ K;
/*****************************************************************************
*************************************/

/***************************** Initial and operation conditions
*********************************************************/
/* it must be the same imformation in the boundary condtions panel */
real tsl = 5400;                       /* slaughter time sec */
real Pt = 101325;                      /* cell total presure */
real To = 279.17;                      /* Inlet air temperature */
real Tofloor = 298.15;                 /* floor air temperature */
real yo = 0.005730549;                 /* water mass composition at the inlet */
real yofloor = 0.012452342;            /* water mass composition at the inlet floor conditions */
real RH = 0.985;                       /* Humidity as a fracction */
real Yomeat = 0.75;                    /* initial water composition of meat */
real Tomeat = 315.55;                  /* initial meat temperature */
/*****************************************************************************
*************************************/

/********************** Time variable and vector to store and solve the water diffusion in meat
************/
real last_ts = -1;                 /*  Global variable.  Time s never <0 used in solve_meatwater */
real last_ts2 = -1;                /*  Global variable.  Time s never <0 used in heat_flux_meat */
int nnodes = 20;                   /* number on nodes per face to calculate the concentration inside the
                                   meat */
int niter;                         /* n iterations */
int nitermax = 19;                 /* niter to calculate fluxes */
int lref = 116;                    /* face reference number to print values */
real Xpos[] = {0, 0.000005, 0.000015, 0.00005, 0.0001, 0.00025, 0.0005, 0.00075, 0.001, 0.0015, 0.002,
0.003, 0.004, 0.005, 0.006, 0.008, 0.01, 0.012, 0.014, 0.016, 0.020, 0.024};/**/
real Tairtab[] = {7.2, 7.4, 7.1, 6.9, 6.9, 6.5, 6.7, 6.4, 6.5, 6, 5.3, 5.2, 5.3, 5.5, 5.4, 5.5, 5.5, 5.4, 5.6, 5.7,
5.7};
real timetab[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
real watercont[] = {0.006287615,0.006318742, 0.006186619, 0.006119355, 0.006077779, 0.005917065,
0.005980425, 0.005884068, 0.005900769, 0.005762241, 0.005430462, 0.00537669, 0.005471543,
```

```
0.005465084, 0.005500349, 0.005551778, 0.005513497, 0.005484137, 0.005583786, 0.005630782,
0.005619935};


/*****************************************************************************/

/************************************** program controls
*****************************************************************/
real aweps = 0.0005;            /* coenvergence criterium to find the aw using newthon method */
int Radiation_Control = 1;      /* 1 when take in account the radiation , 0, when do not */
real alfa = 1;                  /* is the relaxation factor in the UDF that calculate the water on the
                                interface */
real alfasc = 1;                /* is the relaxation factor to calculate the inverse Sc or ycstar */
real yeps = 0.01;               /* tolerance calculating ycstar */
int conwall = 1;                /* 1 define solving the wall, 2 wall approach */
int surf_meat_ID = 4;           /* ID zone of the wall in contact with the meat */
int surf_air_ID = 10;           /* ID zone of the wall in contact with the air */
int meat_ID = 3;                /* ID zone of the meat */
int air_ID = 2;                 /* ID zone of the air */
/*****************************************************************************************
**************************************/
/*****************************************************************************************
**************************************/


real water_activity(real aw, real xm)
{

        real xmg = 0.0565;      /* GAB constant */
        real cg = 4.2396;       /* GAB constant */
        real k = 0.9692;        /* GAB constant */

        real f = 0.0488;        /* Lewicki constant */
        real g = 0.8761;        /* Lewicki constant */
        real h = -34.7794;      /* Lewicki constant */

        real fun = 10;          /* function equal to zero to calculate aw */
        real funder;            /* derivative of the function  */

        do
        {
                if ( aw >= 0.958093)
                {
                        fun = xm - f / pow((1-aw),g) + f/(1+pow(aw,h));
                        funder = -f * g / pow((1-aw),(g+1)) -f * h * pow(aw,(h-1))
/pow((1+pow(aw,h)),2);
                }
                else
                {
                        fun = xm - xmg * cg * k * aw/((1-k*aw)*(1+  (cg-1)*k*aw));
                        funder = xmg * cg * k/( (1- k * aw) * ( 1 + (cg-1) * k * aw))*(-1 + aw * (cg-1)
                        * k / (1+ (cg-1)* k * aw) - k * aw /(1- k * aw));
                }


                aw = aw - fun/funder;

        } while (fabs(fun) >= aweps );
```

```
        return aw;

}


real yair(real time)
{
        real yoair;
        int i;
        int k;


        if (time <= tsl)
        {
                yoair = yofloor;
        }
        else
        {
                time = (time - tsl)/3600;
                if (time <= 20)
                {
                        for (i = 0; i <= 19; i++)
                        {
                                if ( timetab[i+1] < time )
                                {
                                k = i+1;

                                }
                        }
                        yoair = watercont[k] + (watercont[k+1] - watercont[k]) / (timetab[k+1] -
                        timetab[k]) * (time - timetab[k]);

                }
                else
                {
                        yoair = watercont[20];

                }

        }

        return yoair;

}



real tempair(real time)
{
        real tair;
        int i;
        int k;


        if (time <= tsl)
        {
                tair = Tofloor;
        }
```

```
        else
        {
                time = (time - tsl)/3600;
                if (time <= 20)
                {
                        for (i = 0; i <= 19; i++)
                        {
                                if ( timetab[i+1] < time )
                                {
                                k = i+1;

                                }
                        }
                        tair = Tairtab[k] + (Tairtab[k+1] - Tairtab[k]) / (timetab[k+1] - timetab[k]) *
                        (time - timetab[k]);

                }
                else
                {
                        tair = Tairtab[20];

                }
                tair = tair +273.15;


        }

        return tair;

}


real massflux(face_t f, Thread *t)
{

        real hm;                        /* define the hm */
        real Tf;                        /* Face temperature */
        real yf;                        /* define the wat. comp. on the face */
        real Pw;                        /* face vapor presure */
        real x;                         /* molec. compos. water /air */
        real ym;                        /* define the wat. weight composition on the meat face */
        real xm;                        /* water content of the meat on the interface */
        real aw;                        /* water activity */
        real c1 = -5800.2206;           /* cte of vapor presure */
        real c2 = 1.3914993;
        real c3 = -0.048640239;
        real c4 = 0.000041764768;
        real c5 = -0.000000014452093;
        real c6 = 6.5459673;
        real mass_flux;
        real flow_time;                 /* time modeled */
        real yoair;

        flow_time = RP_Get_Real("flow-time");
        Tf = F_T(f,t);
        /* face temperature */
        Pw = exp( c1 / Tf + c2 + c3 * Tf + c4 * pow(Tf , 2) + c5 * pow(Tf, 3) + c6 * log(Tf) );
```

```
                                        /* vapor presure */
        ym = F_UDMI(f, t, 0);           /* define the wat. weight composition on the meat face */
        xm = ym/(1-ym);                 /* water content of the meat on the interface */
        aw = F_UDMI(f,t,23);
        aw = water_activity(aw, xm);    /* water activity */
        F_UDMI(f,t,23) = aw;
        x = aw * Pw / Pt;               * molec. compos. water /air */
        yf = x * Mwater / (x * Mwater + (1 - x) * Mair) ;/* define the wat. weight comp. water / air */
        yoair = yair(flow_time);
        if (flow_time <= 3600 )
        {
                hm = F_UDMI(f,t,25);
        }
        else
        {
                hm = F_UDMI(f,t,22);
        }

        mass_flux = hm * (yf - yoair);


        return mass_flux;

}


real heatconvection(face_t f, Thread *t)
{

        real Tf;                        /* define the face temperature */
        real ht;                        /* ht */
        real heat_flux_conv;
        real Toair;
        real flow_time;

        flow_time = RP_Get_Real("flow-time");
        if (flow_time <= 3600 )
        {
                ht = F_UDMI(f, t, 24);
        }
        else
        {
                ht = F_UDMI(f, t, 21);
        }
        ht = F_UDMI(f, t, 21);
        Tf = F_T(f, t);
        Toair = tempair(flow_time);
        heat_flux_conv = ht * (Tf - Toair);


        return heat_flux_conv;


}


real heattotal(face_t f, Thread *t)
{
        real T;                                 /* Face Temperature  K */
```

```
        real mass_flux;             /* define the mass flux on the wall (face) on Kg Water / s m2 */
        real heat_flux_total;       /* define the total heat flux on the wall (face) on W /  m2 */
        real heat_flux_conv;        /* define the convective heat flux on the wall (face) on W /  m2 */
        real heat_flux_rad;         /* define the radiative heat flux on the wall (face) on W /  m2 */
        real dhvap;
        real Toair;
        real flow_time;

        flow_time = RP_Get_Real("flow-time");
        Toair = tempair(flow_time);
        T = F_T(f,t);                       /* Face Temperature  K */
        mass_flux = massflux(f,t);
        heat_flux_conv = heatconvection(f,t);
        heat_flux_rad =  Radiation_Control * E * sigma * (pow(Toair,4) - pow(T,4));
        dhvap = 2.5e6 -2.5e3*(T - 273.15);
        heat_flux_total = - dhvap * mass_flux - heat_flux_conv + heat_flux_rad;


        return heat_flux_total;
}



DEFINE_ON_DEMAND(demand_initialization)
{
        Domain *d;
        Thread *tmeat;                          /* Thread of meat zone */
        Thread *t;
        cell_t c;
        face_t f;


        d = Get_Domain(1);
        tmeat = Lookup_Thread(d, meat_ID);      /* Thread of meat */

        begin_c_loop(c, tmeat)
        {
                C_T(c, tmeat) = Tomeat;         /* Tomeat  */

        }
        end_c_loop(c, tmeat)

        t = Lookup_Thread(d, surf_meat_ID);     /* Thread of the wall in contact with meat*/
        begin_f_loop(f, t)
          {

                F_T(f, t) = Tomeat;             /* Tomeat  */

          }
        end_f_loop(f,t)


        t = Lookup_Thread(d, air_ID);       /* Thread of air */

        begin_c_loop(c, t)
        {
                C_T(c, t) = To -1;              /* T  */

        }
```

```
        end_c_loop(c, t)


}


DEFINE_ON_DEMAND(dem_init_watercontent)
{
        Domain *d;
        int m;                                  /* integer that define the number of nodes */

        Thread *tm;
        Thread *tair;
        face_t f;

        d = Get_Domain(1);



        tm = Lookup_Thread(d, surf_meat_ID);    /* Thread of the wall in contact with meat*/
        tair = Lookup_Thread(d, surf_air_ID);   /* Thread of the wall in contact with air*/

        begin_f_loop(f, tm)
         {

                for (m = 0; m <= nnodes; m++)
                {
                        F_UDMI(f, tm, m) = Yomeat;
                }

                F_UDMI(f, tm, 23) = 0.9999;

         }
        end_f_loop(f,tm)
        F_UDMI(1,tair,3) = 0;     /* acumulative weight lost 1*/
        F_UDMI(1,tair,4) = 0; /* acumulative weight lost 2*/
        F_UDMI(1,tair,5) = 0; /* acumulative weight lost 3*/
}


DEFINE_PROFILE(heat_flux_meat, tm, j)
{

        face_t  f;          /* face of the face loop , has the same number in the air or meat face*/


        begin_f_loop(f, tm)
                {
                        F_PROFILE(f, tm, j) = heattotal(f,tm);
                }
                end_f_loop(f,tm)



}
```

```
DEFINE_ADJUST(solve_meatwater, domain)
{
        int l = 0;                  /* integer that identify the face */
        int k;                      /* integer tgat identify the node to calculate the meat water diffusion
                                    */
        real Tf;                    /* Face Temperature  K */
        real Tc;                    /* cell temperature */
        real yp;                    /* yp = distance between the point p and the wall */
        real De;                    /* Difusivity on the east */
        real Dw;                    /* Diffusivity on the west */
        real Ypto;                  /* water concentration in node p at time old t*/
        real Yeto;                  /* water concentration in node east at time old t*/
        real Ywto;                  * water concentration in node west at time old t*/
        real ae;                    /* ae*/
        real ap;                    /* ap*/
        real aw;                    /* aw*/
        real awall;                 /* aw in the wall */
        real apo;                   /* apo */
        real su;                    /* su term */
        real dx;                    /* longitud of the element dx */
        real dt;                    /* time interval */
        real dxep;                  /* dx east */
        real dxwp;                  /* dx west */
        real Xe;                    /* X position of the east boundary */
        real Xe1;                   /* X position of the east boundary of the first element*/
        real Xw;                    /* X position of the weast bondary */
        real b;                     /* source term given the mass diffusion */
        real curr_ts;               /* current time */
        real Factor;                /* factor use to solve the TDM */
        real Dface;                 /* Difusivity at the face */
        real Dcell;                 /* Difusivity at the cell node or yp distance from the wall */
        real mass_flux;             /* define the mass flux on the wall (face) on Kg Water / s m2 */
        real A[ND_ND];              /* vector Area of the face */
        real Area;                  /* Magnitud of the vector Area */
        real Aunit[ND_ND];          /* unit vector perpendicular to the area, same direction of the area
                                    vector */
        real Cface[ND_ND];          /* face centroid vector */
        real Ccell[ND_ND];          /* cell centroid vector */
        real dro[ND_ND];            /* vector distance connecting the cell and face centroid */
        real ds;                    /* distance between the cell centroide and face centroide */
        real DD[20];                /* principal diagonal */
        real EE[20];                /* upper diagonal */
        real CC[20];                /* lower diagonal */
        real BB[20];                /*  vector */
        real XX[20];                /* vector solution */
        real flow_time;             /* flow time */
        real Wlossdt1;              /* weight loss in dt with the mass flux average*/
        real Wlossdt2;              /* weight loss in dt with the mass flux at the end*/
        real Wlossdt3;              /* weight loss in dt with the mass flux at the middle*/
        real Zero[ND_ND];

        Thread *tm;                 /* Thread of the wall in contact with meat*/
        Thread *tair;               /* Thread of the wall in contact with air*/
        face_t f;
        Thread *to;
        cell_t co;
```

```
Zero[0] = 0;
Zero[1] = 0;
Zero[2] = 0;

dt = RP_Get_Real("physical-time-step");     /* time interval */
flow_time = RP_Get_Real("flow-time");
tm = Lookup_Thread(domain, surf_meat_ID);          /* Thread of the wall in contact with
                                                   meat*/
tair = Lookup_Thread(domain, surf_air_ID);/* Thread of the wall in contact with air*/
Wlossdt1 = 0;
Wlossdt2 = 0;
Wlossdt3 = 0;

curr_ts = CURRENT_TIME;
if (last_ts != curr_ts)
{
        last_ts = curr_ts;
        niter = 1;


        begin_f_loop(f, tm)
         {

                co = F_C0(f,tm);                          /* cell asociated to the face */
                to = THREAD_T0(tm);                       /* cell thread for co */
                Tf = F_T(f,tm);                           /* Face Temperature  K */
                Tc = C_T(co, to);                         /* cell temperature */
                Dface = 1.09e-6*exp(-2507.52/Tf);         /* Difusivity at the face */
                Dcell = 1.09e-6*exp(-2507.52/Tc);         /* Difusivity at the cell node or
                                                          yp distance from the wall */
                mass_flux = massflux( f, tm);
                F_UDMI(1,tair,1) = mass_flux;             /* mass flux at the begining of the
                                                          interval */

                b =  -mass_flux/denmeat;

                /********** calculation of the distance between the cell node and cell face =
                yp  ********/
                F_AREA(A, f, tm);
                Area = NV_MAG(A);                                  /* Area */
                F_CENTROID(Cface, f, tm);
                C_CENTROID(Ccell, co, to);

                NV_VV(dro, =, Cface, -, Ccell);     /* vector distance between the face and
                                                    centroids */
                NV_V_VS(Aunit, = ,Zero, + , A, *, (1/Area));/* unit vector perpendicular to
                                                    area, same direction of the area vector */
                ds = NV_MAG(dro);                   /* distance between the face and cell
                                                    centroids */
                yp = NV_DOT(dro, Aunit);            /* yp = distance between the point p and
                                                    the wall */

/*****************************************************************************
**************/

/*************calculation of the weight loss ****************************/
                Wlossdt1 = Wlossdt1 + mass_flux * Area * dt;
```

```
/*********************************************************************/

                for (k = 1; k <= nnodes; k++)
                {
                        /***** positions dx, dxep, dxwp ***/
                        switch(k)
                        {
                                case 1:
                                        Xe = (Xpos[k+1] + Xpos[k])/2;
                                        Xe1 = Xe;
                                        Xw = 0;
                                        break;
                                case 20:
                                        Xe = Xpos[k+1];
                                        Xw = (Xpos[k] + Xpos[k-1])/2;
                                        break;
                                default:
                                        Xe = (Xpos[k+1] + Xpos[k])/2;
                                        Xw = (Xpos[k] + Xpos[k-1])/2;
                        }
                        dx = Xe - Xw;
                        dxep = Xpos[k+1] - Xpos[k];
                        dxwp = Xpos[k] - Xpos[k-1];
                        /*******************************/

                        /******************* calculation of De and Dw *********/
                        if (Xe <= yp)
                        {
                                De = Dface + (Dcell - Dface)/(yp - 0)*(Xe - 0);
                        }
                        else
                        {
                                De = Dcell;
                        }
                        if (Xw <= yp)
                        {
                                Dw = Dface + (Dcell - Dface)/(yp - 0)*(Xw - 0);
                        }
                        else
                        {
                                De = Dcell;
                        }

/*******************************************************/

                        apo = dx/dt;
                        aw = Dw/dxwp;
                        ae = De/dxep;
                        Ypto = F_UDMI(f,tm, k);
                        switch(k)
                        {
                                case 1:
                                        Yeto = F_UDMI(f,tm, k+1);
                                        su = ae * Yeto/2 + (apo - ae/2)*Ypto + b;
                                        awall = aw;
                                        aw = 0;
                                        break;
```

```
                    case 20:
                            Ywto = F_UDMI(f,tm, k-1);
                            su = aw * Ywto/2 + (apo -ae/2 - aw/2)*Ypto +
                            ae*Yomeat;
                            break;
                    default:
                            Yeto = F_UDMI(f,tm, k+1);
                            Ywto = F_UDMI(f,tm, k-1);
                            su = ae * Yeto/2 + aw * Ywto/2 + (apo - aw/2 -
                            ae/2)*Ypto;


                }
                ap = 0.5*(aw +ae) + apo;
                DD[k] = ap;
                EE[k] = -ae/2;
                CC[k] = -aw/2;
                BB[k] = su;

        }
        /******************TDM matrix solution
        ******************************/
        /*********** Forward Elimination ****************/
        for (k = 2; k <= nnodes; k++)
        {
        Factor = CC[k] / DD[k- 1];
        DD[k] = DD[k] - EE[k - 1] * Factor;
        BB[k] = BB[k] - BB[k - 1] * Factor;
        }
        /************** Back substitution ********************/
        XX[nnodes] = BB[nnodes] / DD[nnodes];
        F_UDMI(f,tm, nnodes) = XX[nnodes];

        for (k = (nnodes-1); k >= 1; k--)
        {
        XX[k] = (BB[k] - EE[k] * XX[k + 1]) / DD[k];
                F_UDMI(f,tm, k) = XX[k];

        }

/*********************************************************************/
        dxwp = Xpos[1] - Xpos[0];
        F_UDMI(f, tm, 0) = F_UDMI(f, tm, 1) + b/awall;

        l = l+1;

/*********************************************************************/
        mass_flux = massflux( f, tm);
        F_UDMI(1,tair,2) = mass_flux;/* mass flux at the end of the interval */

        Wlossdt2 = Wlossdt2 + F_UDMI(1,tair,2) * Area * dt; /* at the end */
        Wlossdt3 = Wlossdt3 + (F_UDMI(1,tair,1) + F_UDMI(1,tair,2))/2* Area * dt;
                                                /* at the middle */

/*********************************************************************/


    }
  end_f_loop(f,tm)
```

```
        /**************** total accumulative weight loss ************/
        F_UDMI(1,tair,3) = F_UDMI(1,tair,3) + Wlossdt1;   /* acumulative weight lost at
                                                            begining*/
        F_UDMI(1,tair,4) = F_UDMI(1,tair,4) + Wlossdt2;   /* acumulative weight lost at
                                                            begining*/
        F_UDMI(1,tair,5) = F_UDMI(1,tair,5) + Wlossdt3;   /* acumulative weight lost at
                                                            begining*/
        /*************************************************************/
    }

