Heat and Mass Transfer in Food Processing

by

James Leo Spielbauer

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

MASTER OF SCIENCE
(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

1992

Abstract

The Oscar Mayer Foods Corporation has been cooking hot dogs by essentially the same process for years. This has made designing new cooking processes relatively simple as the new process could be based on the design of an existing process. Recently the design procedure has changed. The introduction of new products such as cheese hot dogs and low fat hot dogs has created the demand for a greater understanding of the cooking process and the physical factors that influence the heat and mass transfer in the final product.

The goal of this research is to determine the effects that physical parameters such as temperature, humidity, air velocity, and time have on the temperature profile of the product as it is cooked. The research involved the development of models of a hot dog and the cooking process that were used to simulate a real process. Experiments were conducted on real cooking equipment to determine the convective heat transfer coefficient which was used to model both the heat and mass transfer at the product surface.

The models were then used to simulate cooking of hot dogs under various design conditions to determine the importance and effects various parameters have on the temperature profile of the final product. The model is flexible and can aid the design of cooking processes for new and existing products.

Acknowledgments

I saved the acknowledgments for last because I thought it would be the easiest page to write. But as I look back over the past year and a half and longer I quickly realize that there have been many people who have helped me reach the educational goals I have set for myself. Therefore I'd like to thank all those people who, in one way or another, have helped me on my way.

First off, I'd like to thank John Mitchell, Bill Beckman, and Sandy Klein for allowing me to pursue my goal of a master degree under their guidance. Their combined support has made the last year and a half a worthwhile adventure.

I'd like to thank my parents whose constant support, both emotional and financial!, have helped me to reach the goals I have set for myself. There were times I called home because school wasn't going well, you were both there for me and I love you both very much.

Next, I'd like to thank Jerry Marra and the Oscar Mayer Foods Corporation. Jerry, I remember when you attended my first seminar. It wasn't the best presentation in the world, but your comments and compliments gave me the motivation I needed to make it through the week. Thank you for your constant support and advice!

The next group of people I'd like to thank are the friends I have made at the lab.

Jeff Thornton, for convincing me to come up to the lab in the first place. You said I'd love it, it's great up here! Yah Jeff, you sure were right!

Doug Reindl, for buying the labs first water bong and letting me sky a few. Not to mention giving me advice regarding my interview with "double A, beep-beep, M C O".

Osman Ahmed, for your constant support, advice and friendship. Your ability to always put a smile on your face helped cheer me up on even the worst days.

Robert Lanoue. What can I say Robo Slobber? "Lets get together, have a gumby's pizza, and slam a few POINTS!" Oh, and if anyone ever insults the TLC, remember to "kick 'er in the head!"

Bernd Gebbeken, for your willingness to go tip a few on Thursdays at Monday's, for going to see Washington D.C. and Petersburg, for taking me sailing, and for becoming a life long friend.

Finally, I'd like to thank all those people I have not mentioned by name, you know who you are. You're the people who have always been there. You're the friends from high school, college, camp Onaway, Cabin Fever, hunting old Mossy Horns, Stevens Point trivia, old roommates, and drinkin' buddies. Thanks for all the good times!