}
```

ELSEVIER

International Journal of Refrigeration 26 (2003) 224–231

REVUE INTERNATIONALE DU FROID
INTERNATIONAL JOURNAL OF
refrigeration

www.elsevier.com/locate/ijrefrig

Modelling the chilling of the leg, loin and shoulder of beef carcasses using an evolutionary method

F.J. Trujillo*, Q.T. Pham

Department of Chemical Engineering, University of New South Wales, Sydney 2052, Australia

Received 20 January 2002; accepted 20 July 2002

Abstract

An evolutionary algorithm was used to adjust unknown parameters during the beef cooling process. These parameters are the equivalent diameter and the initial temperature profile, which are difficult to estimate given the irregular geometry, the elapsed time after slaughter and variations in both the air temperature and velocity. The adjusted parameters produced accurate predictions of the center and surface temperature profiles of the leg, loin and shoulder. The adjusted dimensions agreed very well with the measured carcass dimensions. Empirical equations were obtained to correlate this diameter with the weight and fat grade of beef carcasses.

Keywords: Chilling; Beef; Modelling; Process

Modélisation du refroidissement des cuisses, de l'aloyau et de l'épaule de bœuf à l'aide d'une méthode évolutive

Mots clés : Refroidissement ; Bœuf ; Modélisation ; Procédé

1. Introduction

The correct chilling of beef carcasses after slaughter is important from many points of view. The rate of chilling affects microbial growth, weight loss, energy consumption, and quality factors such as meat tenderness. Therefore, the design of chilling processes should rely on an accurate physical model.

Simplified geometric models have been popular for a long time [1,2]. They make use of a simple shape (like a slab, infinite cylinder or sphere) to represent parts of a carcass such as the leg, loin and shoulder. The reason for these models' popularity is that the equations of heat conduction can be readily solved, analytically [3] or by finite differences [4,5]. Finite element models [6–8], while accurately representing the complex geometry and composition of a carcass, are time consuming to construct (especially in three dimensions) and the gain in accuracy and precision often does not justify the extra effort in the case of carcass chilling, where there are large uncertainties and variability in both the product (size, shape, composition) and the processing conditions (temperature and heat transfer coefficients). The shape factor approach [9–11], is a simple alternative but is not suitable for generating cooling curves or for calculating surface temperature.

For a simplified model to work, it is essential that the model's equivalent dimension (diameter) be judiciously chosen. However, it is not easily determined by direct measurement because the shapes of carcass parts are not simple. The modeller must therefore try different equivalent dimensions until a good fit is obtained.

A classical curve fitting method consists in plotting the logarithm of the residual temperature difference against time. From the slope in the linear region, which occurs after a certain time, the equivalent diameter or

* Corresponding author.

0140-7007/03/$20.00
PII: S0140-7007(02)00036-1

Nomenclature

c_P	specific heat (J kg^{-1} K^{-1})
D	diameter of the section (m)
D_i	diameter i of the section (m)
F	AUS-MEAT P8 fatness measurement (mm)
k	thermal conductivity (W m^{-1} K^{-1})
P	perimeter (m)
r	position between the center and the surface (m)
R	cylinder radius or half diameter (m)
R^2	R-squared value of regression
t	time (s)
T	temperature (K)
T_C^0	initial center temperature (K)
T_r^0	initial temperature at position r (K)
T_S^0	initial surface temperature (K)
W	weight of the carcass (kg)

Greek letter

ρ	Density (kg m^{-3})

thickness can be found [12]. This approach is often not applicable in practice because:

- Some freezing may have occurred.
- The ambient temperature may not be constant.
- The heat transfer coefficient may not be constant, due to changes in evaporation rate or air velocity.
- The initial temperature profile in the product may not be uniform, because some time has elapsed (often half to 1 h) between slaughter and entry into the cooling room.

The objectives of this work were: firstly, to determine both the model diameter and the initial temperature profile from the temperature data, using a stochastic curve fitting procedure; secondly, to compare the regressed diameter with the diameter calculated by Davey and Pham [5] from the same carcasses; and thirdly, to find an empirical relationship between the diameter and the weight and fatness of carcasses.

2. Theory

2.1. Heat transfer

Heat conduction inside a solid is governed by the Fourier equation:

$$\rho c \frac{\partial T}{\partial t} = \nabla(k \nabla T) \tag{1}$$

The equation can be solved provided we know the material properties (c_P, the specific heat, ρ the density and k, the thermal conductivity), the environmental temperature T_a and the surface heat transfer coefficient h. Material properties can be calculated from composition [13]. The heat transfer coefficient was calculated with the same procedure established by Davey and Pham [5], which account for free convection, forced convection, radiation and evaporative heat loss.

The heat transfer equation was solved using a finite difference procedure with Crank–Nicolson stepping scheme. The leg was modeled as an infinite cylinder and the loin and shoulder as slabs. Full details of the finite difference procedure can be found in Davey and Pham [5].

2.2. Curve fitting

In Davey and Pham's work [5], the time spent by the carcass on the slaughter floor, during which temperature could not be monitored, was unknown, neither were the slaughter floor temperature and humidity accurately measured. The initial carcass temperature on slaughtering was also subject to some uncertainties due to postmortem heating. Davey and Pham [5] estimated the duration of the slaughter floor time by comparing the calculated and measured values of the total heat release in the chiller, and adjusting the slaughter floor time until the two agree.

In the present work, no assumption was made about slaughter floor time, conditions or postmortem heating. Instead, the equivalent diameter and temperature profile on entry to the chiller were adjusted by minimizing the sum of square errors using the evolutionary minimization method of Pham [14].

Many minimization methods have been proposed over the years. Classical deterministic methods (Powell, Newton, etc.) start out with a guess and advance in the general downhill direction, as judged from the local gradient. For the problem at hand, it was thought that these methods might have some drawbacks. Because they need local gradient information, many of them are not good at dealing with objective functions that contain random errors arising out of measurements or numerical calculations. They follow a single search path, thus, they may end up at a local minimum instead of at the desired global one. They may also "stall" in regions far away from the minimum, where the gradient is negligible and subject to errors. It is for that reason that an evolutionary stochastic curve fitting method was tried.

Evolutionary algorithms [15], of which genetic algorithms are a well known variety, start with a "population"

of trial points ("organisms") chosen randomly, and they "evolve" towards the optimum using various mechanisms such as "reproduction", "mutation" and "selection". Reproduction is the creating of a new and hopefully better trial point, starting from two or more existing trial points. Mutation is the creation of a new trial point by randomly modifying some features of one existing trial point. Selection is the elimination of one or more trial points in order to get a "fitter", or more optimal, population. All these steps are highly randomized. Evolutionary algorithms differ in the nature and frequency of these basic operations. The algorithm used for the present work is Pham's evolutionary algorithm for solving real valued problems [14].

The temperature profile at the beginning of measurements (soon after entry into the cooling room, but about an hour after slaughter) was assumed to follow a power law equation. Thus, it was completely determined by three (unknown) parameter n, T_C^0 and T_S^0:

$$T_r^0 = T_C^0 - \left(T_C^0 - T_S^0\right)\left(\frac{r}{R}\right)^n \tag{2}$$

where T_C^0 is the initial center temperature, T_S^0 is the initial surface temperature, T_r^0 is the initial temperature at the position r placed between the center and the surface, and R is the half dimension of the model (cylinder radius in the case of the leg, half thickness in the case of the loin and shoulder). The validity of this approach was checked by Pham and Coulter [16].

The evolutionary curve fitting method requires that the search range for each unknown parameter be given. The following ranges were used and it was checked that the optimal values did not lie on the extreme of the range:

$T_C^0 = 31–46\ ^\circ C$

$T_S^0 = 4–30\ ^\circ C$

$n = 0–7$

$R = 0.001–0.8$

For those interested in evolutionary algorithms, there is a brief description of the method in Fig. 1. The type of evolution algorithm shown in this table is known as a steady-state population replacement, zero generation gap, random tournament selection. Steady-state means that as soon as a new member is generated, an existing one is eliminated. Zero generation gap means that new and old members are equally likely to be selected for reproduction and elimination (other strategies may involve automatically eliminating the least fit members of the old population). Random tournament selection refers to the way candidates are chosen during the

Overall pseudocode:

Create N random starting vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$
Repeat
  Introduce a new trial point by Reproduction-mutation
  Eliminate an existing trial point by Selection
until convergence criteria are met.

Pseudo-code of reproduction-mutation operation:

Choose at random two members $\mathbf{x}_A$ and $\mathbf{x}_B$ from the population
Decide whether to mutate
If decided to mutate then
  $\Delta\mathbf{x} = (\mathbf{x}_{max}-\mathbf{x}_{min})/4$ (where $\mathbf{x}_{max}$, $\mathbf{x}_{min}$ denote the range of $\mathbf{x}$)
else
  $\Delta\mathbf{x} = \mathbf{x}_A - \mathbf{x}_B$ (where $\mathbf{x}_A$ denotes the "fitter" member of the pair)
end if*
$\mathbf{x}_{N+1} = \mathbf{x}_A + \mathrm{Gauss}(r,s).\Delta\mathbf{x}$

Pseudo-code of selection operation:

Choose at random two members from the population.
Eliminate the less fit member.

Fig. 1. Pseudo-code of the evolutionary method to find the value of a vector $\mathbf{x}$ $(=x_1,x_2\ldots,x_{Nvar})$ that will maximize a function $f(\mathbf{x})$.

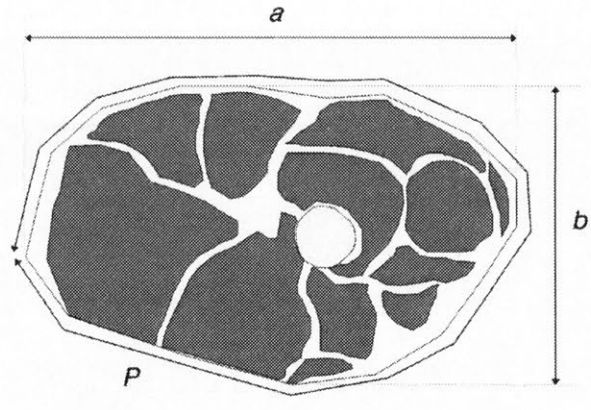


Fig. 2. Example of geometry measurement location for the leg (cross-section 3), a roughly elliptical section: a = largest dimension, b = dimension along direction perpendicular to a, p = perimeter.

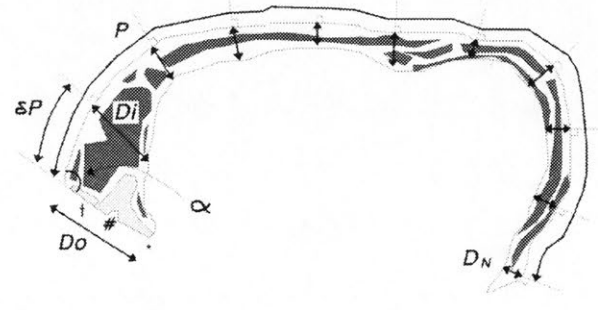


Fig. 3. Example of geometry measurement location for Section 7 (loin).

Table 1
Description of the location of the cross-section positions

Section	Location of the cross-section on the beef side
1	At the base of the Achilles tendon.
2	Midway between Section 1 and Section 3 positions.
3	At the widest part of the rump.
4	At the uppermost tip of the Coccygeal vertebrae.
5	At the narrowest part of the side, between the 1st Sacral vertebra and the 5th Lumbar vertebra.
6	At the base of the 4th Lumbar vertebra.
7	Between the 10th and 11th ribs, taken from the point where the ribs join to the Feather bones on the spine.
8	At the top of the Sternum, below the Xiphoid cartilage (between the 7th and 8th ribs taken from the point where the ribs attach to the sternum).
9	At the base of the 3rd rib, taken from the point where the rib attaches to the Feather bone on the spine.
10	At the base of the 1st rib, taken from the point where the rib attaches to the Feather bone on the spine.
11	At the bottom tip of the 1st Cervical Vertebra.
12	At the position just above the "elbow", where the Humerus bone attaches to the Radius bone.
13	At the position just above the bulge at which the Ulna and Radius bones attach to the Carpus.

elimination step. Gauss(r,s) is a random normal variable with mean r and standard deviation s. The parameter r indicates the extent to which the offspring is biased towards the fitter parent. In our implementation we use $r \approx 1.5$, which means that we *extrapolate* past the fitter parent by about 50% of the difference between $\mathbf{x}_A$ and $\mathbf{x}_B$. s is a "fuzziness factor" that measures the spread of the random deviations of offspring from their expected values, as a fraction of $\Delta\mathbf{x}$.

A population size of 10 and a mutation frequency of 0 was used, but the mutation rate was increased to 0.5 (or 50% of the time) when the "parents" are very close together (1% of search range), to ensure that the procedure escapes from local minima.

3. Experimental details

Geometric measurement and chilling trials were carried out by Davey in an industrial processing plant [5,7]. The beef side was divided into thirteen cross-sections. The location of each cross-section is described in Table 1. Geometric measurements were taken at each cross-section. The leg was treated as an ellipse, for

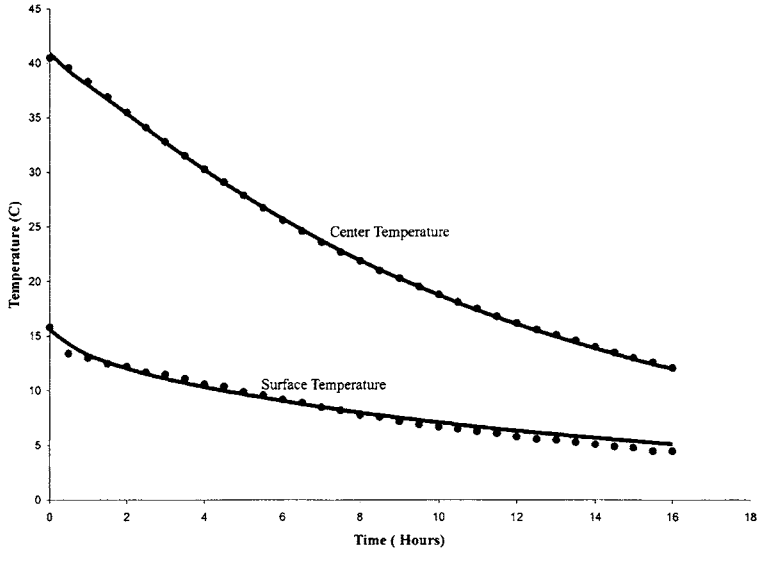


Fig. 4. Experimental and curve-fitted temperature profile for Shoulder. Run 33, carcass weight = 135.50 kg, fat in mm = 23, air velocity = 0.685 m/s, average air temperature = 1.6 °C.

which the minor diameter, major diameter and outer perimeter were measured (Fig. 2). For each loin and the shoulder cross-section, the outer length P was measured (Fig. 3). Then it was divided into equal intervals δP (11 and five intervals for the loin and shoulder respectively). For each interval of δP the thickness of the meat perpendicular to the outer surface, D_i, was measured [5] (Fig. 3).

The carcass was then chilled inside a portable wind tunnel with rectangular cross section (650 mm×1100 mm), placed inside a chiller room [18]. Measurement of surface and center temperatures for the leg, loin and shoulder of the beef side started a few minutes after entry to the chiller and continued for up to 24 h. The leg and shoulder thermocouple positions were located at the cross-section positions 3 and 10 of Table 1 respectively. The loin temperature was measured in the cross-section located at the base of the feather bone of the 13th rib which is placed between the cross-section positions 6 and 7 of Table 1. The air temperature and the air velocity parallel to the carcass were also measured.

4. Results and discussion

The evolutionary minimization method was successful for all the temperature curves considered (36 for leg, 27 for loin and 34 for shoulder). We did not take into

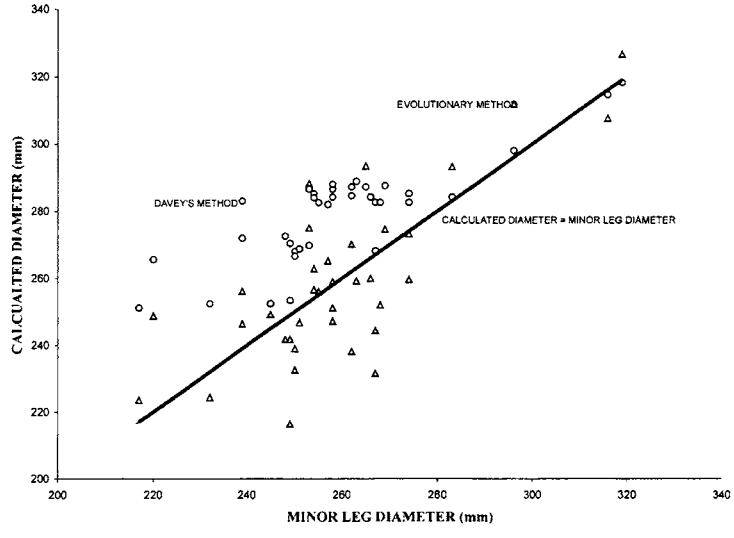


Fig. 5. Leg regressed diameter against minor leg diameter.

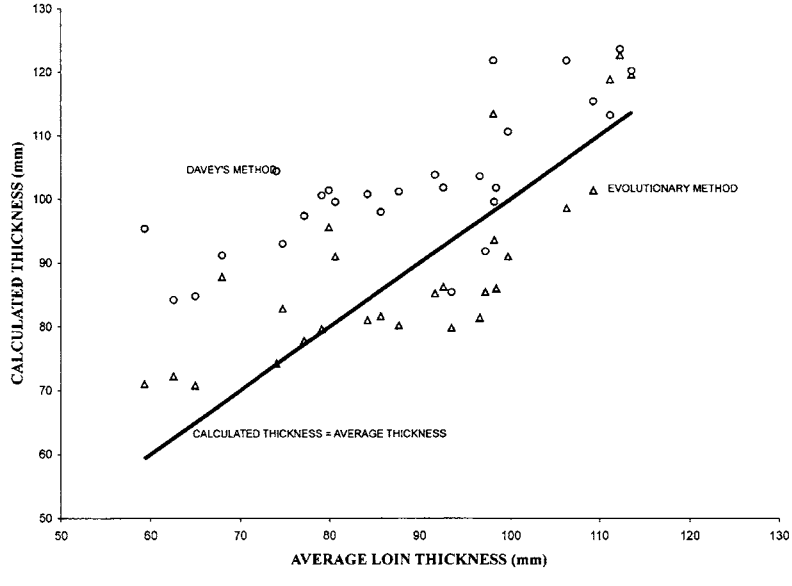


Fig. 6. Loin regressed thickness against average loin thickness.

account the trials that presented grossly unexpected temperature profiles that were likely to be caused by difficulties in placing thermocouples in the correct position. Fig. 4 shows an example of the fit obtained.

The calculations were time-consuming but not overly so. For each curve fit, 3000 finite-difference calculations were carried out (although the search converged within 500–1000 trials), taking 5 min on a 300 MHz Pentium II computer running Windows NT. The finite difference procedure was programmed in the Delphi 4 language and used a Crank–Nicolson finite difference procedure with a time step of 60 s.

Much research on evolutionary algorithms is devoted to "fine tuning" the procedure by adjusting evolutionary parameters such as the rate of mutation, the reproduction operators, the population size, selection technique, etc. In this work, an existing evolutionary program was used without modification, which shows that it is a powerful general-purpose program.

Because several parameters (equivalent diameter and initial temperature profile) were curve-fitted simultaneously, there was a possibility that errors in one parameter were compensated by errors in another. Furthermore even a physically wrong model could still yield good predictions over a certain range if model parameters were given physically implausible values. To see if this could have happened, we plotted the minor leg diameter against the calculated diameter in Fig. 5 using both the evolutionary method and Davey and Pham's [5]. The leg diameter calculated by the evolutionary method agreed better with the measured minor diameter than with the equivalent diameter obtained by Davey's method. In fact, regression line with the intercept set to zero gives

$$\text{Calculated Diameter} = 0.9977 \text{ Measured Diameter}$$

Fig. 6 shows the calculated loin thickness against the average loin thickness. Because the thermocouple was placed on the base of the feather bone of the 13th rib, the average diameter was calculated as the mean of the D0, D1, D2, D3 and D4 diameters of the cross-sections 6 and 7. There is also a very good match between the calculated and measured thicknesses:

$$\text{Calculated Thickness} = 0.9973 \text{ Measured Thickness}$$

Fig. 7 shows the calculated diameter and the D2 diameter of the shoulder. There is also a good correlation between the calculated diameter by the evolutionary method and the D2 diameter.

The good correlation between the adjusted diameter and the geometry, and the good fitting obtained on the cooling curves on the three sections: leg, loin and shoulder, suggest that evolutionary methods are able to give reasonable estimates of the unknown parameters. Furthermore, because the curve-fitting procedure automatically compensates for uncertainties in the initial temperature profile, in the thermal properties, in the calculation of heat transfer coefficient, in the calculation of evaporative effects, and in the irregular geometry, its

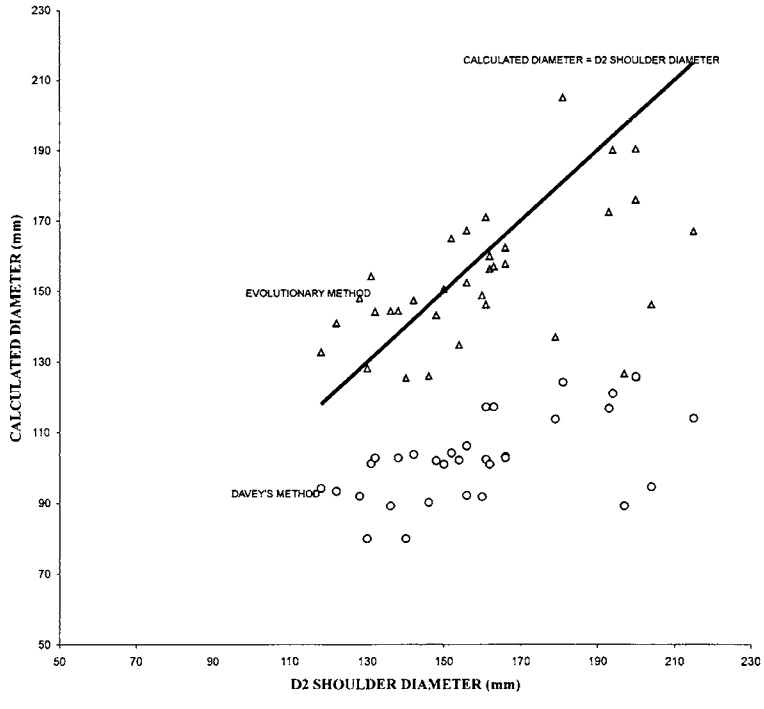


Fig. 7. Shoulder regressed diameter against D2 diameter.

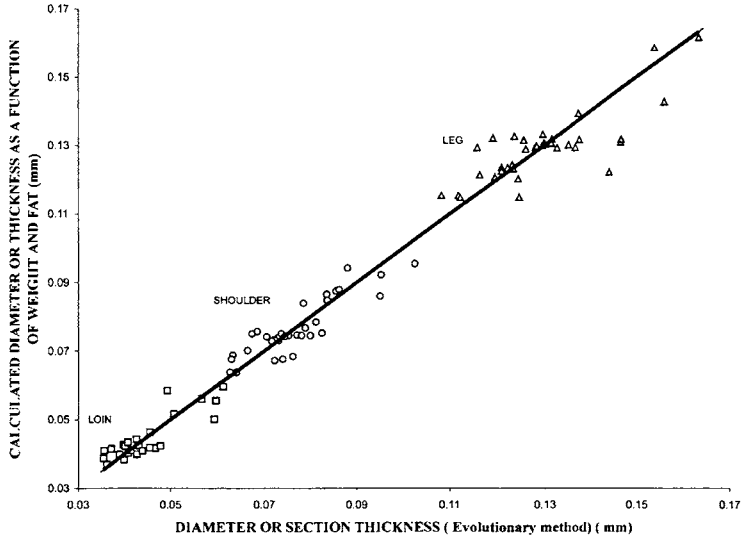


Fig. 8. Calculated diameter against regressed diameter.

results (the curve-fitted half dimensions) can be used with confidence for cooling curve prediction.

Empirical equations were then found to correlate the equivalent dimension with the weight and fatness of beef carcasses:

Leg:

$$D = 0.1025 + 1.55 \times 10^{-6} W^2 + 6.9 \times 10^{-7} F^3$$
$$R^2 = 60.2\% \quad (3)$$

Loin:

$$D = 4.09 \times 10^{-2} + 1.145 \times 10^{-5} W.F - 1.315 \times 10^{-3} F$$
$$R^2 = 73.7\% \quad (4)$$

Shoulder:

$$D = 4.09 \times 10^{-2} + 2.05 \times 10^{-4} W + 7.03 \times 10^{-4} F$$
$$R^2 = 77.7\% \quad (5)$$

where D is the diameter or thickness of the section, W is the weight of the carcass in kilograms and F is the AUSMEAT P8 fatness measurement in millimeters. Fig. 8 plots the calculated diameter with Eqs. (3)–(5) against the adjusted diameter.

5. Conclusion

The center and surface temperature of the leg, loins and shoulder during the cooling of beef carcasses can be modeled accurately with simple geometric models with the half dimensions found by curve fitting using Pham's evolutionary algorithm. It appears that evolutionary methods are well suited for determining unknown parameters from experimental data. They are particularly good at filling gaps in incomplete data as often obtained in industrial situations. They can handle large uncertainties associated with errors in measuring temperatures and truncation errors in numerical calculations, and are good at avoiding local minimum. They are potentially applicable to a wide variety of parameter-determination situations where the mathematical model has to be solved numerically.

References

[1] Bailey C, Cox RP. The chilling of beef carcasses. Proceeding of the Institute of Refrigeration 1976;72:76–90.
[2] Earle RL, Fleming AK. Cooling and freezing of lamb and mutton carcasses. Food Technology 1967;21:79–84.
[3] Carslaw HS, Jaeger JC. Conduction of heat in solids. 2nd ed. Oxford: Clarendon Press; 1959.
[4] Cleland AC. Rood refrigeration processes analysis, design and simulation. London: Elsevier; 1990.
[5] Davey LM, Pham QT. Predicting the dynamic product head load and weight loss during beef chilling using a multi-region finite diffence approach. Int J Refrig 1997; 20(7):470–82.
[6] Arce A, Potluri PL, Schneider KC, Sweat VE, Dutson TR. Modelling beef carcass cooling using a finite element technique. Trans ASAE 1983, 26, 950–4, 960.
[7] Davey LM, Pham QT. A multi-layered two-dimensional finite element model to calculate dynamic product heat load and weight loss during beef chilling. Int J Refrig 2000;23:444–56.
[8] Mallikarjunan P, Mittal GS. Heat and mass transfer during beef carcass chilling- modelling and simulation. J Food Eng 1994;23:277–92.

[9] Crocombe JP, Lovatt SJ, Clarke RD. Evaluation of chilling time shape factors through the use of three-dimensional surface modelling. In: 20th Int Congress of Refrigeration, Sydney, 19–24 September 1999 [Paper 353].

[10] Cleland DJ, Cleland AC, Earle RL. Prediction of freezing and thawing times for multidimensional shapes by simple formulae. Int J Refrig 1987;10:234–40.

[11] Lin Z, Cleland AC, Sellarach GF, Cleland DJ. A simple method for prediction of chilling times: extension to three-dimensional irregular shapes. Int J Refrigeration 1996;19: 107–14.

[12] Pflug IJ, Blaisdell JL, Kopelman IJ. Developing temperature-time curves for objects that can be approximated by a sphere, infinite plate or infinite cylinder. ASHRAE Transactions 1965;71:238–48.

[13] Pham QT. Prediction of thermal conductivity of meats and other animal products from composition data. In: 5th International Congress of Engineering and Foods (ICEF 5), Cologne, 1989.

[14] Pham QT. Competitive evolution: a natural approach to operator selection. AI 94 Evolutionary Computational Workshop, 21 Nov, Armidale, Australia, 1994.

[15] Michalewicz Z. Genetic algorithms + data structures = evolution programs. New York: Springer Verlag; 1992.

[16] Pham QT, Coulter S. Modelling the chilling of pig carcasess using and evolutionary method. In: Proceedings of the 19th International Congress of Refrigeration, The Hague, vol. 3a, 20–25 Aug 1995. p. 676–83.

[18] Davey LM. Measurement and prediction of product heat load and weight loss during beef chilling. PhD thesis, University of New South Wales, Sydney, 1998.

ELSEVIER

Journal of Food Engineering 60 (2003) 357–366

JOURNAL OF FOOD ENGINEERING

www.elsevier.com/locate/jfoodeng

Moisture sorption isotherm of fresh lean beef and external beef fat

Francisco Javier Trujillo, Pei Ching Yeow, Q. Tuan Pham *

School of Chemical Engineering and Industrial Chemistry, University of New South Wales, Sydney 2052, Australia

Received 12 July 2002; accepted 5 February 2003

Abstract

The moisture sorption isotherm (MSI) of lean beef and fat beef was experimentally determined. The experimental procedure used was that of the COST 90 project with some modifications to accelerate equilibration. The procedure was validated with the standard reference material microcrystalline cellulose. The MSI of the beef at the highest humidity range was obtained by accelerating equilibration with changes of salts, using a low water activity salt for some time. This procedure was reliable for beef samples but not for the fat samples. No significant changes were found for lean beef in the temperature range 5–40 °C. Three models, GAB, Peleg and Lewicki, were used to fit the experimental data. The best fit was obtained with the GAB equation. The fat MSI was determined at 5, 15 and 25 °C and it was best fitted with the Lewicki model.

Keywords: Beef; Fat; Moisture sorption isotherm; Water activity; Desorption

1. Introduction

Water activity is one of the most important factors that affect the transfer of moisture in meat thermal processing operations: drying, chilling, freezing, heating, cooking, storage and transport. It is also important in the preservation and quality of meat. For instance, microbial growth on the surface of foods is mainly controlled by water activity, temperature and pH (Ross, 1999).

When food is exposed to air, the surface water content is the determined by the equilibrium between evaporation and internal water migration (Baucour & Daudin, 2000). For a given food, water activity is function of the water content (X) and the temperature. Knowledge of this relationship (the moisture sorption isotherm, MSI) is essential for the prediction of evaporative losses and potential for microbial growth during meat chilling.

Previous work has been done to determine the MSI of cooked and raw meats (Delgado & Sun, 2002a; Iglesias & Chirife, 1982). Palnitkar and Heldman (1971) determined the adsorption and desorption isotherm of freeze-dried precooked beef at 21.1 °C. Saravacos and Stinchfield (1965) determined the adsorption isotherm of freeze-dried meat in the range −20 to 50 °C. For fresh raw beef, few results has been reported, although Taylor (1961) determined the desorption moisture isotherm of raw beef at 19.5 °C. Also, many difficulties are frequently encountered with published moisture sorption data. For example, information on the history and pretreatment of the food sample is frequently not properly reported.

Davey (1998) reported that the amount of fat that cover the meat carcasses directly affect the weight lost during beef chilling. Thus, it was decided to determine the MSI of lean meat and external beef fat.

The water activity of the meat was determined in the range of $a_w = 0.3$–0.98. The method of the COST 90 project (Spiess & Wolf, 1987) was modified to accelerate the process and avoid spoilage or fungal growth. Determining the MSI at high values of a_w (over 0.90) presents a big problem, since a long time is required to reach the equilibrium (due to the low driving force and large moisture content to be evaporated) and spoilage becomes inevitable (Baucour & Daudin, 2000; Wolf, Spiess, & Jung, 1985). However, this range must be considered because it is important in the chilling of beef carcasses where the meat surface is nearly fully wetted for considerable periods (Herbert, Lovett, & Radford, 1978).

* Corresponding author.
E-mail address: tuan.pham@unsw.edu.au (Q.T. Pham).

0260-8774/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0260-8774(03)00058-X

Nomenclature

a_w	water activity
a, b, c, d	constants of the Peleg equation
C_g	constant of the GAB equation
d_d	diffusion distance
D	diffusivity
D_{crit}	critical difference between experimental and literature values
F_t	F-statistic
F, G, H	constants of the Lewicki equation
K	constant of the GAB equation
K_b	Boltzmann constant
RMS	root mean square
T	temperature (Kelvin)
X	water content (g water/g dry solid)
X_{exp}	experimental value of X
X_{mean}	mean water content
X_{mg}	constant of the GAB equation
X_{model}	value of X given by the mathematical model
X_{lit}	literature water content

Greek symbols

μ	viscosity
σ	standard deviation
τ	characteristic time for molecular movement

There are several equations available to describe the MSI of food materials (Berg & Bruin, 1981). However, no single model fits the whole range of a_w accurately. The obtained data was fitted using the following models: GAB (Timmerman, Chirife, & Iglesias, 2001), Peleg (Peleg, 1992) and Lewicki (Lewicki, 1998).

2. Material and methods

2.1. Previous techniques of sorption measurements

Current methods to determine the MSI require the food samples to reach equilibrium with the surrounding atmosphere and then the water content X of the sample is obtained by weighing. The most accepted and standardized method is that of the COST 90 project (Spiess & Wolf, 1987), which uses a standard sorption apparatus, procedure and reference material (microcrystalline cellulose or MCC) (Wolf et al., 1985).

The COST 90 equipment consists of a 1 litre sorption flask containing a weighing bottle which contains 1 g of food. At the bottom of the flask is a slurry of analytical grade salts with known a_w. The time recommended by the COST 90 project to reach equilibrium is four days for MCC powder (Spiess & Wolf, 1987) and at least 14 days for real food materials (Wolf et al., 1985). Sing, Rao, Anjaneyulu, and Patil (2001) reached the equilibrium of smoked chicken sausages in 2–3 weeks following the COST 90 project. With a perishable food such as raw beef we cannot afford to take such long equilibration times because of bacterial and fungal degradation. To avoid spoilage, it has been suggested to add small amounts of fungicides like phenyl mercury acetate and thymol; however the consequences on the product on the isotherm are difficult to assess (Baucour & Daudin, 2000). Therefore, some modifications of the technique were made to accelerate the mass transfer process and reduce the equilibration time.

2.2. Our modifications to the COST 90 method

The geometry of the sorption container, the distance between the sample and the liquid surface and the sample geometry all affect the rate of mass transfer.

According to Lang, McCune, and Steinberg (1981), in an equilibrium environment between a food product and saturated salt slurry, the driving force is the difference in vapor pressure. Thus, the faster the vapor space reaches equilibrium with the saturated salt slurry, the quicker the maximum driving force for water desorption will be applied to the sample. A reduction in the size of the sorption container would reduce the diffusion distance and accelerate the process, since according to Boltzmann's equation (Labuza & Hyman, 1998), the effective diffusion time decreases inversely with the square of distance:

$$D = \frac{K_b \cdot T^{\tau}}{3\Pi\mu d_d^2} \tag{1}$$

The container size should be just enough to accommodate a single sample to ensure the highest possible area–volume ratio (Lang et al., 1981), and the container height should be approximately equal to container diameter (Spiess & Wolf, 1987). Thus the sorption jars used in this experiment were 5.2 cm tall by 5.3 cm diameter (Fig. 1).

If the jar is not hermetic, water vapor loss will cause a humidity gradient. Yeow (2001) made some preliminary experiments to determine whether this would cause a significant humidity difference between the salt surface and the sample location. The rate of weight loss from the whole jar was determined, and the air between the salt slurry surface and the lid of the jar was modeled as a

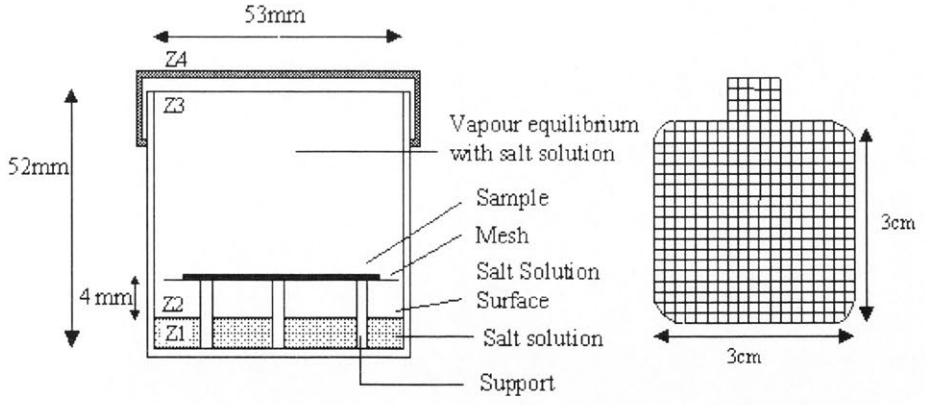


Fig. 1. Schematic diagram of sorption jar and sample holder used in the experiments.

stagnant film (Bird, Steward, & Lightfood, 1960), from which the humidity gradient can be calculated. To avoid the sample getting wet, it was placed 4 mm above the liquid surface. Calculations showed that at this distance, the deviation of the vapor pressure from equilibrium is less than 1% and can be neglected.

In the COST 90 method, the sample is held in a weighing bottle that resembled a miniature kettle. The weighing bottle is placed above the saturated salt in a closed container and is weighed regularly to detect the first sign of equilibration. During weighing the bottle is closed to minimize water adsorption. The design of the weighing bottle imposes great resistance to the mass transfer of the desorption process and requires a long time to equilibrate with the salt. Hence, in our experiments a mesh was used to hold the sample, and thus minimize the external resistance (Fig. 1).

The preparation of the sample also affects the mass transfer rate. The diffusion process may be accelerated by increasing the external area and reducing the thickness of the sample. Thus, the samples of meat were cut into thin strips.

2.3. Measurement of the moisture sorption isotherm

The COST 90 method was followed with the modifications described before. Samples of eye fillet and external beef fat were used on the test. The eye fillet was obtained from the butcher the same day that the beef was slaughtered. Pham and Karuri (1999) reported that during chilling, beef carcasses lose moisture only in the first few centimeters below the surface. Thus, slices were cut from the inside of the piece of beef to obtain samples with high initial water content. The slices were cut into thin strips of approximately 2 mm × 2 mm × 25 mm along the direction of the fibers to avoid rupturing the fibers. 0.3 g samples were taken from the strips to conduct the experiments.

The cut samples were put on stainless steel mesh and placed into the jar described in Fig. 1. Ten analytical grade salts (Table 1) were used to make saturated slurries of known a_w (Greenspan, 1976). The slurries were placed into a water bath to keep the temperature at 25 and 40 ± 1 °C. A refrigerator with on–off controller was used to keep the temperature in 5 and 15 ± 2 °C. The samples were weighed at regular intervals of time until equilibrium was reached. The criteria to determine the equilibrium were when the slope of curve of weight vs. time reached a value close to zero and the difference between three consecutive weighings were less than 0.5 mg. The tests were done in triplicate and the ash, protein, fat and moisture content of the batch of meat were determined following the procedure of Greenfield, Arcot, Emerson, Hutchison, and Wills (1998).

The samples of fat were of about 1.0 g and were cut in small slices because it was not possible to cut it in small strips. The temperature was fixed at 5, 15 or 25 °C. It was not possible to carry out experiments at 40 °C because the fat melted.

2.4. Equilibration of high humidity samples