Table of Contents

Abstract ..... *ii*

Acknowledgments ..... *iii*

List of Figures ..... *viii*

List of Tables ..... *ix*

Nomenclature ..... *x*

I. Introduction *1*

I.1 Background 2

I.2 A Detailed Description of the Hot Dog Cooking Process 2

I.3 A Detailed Description of a Typical Cooking Zone 5

I.4 Temperature Profile of Cooking Process 8

I.5 Organization 10

II. Development of the Product Model *12*

II.1 Transient Conduction Within the Product 12

II.1-1 Governing Partial Differential Equation 13

II.1-2 Simplifying Assumptions 14

II.1-3 Boundary and Initial Conditions Defined 16

II.1-4 Analytical Solution 17

II.1-5 Finite-Difference Approximation 18

II.2 Product Model Modified for Mass Transfer 23

II.2-1 Boundary Layer Assumptions 24

II.2-2 Mass Flux at the Product Surface 25

II.2-3 Energy Flux at the Product Surface 29

II.2-4 Lewis Number of Unity Assumption 32

II.2-5 Mass Transfer Conductance and Driving Force 35

II.2-6 Simplified Mass Flux Equations 37

II.2-7 Heat Flux to Product 39

II.2-8 Surface Node Modified for Mass Transfer 40

II.2-9 Mass Diffusion Within the Product 42

II.2-10 Mass Transfer Model Modified for Hide Formation 44

III. Heat Transfer Coefficients for Food Processing 46

III.1 Designing the Heat Transfer Coefficient Experiment 46

III.1-1 Experiment Design 47

III.1-2 Design Equations for a Lumped Solid 49

III.1-3 Uncertainty Analysis 54

III.2 Process Heat Transfer Coefficient Experiment 63

III.2-1 Heat Transfer Coefficient Data Acquisition 64

III.2-2 Experimental Results 67

IV. Dry Bulb Temperature Variation Models 74

IV.1 Sensible Heat Exchanger Model 75

IV.1-1 Effectiveness-NTU Heat Exchanger Analysis 76

IV.1-2 Modifications for Flow of Hot Dogs in an Oven 81

IV.2 Cooling Tower Analogy 86

V. Model Validation *95*

V.1 Simulation of Individual Zones 96

V.2 Internal Temperature Profile 105

VI. Conclusions and Recommendations *108*

Appendix *114*

A.1 Simulation Programs 114

A.1-1 House Specification Program 115

A.1-2 Product Specification Program 121

A.1-3 Process Specification Program 128

A.1-4 Main Simulation Program 136

A.2 Analytical Solution 172

A.2-1 Bessel Function Root Finder (Written by Doug Reindl) 172

A.2-2 EES Code Used to Solve Analytical Solution 181

A.3 Heat Transfer Coefficient Experiment Programs 182

A.3-1 EES Program Used to Design the Aluminum Hot Dog 182

A.3-2 Program Used to Sort Data Trace Data 183

A.3-3 Heat Transfer Coefficient Calculation Program 186

References *191*

Bibliography *192*

List of Figures

Figure 1.1 Process Cross Section 3

Figure 1.2 Convection Oven Cross Section 6

Figure 1.3 Process Temperature Profile 9

Figure 2.1 Control Volume in Cylindrical Coordinates 13

Figure 2.2 Interior Node for Finite-Difference Approximation 19

Figure 2.3 Surface Node for Finite-Difference Approximation 21

Figure 2.4 Center Node for Finite-Difference Approximation 22

Figure 2.5 Fluid Flow Over a Hot Dog 24

Figure 2.6 Boundary Layer Flow Over a Flat Plate 25

Figure 2.7 Mass Flux of Water From the Product Surface 26

Figure 2.8 Energy Flux From Product Surface 30

Figure 2.9 Surface Node With Mass Transfer 41

Figure 3.1 Solid Used in Lumped Capacitance Analysis 47

Figure 3.2 Design Graph for Aluminum Alloy 2024 52

Figure 3.3 Aluminum Hot Dog Schematic 54

Figure 3.4 Data Trace Temperature Recorder 56

Figure 3.5 Data Trace Precision Plot 57

Figure 3.6 Natural Convection Heat Transfer Coefficient 60

Figure 3.7 Error in Heat Transfer Coefficient 61

Figure 3.8 Natural Convection Data Re-analyzed 63

Figure 3.9 Stick of Product With Data Traces Attached 65

Figure 3.10 Raw Data Temperature Profile 66
Figure 3.11 Heat Transfer Coefficient Plot for Smoke Zone 69
Figure 3.12 Measured h as a Function of Average Air Velocity 72
Figure 3.13 Average Heat Transfer Coefficient Labeling Code 72
Figure 4.1 Hot Dog "Flow" Within a Baking Zone 75
Figure 4.2 Counter Flow Heat Exchanger 76
Figure 4.3 Pass Height and Pass Spacing 82
Figure 4.4 Cabinet Width 82
Figure 4.5 Flow of "Wet" Product in a Cook Zone 87
Figure 5.1 Smoke Zone Simulation Comparison 97
Figure 5.2 Drying Zone Simulation Comparison 99
Figure 5.3 Dry-Cook Zone Simulation Comparison 101
Figure 5.4 Final Cook Zone Simulation Comparison 102
Figure 5.5 Smoke Zone Product Temperature 103
Figure 5.6 Final Cook Zone Product Temperature 104
Figure 5.7 Verification of Finite-Difference Approximation 106

List of Tables

Table 3.1 Measurement Uncertainties 58

Nomenclature

Roman Letters

A	Area, m^2
A'	Area per unit length, m
B	Mass transfer driving force
Bi	Biot number
C	Specific heat, J/kg-K
C_p	Specific heat at constant pressure, J/kg-K
C_r	Capacitance ratio
C_s	Effective specific heat, J/kg-K
D	Diameter, m
D_{AB}	Mass diffusivity of component A in component B, kg/m^2-s
D_w	Mass diffusivity of water in air, kg/m^2-s
$\dot{E}_e$	Rate of energy transfer due to evaporation, W
$\dot{E}_{gen}$	Rate of energy generation, W
$\dot{E}_{in}$	Rate of energy transfer into a control volume, W
$\dot{E}_{out}$	Rate of energy transfer out of a control volume, W
$\dot{E}_{store}$	Rate of increase of energy stored within a control volume, W
g	Mass transfer conductance, kg/m^2-s
g^*	Value of mass transfer conductance at very small mass transfer rates, kg/m^2-s
g'''	Energy generation rate per unit volume, W/m^3

$G_{diff,w}$	Mass flux of water transported by diffusion, kg/m²-s
h	Convective heat transfer coefficient, W/m²-K
i	Enthalpy, J/kg-K
i_{fg}	Latent heat of vaporization, J/kg
i_j	Partial enthalpy of component j of a mixture, J/kg
J_0	Bessel function of the first kind of order zero
J_1	Bessel function of the first kind of order one
k	Thermal conductivity, W/m-K
L	Length, m
L_c	Characteristic length, m
Le	Lewis number
M	Mass, kg
m_w	Mass concentration of water in air, kg_{water}/kg_{air}
$\dot{m}$	Mass flow rate, kg/s
$\dot{m}_e$	Mass flow rate of water due to evaporation, kg/s
$\dot{m}''$	Total mass flux at surface or phase interface, kg/m²-s
NTU	Number of transfer units
Nu	Nusselt number
O	Order of magnitude of error due to neglecting higher order terms
p	Pressure, kPa
P	Conserved property, m_w or i
Pr	Prandtl number
q	Heat transfer rate, W
q''	Heat transfer rate per unit area (heat flux), W/m²

r, ø, z	Cylindrical Coordinates
R	Outer radius of product, m
Ra	Rayleigh number
R_{cond}	Resistance to conductive heat or mass transfer, m^2-K/W
R_{conv}	Resistance to convective heat or mass transfer, m^2-K/W
Re	Reynolds number
t	Time, s
T	Temperature, °C
u, v, w	Mass average flow velocity components, m/s
U	Overall heat transfer coefficient, W/m^2-K
V	Volume, m^3
V_a	Average air velocity, m^2/s
V_{chain}	Velocity of conveyance chain, m^2/s