Our preliminary tests confirmed previous reports (Baucour & Daudin, 2000; Wolf et al., 1985) that, at high humidities, the mass transfer is very slow, making it difficult to reach equilibrium in the range 0.9–1.0 a_w. Moreover, at these high a_w fungal and bacterial growth proceed quickly. For that reason it was necessary to accelerate the process on this range. Because the driving force is the a_w difference between the surrounding air and the sample, the first stage of the desorption process was accelerated by putting the sample in $MgCl_2$, which have the lowest humidity (a similar procedure was used by Delgado and Sun (2002b) on chicken meat to accelerate the equilibrium process). The sample was weighed at constant time and the sample was transferred to a high-humidity jar when its weight is within about 12% of the expected equilibrated value.

Table 1
Experimental moisture sorption isotherms of lean beef at 5, 15, 25 and 40 °C

Salt	$T = 5$ °C		$T = 15$ °C		$T = 25$ °C		$T = 40$ °C	
	a_w	X	a_w	X	a_w	X	a_w	X
K_2SO_4	0.9842	1.2842 ± 0.1288	0.9789	–	0.9730	1.1681 ± 0.0355	0.9541	0.9949 ± 0.0101
KNO_3	0.9627	0.7670 ± 0.0335	0.9541	0.6209 ± 0.0328	0.9358	0.5170 ± 0.0573	0.8903	0.3691 ± 0.0418
KCl	0.8767	0.3534 ± 0.0386	0.8592	0.3039 ± 0.0283	0.8434	0.2763 ± 0.0226	0.8232	0.2775 ± 0.0116
KBr	0.8509	0.3070 ± 0.0152	0.8262	0.3173 ± 0.0825	0.8089	0.2337 ± 0.0100	0.7942	0.2304 ± 0.0070
NaCl	0.7565	0.2073 ± 0.0334	0.7561	0.2095 ± 0.0210	0.7529	0.2024 ± 0.0267	0.7468	0.2062 ± 0.0083
KI	0.7330	0.1884 ± 0.0273	0.7098	0.1629 ± 0.0321	0.6886	0.1574 ± 0.0246	0.6609	0.1470 ± 0.0034
NaBr	0.6351	0.1371 ± 0.0219	0.6068	0.1015 ± 0.0318	0.5757	0.1023 ± 0.0435	0.5317	0.0923 ± 0.0048
$Mg(NO_3)_2$	0.5886	0.1230 ± 0.0223	0.5587	0.1002 ± 0.0101	0.5289	0.0999 ± 0.0184	0.4842	0.0724 ± 0.0101
NaI	0.4242	0.0955 ± 0.0162	0.4088	0.0807 ± 0.0042	0.3817	0.0545 ± 0.0148	0.3288	0.0437 ± 0.0029
$MgCl_2$	0.3360	0.0861 ± 0.0213	0.3330	0.0792	0.3278	0.0517 ± 0.0237	0.3160	–

The a_w values were obtained from Greenspan (1976).

3. Results and discussion

3.1. Validation of the method with microcrystalline cellulose

The method was validated with MCC (Avicel™ PH101 50 micron powder) following the recommendations of Spiess and Wolf (1987). Because the MCC came as a micron powder and the mesh could not hold such fine powder, the mesh was lined with aluminum foil.

With our reported modifications to the method, the time to reach the equilibrium for MCC was reduced from four days to about 20 h, except the sample at $a_w = 0.9710$ which took 40 h (see Fig. 2). Thus, we have accelerated the equilibration rate by 3–4 times compared to the COST 90 method.

The four replicate experimental results of MCC sorption isotherm were compared to the 32 collaborative participating laboratories COST 90 project (Fig. 3). The literature results given by COST 90 were reported in the form of the GAB equation (Spiess & Wolf, 1987).

According to Spiess and Wolf (1987), if n replicate determinations are performed, then good agreement, on a 95% probability level, is obtained if the difference $|X_{mean} - X_{lit}|$ between the result and the reference value is equal to or smaller than a critical difference $D_{crit} = 0.005$. Our MCC results for the nine salts with the lower a_w meet the criterion of $|X_{mean} - X_{lit}| < D_{crit}$,

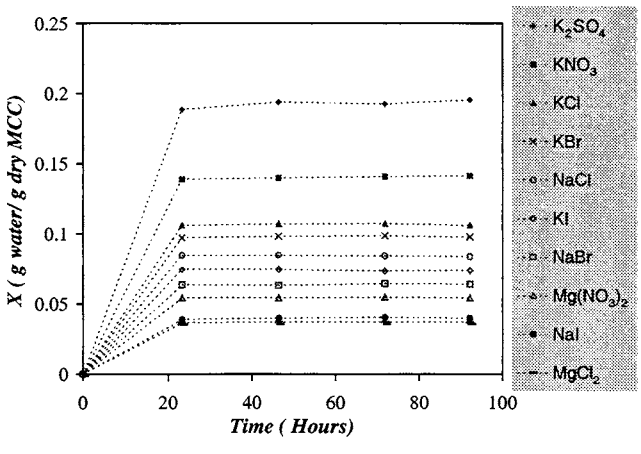


Fig. 2. Kinetic of the water adsorption of MCC.

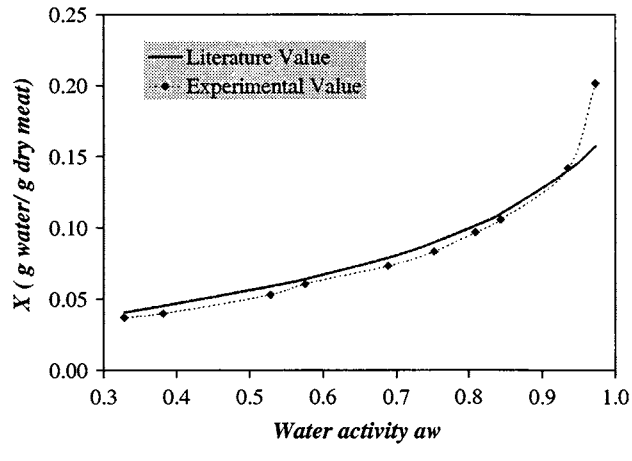


Fig. 3. Moisture sorption isotherm of MCC.

therefore, the experimental results obtained with this method are on average within the 95% confidence interval. The highest a_w (K_2SO_4, $a_w = 0.9730$) did not meet that criteria but this was attributable to the fact that the GAB literature equation was fitted only in the interval 0.1115–0.9026. It has been reported that this equation gives a good fit of food MSI in the interval $0.10 < a_w < 0.90$ (Timmerman et al., 2001). Thus, the reported equation for MCC is not reliable for $a_w > 0.9026$.

3.2. Experimental results for meat moisture isotherm

Figs. 4 and 5 show the change of the water content of meat with time at 25 and 5 °C. All samples reached equilibrium within 30–80 h, except at high air humidities (K_2SO_4 at 25 °C, K_2SO_4 and KNO_3 at 5 °C). However, when desorption was accelerated by first exposing the sample to low humidity (over $MgCl_2$ for 2 h), equilibration was achieved within 20 h (Fig. 6).

To see if this acceleration method affected the equilibrium moisture content, the final moisture content of meat samples was determined using KCl, KBr and $Mg(NO_3)_2$ at 25 °C and KNO_3 at 40 °C. For each salt, three replicate meat samples were allowed to reach the equilibrium without acceleration and another three replicates were placed first over $MgCl_2$ during the first hours. Analysis of variance showed that there is no significant difference at 95% confidence between the two treatments (Table 3). Therefore the acceleration method was considered acceptable.

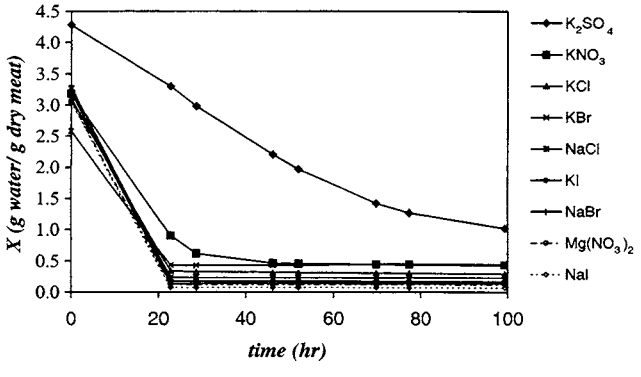


Fig. 4. Change of the water content with time at 25 °C with no change of salt. The sample at high humidity (K_2SO_4) could not reach the equilibrium but the others did.

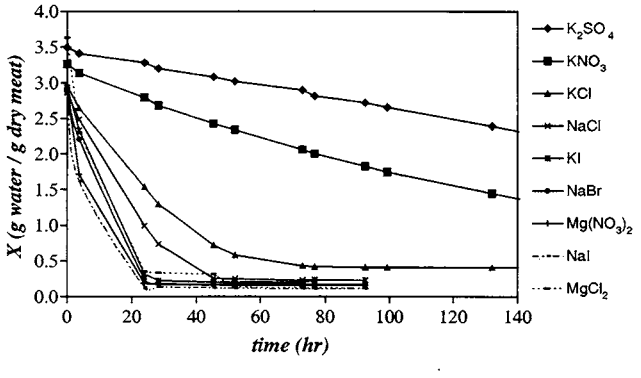


Fig. 5. Change of the water content with time at 5 °C with no change of salt. The samples reach the equilibrium except at the two higher humidity values (K_2SO_4 and KNO_3).

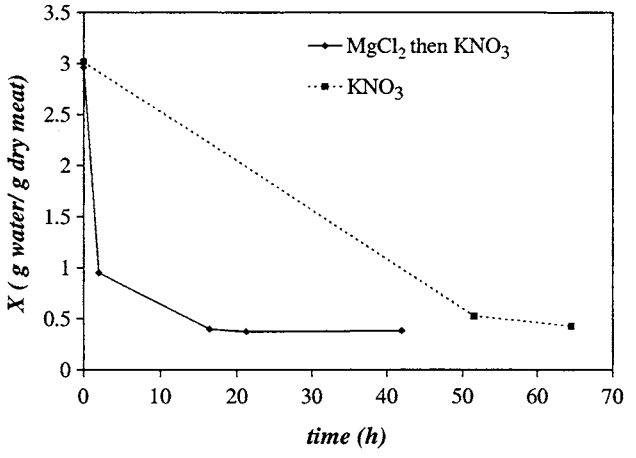


Fig. 6. Accelerated desorption process of meat. $MgCl_2$ used during the fist 2 h then KNO_3.

The desorption isotherm of lean meat, at 5, 15, 25 and 40 °C, are shown in Table 1 and Fig. 7. The influence of salts, sample composition and temperature on the MSI was subjected to analysis of variance. No significant difference was found for sample composition at a confidence level of 95%. Table 2 shows the average composition of the meat. The factor that most affects the variance is the type of salt i.e. water activity. No effect was found for temperature. Thus, it was assumed that the MSI of lean beef is independent of temperature in the range 5–40 °C. Saravacos and Stinchfield (1965) found that the adsorption moisture isotherm of freeze-dried beef shows a maximum between 10 and 20 °C. Their isotherms at 0, 10 and 20 °C overlap in most of the range of humidity.

3.3. Curve fitting the experimental results

Several equations are available on the literature to fit MSI data (Berg & Bruin, 1981). The following three were used:

Peleg:

$$X = a \cdot a_w^b + c \cdot a_w^d \tag{2}$$

GAB:

$$X = \frac{X_{mg} \cdot C_g \cdot K \cdot a_w}{(1 - K \cdot a_w)[1 + (C_g - 1)K \cdot a_w]} \tag{3}$$

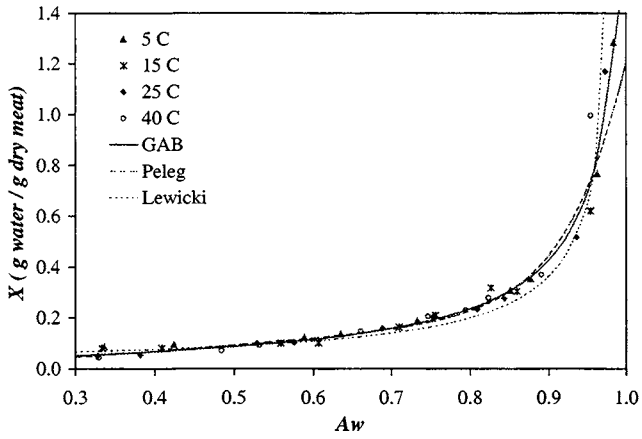


Fig. 7. Moisture sorption isotherm of beef in the range 5–40 °C.

Table 2
Average composition of the lean meat

	Units	Average	σ
Fat	g fat/ g wet sample	0.156	0.282
Ash	g ash/ g wet sample	0.018	0.018
Water content	g water/ g dry material	2.779	0.171
Protein	% protein of dry mass basis	77.87	4.54

Lewicki:

$$X = \frac{F}{(1-a_w)^G} - \frac{F}{1+a_w^H} \tag{4}$$

where X is the water content in solid; a_w the water activity; and all other symbols represent constants in the equations.

The Peleg equation can predict both sigmoidal and non-sigmoidal isotherms. According to Peleg (1992), this model fitted as well as or better than the GAB model but its constants have no physical meaning.

The GAB equation is a semi-theoretical multilayer sorption model with a physical meaning for each constants (Timmerman et al., 2001). The model is applicable to a wide range of a_w (0.1–0.9) but it has been reported that the error increased sharply for values of a_w above 0.9. At higher water activities, the GAB plot presents a downward deviation due to the appearance of a third sorption stage (Timmerman & Chirife, 1991), an effect that determines the upper limit of application of the GAB equation. The main difficulty of GAB model for practical purpose is to extend its use up to a_w of 1 (Rahman, Perera, & Thebaud, 1998).

Both the GAB model and most of its modifications as well as the Peleg's four parameter model have the same weakness in that they predict a finite adsorption at water activity 1 (Lewicki, 1998).

The Lewicki equation was developed to make it applicable to the high range of a_w. It was assumed that $X = 0$ when $a_w = 0$, and $X = \infty$ when $a_w = 1$ (Lewicki, 1998). This equation fits well the MSI data at high humidities and predicts that the water content X tends to ∞ when $a_w \to 1$, condition that have been found in many fresh tissue foods.

The three models were used to fit the experimental MSI data and their constants were calculated minimizing the root mean square percent error (% RMS) defined as

$$\%\text{RMS} = \frac{\sqrt{\sum\left(\frac{X_{\exp}-X_{\text{model}}}{X_{\exp}}\right)^2}}{n-1} \times 100\% \tag{5}$$

Table 4 shows the fitted parameter of the models on lean beef at each temperature and for all temperatures. Overall the GAB model fitted the experimental data best, followed by the Peleg and Lewicki Models. However, the Lewicki model shows a better fit in the high a_w range, although it is the poorest on the intermediate range (0.60–0.90). The only disadvantage of the GAB

Table 3
Effect of the change of salts on the MSI of lean beef and ANOVA table

Salt	Temperature	With change of salt		Without change of salt	
		X	σ	X	σ
KCl	25	0.2736	0.00016	0.2903	0.00013
KBr	25	0.2170	0.00149	0.2574	0.00000
$Mg(NO_3)_2$	25	0.0949	0.00009	0.1051	0.00005
KNO_3	40	0.3825	0.00167	0.3558	0.00216

ANOVA

	SS	Dof	MSS	Ft	F-critic	Result
Treatments	0.0006163	1	0.0006163	0.78025877	4.38	No significant difference
Salts	0.2271	3	0.0756917	95.83478	3.13	Significant difference
Error	0.0150	19	0.0007898			
Total	0.2427	23				

Table 4
Parameter of the GAB, Peleg and Lewicki equations for lean meat

	$T = 5$ °C	$T = 15$ °C	$T = 25$ °C	$T = 40$ °C	Global
GAB					
X_{mg}	0.0522	0.0497	0.0540	0.0527	0.0565
C_g	4.2694E+13	3.2332E+13	3.8212	3.0732	4.2396
K	0.9721	0.9688	0.9750	0.9878	0.9692
RMS%	0.2956	1.3919	0.8208	1.6601	1.8778
Peleg					
a	1.0848	1.0410	1.9641	2.6280	0.9650
b	13.3198	9.6905	31.0390	28.9194	13.3039
c	0.2011	0.1153	0.3137	0.3456	0.2427
d	0.8855	0.4031	2.0065	1.9638	1.3886
RMS%	2.0627	3.5093	6.2678	0.9974	2.3377
Lewicki					
F	0.1209	0.1398	0.0787	0.0428	0.0488
G	0.5816	0.5474	0.7386	1.0235	0.8761
H	0.5273	2.7044	1.4549	−10.5667	−34.7794
RMS%	2.0165	5.3498	5.3018	0.0249	4.1227

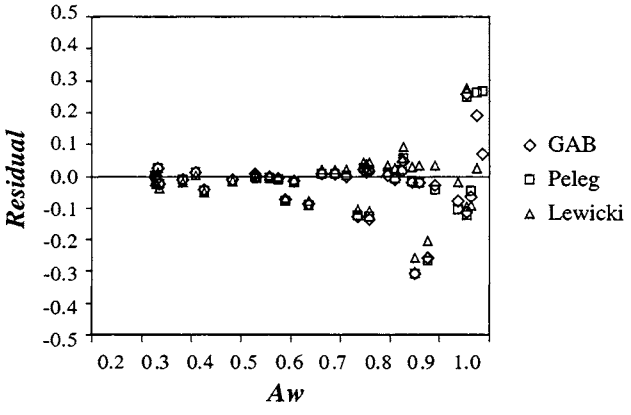


Fig. 8. Distribution of residual errors for GAB, Peleg and Lewicki models for lean beef isotherm, all temperatures, showing that no residual trend is evident.

model, as well as the Peleg, is that it predict a finite water content X ($X = 1.82$) at water activity 1. Fig. 7 shows the fitting of the three equations and Fig. 8 the distribution of residual errors.

The mean repeatability and the standard deviation of the lean beef moisture content X are 0.14 and 0.30 respectively, which are smaller than those of Wolf et al. (1985) under the COST 90 project (mean repeatability of 0.28 and standard deviation of 0.52, even though the latter measured the isotherm at only one temperature). The temperature-independent GAB equation produced a RMS error of only 1.88%. Thus, it can be concluded that the lean beef data can be accurately fitted with a model independent of temperature.

3.4. Experimental results of the external fat moisture isotherm

The desorption isotherm of beef fat at 5, 15 and 25 °C are shown in Table 5. The data was statistically analyzed and, as in the case of the lean meat, no significant difference was found with changes of composition at a

Table 5
Experimental moisture sorption isotherms of external fat at 5, 15 and 25 °C (means and standard deviations)

Salt	$T = 5$ °C		$T = 15$ °C		$T = 25$ °C	
	a_w	X	a_w	X	a_w	X
K_2SO_4	0.9842	–	0.9789	–	0.9730	0.0260
KNO_3	0.9627	–	0.9541	0.0218 ± 0.0061	0.9358	0.0170 ± 0.0183
KCl	0.8767	0.0502 ± 0.0340	0.8592	0.0182 ± 0.0129	0.8434	0.0089 ± 0.0096
KBr	0.8509	0.0223 ± 0.0042	0.8262	0.0075 ± 0.0015	0.8089	0.0068 ± 0.0070
NaCl	0.7565	0.0139 ± 0.0025	0.7561	0.0083 ± 0.0025	0.7529	0.0093 ± 0.0078
KI	0.7330	0.0192 ± 0.0136	0.7098	0.0064 ± 0.0033	0.6886	0.0069 ± 0.0075
NaBr	0.6351	0.0123 ± 0.0056	0.6068	0.0063 ± 0.0043	0.5757	0.0056 ± 0.0058
$Mg(NO_3)_2$	0.5886	0.0094 ± 0.0084	0.5587	0.0041 ± 0.0026	0.5289	0.0062 ± 0.0075
NaI	0.4242	0.0084 ± 0.0063	0.4088	0.0064 ± 0.0041	0.3817	0.0100 ± 0.0116
$MgCl_2$	0.3360	0.0060 ± 0.0044	0.3330	0.0039 ± 0.0003	0.3278	0.0058 ± 0.0096

Table 6
Average composition of fat

	Units	Average	σ
Fat	g fat/ g wet fat sample	0.898	0.041
Ash	g ash/ g wet fat sample	0.0022	0.00084
Water	g water/ g dry material	0.118	0.027

confidence level of 95%. Table 6 shows the average composition of the fat. The factor that mostly affects the variance is also the change of salt. Temperature was found to have a significant effect. As it seen in Fig. 9, there is no clear difference between 15 and 25 °C but the MSI at 5 °C is clearly higher than the others.

Because fat is less susceptible to bacteria degradation than meat, it was possible to obtain the moisture content of the samples in equilibrium with KNO_3 at 15 and 25 °C. The two high humidity points, with K_2SO_4 and KNO_3, were accelerated with the salt change technique. However, the three replicates of the fat in KNO_3 using the acceleration process were statistically lower than the three replicates with the standard procedure. Thus, this procedure cannot be considered reliable in the case of fat.

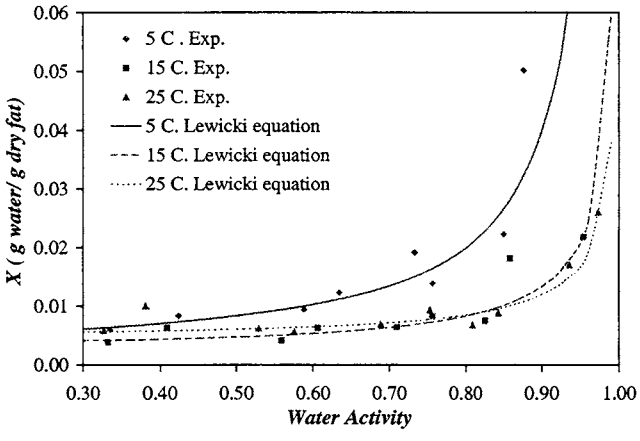


Fig. 9. Moisture sorption isotherm of fat at 5, 15 and 25 °C.

The different behavior of fat under accelerated desorption may be attributed to the hydrophobic character and resulting low moisture content of the fat. Under accelerated desorption at the beginning of the process, the external part of the sample loses most or all its moisture due to the low surrounding humidity. After changing to the second (high humidity) salt, moisture migrates from the centre to the surface but cannot be reabsorbed by the desiccated layers of fat because these has become denatured and lost hydrophilic sites to which water molecules can be attached.

The three previous models were also used to fit the fat data. The Lewicki model fitted the data better than the others. Table 7 shows the fitted constants of the GAB, Lewicki and Peleg equations at 5, 15 and 25 °C. The Lewicki equation was expressed as a function of temperature to make it more general. This modification was made by correlating the parameters of the equation as an exponential function of the temperature, similar to the modification of the GAB model as a function of temperature (Weisser, 1985). The Lewicki constants as function of temperature are

$$F = 6.3828 \times 10^2 \exp\left(\frac{-3.3153 \times 10^3}{T}\right)$$
$$G = 7.7994 \times 10^{12} \exp\left(\frac{7.1235 \times 10^3}{T}\right) \quad (6)$$
$$H = 9.0851 \times 10^9 \exp\left(\frac{-1.5453 \times 10^7}{T}\right)$$

The RMS error using these equations is 6.616 which mean that the fitting at each independent temperature is better.

Table 7
Parameter of the GAB, Peleg and Lewicki equations for fat

Temperature	GAB		Peleg		Lewicki	
5 °C		4.1402E−03	*a*	86.4670	*f*	4.2833E−03
		8.0957E+13	*b*	59.8530	*g*	0.9868
		0.9944	*c*	2.0115E−02	*h*	−5.0762
			d	1.1144		
	RMS%	4.102	RMS%	1.5937	RMS%	3.984
15 °C		2.4108E−03	*a*	−9.6562E+02	*f*	3.6611E−03
		8.7689E+13	*b*	1.0324E+03	*g*	0.6129
		0.9205	*c*	1.0661E−02	*h*	−1.7028
			d	1.0692		
	RMS%	6.251	RMS%	11.250	RMS%	5.243
25 °C		3.2535E−03	*a*	−9.3502E+02	*f*	4.9701E−03
		1.0659E+14	*b*	1.7761E+03	*g*	0.4544
		0.8167	*c*	9.5742E−03	*h*	−2.5007
			d	0.5364		
	RMS%	8.320	RMS%	10.978	RMS%	3.708

Although the GAB model does not fit the data as well as the Lewicki model, it is interested to note that the C_g value of the GAB equation is very high. Thus, the GAB equation can be reduced to a two parameter equations:

$$X = \frac{X_{mg}}{(1 - K \cdot a_w)} \tag{7}$$

4. Conclusion

The moisture sorption isotherm of fresh lean beef and external beef fat has been successfully measured by using improved techniques to avoid spoilage at high relative humidities. The experimental procedure used, based on COST 90 with some geometrical modifications (sorption container geometry, position of the sample and geometry of the sample) was validated with the standard reference material MCC. The MSI of the beef at the two highest humidities was obtained by accelerating the process with a change of salt. This procedure was reliable for beef samples but not for the fat samples, possible caused by strong hysteresis given by the hydrophobic characteristics of the fat.

In both cases, lean beef and fat, no significant differences of the MSI with changes in composition was found. The MSI of the former was measured at 5, 15, 25 and 40 °C. The experimental data was accurately fitted using a model independent of temperature. The GAB equation fitted the data best, although it predict a constant value of moisture content at $a_w = 1$. The Lewicki model does not fit well the data in the range (0.60–0.90) but gives the best fit at high humidity (0.90–1.00) and predict that water content approaches infinity when $a_w \rightarrow 1$.

The MSI of fresh beef surface fat was measure at 5, 15 and 25 °C. The experimental water content of the MSI at 5 °C is higher than that at 15 and 25 °C. However, no significant difference in the MSI was found between 15 and 25 °C. The experimental data was fitted using the GAB, Lewicki and Peleg models. The best fit was obtained with the Lewicki equation and the parameters were expressed as a function of temperature to make the equation more general, at a slight cost in accuracy.

Acknowledgements

The authors wish to acknowledge the support of the Australian Government which is supporting this work under an Australian Research Council Large Grant.

References

Baucour, P., & Daudin, J. D. (2000). Development of a new method for fast measurement of water sorption isotherms in the high humidity range validation gelatine gel. *Journal of Food Engineering, 44*, 97–107.

Berg, C., & Bruin, S. (1981). Water activity and its estimation in food systems: theoretical aspects. In L. B. Rockland, & G. F. Steward (Eds.), *Water activity: influences on food quality*. New York: Academic Press.

Bird, R. B., Steward, W. E., & Lightfood, E. N. (1960). *Transport phenomena*. New York, NY: Wiley.

Davey, L. M. (1998). Measurement and prediction of product heat load and weight loss during beef chilling. Ph.D. Thesis. Sydney, Australia: University of New South Wales.

Delgado, A. E., & Sun, D. W. (2002a). Desorption isotherms for cooked and cured beef and pork. *Journal of Food Engineering, 51*, 163–170.

Delgado, A. E., & Sun, D. W. (2002b). Desorption isotherms and glass transition temperature for chicken meat. *Journal of Food Engineering, 55*, 1–8.

Greenfield, H., Arcot, J., Emerson, E., Hutchison, G. L., & Wills, R. B. H. (1998). *Laboratory Instruction for Food Composition Studies*. Department of Food Science and Technology, University of New South Wales.

Greenspan, L. (1976). Humidity fixed points of binary saturated aqueous solutions. *Journal of Research of the National Bureau of Standards, 81A*(1), 89–96.

Herbert, L. S., Lovett, D. A., & Radford, R. D. (1978). Evaporative weight loss during meat chilling. *Food Technology in Australia* (April), 145–148.

Iglesias, H. A., & Chirife, J. (1982). *Handbook of food isotherms*. New York: Academic Press.

Labuza, T. P., & Hyman, C. R. (1998). Moisture migration and control in multidomain foods. *Trends in Food Science & Technology, 9*, 47–55.

Lang, K. W., McCune, T. D., & Steinberg, M. P. (1981). A proximity equilibration cell for rapid determination of sorption isotherms. *Journal of Food Science, 46*, 936–938.

Lewicki, P. P. (1998). A 3 parameter equation for food moisture sorption isotherms. *Journal of Food Process Engineering, 21*, 127–144.

Palnitkar, M. P., & Heldman, D. R. (1971). Equilibrium moisture characteristics of freeze-dried beef components. *Journal of Food Science, 36*, 1015–1018.

Peleg, M. (1992). Assessment of a semi-empirical 4 parameter general model for sigmoid moisture sorption isotherms. *Journal of Food Process Engineering, 16*, 21–37.

Pham, Q. T., & Karuri, N. W. (1999). A computationally efficient technique for calculating simultaneous heat and mass transfer during food chilling. In *20th international congress of refrigeration*, IIR/IIF, Sydney, Australia.

Rahman, M. S., Perera, C. O., & Thebaud, C. (1998). Desorption isotherm and heat pump drying kinetics of peas. *Food Research International, 30*, 485–491.

Ross, T. (1999). *Predictive microbiology model in the meat industry*. Meat and Livestock Australia, Sydney.

Saravacos, G. D., & Stinchfield, R. M. (1965). Effect of temperature and pressure on the sorption of water vapor by freeze-dried food materials. *Journal of Food Science, 30*, 779–786.

Sing, R. R. B., Rao, K. H., Anjaneyulu, A. S. R., & Patil, G. R. (2001). Moisture sorption properties of smoked chicken sausages from spent hen meat. *Food Research International, 34*, 143–148.

Spiess, W. E. L., & Wolf, W. (1987). Critical evaluation of methods to determine moisture sorption isotherms. In *Water activity: Theory and applications in food* (pp. 215–233). New York: Academic Press.

Taylor, A. A. (1961). Determination of moisture equilibria in dehydrated foods. *Food Technology, 15*, 536–540.

Timmerman, E. O., & Chirife, J. (1991). The physical state of water sorberd at high activities in starch in terms of the GAB sorption equation. *Journal of Food Engineering, 13*, 171–179.

Timmerman, E. O, Chirife, J., & Iglesias, H. A. (2001). Water sorption of foods and foodstuffs: BET or GAB parameter? *Journal of Food Engineering, 48*, 19–31.

Weisser, H. (1985). Influence of temperature on sorption equilibria. In D. Simatos, & J. L. Multon (Eds.), *Properties of water in food: in relation to quality and stability*. Dordrecht: Martinus Nijhoff Publishers.

Wolf, W., Spiess, W. E. L., & Jung, G. (1985). Standardization of isotherm measurements (Cost-Project 90 and 90 bis). In D. Simatos, & J. L. Multon (Eds.), *Properties of water in food: in relation to quality and stability*. Dordrecht: Martinus Nijhoff Publishers.

Yeow, P. C. (2001). Measurement of food moisture isotherms. Bachelor Thesis Chemical Engineering. Sydney, Australia: University of New South Wales.

CFD MODELING OF HEAT AND MOISTURE TRANSFER ON A TWO-DIMENSIONAL MODEL OF A BEEF LEG

FRANCISCO JAVIER TRUJILLO*, Q. TUAN PHAM**
School of Chemical Engineering and Industrial Chemistry
University of New South Wales, Sydney 2052, Australia
*francisco.trujilllo@student.unsw.edu.au
**tuan.pham@unsw.edu.au

ABSTRACT

A numerical simulation of simultaneous heat and mass transfer in an ellipse model of a beef leg was carried out using the CFD software FLUENT 6.0. Special techniques were used to solve the heat and mass transport equation for both the air and meat phases, due to some limitations in the software. The meat was treated as a sub-region of the fluid with zero momentum transport, and the mass transport in the air and the meat respectively were modelled by different field variables, which are however linked at the interface. In the air, turbulent flow was modeled with the RNG $\kappa - \varepsilon$ model and the boundary layer was fully solved using the FLUENT 6.0 enhanced wall treatment. The model predicted local variations in the heat and mass transfer coefficients and temperature and water activity around the ellipse's surface.

INTRODUCTION

During the chilling of beef carcasses after slaughter, cooling and evaporation proceed together and interact with each other to influence surface water activity, microbial growth, weight loss, meat temperature and meat tenderness, all of which are important economically. As the meat cools, heat is conducted through the meat and carried away by the air. The meat surface is warmer and more humid that the air, resulting in surface evaporation. Water from inside diffuses towards the surface to make up for evaporation. The balance between evaporation and diffusion governs the water activity near the surface, which together with temperature determines the potential for microbial growth.

Both processes, heat and mass transfer, are affected by the flow characteristics and the development of the momentum, heat and mass boundary layers. They are function of the air properties, geometry of the product, chiller room, and the flow characteristics (temperature, humidity, velocity and turbulence). Therefore, local variations in the heat and mass transfer coefficients are expected along the surface (Verboven *et al.*, 1997) producing local differences in temperature and water activity.

Traditional computer models (Davey and Pham, 1997, 2000; Mallikarjunan and Mittal, 1994) have focused on solving the conduction equation in the meat, but the average heat (htc) and mass transfer (mtc) coefficients are calculated with empirical equations. Nguyen and Pham (1999) used CFD to simulate the heat transfer process in a beef carcass chilling, taking in account the conduction inside the meat and the convection in the air face, but they did not take in account the evaporation, radiation and mass transfer. Hu and Sun (2000) modeled the heat and mass transfer on the air side during the cooling of cylindrical shaped cooked meat. They used the CFD software CFX to calculate the average htc but they did not predict local htc variations. Hu and Sun (2001) modelled the heat and moisture transfer. The former was modeled in both the solid and air phases using CFD, but the mass transfer was not treated rigorously via CFD modeling. For example, the Lewis relationship was used to calculate mass transfer from heat transfer, and the surface water activity of the meat was assumed to be equal to the relative humidity of the air (on a scale of 0 to 1), which was not necessary true.

The modeling of mass transfer during cooling is still very approximate. Most models calculate the mtc using the Lewis relationship and assume a constant value for water activity (Davey and Pham, 1997, 2000; Hu and Sun, 2000). A source of difficulty is that heat and mass transfer happen on vastly different scales due to the big difference in heat and mass diffusivity. While the whole product is cooled, only the surface layer (a few mm) loses water. This makes it difficult to model the two processes accurately using and homogeneous grid configuration. To solve this

problem, Pham and Karuri (1999) used a separate discretization grid for temperature and moisture calculations. However, they solved the equations only for the solid phase and not for the air phase, relying instead on an empirical htc and the Lewis relationship to calculate the average mtc.

No previous work has attempted to solve simultaneously the equations for mass diffusion, fluid flow and heat transfer in both phases together. This is the objective of the present paper.

1. PROBLEM STATEMENT

A beef leg undergoing chilling is modeled as an ellipse (Davey and Pham, 2000), with minor and mayor diameters of 0.22m and 0.29m respectively, placed inside a wind tunnel 1.5m wide by 2.3m long. Air enters the tunnel at 277.95°K, 98% relative humidity, atmospheric pressure, 0.54m/s normal to the inlet plane, with a turbulence intensity of 10%. The product is initially at 315.15°K with a moisture content of 75% wet basis. The properties of air were assumed constant except the density that was expressed as function of temperature and Pressure. For the meat, we assume a density of 1111 kgm^{-3}, specific heat of 1006.87 $Jkg^{-1}K^{-1}$, thermal conductivity of 0.0263 $Wm^{-1}K^{-1}$ and the water diffusivity is given by the equation of Herbert *et al.* (1978):

$$D_m = 1.10\times10^{-6} e^{\left(\frac{2300}{T}\right)} \tag{1}$$

At the meat surface, the water activity (relative humidity on a scale of 0 to 1) is given by a Lewicki-type isotherm equation based on the authors' data (Trujillo *et al,* 2003). The moisture content of the air next to the surface is determined from the surface temperature and water activity.

2. MATHEMATICAL MODEL

2.1 Transport equations in the air

In the air, a set of six transport equations are solved. Each has the general form shown in eq.1:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla\cdot(\rho\phi\vec{v}) = \nabla\cdot(\Gamma\nabla\phi) + S \tag{2}$$

where the terms on the left represent accumulation rate and convection, while those on the right represent diffusion and creation/destruction of the field variable ϕ. ϕ can be 1 (in the continuity equation), velocity (in the momentum equation), turbulent kinetic energy, turbulent dissipation rate (in the k and ε transport equations), temperature (in the thermal energy equation), or moisture content (in the moisture transport equation).

The airflow is calculated from the equations of continuity, momentum, turbulent energy (k) and turbulent dissipation (ε). The turbulent model used is the RNG (Re-Normalization Group) k-ε model, which uses a rigorous statistical technique to improve the accuracy of the standard k-ε model. It includes an analytically - derived formula for effective viscosity that accounts for low-Reynolds number effects (Fluent Inc., 2001a). In addition, the transport equations for thermal energy and water vapor are also solved. All transport properties in the air phase (μ, ?, D) are effective values which are the sums of intrinsic and turbulent components.

2.2 Transport equations in the meat

In the meat, only the transport equations for thermal energy and moisture need to be solved. The convection and source terms are dropped from these equations.

2.3 Boundary conditions

Conditions at the tunnel's inlet are as given under "Problem specifications". At the tunnel outlet, zero normal gradients are assumed for all variables: v, T, Y, ε, κ. At the walls of the tunnel, zero velocity, zero heat flux and zero water flux are assumed.

At the meat surface, thermal, species equilibrium and conservation of heat and mass apply. With regards to conservation of heat and mass, special techniques have to be applied to balance the heat and mass fluxes coming out

of the solid with those entering the air phase. After the transport equations in the air have been solved, the water flux that enters the air was calculated cell by cell using the concentration profile in the air control volume next to the solid surface, using the equation:

$$\dot{m} = \rho D \frac{(Y_f - Y_c)}{ds} \tag{3}$$

The heat flux entering the air was calculated as the sum of convection, evaporative and radiative components:

$$q = q_{conv} + q_{evap} + q_{rad} \tag{4}$$

$$q_{evap} = \dot{m}\Delta H_{vap} \tag{5}$$

$$q_{rad} = \sigma \varepsilon_r \left(T_{m-s}^4 - T_a^4\right) \tag{6}$$

and q_{conv} was calculated using the temperature profile of the air control volume next to the surface:

$$q_{conv} = k \frac{(T_f - T_c)}{ds} \tag{7}$$

The following equilibrium condition between the air in contact with meat and the meat surface must also hold:
Thermal equilibrium:

$$T_{a-s} = T_{m-s} \tag{8}$$

Chemical potential equilibrium:

$$a_{w-m-s} = a_{w-a-s} \tag{9}$$

Where a_{w-m} is function of the water content in the meat surface (a function of meat moisture content as given in Trujillo *et al.*, 2003).

2.4 Initial conditions:

At zero time, the meat temperature and water composition are constant as per "Problem Statement". It was also assumed that the airflow was fully developed. Thus, a steady state solution for the air phase was done as a preliminary step, with the air temperature near the meat surface kept at 315.15°K and the humidity corresponding to equilibrium with the meat.

2.5 Boundary layer treatment:

Because the mass and heat fluxes between meat and air are calculated from the temperature and concentration gradient next to that surface, these profiles must be accurately known. Therefore we used FLUENT 6.0's *enhanced wall treatment* (Fluent Inc., 2001a) which is a near-wall modeling method that combines a two-layer model (the viscosity affected near wall region is completely resolved all the way to the viscous sub-layer) with enhanced wall functions. The near-wall mesh must be fine enough to resolve down to the laminar sub-layer ($y^+ \approx 1$).

3. DETAILS OF NUMERICAL SOLUTION

The equations were solved using FLUENT 6.0. FLUENT's *segregated method*, where the governing equations are solved sequentially. In that method, each discretized transport equation is linearized implicitly with respect to the equation's dependent variable. Because the equations are non-linear and coupled, iterations must be performed before a converged solution is obtained. A point implicit (Gauss-Seidel) linear equation solver is used by FLUENT in conjunction with an algebraic multigrid (AMG) method. The pressure – velocity coupling method used was PISO which is recommended for unsteady problems. Time steps of 1 second were used at the beginning, gradually increasing up to 10 minutes at the end of the simulation. Up to 40 iterations were done for each time step.

In the meat, the energy and mass transfer processes could not be solved using FLUENT 6.0's existing energy and mass transfer equations because:

- FLUENT6.0 cannot solve the mass transfer equation in a solid (the meat). Thus the meat had to be defined as a fluid phase (this is purely a formal definition for FLUENT – since the equations of continuity, momentum, turbulent kinetic energy and turbulent dissipation rate are dropped, the physical behavior is in effect that of a solid).
- Once the meat was defined as a fluid phase to satisfy the previous requirement, FLUENT 6.0 then requires that the physical properties (c_p, μ, D, k, ρ) be the same in all parts of the solution domain (i.e. both air and meat phases are modeled as the same substance). Since these properties were in fact different in the two phases, new transport equations involving new field variables must be defined and solved in the meat.