Greek Letters

α	Thermal diffusivity, m^2/s
Δ	Designates a difference when used as a prefix
δ	Uncertainty in a measurement
ε	Emissivity; heat exchanger effectiveness
θ	Temperature difference, K
λ	Eigenvalue; diffusion coefficient ρD_w or k/C_p, kg/m-s
ν	Kinematic viscosity, m^2/s
ρ	Mass density, kg/m^3

σ	Stefan-Boltzmann constant
$\varnothing$	Azimuthal angle, rad
ω	Humidity ratio

Subscripts

0	Evaluated within the boundary layer at the surface
1, 2, 3	Node numbers
a	Air water-vapor mixture
c	Cold fluid
cond	Conduction
conv	Convection
db	Dry bulb
e	Evaporation
eff	Effective
f	Saturated liquid condition
g	Saturated vapor condition
h	Hot Fluid
hd	Hot dog
i	Initial condition
j	Refers to substance j, which is a component of a mixture
L	Evaluated at the surface but within the liquid phase
max	Maximum
min	Minimum

n	Index used to indicate node number within product
N	Surface node number
rad	Radiation
ref	Reference state
s	Surface; saturated
T	Evaluated at the transferred substance state
w	Water
wb	Wet bulb
∞	Free stream condition

Superscripts

$'$	Per unit length
$''$	Per unit area
p	Index used to indicate time in finite-difference approximation

Overbar

$\bar{}$	Surface average conditions

Chapter I

Introduction

Oscar Mayer has been cooking hot dogs by essentially the same process for years. This has made designing new cooking processes relatively simple as the new process could be based on the design of an existing process. Recently the design procedure has changed. The introduction of new products such as cheese hot dogs and low fat hot dogs has created the demand for a greater understanding of the cooking process and the physical factors that influence the heat and mass transfer in the final product.

The goal of this research is to determine the effects that physical parameters such as temperature, humidity, and time have on the product as it is cooked. The research involved the development of models of a hot dog and the cooking process that were used to simulate a real process. The models were used to simulate cooking of hot dogs under various design conditions to determine the importance and effects various parameters have on the temperature profile of the final product. The model is flexible and can aid the design of cooking processes for new and existing products.

I.1 Background

Cooking of food products is familiar to just about everyone as it is a task encountered during everyday life. Yet as important as cooked food is to everyday life, there is not a clear understanding of the cooking process and the factors that influence the quality of the final product. Part of this lack of understanding is due to the complexity of the cooking process. There can be many phenomena such as melting of fats, denaturation of proteins, and water evaporation, all of which can affect the final quality of the food product. Moreover, even if the cooking process was completely understood, the material properties of food poses another problem. Thermal properties are usually functions of temperature and foods that exhibit large variability in their composition will exhibit the same degree of variability in their thermal properties [1].

Even with all the problems and difficulties associated with food processing, it is possible to model food processing systems by making simplifying assumptions. These assumptions will limit the scope of the solution, but at the same time lead to models which will aid in the understanding of the heat and mass transfer that occurs during cooking.

I.2 A Detailed Description of the Hot Dog Cooking Process

Before the cooking process can be explained it is important to understand how the hot dogs are formed and loaded into the processing equipment. Initially the hot dog meat is in the form of a liquid batter. The batter is stuffed into a synthetic casing that forms the shape of the hot dog and holds the meat together until it can be cooked. The

casing is twisted to form a long chain of hot dogs. The hot dogs are then looped onto a stainless steel rack referred to as a stick. The stick is loaded onto a conveyance chain that transports the hot dogs through the entire cooking process. The chain speed or conveyance rate can be varied but is typically set at 5.5 ft/min. At this conveyance rate it takes approximately 96 minutes for a stick of hot dogs to go through the entire process.

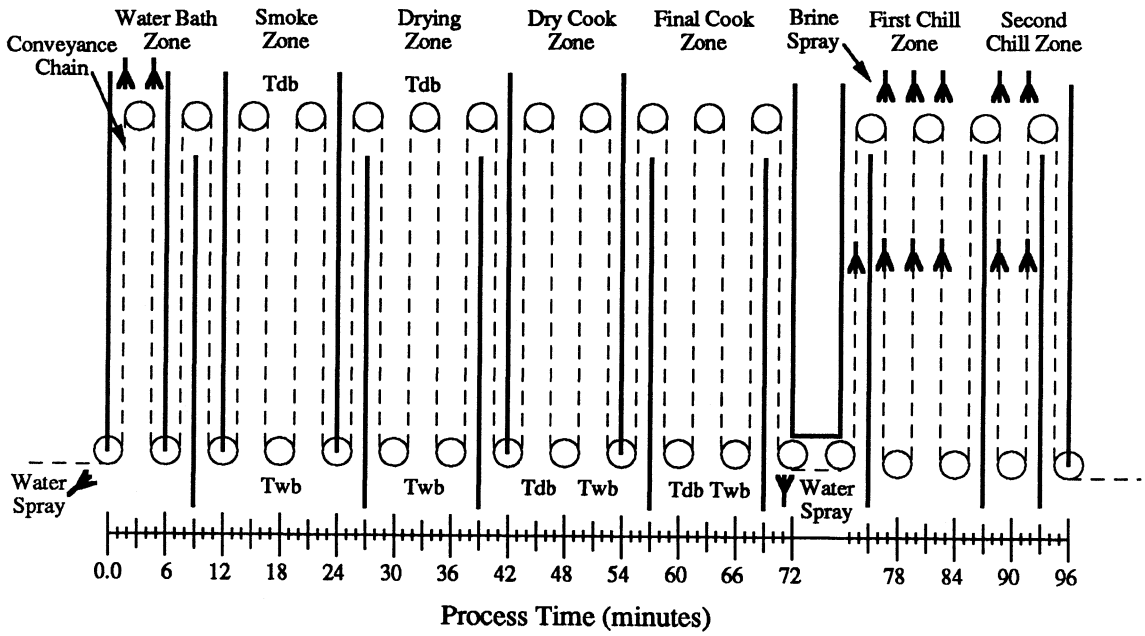

Figure 1.1 Process Cross Section

The cooking process, shown in Fig. 1.1, consists of seven different physical zones. Each zone is approximately twelve minutes long. The product typically makes four passes in each zone, where a pass is defined as the distance from the top of the zone to the bottom of the zone. Each zone is responsible for a different aspect of the cooking process and therefore has different free stream conditions such as dry bulb temperature,

wet bulb temperature and average air velocity. The zones are often times separated by what is called a transition zone. The function of the transition zone is to prevent the conditions in one zone from adversely affecting the conditions in an adjacent zone. This is accomplished by controlling the air pressures in each zone, thus minimizing the flow of air from one zone to the next.

The first zone is the water bath zone. The primary function of this zone is to quickly bring the product temperature up to approximately 90°F. The temperature increase is accomplished by spraying warm water on the product as it passes through the zone. The second half of this zone is an area in which water on the product surface is allowed to evaporate and drip off the product. The drying of the product surface is important because water droplets on the surface of the product will adversely effect the final surface color and quality of the product.

The next zone is called the smoke zone. The main function of this zone is to give the product its smoked flavor. The smoke is produced by burning wood chips and then entraining the resulting smoke into the main air flow for this zone. The smoke then diffuses through the surface of the product and into the meat.

The next zone, called the drying zone, is probably the most important zone as far as final product characteristics are concerned. The product undergoes both physical and chemical changes during processing in this zone. The first and most obvious function of this zone is to dry the product which brings the product to the proper weight as specified on the consumer package. But equally important, the drying process also brings meat proteins to the product surface. As these proteins accumulate at the product surface they form what is called the hide. The formation and firmness of this hide

is important in producing what is referred to as the "bite" of the product. These proteins also react with the smoke that has diffused into the product from the previous zone to give the product its color. If the product is over-dried it will be under-weight and dark in color. If the product is under-dried it will have a non-uniform surface color and will therefore fail to meet quality requirements.

The next two zones which are the dry cook zone and the final cook zone, basically just cook the product. In the final cook zone the product core temperature must reach at least 155°F to satisfy government health and safety regulations.

The last two zones are chilling zones. Their main function is to quickly reduce the product temperature to help reduce damages that might occur during packaging, to prolong shelf life, and to reduce the cooling load on the refrigerators where the product is stored. Initially the product temperature is reduced with a water spray which is located in the transition zone that lies between the final cook zone and the first chill zone. Further cooling is accomplished by spraying the product with a chilled salt brine as the product continues through the two chill zones.

I.3 A Detailed Description of a Typical Cooking Zone

The term baking is defined by the COST 91 terminology as, "cooking of food in an oven in air to which water vapor may or may not be added" [2]. Thus cooking involves both heat and mass transfer.

The majority of the baking process occurs in the smoke, drying, dry-cook, and final cook zones. These zones, see Fig. 1.2, are all very similar in design and are referred to as convection ovens. Convection ovens have four basic process parameters: oven design, air temperature, velocity, and humidity. Oven design refers to the physical shape of the oven and how it relates to the air distribution and flow patterns that develop inside the oven.