- FLUENT6.0 will only allow the boundary conditions of the mass transport equations to be zero flux or a fixed concentration, while we want to calculate the flux at the meat-air boundary according to equation (3).

To overcome the above problems, FLUENT allows the user to define new field variables called UDS (User Defined Scalars). The moisture and temperature inside the meat are considered as new field variables or UDS with their own associated transport properties (diffusivity and density). Defining a UDS simply involves specifying whether there are convective, diffusive and transient terms in the transport equations, and specifying expressions for the transport properties mentioned above.

The FLUENT default menu allows only fixed boundary conditions. Because the boundary conditions at the air- meat interface change with time and are dependent on the values of the field variables, they were modeled using UDF's (User Defined Functions), which are functions programmed by the user in C++ that can be automatically linked with the FLUENT Solver (Fluent Inc., 2001b). With a UDF, we can take the present values of the local field variables (T, Y, etc.) and use them to calculate boundary conditions applying at that particular instant. The UDFs are incorporated in the set of equations solved by the segregated solver, and the values predicted by the UDF were updated in each iteration. The UDF's programmed boundary conditions were:

1. DEFINE_PROFILE(mass_flux_meat, tm, j) : Calculate the water flux leaving the meat in the interface caused by the mass convection given the development of the mass boundary layer. The calculation is made with the equation (3). It is used as a boundary condition of the UDS-2 that defines the mass transport equation into the meat.
2. DEFINE_PROFILE(heat_flux_meat, tm, j) : Calculate the heat flux leaving the meat in the interface caused by convection, radiation and evaporation according to the equation (4). It is used as a boundary condition of the UDS-1 that defines the energy equation into the meat.
3. DEFINE_PROFILE(temperature_air_interf, t, n) : makes the temperature of the air on the air meat interface equal to the temperature of the meat surface (Equation 8). It is used as a boundary condition of the air phase energy equation.
4. DEFINE_PROFILE(water_air_interf, t, n): Calculate the air water mass concentration on the interface with the following procedure:

   - Read the meat surface temperature (Equation 8).
   - Read the water mass concentration in the meat interface and calculate the water content.
   - Calculate the water activity with the Lewicki equation (Trujillo *et al.* (In Press)) using the Newton-Raphson method.
   - With the water activity and the meat surface temperature, calculate the air water mass concentration using a vapor pressure equation.

   This UDF is used as a boundary condition of the mass equation of the air phase.

   On each iteration the solver segregated method does the following:
   - Solve the linearized discretized momentum equation in the air phase.
   - Solve mass conservation in the air phase and update velocities.
   - Solve energy equation in the air phase– this involves calls to UDF 3.

- Solve mass equation in the air phase – this involves calls to UDF 4.
- Solve turbulent kinetic energy in the air phase.
- Solve Eddy dissipation in the air phase.
- Solve energy equation in the meat phase using UDS1. This involves calls to UDF 2.
- Solve mass equation in the meat phase using UDS2. This involves calls to UDF 1.
- Update all properties
- Check for convergence.

That procedure is done each time step until convergence is obtained or 40 iterations have been done.

4. RESULTS

Figure 1 shows the average surface and center temperature as function of time. The surface temperature is in good agreement with the experimental data suggesting that the model is accurate. The measured center temperature drops faster than the predicted values but this may be caused by difficulties in experimentally locating the center. Figure 2 shows the changes of the surface temperature with the position over the sphere (given by the angle) at 1 and 5 hours. Temperature could vary by up to 5C around the ellipse.

Figure 3 shows the change of the average surface water activity with time showing that it drops faster during the first 4 hours and then increases caused by the internal water diffusion that rewets again the meat surface. This is in agreement with the experimental observation of Herbert *et al.* (1978). Figure 4 shows the local variations of the water activity with position and Figure 5 shows the water concentration profile deep inside the meat at 180° (the impact or stagnation point) It shows that the mass transfer inside the meat is noticeable only in a 25 mm surface layer.

Figure 6 shows the relationship between the local and global heat and mass transfer coefficients. It is interesting to see that this relationship is almost constant with time and is almost the same between heat and mass transfer. The lowest value is at about 50° and the maximum between 160° and 180° (0° being the point furthest downstream in the ellipse). This result is in agreement with the highest and lowest temperature and water activity zones in figures 2 and 4. Those local variations can be explained by looking at Figure 7 which shows the air velocity profile around the ellipse. It is seen that on the upstream side of the ellipse, between 180° and 90°, the htc and mtc are higher, while after the de-attachment of the boundary layer, the htc and mtc are lower, the lowest value being at about 60° where there is a recirculation zone.

CONCLUSIONS

For the first time a couple solution of heat and mass transfer during meat chilling has been presented, with the transport equations on both the air and meat phases being solved simultaneously. Even though FLUENT 6.0 is a very powerful CFD software, it has difficulties dealing with simultaneous heat and mass transfer in two different phases, and special techniques had to be used to solve the mass transport equation in the meat. The solid has to be treated as a sub-region of the fluid phase where convection does not occur. The temperature and moisture fields in the solid are represented by new user-defined field variables. At the solid-air interface, the temperature and moisture fields in the fluid and the user-defined fields in the solid are linked by special user-defined equations, representing interfacial equilibria and transfer. The model gave plausible predictions of local variations

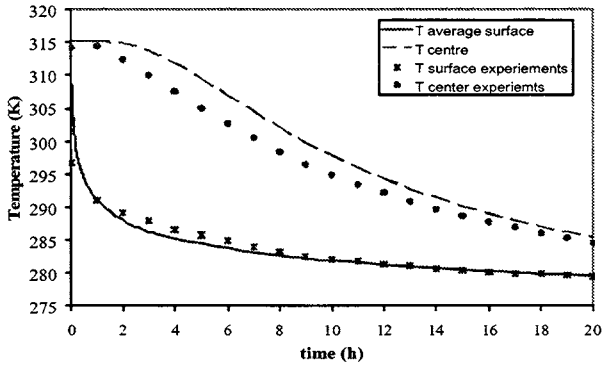


Figure 1. Average surface and center temperature of the meat ellipse.

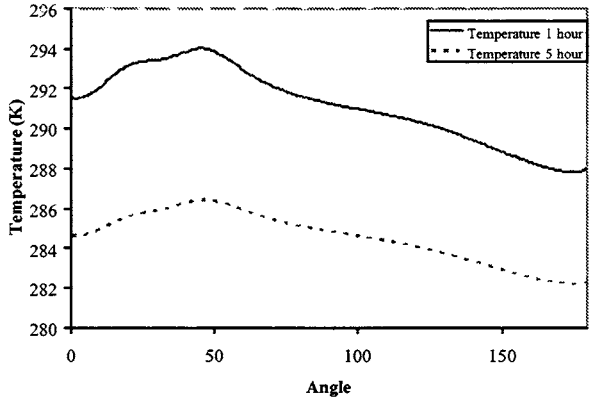


Figure 2. Surface temperature profile as a function of the angle at 1 and 5 hours.

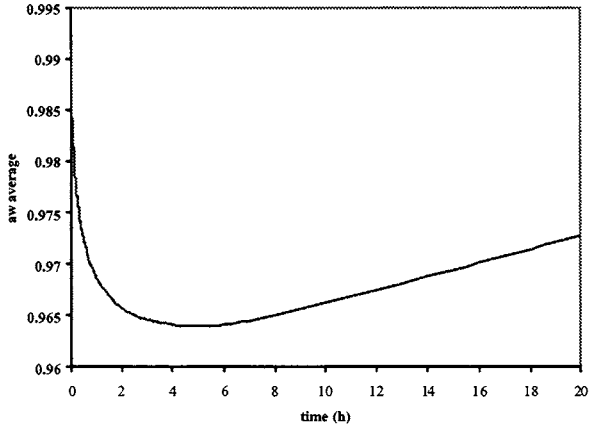


Figure 3. Average surface water activity vs. time.

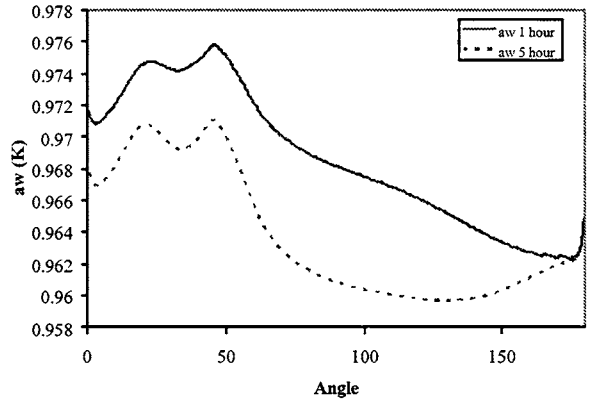


Figure 4. Surface water activity as a function of the angle at 1 and 5 hours.

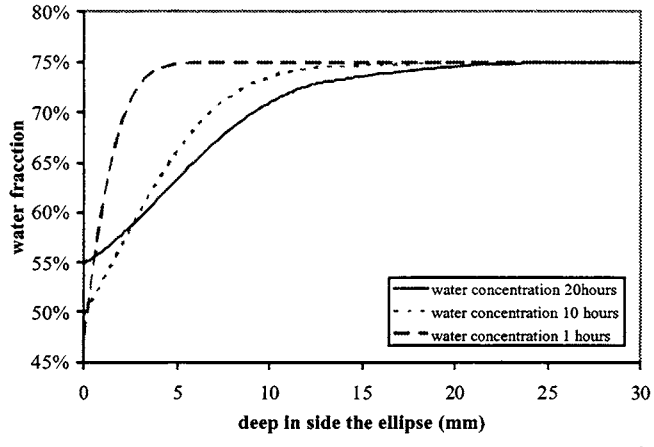


Figure 5. Water concentration profile deep inside the meat at 180° at 1, 10 and 20 hours.

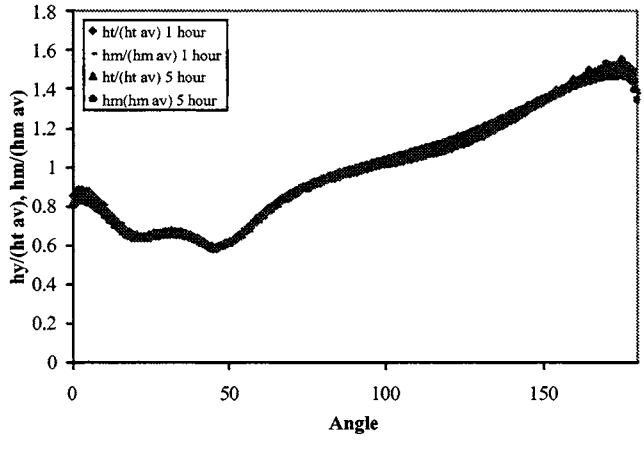


Figure 6. Relation between the local and global heat transfer and mass transfer coefficient as a function of the angle at 1 and 5 hours.

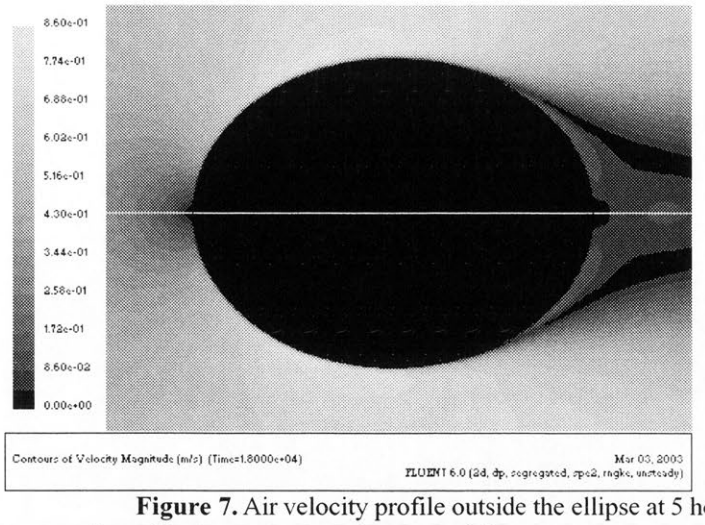


Figure 7. Air velocity profile outside the ellipse at 5 hours.

in heat and mass transfer and in temperature and water activity.

It is also important to take in account the time difficulties involved on CFD simulations. To model 20 hours of chilling took about 6 days in a Pentium 1.5 GHz Computer, making it unpractical for normal industrial calculations.

NOMENCLATURE

a_w :	Water activity.
C_p :	Air heat capacity (J / Kg / K).
ds :	Distance node face and node cell (m).
D :	Water Diffusivity (m^2 /s).
$\dot{m}$:	Water mass flux (Kg / m^2 /s).
q :	Heat flux (W / m^2).
S:	Source term.
t :	Time (s)
T :	Temperature (K)
$\vec{v}$:	Velocity vector.
Y :	Water weight composition.
y^+ :	Non-dimensional number.

Greek symbols:

ΔH_{vap} :	Water vaporization heat (J /Kg).
ε :	Turbulent energy dissipation rate (m^2 /s^3)
ε_r :	Radiation emissivity.
ϕ :	Field transport variable.
κ :	Thermal conductivity (W/m/s)
μ :	Air viscosity (kg/m/s)
ρ :	Density (Kg/m^3)
σ :	Stefan-Boltzmann constant
Γ :	Diffusivity (m^2 /s)

Subscripts:

a:	Air phase.
a-s:	Air phase at the surface.
c:	cell node.
conv:	Convective.
evap:	Evaporation.
f:	face node.
m:	Meat.
m-s:	Meat at the surface.
rad:	Radiation.

REFERENCES

1. Davey, L. M., Pham, Q. T., 1997, Predicting the dynamic product heat load and weight loss during beef chilling using a multi-region finite difference approach, Int. J. Refrig., Vol. 20, no. 7: p. 470-482.
2. Davey, L. M., Pham, Q. T., 2000, A multi-layered two dimensional finite element model to calculate dynamic product heat load and weight during beef chilling, Int. J. Refrig., vol. 23, p. 444-456.
3. Fluent Inc., 2001a, FLUENT 6.0 User's guide, vols. 1-5, Fluent Inc., Lebanon NH USA.
4. Fluent Inc., 2001b, FLUENT 6.0 UDF manual, Fluent Inc., Lebanon NH USA.
5. Herbert, L.S., Lovett, D. A., Radford, R. D., 1978, Evaporative weight loss during meat chilling, Food Technol. Australia, vol. 30, p. 145-148.

6. Hu, Z., Sun, D. W., 2000, Computational fluid dynamics simulation of heat and moisture transfer for predicting cooling rate and weight loss of cocked meats, J. Food Eng., vol. 46, no.3: p. 189-197.
7. Hu, Z., Sun, D. W., 2001, Predicting local surface heat transfer coefficients by different turbulent $\kappa - \varepsilon$ models to simulate heat and moisture transfer during air-blast chilling, Int. J. Refrig., vol. 24, no.7: p. 702 - 717.
8. Mallikarjunan, P., Mittal, G. S., 1994, Heat and mass transfer during beef carcass chilling – modeling and simulation, J. of food Eng., vol. 23, p. 277-292.
9. Nguyen, A., Pham, Q. T., A computational Fluid dynamic model of beef chilling, 20th International Congress of Refrigeration, IIR/IIF, Sydney, CDROM.
10. Pham, Q. T., Karuri, N.W., 1999, A computational efficient technique for calculating simultaneously heat and mass transfer during food chilling, 20th International Congress of Refrigeration, IIR/IIF, Sydney, CDROM.
11. Trujillo, F. J., Yeow, P. C., Pham, Q. T., 2003, Moisture sorption isotherm of fresh lean beef and external beef fat, Journal of Food Engineering, (in press).
12. Verboven, P., Scheerlinck, N., Baerdemaeker, J. D., Nicalai, B. M., 1997, The local surface transfer coefficient in thermal food process calculations: A CFD approach., J. Food Eng., vol. 33, p. 15-35.

Third International Conference on CFD in the Minerals and Process Industries
CSIRO, Melbourne, Australia
10-12 December 2003

CFD MODELLING OF THE HEAT AND MASS TRANSFER PROCESS DURING THE EVAPORATION OF WATER FROM A CIRCULAR CYLINDER

Francisco Javier TRUJILLO[1], Simon J. LOVATT[2], Mark B. HARRIS[2], Jim WILLIX[2], and Q. Tuan PHAM[1]

[1] *School of Chemical engineering and Industrial Chemistry,* University of New South Wales, Sydney 2052, Australia

[2] AgResearch MIRINZ Centre Private Bag 3123, Hamilton, New Zealand

ABSTRACT

The purpose of this work is to simulate the heat and mass transfer during the evaporation of water from a cylinder at moderate Reynolds numbers. The process was modelled on FLUENT 6.1.18 and the results were compared with experimental data. Different mathematical approaches were used to assess which model fit the data better. Four turbulent models were used and compared: the laminar, standard κ-ε, RNG κ-ε and the SST κ-ω. Two wall treatment approaches were followed: the standard wall function and the enhanced wall treatment. The model that best fits the experimental data was the RNG κ-ε model using the enhanced wall treatment and taking in account the effect of radiation. This model was able to predict the changes of temperature around the cylinders as well as the heat (h) and mass (h_m) transfer coefficients. The ratio between h and h_m was calculated with the CFD modelling, showing that this ratio is not constant at low air velocities reaching a minimum of up to 19% of difference, at the point of separation of the boundary layer, comparing with the Chilton-Colburn analogy. It was found that this difference may be caused by the heat of radiation that becomes important at low Reynolds numbers.

NOMENCLATURE

C_P	Heat capacity
C_μ	RNG κ-ε model constant
D	Diffusivity of water in air
ds	distance from node face to node cell
E	Empirical Constant
ε	Radiation emissivity
h	Local convective heat transfer coefficient
h_m	Local mass transfer coefficient
κ	Von Karman constant
L_e	Lewis number
J_w	Local mass flux on the wall
L_{vap}	Vaporization heat of water
P_c	Number function of Sc and Sc_t
RH	Relative Humidity (%)
Sc	Schmidt number
T	Temperature
$\dot{q}_w$	Wall heat flux
Y	Mass weigh composition
Y^*	Non-dimensional water composition on the near wall cell
y^*	Non-dimensional distance to the wall
y^+	Non-dimensional distance to the wall
y_C^*	Mass sub-layer thickness
μ	Dynamic viscosity
ρ	Density
σ	Stefan-Boltzmann constant
$\bar{x}$	Overall or global value of x

Subindices

a	Air
c	Cell
eff	Effective value
mol	Molecular value
t	Turbulent
w	Wall

INTRODUCTION

Heat and mass transfer between air and food products are involved in many food processing operations, such as freezing, drying and chilling. To control and optimize these processes, it is necessary to accurately know the local heat and mass transfer coefficients. They also determine the local temperature and surface water activity, which are two of the most important properties in the control of the bacterial growth.

Industrial meat chilling operations use low air velocities but may have high turbulence intensity. Due to the complex shape of the product, the boundary layer changes rapidly along the surface affecting the local heat and mass transfer coefficients. A circular cylinder under cross flow conditions was used to experimentally determine and model the heat and mass transfer coefficients. The air velocities used were 0.5, 1.5 and 3 m/s and a turbulence intensity of 2%.

EXPERIMENT PROCEDURE

Kondjoyan and Daudin's (1993b) approach of using a steady-state technique to estimate average surface heat and mass transfer coefficients was followed. Two wet plaster samples were placed into a wind tunnel (one for temperature measurement, one for weight measurement) and held under constant air conditions. Plaster samples were used, as it was claimed that the plaster surface remained fully wet for long periods of time. When steady state was reached, the heat extracted from the sample by

evaporation was balanced by that provided from the air stream through convection and radiation to the surface. Under these conditions, the surface of the sample approached the wet-bulb temperature of the air. The average sample heat and mass transfer coefficients were then computed based on the average surface temperature and the rate of evaporation (weight loss) measured from the samples.

The apparatus used comprised a 100mm diameter plaster cylinder, with a length also of 100mm. To reduce aerodynamic edge effects the plaster cylinder was located between two extra wet plasters as it seems in figure 1.

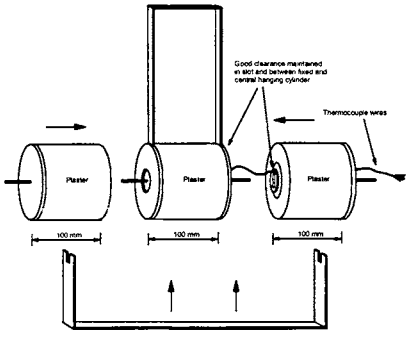


Figure 1: Experimental apparatus to minimise aerodynamic, heat and mass transfer edge effects.

The two plaster assemblies were hung in a controlled environment wind tunnel. One of the samples was weighed during the weight loss experiment, while the other had T type thermocouples inserted near the surface of the plaster to measure the surface temperature (Figure 2). Thus, any influence of the thermocouple wires on sample weight measurement was reduced.

During drying experiments, the air stream temperature, humidity and velocity were maintained at constant pre-determined levels. Plaster surface temperatures, air stream temperature, humidity, and sample weight were recorded versus time. At the start of each run, the plaster surface temperature took a little time to reach a steady state temperature distribution. This value was within 0.2 – 1.8 degrees of the wet bulb temperature, depending on the air velocity over the cylinder. Once this steady temperature was achieved, the rate of evaporation from the cylinder surface remained steady until most of the water in the plaster had evaporated (constant rate dying). After this point the surface temperatures would begin to rise, as mass transfer within the plaster began to limit the drying rate. The recorded data used was that obtained during the steady state stage.