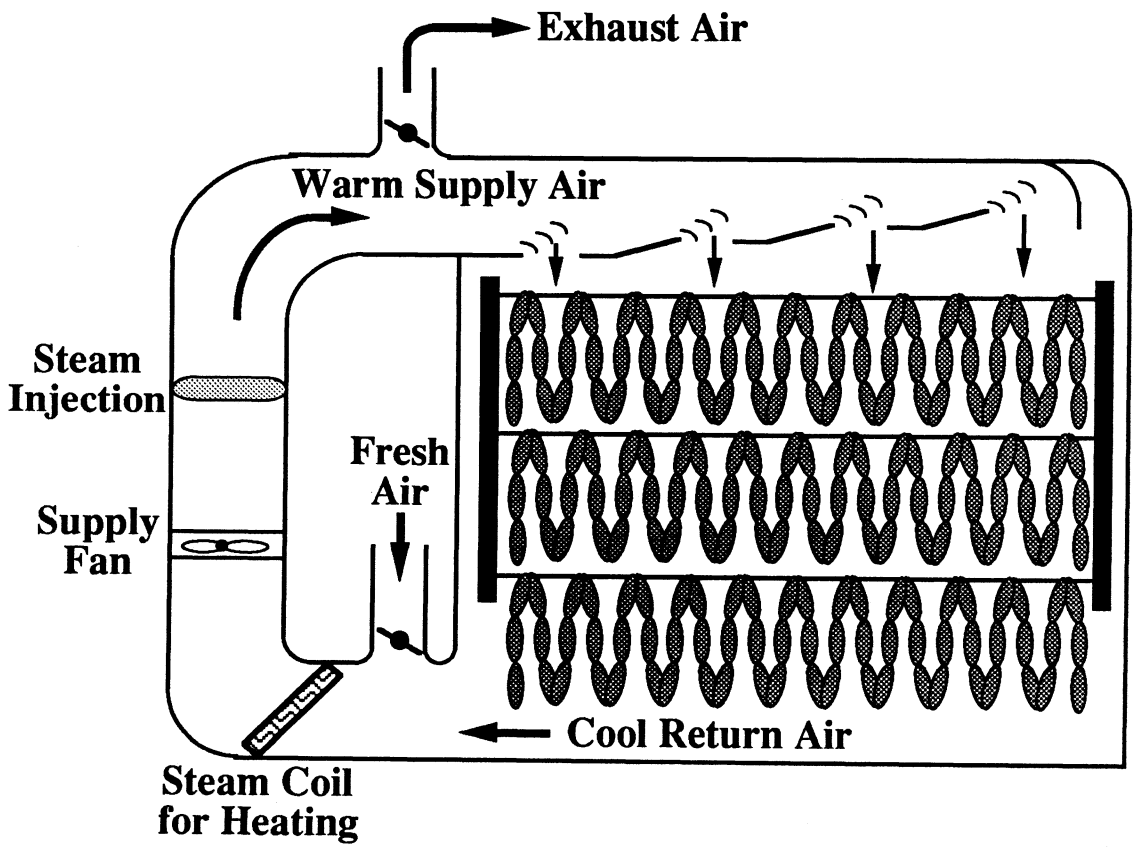

Figure 1.2 Convection Oven Cross Section

Each of the baking zones contains a steam heating coil and a steam injector. The steam coil provides a means of heating the oven air, while the steam injector is used to humidify the air. There is an exhaust duct and a fresh air duct which are used to further control air temperature and humidity in the oven. There is no means of controlling the conditions in the transition zones other than the flow of air from adjacent zones.

Air is supplied to the top of the zone and passes through a row of diffusers. The warm air then flows down past the product to the bottom of the zone where it is returned to the top of the zone. Such an air flow pattern can lead to variations in the free stream conditions from the top of the zone to the bottom of the zone. The top to bottom variation is a function of how full the zone is with product. The more full the zone is, the greater the top to bottom variation. For example, with the zone almost full, the free stream dry bulb temperature can differ from top to bottom by as much as 25°F. It is important to realize that this variation is indeed real and that it is not due to the zone control but rather a condition that arises due to the heat transfer to the product.

The water bath and chill zones are physically similar to the baking zones except instead of air ducts there are headers located at the top and middle of the zone. These headers have nozzles that spray the product with water or brine as the product passes through the zone. The water or brine is then collected at the bottom of the zone.

Each zone is equipped with sensors that keep track of the dry bulb temperature, the wet bulb temperature, fan speed, control valve positions, and exhaust and fresh air damper positions. The set point temperatures for each zone can be programmed into a controller. The controller maintains the zone temperatures at these set points with a dead band of plus or minus 1°F. The wet bulb temperature is controlled, for all zones,

at the bottom or return of the zone. The dry bulb temperature is controlled from the top of the smoke and drying zones, but from the bottom of the dry cook and final cook zones. See Fig. 1.1 for approximate temperature sensor locations.

I.4 Temperature Profile of Cooking Process

Now that a physical description of the hot dog process has been presented, it is time to look at the temperature history of a product as it is cooked. Fig. 1.3 is a graph of the local dry bulb, wet bulb, and product temperature as functions of time. The temperatures are plotted as functions of time because the temperature probes record and store temperature data as a function of time. The time scale shown on Fig. 1.3 can be related to the location of the probes within the process at any particular time by referring back to Fig. 1.1. For example, at time t = 18.0 min. the temperature probes are at the bottom of the second pass in the smoke zone. The method of correlating the time to the approximate probe location is not always accurate because sometimes the conveyance chain is stopped due to a processing problem with the cooking equipment. Even though the chain is stopped, the temperature probes keep recording temperatures as a function of time. Therefore, if the delay is long, the elapsed time will no longer agree to the probe position within the process.

The important characteristics of Fig. 1.3 need to be discussed. First the graph shows that there is quite a large variation in the dry bulb temperature from the top of the zone to the bottom of the zone. The variation in dry bulb temperature is caused by the heat transfer from the air as it flows past the product. The wet bulb temperature, on

the other hand, remains constant throughout the zone. The other important characteristic of Fig. 1.3 is that in the smoke zone and first half of the drying zone the product tends to follow the wet bulb temperature, which indicates a large rate of mass transfer. Then about 33 minutes into the cooking process the product breaks away from the wet bulb and begins to follow the dry bulb temperature. The cause for this break away, or change in thermal behavior, is linked to the formation of a hide at the product surface. The hide, it is believed, reduces the rate of mass transfer from the product and therefore changes the boundary conditions at the surface of the product.

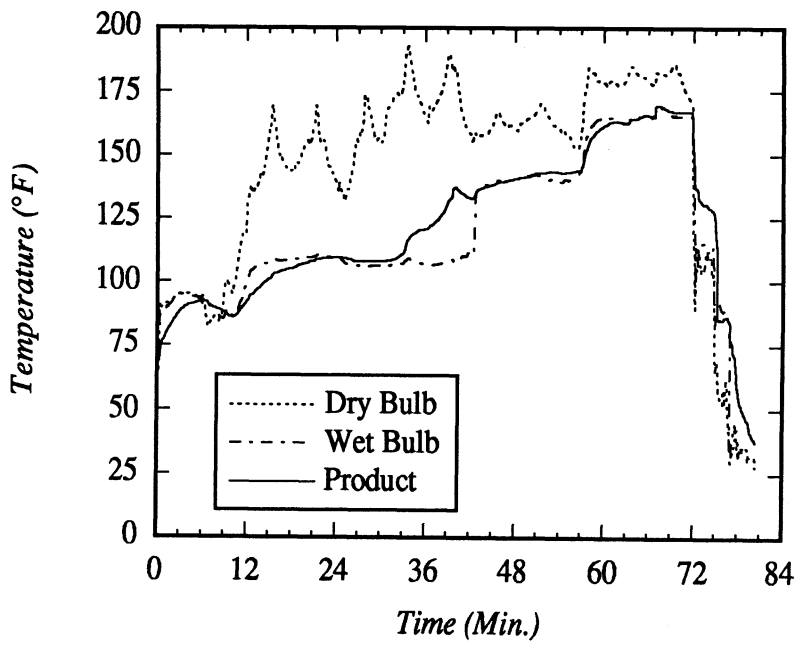

Figure 1.3 Process Temperature Profile

The importance of these two observations is that there are two distinct realms of thermal behavior for the hot dog. In the beginning of the process, before the hide

forms, the mass transfer rate from the product will be high and it will be important to include it in the model. Once the hide has formed, the mass transfer rate from the product will be greatly reduced and it will no longer be necessary to include mass transfer in the model. There is a period of time between these two limits during which time the hide is forming and the mass transfer rate from the product is decreasing. Modeling of this zone lies somewhere between the two limits.

I.5 Organization

Chapter II discusses the methods used to model the heat and mass transfer from the product surface and the finite-difference equations for transient conduction within the product.

Chapter III describes the design and results of experiments used to estimate the heat transfer coefficient in the baking zones of the cooking process, including a discussion of the uncertainty involved in measuring the heat transfer coefficient. The chapter concludes with a discussion of existing correlations for flow over surfaces which are used to estimate the heat transfer coefficient based on the average air velocity within the zone.

Chapter IV discusses two methods used to model the top to bottom variation in the dry bulb temperature for the zone. The first method involves using a sensible heat exchanger model to predict the exiting free stream dry bulb temperature for those zones where mass transfer from the product is negligible. The second method involves using a cooling tower model to predict the exiting free stream dry bulb temperature for those

zones where the mass transfer from the product is significant.

Chapter V compares the predicted results from the simulation program to actual data collected on the cooking process.

Chapter II

Development of the Product Model

Modeling of the product can be broken down into two main parts. The first part of the model accounts for the heat transfer at the surface of the product and conduction within the product. The second part of the model accounts for the mass transfer at the surface of the product and mass diffusion within the product. Although the heat and mass transfer are coupled to each other, they will be discussed separately. First a heat transfer model will be developed neglecting mass transfer, then the model will be modified to account for mass transfer at the surface of the product.

II.1 Transient Conduction Within the Product

The heat transfer model is developed in steps. First the governing partial differential equation is presented. Then various assumptions are made such that the governing equation is reduced to a transient one dimensional radial conduction problem with constant properties. Next the analytical solution is presented. And finally a finite-difference solution, which can later be modified to include mass transfer, is discussed.

II.1-1 Governing Partial Differential Equation

The first step in modeling transient conduction within the product is to select the appropriate coordinate system suited for the geometry of the product. The product, a hot dog in this case, can most easily be represented in cylindrical coordinates. Performing an energy balance on the control volume shown in Fig. 2.1 results in Eqn. 2.1-1.

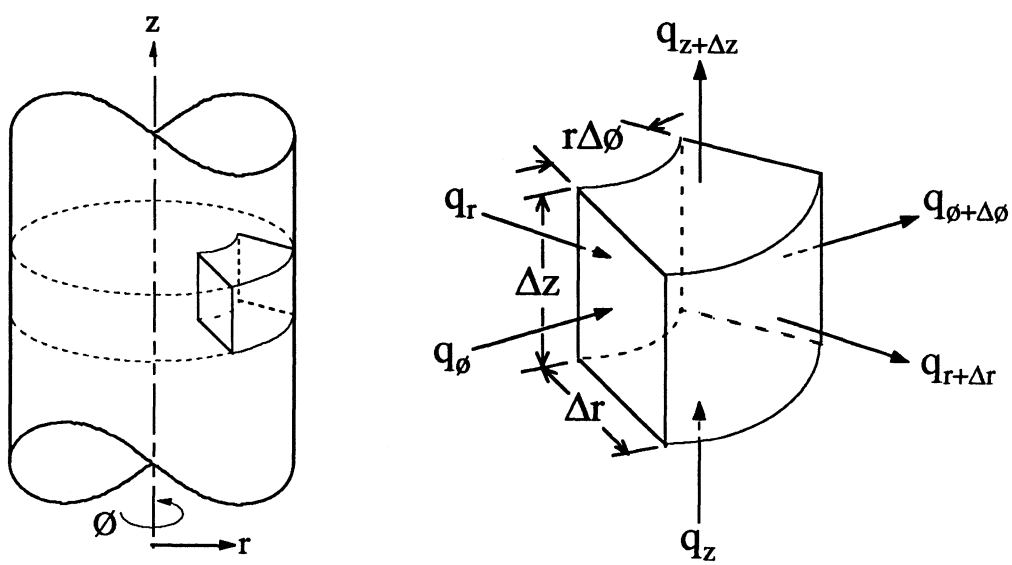

Figure 2.1 Control Volume in Cylindrical Coordinates [3]

$$\dot{E}_{store} = \dot{E}_{in} - \dot{E}_{out} + \dot{E}_{gen} \qquad (2.1\text{-}1)$$

where:

$\dot{E}_{in}$ = Rate of energy conducted into the control volume

$\dot{E}_{out}$ = Rate of energy conducted out of the control volume

$\dot{E}_{gen}$ = Rate of energy generation within the control volume

$\dot{E}_{store}$ = Rate of energy storage within the control volume

Substituting the appropriate rate equations into Eqn. 2.1-1 and taking the limit as Δz, Δr, and $\Delta \emptyset$ vanish results in the governing partial differential equation for transient conduction within a cylinder.

$$\rho C\frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial}{\partial \emptyset}\left(k\frac{\partial T}{\partial \emptyset}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + g''' \qquad (2.1\text{-}2)$$

where:

ρ = Mass density

k = Thermal conductivity

C = Specific heat

T = Temperature

t = Time

g''' = Energy generation rate per unit volume

II.1-2 Simplifying Assumptions

Three assumptions regarding conduction within a hot dog can be made which will allow for the simplification of Eqn. 2.1-2.

The first assumption is that the radial temperature gradients are much greater than the axial or circumferential temperature gradients. This implies that end effects are negligible, the heat transfer coefficient is uniform around the product surface, and the material properties are spatially uniform throughout the product. The governing partial differential equation can then be reduced to a one dimensional transient conduction problem in the radial direction.

The physical, chemical, and thermal changes that occur during cooking can have effects on the properties that influence the heat and mass transfer in the product. Thermal conductivity and specific heat can be temperature dependent. Such things as fat liquefaction, protein denaturation, and the phase change of water within the product can influence the specific heat, density, and thermal conductivity of the product [2,4]. The evaporation of water from the product can cause the physical size, shape, and density of the product to change. As water evaporates from the product surface, a concentration gradient is produced within the product causing water to diffuse to the product surface. As water diffuses to the surface it can bring with it proteins and fats which may affect the rate of mass transfer at the surface.

These are just a few of the many chemical, physical, and thermal changes that can occur during the cooking process, and their potential effects on the heat and mass transfer from the product. It would be extremely difficult, if not impossible, to accurately model all of these factors and properly account for their influence on the material properties of the product. Therefore, as a simplifying assumption, the effects that physical, chemical, and thermal changes have on the material properties of the product will be neglected. The material properties of the product can therefore be assumed to remain constant throughout the cooking process.

The final assumption is that there is no energy generation within the product. Although an energy generation term might be included to model the melting of fats or the freezing of water, these phase changes will be neglected.

With these three main assumptions the governing partial differential equation for transient conduction within a cylinder reduces to

$$\rho C\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial r^2} + \frac{k}{r}\frac{\partial T}{\partial r} \qquad (2.1\text{-}3)$$

In summary, Eqn. 2.1-3 represents one dimensional transient conduction within the product. The assumptions that are built into Eqn. 2.1-3 are that the radial temperature gradient is much larger than the axial or circumferential gradients, the material properties of the product remain constant throughout the cooking process, and that there is no energy generation within the product.

II.1-3 Boundary and Initial Conditions Defined

To solve Eqn. 2.1-3 for the temperature distribution within the product, it is necessary to define the physical conditions at the surface of the product and the temperature distribution within the product at the initial time. At the surface of the product there can be heat transfer due to convection and radiation. It will be assumed that the heat transfer due to radiation is negligible as each individual product will tend to "see" only other products at approximately the same temperature. Therefore the only heat transfer that occurs at the surface of the product is due to convection. The other boundary condition is that the temperature at the center of the product is finite. The initial condition is that the product is at some uniform temperature at time zero. The two boundary conditions and the initial condition can be represented by Eqn. 2.1-4, Eqn. 2.1-5, and Eqn. 2.1-6 respectively.

$$\text{B.C.} \quad \left(\frac{\partial T}{\partial r}\right)_{r=R} = \frac{h}{k}\left[T_\infty - T(R,t)\right] \tag{2.1-4}$$

$$\text{B.C.} \quad T(0,t) < \infty \tag{2.1-5}$$

$$\text{I.C.} \quad T(r,0) = T_i \tag{2.1-6}$$

where:

h = Convective heat transfer coefficient

T_∞ = Free stream temperature

R = Outer radius of cylinder

T_i = Initial temperature of cylinder

II.1-4 Analytical Solution

Eqn. 2.1-3 through Eqn. 2.1-6 can be solved analytically for the transient temperature distribution within the product. The solution is represented by Eqn. 2.1-7 through Eqn. 2.1-10.

$$\text{I.C.} \quad T(r,0) = \frac{T(r,t) - T_\infty}{T_i - T_\infty} = 2Bi \sum_{n=1}^{\infty} \frac{J_0(\lambda_n r)\, \mathrm{Exp}(-\alpha\lambda_n^2 t)}{\left[(\lambda_n R)^2 + Bi^2\right] J_0(\lambda_n R)} \tag{2.1-7}$$

$$\lambda_n R = \frac{Bi J_0(\lambda_n R)}{J_1(\lambda_n R)} \tag{2.1-8}$$

$$Bi = \frac{hR}{k} \tag{2.1-9}$$

$$\alpha = \frac{k}{\rho C} \tag{2.1-10}$$

where:

Bi = Biot number

α = Thermal Diffusivity

J_0 = Bessel function of first kind of order zero

J_1 = Bessel function of first kind of order one

λ = Eigenvalue

The evaluation of Eqn. 2.1-7 involves finding the eigenvalues which satisfy the eigenfunction given by Eqn. 2.1-8. Although Eqn. 2.1-7 is not difficult to use it does not readily lend itself to modifications for variable free stream temperatures. Therefore a finite-difference solution of Eqn. 2.1-3 through Eqn. 2.1-6 will be developed under the same assumptions as the exact solution. The finite-difference solution can then be modified to account for a variable free stream temperature and mass transfer at the surface.