This method has been used before to determine local heat and mass transfer coefficients on elliptical cylinders (Kondjoyan and Daudin's (1993a)) and at the surface of a pork hindquarter (Kondjoyan and Daudin's (1997)).

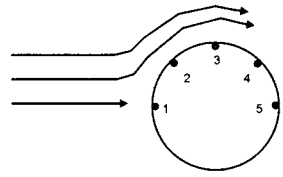


Figure 2: Thermocouples inserted just below the surface of the cylinder.

Calculation of experimental heat and mass transfer coefficient

A heat balance over the surface of the plaster during the constant rate drying period may be written:

$$h_m L_{vap}\left(Y_a - Y_w\right) + \varepsilon\sigma\left(T_a^4 - T_w^4\right) = h\left(T_w - T_a\right) \quad (1)$$

and the rate of mass transfer from the surface, J_w, as

$$J_w = h_m\left(Y_a - Y_w\right) \quad (2)$$

Rearranging these equations and integrating them over the surface of the plaster cylinder enables the overall heat and mass transfer coefficients to be calculated from measurements of the rate of weight loss, the plaster surface temperatures, air stream temperature and humidity:

$$\overline{h}_m = \frac{\overline{J}_w}{\left(Y_a - \overline{Y}_w\right)} \quad (3)$$

$$\overline{h} = \frac{-M_m L_{vap}}{\left(T_a - \overline{T}_w\right)} - \frac{\varepsilon\sigma\left(T_a^4 - \overline{T}_w^4\right)}{\left(T_a - \overline{T}_w\right)} \quad (4)$$

These equations incorporate some small spatial averaging errors. However, these errors were shown to be relatively insignificant by Kondjoyan and Daudin (<3% effect on h and h_m).

The mass transfer coefficients can also be calculated from the heat transfer coefficient if the relationship between heat and mass transfer is known. For laminar flow over flat plates it can be shown that:

$$\frac{h}{h_m} = C_P L_e^{2/3} \quad (5)$$

According to Lewis (1971) Equation 5 may be used to predict heat transfer values from mass transfer measurement. The error involved is much less than the deviation caused by such factors as free stream turbulence and wind tunnel blockage. Kestin and Wood (1971) used equation 5 to calculate the wall temperature and local mass transfer coefficient before and after the detachment of the boundary layer. This relationship (Chilton-Colburn analogy) has also been shown to hold for more complicated boundary layer flows including turbulent flow over a flat plate, laminar and turbulent flow over cylinders.

Introducing the variable, K, representing the ratio of heat to mass transfer:

$$K = \frac{h}{h_m L_{vap}} = \frac{\overline{h}}{\overline{h}_m L_{vap}} \quad (6)$$

the local heat and mass transfer coefficients may be calculated directly from the surface temperature measurements as:

$$h = \frac{\varepsilon\sigma\left(T_a^4 - T_w^4\right)}{\left[\left(T_w - T_a\right) + \left(Y_w - Y_a\right)/K\right]} \quad (7)$$

and

$$h_m = \frac{\varepsilon\sigma\left(T_a^4 - T_w^4\right)}{L_{vap}\left[K\left(T_w - T_a\right) + \left(Y_w - Y_a\right)\right]} \quad (8)$$

where Y_w can be calculated from the saturation curve knowing the local temperature.

MATHEMATICAL MODEL

Four mathematical models were used to calculate the turbulence and two different approaches were followed to solve the wall. The continuity, velocity, energy, moisture content, and additional turbulent transport equations were solved using FLUENT 6.1.18. The used turbulent models are:

1) Standard κ-ε model: it is a semi-empirical method based on the transport equation for turbulent kinetic energy (κ) and its dissipation rate (ε) (Launder and Spalding, 1974). This model was developed assuming that the flow is fully turbulent and the effects of molecular viscosity are negligible.

2) The RNG- κ-ε model: This model is derived from the Navier-Stokes equations using the "renormalization group" method that results in a model with constants different from those in the standard κ-ε model. It also results in a differential equation for turbulent viscosity that is integrated to obtain an accurate description of how the effective turbulent viscosity varies with the effective Reynolds number, allowing the model to better handle low-Reynolds number and near wall flows. Another advantage of this model is that it calculates the effective inverse Prandtl number (or Smith in the case of mass transfer) as a function of μ_{mol} / μ_{eff}, which is consistent with experimental evidence and allows heat and mass transfer to be calculated in low Reynolds number regions (Fluent Inc. (2003)).

3) The SST κ-ω model: it is an improved version of the standard κ-ω model based on model transport equations for the turbulence kinetic energy (κ) and the vorticity fluctuation of turbulence (ω). This model was designed to be applied throughout the boundary layer, provided that the near-wall mesh resolution is sufficient.

4) Laminar model: It assumes that the flow is laminar and it does not take into account turbulent effects.

Two wall treatment approaches were used:

1) Enhanced wall treatment (EWT): It is a near-wall modelling method that combines a two-layer model (the viscosity affected near wall region is completely resolved all the way to the viscous sub-layer) together with enhanced wall functions. The near-wall mesh created was fine enough to resolve down to the laminar sub-layer ($y^+ \approx 1$); the created fine mesh started at 0.1 mm on the wall and it was gradually increased. All four turbulent models were solved using the fine near wall mesh.

2) The standard wall functions (SWF): They are the default wall functions in Fluent and are based on the proposal of Launder and Spalding (1974). This approach is valid for full developed turbulent flow. The near wall mesh was fixed at 2 mm giving a y^+ between 6 and 25. Only the RNG turbulent model was tested with the two different wall approaches.

Additionally, the radiation heat on the cylinder surface was taken in account within the four turbulent models and the two wall approaches combinations. The RNG κ-ε model was also tested under zero radiation conditions. Table 1 summarizes all the different models characteristics.

Model	Turb.	Wall mesh	Wall App.	Rad
A	RNG κ-ε	Fine	EWT	Yes
B	RNG κ-ε	Coarse	SWF	Yes
C	Laminar	Fine	EWT	Yes
D	Std. κ-ε	Fine	EWT	Yes
E	SST κ-ω	Fine	EWT	Yes
F	RNG κ-ε	Fine	EWT	No

Table 1. Model characteristics

Boundary conditions

The inlet conditions for the different experimental trials are in Table 2.

At the tunnel outlet, zero normal gradients are assumed for all variables except pressure. At the walls of the tunnel, zero velocity, heat flux and water flux are assumed.

Exp	V (m/s)	Re.	T (K)	Tu .	RH (%)	P (Pa)
1	0.5	2759	293.26	2%	39.5	102103
2	1.5	8275	293.36	2%	39.8	102230
3	3	16531	293.76	2%	40.4	102070

Table 2. Inlet conditions

At the cylinder surface, both heat flux and water concentration change with the position around the cylinder. The mass transfer coefficient changes point by point depending of the development of the boundary layer. Thus, the heat flux, which depends of the evaporation rate, also changes around the surface. On the other hand, the mass flux at the surface depends on the wall water concentration gradient, which is calculated with the vapor pressure at the wall temperature. Thus, the concentration gradient is function of the wall temperature.

To establish the thermal and mass boundary conditions, conservation of heat and water equilibrium apply. There are two calculation procedures depending of the wall approach used.

Wall enhanced Treatment

In this case the mesh is fine enough and the mass flux in the interface can be calculated with the water concentration at the surface and in the cell next to the surface with the equation:

$$J_w = \rho D \frac{(Y_w - Y_c)}{ds} \qquad (9)$$

The heat flux at the wall is calculated with the heat of vaporization and the radiation given the equation:

$$\dot{q}_w = J_w L_{vap} + \varepsilon\sigma\left(T_a^4 - T_w^4\right) \qquad (10)$$

The water concentration in the interface is calculated point by point around the cylinder as a function of the surface temperature as follow:

1) The vapor pressure at the interface is calculated as a function of temperature using a vapor pressure–temperature equation.

2) The molar concentration is calculated from the relation between the vapor pressure and the total pressure.

3) The mass water concentration is calculated from the molar water concentration and the molecular weights.

Standard Wall Function

It was used in model B (modelling the turbulence with the RNG κ-ε model). In this case the surface mass flux was calculated with the equations proposed by Launder and Spalding (1974) assuming that species transport behaves analogously to heat transfer. The mass flux at the wall J_w is calculated from the equation:

$$Y^* = \frac{(Y_w - Y_c)\rho C_\mu^{1/4}\kappa_P^{1/2}}{J_w} = \begin{Bmatrix} Sc.y^* \quad for \quad (y^* < y_C^*) \\ Sc_t\left[\frac{1}{\kappa}\ln(E.y^*) + P_c\right] for(y^* < y_C^*) \end{Bmatrix} \qquad (11)$$

Details of how calculate P_c, y_c^* and Sc_t are given in Fluent Inc. (2003). The heat flux at the wall and the water concentration at the surface are calculated in the same way as above.

Radiation effects

The heat of radiation was taken in account in modes A to E. Model F neglects the effect of radiation. Thus, equation 10 becomes:

$$\dot{q}_w = J_w L_{vap} \qquad (12)$$

Boundary condition implementation

The thermal and mass boundary conditions are not constant and change point by point around the cylinder. They also depend on the solution of the other transport equation. Heat flux on the wall depends of the mass flux; surface water concentration depends on surface temperature. Therefore, both boundary conditions were established programming a Fluent UDF (User Defined Function). The Programmed UDF's are:

1) DEFINE_PROFILE(heat_vaporization, t, j): It calculates the heat flux leaving the cylinder interface caused by evaporation and radiation according to the equation 10. The calculation is done over each face element around the cylinder. The radiative term is excluded only on model F. The wall mass flux J_w was calculated with the equation 9 or 11 depending of the wall approach used.

2) DEFINE_PROFILE(water_interface, t, n): It calculates the surface water concentration as a function of temperature following the procedure explained under "Wall enhanced treatment". The calculation is also done over each face element around the cylinder.

Because it is necessary to know the concentration profile to start the calculation of the wall heat flux, the modelling is first done with constant boundary conditions. The wall temperature is fixed as the wet bulb temperature and the mass concentration as the calculated in equilibrium at the wet bulb temperature. After getting convergence, the above UDF's were activated and iterations started again until it converges. It was also necessary to use a relaxation factor of 0.1 in the water_interface UDF to get convergence.

RESULTS AND ANALYSIS

Table 3 shows the root mean square percentage error (%RMS) of the different models. The calculation was done comparing the experimental global mass flux $\bar{J}_w$ and temperatures with the corresponded values obtained by modeling. It is clearly seen that the model A gets the better experimental data fit. Figure 3 shows the experimental and calculated mass flux $\bar{J}_w$ using the models A to F. The enhanced wall function method gives better results than the standard wall function as can be seen comparing models A and B. This is because the wall functions were developed for full turbulent flows and not for low Reynolds numbers. The RNG κ-ε model gives slightly better results in the calculation of the mass flux than the standard κ-ε model (model D). That can be explained given the improvement in the RNG model that allows calculations at low Reynolds numbers. Model A gives the best result matching almost perfectly the experimental data.

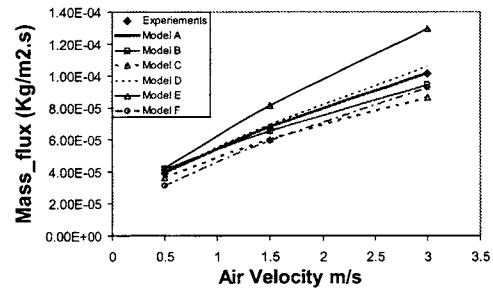


Figure 3. Mass flux vs. air velocity

Model	A	B	C	D	E	F
%RMS	0.51%	2.08%	7.41%	3.90%	7.15%	4.82%

Table 3. Global %RMS of the models.

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	286.55	286.16	286.32	
A	286.65	286.32	286.38	0.03%
B	286.68	286.30	286.32	0.03%
C	286.81	286.43	286.45	0.07%
D	286.61	286.23	286.25	0.02%
E	286.57	286.11	286.13	0.04%
F	285.37	285.58	285.88	1.24%

Table 4. Temperature (K) vs. air velocity

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	0.0104	0.0198	0.0307	
A	0.0107	0.0200	0.0312	1.48%
B	0.0111	0.0192	0.0292	4.51%
C	0.0095	0.0176	0.0261	10.38%
D	0.0108	0.0207	0.0332	5.00%
E	0.0115	0.0247	0.0416	22.38%
F	0.0106	0.0200	0.0313	1.51%

Table 5. $\overline{h}_m$ (Kg /m^{2}s) vs. air velocity

Model	V =0.5	V=1.5	V=3	%RMS
Exp.	9.22	17.94	28.53	
A	9.71	18.80	28.92	3.67%
B	10.52	17.74	26.15	8.20%
C	8.73	16.52	24.08	9.46%
D	9.68	18.93	29.60	4.18%
E	10.52	22.51	36.69	20.43%
F	9.76	18.81	28.90	3.87%

Table 6. $\overline{h}$ (W/m^{2}K) vs. air velocity

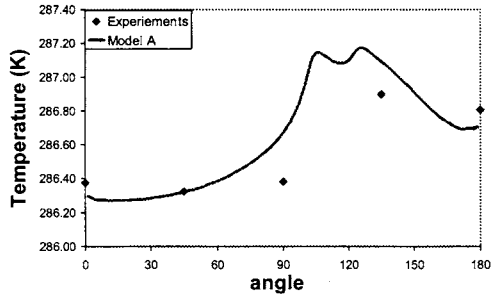


Figure 4. Temperature vs. angle at V = 0.5 m/s (model A)

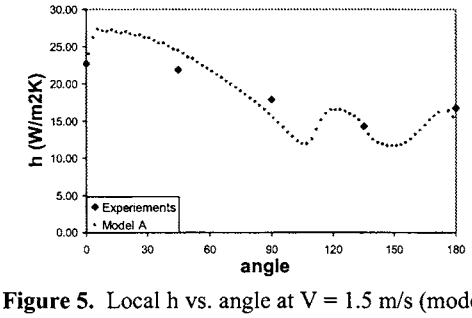


Figure 5. Local h vs. angle at V = 1.5 m/s (model A)

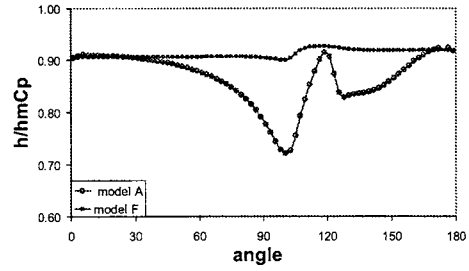


Figure 6. Relation $h/h_m\ C_P$ vs. angle at V = 0.5 m/s

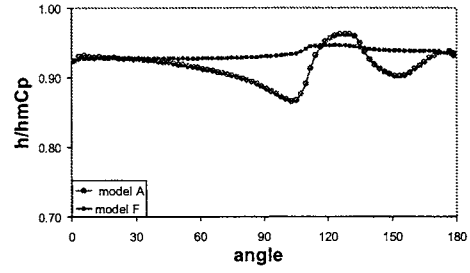


Figure 7. Relation $h/h_m\ C_P$ vs. angle at V = 1.5 m/s

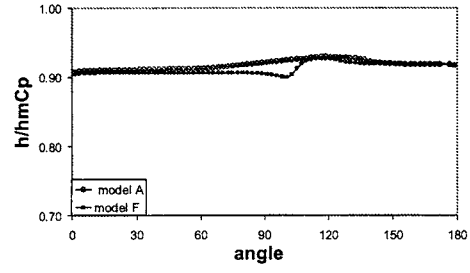


Figure 8. Relation $h/h_m\ C_P$ vs. angle at V = 3 m/s

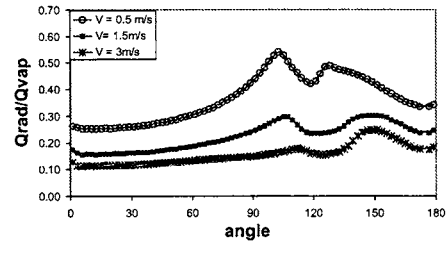


Figure 9. Relation Q_{rad}/Q_{vap} vs. angle

Although the SST κ-ω model (model E) was designed to be applied throughout the boundary layer, it did not fit the data better, as can be seen in Figures 3 and tables 4 to 6. The laminar model (model C) shows the lowest estimate of the mass flux, $\overline{h}$ and $\overline{h}_m$. That was caused because it did not take in account the effect of turbulence, which is important even though the turbulence intensity was just 2%.

Ignoring the effect of radiation (model F) causes underestimation of the mass flux (Figure 3) and temperature (Table 4). That can be explained with equation 10. Radiative heat is in opposite direction to evaporative heat flux and reduces the absolute value of the total wall heat flux. Thus, if radiation is not taken into account, the wall heat flux is higher and the wall

temperature is lower. Lower wall temperature means lower water surface concentration (given the dependence of temperature with vapour pressure), causing a lower water concentration gradient (or mass flux) in the wall.

The local prediction of temperature and h around the cylinder obtained with model A is good as it can be seen in Figures 4 and 5 (the angle is taken from the stagnant point).

The relationship between the global heat and mass transfer coefficients ($\bar{h}/\bar{h}_m C_p$) is almost constant and close to the value of $L_e^{2/3} \approx 0.892$ agreeing with the Chilton-Colburn analogy (Equation 5). However, there were differences found in the local relationship ($h/h_m C_p$) at low velocities when the radiative heat was taken in account. Figures 6, 7 and 8 show the local relationship between the heat and mass transfer coefficient as a function of the angle using models A and F. Figure 6 (made at V = 0.5 m/s) shows that using model A the relationship strongly deviates from a constant value reaching a minimum (19% deviation) at the separation point of the boundary layer. If the radiative heat is not taken in account (model F) the relation is constant and very close to $L_e^{2/3}$ as in Equation 5. The deviation decreases when the air velocity increases (figures 7 and 8), almost getting a constant value at V = 3 m/s. Figure 9 shows that the radiative heat becomes less important compared with the evaporative heat at higher air velocities. The deviation of the Chilton-Colburn equation at low velocities, or low Reynolds numbers, seems to be influenced by the heat of radiation that becomes important at low air velocities. That effect is stronger at the detachment of the boundary layer where the heat transfer coefficient is decreasing while the mass transfer coefficient is increasing, causing the ratio heat to mass transfer coefficients to decrease. At this point, the convection effects decrease and the radiation effect becomes more important causing the surface temperature to rise (Figure 4). The normal temperature gradient decreases making the wall heat flux reduce. Therefore, the local heat transfer coefficient decreases. Cess (1962) analytically shows that the local Nusselt number (non-dimensional heat transfer coefficient) in a laminar boundary layer of a fluid along a flat plate decreases by effect of radiation. The effect of radiation on the mass transfer coefficient is opposite to heat transfer coefficient. It makes the mass transfer coefficient to increase. This is because the wall temperature increases, as it was established above, making the vapour pressure and the water composition on the wall to increase. This increment on the wall water composition makes the mass gradient at the wall to rise and therefore, the mass transfer coefficient increases. This work shows that the Chilton-Colburn analogy loses accuracy to relate the local heat and mass transfer coefficients at low air velocities. The accuracy of this analogy (Equation 5) to correlate heat and mass transfer coefficients on evaporation and drying processes has been questioned before (Chen *et all* (2002))

CONCLUSION

1) The psychrometric method of Kondjoyan and Daudin (1993b) was followed to experimentally determine local and global heat and mass transfer coefficients around a cylinder at Reynolds numbers between 2700 and 17000.

2) The experiments were modelled with different turbulence models and two wall treatment approaches. It was found that the RNG κ-ε model using the enhanced wall treatment and taking in account the effect of radiation fits better the experimental data.

3) The CFD modelling shows that at low Reynolds numbers when the radiative heat is taken in account the relation $h/h_m C_P$ is not constant around the cylinder, reaching a minimum value at the detachment of the boundary layer.

ACKNOWLEDGEMENTS

The authors whish to acknowledge the support of the Australian Government through the Australian Research Council Large Grant A00103716.

REFERENCES

CESS, R. D., (1962), "The effect of radiation upon forced-convection heat transfer", Appl. Sci. Res., Section A, **10**, 430-438,.

CHEN X. D., LIN S. X. Q. and CHEN G., (2002), "On the ratio of heat to mass transfer coefficient for water evaporation and its impact upon drying modelling". Int. J. Heat and Mass Transfer, **45**, 4369-4372.

FLUENT INC., (2003), Fluent 6.1.18 user's guide, Vols. 1-5, Fluent Inc., Lebanon NH USA.

KESTIN J. and WOOD R. T., (1971), "The influence of turbulence on mass transfer from cylinders", J. Heat Transfer, **93**, 321—327.

KONDJOYAN, A. and DAUDIN, J.D., (1993a), "Heat and Mass Transfer Coefficients at the surface of Elliptical Cylinders placed in a Turbulent air flow", J. of Food Eng., **20**, 339-367.

KONDJOYAN, A. and DAUDIN, J.D., (1993b), "Determination of transfer coefficients by psychrometry", Int. J. Heat Mass Transfer, 36, No. **7**, 1807-1818.

KONDJOYAN, A. and DAUDIN, J.D., (1997), " Heat and Mass Transfer-Coefficients at the surface of a Pork Hindquarter", J. of Food Eng., **32**, 225-240.

LAUNDER, B.E. and SPALDING, D.B., (1974), 'The numerical computation of turbulent flows", Computer Methods in Applied Mechanics and Engineering, **3**, 269-289.

LEWIS J. S., (1971), "Heat transfer predictions from mass transfer measurements around a single cylinder in cross-flow", Int. J. Heat and Mass Transfer, **14**, 325-329.

COMPARISON OF THREE METHODS OF ESTIMATING THE EFFECTIVE DIFFUSIVITY OF MOISTURE IN MEAT FROM DRYING DATA

Francisco Javier Trujillo (1), Chaiyan Wiangkaew (2) and Q. Tuan Pham (3)

School of Chemical Engineering and Industrial Chemistry, University of New South Wales
Sydney 2052, Australia

(1) francisco.trujillo@student.unsw.edu.au
(2) z3015341@student.unsw.edu.au
(3) tuan.pham@unsw.edu.au

ABSTRACTS

The diffusivity of water in meat was experimentally determined using a drying technique. Three different mathematical methods were used to determine the diffusivity from the drying data. The first assumes constant diffusivity, volume, temperature and surface moisture concentration. The second assumes a convective boundary condition. The last also takes into account shrinkage. Important differences on the calculated diffusivity were found using these three different methods.

Keywords: Moisture diffusivity, drying technique, meat, shrinkage

INTRODUCTION

Accurate data on the diffusivity (D) of water in meat is very important for predicting the weight loss of beef carcasses during the chilling process. The diffusivity also affects bacterial growth, as it controls the movement of water to the meat surface. However, few works on moisture diffusivity in beef have been found in the literature. Lomauro *et al.* (1) reported a figure for ground beef moisture diffusivity and Motarjemi (2) reported the diffusivity of raw minced beef at various temperatures and moisture content. Merts *et al.* (3) developed a procedure to measure the drying curve of a cylindrical meat sample and estimated D using several different techniques. Radford *et al.* of the CSIRO Meat Research Laboratory used the drying technique to determine the moisture diffusivity (4). The experimental data was fitted to a heat and mass transfer finite difference model and an equation was obtained that expresses diffusivity as a function of temperature and water concentration.

Since the CSIRO work in the 1970's, there has not been any detailed research on moisture diffusivity on beef. Besides, there has been considerable variation between the values of D reported in the literature. For instance Merts *et al.* (3) reported a diffusivity of 6.54E-11 m2/s at 10 C while the diffusivity obtained with Radford's equation (4) at the same temperature is 3.26E-10 m2/s. Merts *et al.* (3) also reported variations in the calculated diffusivity using different methods. Thus, there is a clear need to carry out some more experimental diffusivity measurements and to determine whether the method of estimation affects the diffusivity calculation.

EXPERIMENTAL METHOD

The drying method was selected among different procedures reported in the literature to determine the moisture diffusivity. According to Zogzas *et al.* (5) this is the most popular technique because drying process are of major importance in the industry and the recorded data can be used with confidence for scaling up industrial driers. Another reason is that the equipment can be easily constructed and controlled.

The method consist of drying a product sample of a standard geometry, such as a slab or cylinder, in a drying tunnel or a controlled environmental chamber where the air velocity can be controlled with a fan, relative humidity with a humidifier, and dry bulb temperature with a heater. The dry and wet bulb temperatures, as well as the weight of the sample, can be recorded at regular intervals. From this data the diffusivity of the sample is determined.

Our equipment consists of an environmental chamber with temperature and humidity control. Inside the chamber, a flat piece of meat contained in a circular dish was placed on an analytical balance. A small fan blew air continuously over the surface of the meat to ensure a high surface mass transfer coefficient, and the balance reading was automatically recorded over time. The dry and wet bulb temperatures were also recorded. The external mass transfer coefficient was determined in a second experiment by replacing the meat by water under the same airflow and temperature conditions and similar humidity conditions.

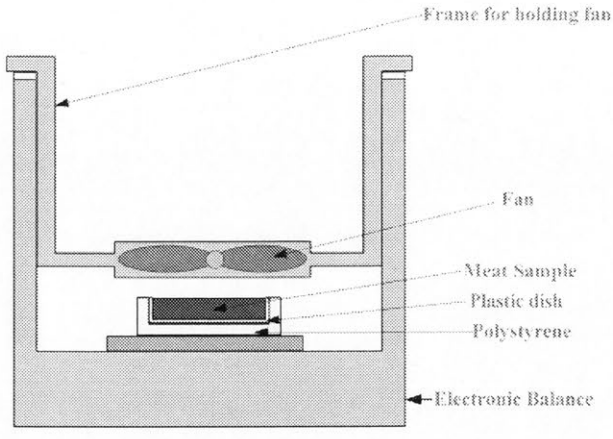


Figure 1: Experimental setup.

MATHEMATICAL MODEL

Drying is a complicated process where heat and mass transfer occurs simultaneously. The mechanism of water diffusion within the solid is complex. Different mechanisms for moisture transportation have been proposed to explain the drying phenomena (e.g. molecular diffusion, capillary motion, and diffusion through solid pores). However, no single mechanism prevails throughout the drying process (6). Thus, an effective diffusivity is used to compensate for the combination of mechanisms. Mathematical modeling is a useful way to validate mechanisms of drying and calculate the effective diffusivity. Models with different level of complexity can be used to find the diffusivity that better fit the experimental data. The degree of complexity depends on consideration of heat transfer and temperature variations, assumptions on the boundary conditions, diffusivity dependence on moisture and temperature, shrinkage and changes on volume. According to Mulet (7), the effective diffusivity may be wrong if the model does not take into accout all the physically important factors. In this work, three models with different levels of complexity will be used and compared. All of them are estimated from the Fick's law according to the equation:

$$\frac{\partial(\rho_s \cdot X)}{\partial t} = \nabla(D \cdot \rho_s \cdot \nabla X) \tag{1}$$

where X is the moisture content (kg water/ kg dry material), ρ_s is the density of dry solids (kg solids/m^3), t is the time (s) and D is the moisture diffusivity (m^2/s).

Model A (Simplified Model):

This model assumes constant moisture diffusivity, no volume change, uni-dimensional moisture movement, uniform initial moisture distribution, negligible external resistances, an isothermal process. With these simplifications, equation 1 can be solved analytically for the case of an infinite slab (8):

$$X^* = \frac{\overline{X} - X_e}{X_o - X_e} = \frac{8}{\pi^2} \sum_{n=0}^{\infty} \frac{1}{(2n+1)^2} e^{\left(-(2n+1)^2 \frac{\pi^2 Dt}{L^2}\right)} \tag{2}$$

where X^* is the dimensionless moisture content ratio, $\overline{X}$ is the average moisture content of the material at time t, X_o is the initial moisture content, X_e is the equilibrium moisture content and L is the thickness of the slab.

Model B:

This model assumes constant moisture diffusivity without volume change, uni-dimensional moisture movement and uniform initial moisture distribution, but it takes in account temperature changes caused by the evaporation and the effects of the surface resistance on the drying process, as expressed by the equation:

$$F = h_m(Y_a - Y_s) \tag{3}$$

where h_m is the external mass transfer coefficient, Y_a is the humidity of the air and Y_s is the humidity of the air in equilibrium with the meat surface. With these changes, equation 2 has to be solved numerically. A finite difference method was used to solve equation 2.

Model C:

This model also takes in account the shrinkage of the meat sample. Shrinkage was observed on the experimental trials, the meat samples volume being reduced by up to 30% of the original volume at the end of the drying process.

Model C assumes that the meat shrinks isotropically during the drying process. Uni-dimensional moisture movement, uniform initial moisture distribution and constant moisture diffusivity are still assumed. Changes in temperature and the effects of the external resistance are taken into account.

The specific volume of the sample was assumed to vary with moisture content according to (6):

$$v = \frac{1 + \beta \cdot X}{\rho_{bo}} \tag{4}$$

where v is the specific volume of the sample (m^3/kg dry material), β is the volume-shrinkage coefficient, and ρ_{bo} is the bulk density of the sample at zero value of moisture content (kg dry meat/ m^3). A finite volume method was used to model the process because it can be easily adapted to take shrinkage into account.

Determination of the Diffusivity:

The diffusivity was determined by minimizing the sum of squared differences between the weight data and the prediction by the model, using an evolutionary algorithm program written by Pham (9). This minimization method, unlike classical gradient methods, starts with a random "population" of trial points which "evolve" towards the optimum using various mechanisms such as "reproduction", "mutation" and "selection". The advantage of this stochastic method is that it deals better with objective functions that contain random errors arising out of measurements or numerical calculations. Classical methods generally follow a single search path, thus, they may end up at a local minimum instead of at the desired global one, and they are easily confused by errors.

RESULTS AND ANALYSIS

Figure 2 show drying curves at 19.92^{0}C and compare the experimental data with simulated results using models A, B and C. It is clearly seen that Model C (Shrinkage – Finite Volume) has the best fit followed by model B (Finite Difference - Convective boundary). Model A has the worst fit.

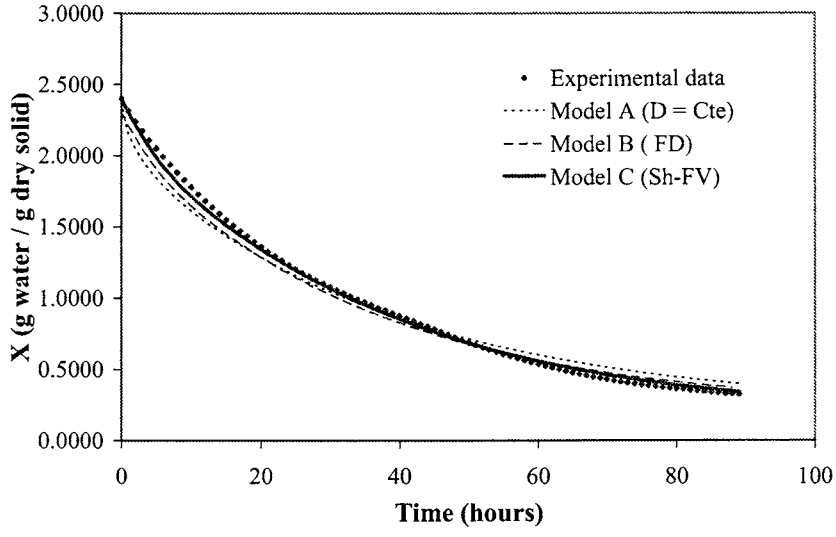


Figure 2: Comparison of drying curves for model A, B and C at 19.92 C, 79.6% relative humidity.

Figure 3 plots the calculated diffusivity as a function of temperature. The values obtained with model C can be up to 50% less than those obtained with models A or B. Mulet (7) reported similar differences for carrots when taking shrinkage into account. In the graph it is seen that the difference between models A and B is bigger at low temperatures but decreases at higher temperatures. That can be explained with Figures 4 and 5 where the average (X_m) and surface (X_s) water content are plotted as a function of time. In these graphics it is seen that at the lowest temperature (6.75 C) the surface water content is slowly decreasing to reach the equilibrium (Figure 4). Meaning that convection on the boundary is important. That explains the higher difference comparing with model A (that assumes that the surface is already on equilibrium). On the other hand, at the highest temperature (40.41 C) the surface water activity drops to the equilibrium almost immediately after starting the drying process (Figure 5). Therefore, the calculated diffusivity with models A and B must be similar.

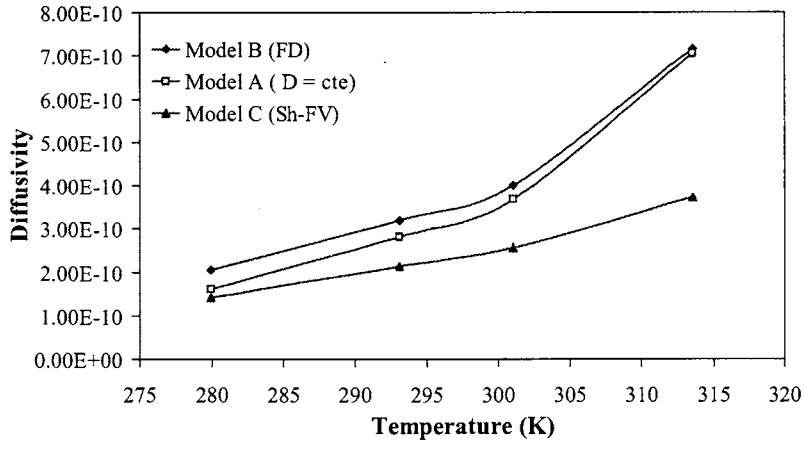


Figure 3: Diffusivity vs temperature, using models model A, B and C.

Table 1 shows the average root mean squares percent error (%RMS) using models A, B and C. It is seen that the %RMS reduces from model A to C showing that the model that better fit the data is the model C. That is expected since the meat samples suffered strong shrinkage during the dryings as is seen in Figures 6. Therefore, model C is more accurately describing the physical phenomenon. Mulet (7), who tested several detailed models with different levels of complexity, concluded that the main factor that must be considered to accurately determine the diffusivity is the product shrinkage. According to Hernandez *et al.* (10) food shrinkage is extremely important in diffusional drying because it produces a variation in the distance required for the movement of water molecules.

Model	%RMS Average
Model A	1.269%
Model B	0.874%
Model C	0.564%

Table 1. Average %RMS percent error fitting the experimental data with models A, B, and C.

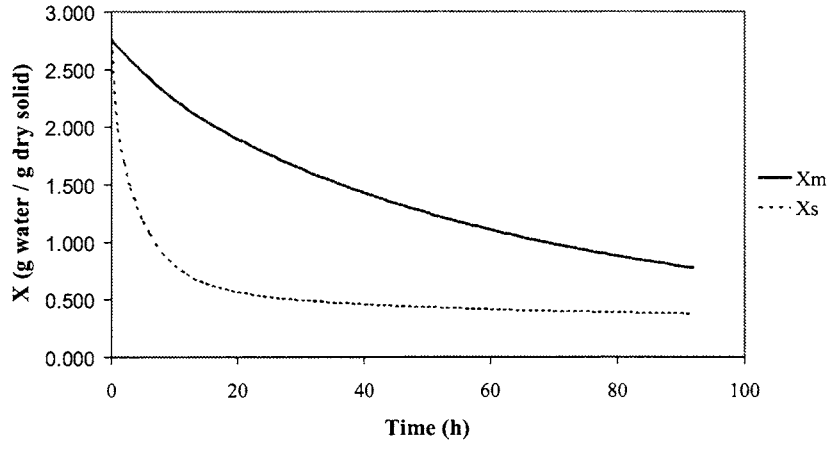


Figure 4. Average moisture (X_m) and surface moisture (X_s) vs time, calculated with model B at 6.75^{0}C, 92% relative humidity.

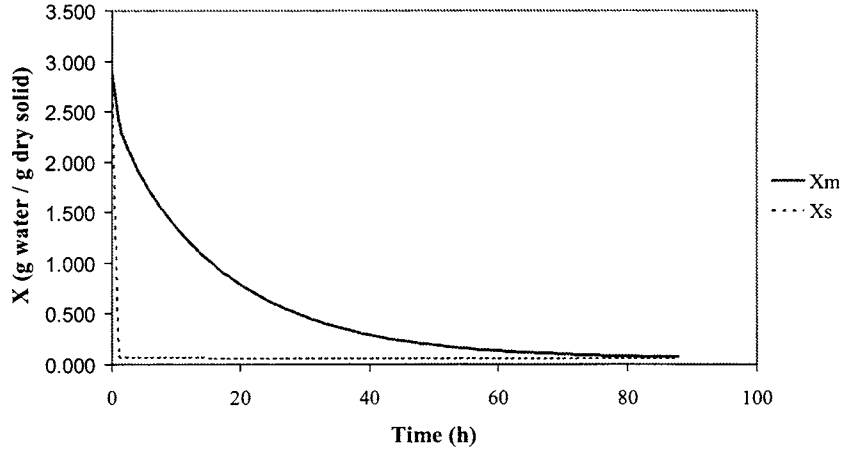


Figure 5. Average moisture (X_m) and surface moisture (X_s) vs time calculated with model B at 40.4 C, 78.1% relative humidity.

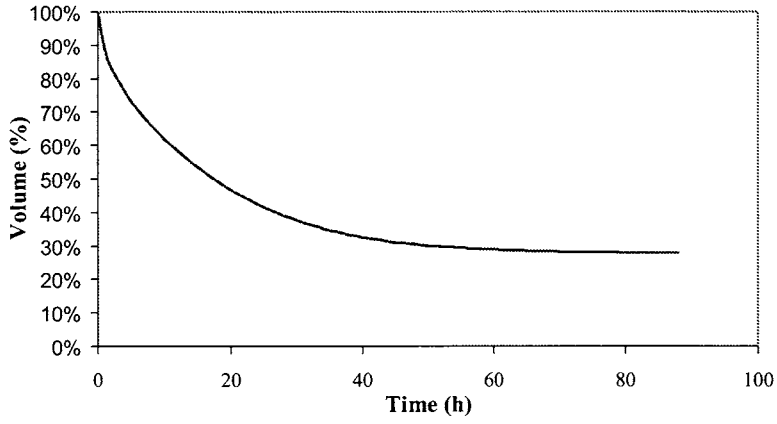


Figure 6: Percentage change in volume during drying at 40.4 C, 78.1% relative humidity.

CONCLUSIONS

- When calculating effective diffusivity from a drying curve, significantly different results are obtained depending on whether surface resistance, non-isothermality (Model B) and shrinkage (Model C) are taken into account. The difference between the diffusivity obtained with models A (none of the above factors taken into account) and B decreases when the temperature increases. This is caused by the external resistance to the mass transfer that becomes relatively more important at lower temperatures.
-
- The model that best fits the experimental data is model C (which takes into account shrinkage) because it is a better representation of the actual physical situation.

BIBLIOGRAPHY

1. Lomauro C. J., Bakshi A. S., Labuza T. P. Moisture transfer properties of dry and semimoist foods. Journal of Food Science, 55, 218-223, 1990.
2. Motarjemi Y. "A Study of some Physical Properties of Water in Foodstuffs." Ph.D. Thesis, Department of Food Engineering, Lund University, Lund, Sweden, 1998.
3. Merts I., Lovatt S. J., Lawson C. R. Diffusivity of moisture in whole muscle meat measured by a drying curve method. IIR Proceedings Series "Refrigeration Science and Technology". Sofia, Bulgaria, 1998.
4. Radford R. D., Herbert C., Lovett D. A. Chilling of meat- A mathematical model for heat and mass transfer. Proc. Comm. C2 Meeting, IIR Bull. Annex 1976-1, pp. 323-330, 1976.
5. Zogzas N. P., Maroulis Z. B. Effective moisture diffusivity estimation from drying data. A comparison between different methods of analysis. Drying Technology. 14, 7 and 8, 1543-2573, 1996.
6. Zogzas N. P., Maroulis Z. B., Marinos-Kouris D. Moisture diffusivity methods of experimental determination – A review. Drying Technology. 12, 3, 483-515, 1994.
7. Mulet A. Drying modeling and water diffusivity in carrots and potatoes. Journal of Food Engineering. Vol. 22, pp. 329-348, 1994.
8. Crank J. "The Mathematics of Diffusion." 2nd Ed., Oxford University Press, Oxford, 1975.
9. Pham, Q. T. Competitive evolution: A natural approach to operator selection. In: "Progress in Evolutionary Computation, Lecture Notes in Artificial Intelligence", Vol. 956, p.49-60. X. Yao (ed.), Springer-Verlag, Heidelberg. 1995.
10. Hernandez J. A., Pavon G., Garcia M. A. Analytical solution of mass transfer equation considering shrinkage for modeling food-drying kinetics. Journal of Food Engineering, 45, 1-10, 2000.