II.1-5 Finite-Difference Approximation

The finite-difference representation of Eqn. 2.1-3, for the interior node shown in Fig. 2.2, will be developed using Taylor series expansion to approximate the partial derivative terms. The spacial derivatives are approximated by the central differences represented by Eqn. 2.1-11 and Eqn. 2.1-12.

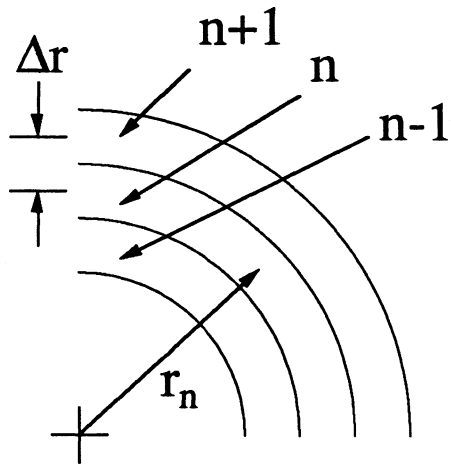

Figure 2.2 Interior Node for Finite-Difference Approximation

$$\frac{\partial T}{\partial r} = \frac{T^p_{n+1} - T^p_{n-1}}{2\Delta r} + O(\Delta r)^2 \tag{2.1-11}$$

$$\frac{\partial^2 T}{\partial r^2} = \frac{T^p_{n+1} - 2T^p_n + T^p_{n-1}}{\Delta r^2} + O(\Delta r)^2 \tag{2.1-12}$$

where:

p = Index used to indicate time

n = Index used to indicate node position within product

O = Order of magnitude of error due to neglecting higher order terms

A forward difference approximation, represented by Eqn. 2.1-13, is used to approximate the time derivative.

$$\frac{\partial T}{\partial t} = \frac{T^{p+1}_n - T^p_n}{\Delta t} + O(\Delta t) \tag{2.1-13}$$

Substituting Eqn. 2.1-11, Eqn. 2.1-12, and Eqn. 2.1-13, neglecting higher order terms, into Eqn. 2.1-3 and solving for the temperature of node n at time p+1 results in

$$T_n^{p+1} = \frac{k\Delta t}{\rho C}\left[\left(\frac{T_{n+1}^p - 2T_n^p + T_{n-1}^p}{\Delta r^2}\right) + \frac{1}{r_n}\left(\frac{T_{n+1}^p - T_{n-1}^p}{2\Delta r}\right)\right] + T_n^p \qquad (2.1\text{-}14)$$

Eqn. 2.1-14 is explicit because the unknown temperature at node n and time p+1 is determined exclusively by known node temperatures at time p. Solving for the unknown node temperature is therefore straight forward. The temperature of all the interior nodes are known at time zero and therefore Eqn. 2.1-14 can be applied to each interior node to find its temperature at time p+1. The temperature distribution can then be determined by "marching" through time at intervals of Δt.

The solution represented by Eqn. 2.1-14 is still incomplete as the conditions at the surface and center of the product have not been included. To account for the boundary condition at the surface of the product it is necessary to perform an energy balance, represented by Eqn. 2.1-15, on the surface node shown in Fig. 2.3.

$$\dot{E}_{store} = q_{conv,in} + q_{cond,in} \qquad (2.1\text{-}15)$$

The rate of energy transfer due to conduction can be calculated by Fourier's Law shown in Eqn. 2.1-16.

$$q_{cond} = -kA_{N-1}\left(\frac{\partial T}{\partial r}\right) \qquad (2.1\text{-}16)$$

where:

A_{N-1} = Surface area of the plane midway between node N and N-1

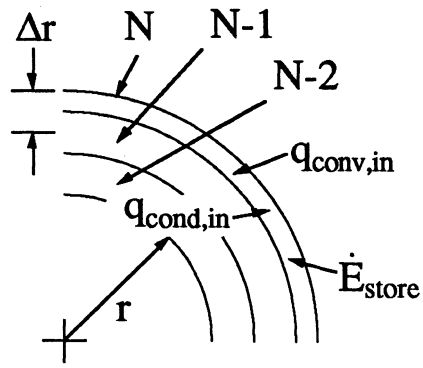

Figure 2.3 Surface Node for Finite-Difference Approximation

The rate of energy transfer due to convection at the product surface can be calculated by Newton's Law of cooling shown below.

$$q_{conv} = hA_N(T_\infty - T_N) \tag{2.1-17}$$

where:

A_N = Surface area of product

Substituting Fourier's Law, Newton's Law of Cooling, and a mathematical expression for the rate of energy storage into Eqn. 2.1-15, results in

$$\rho V_N C\frac{\partial T}{\partial t} = hA_N(T_\infty - T_N) - kA_{N-1}\frac{\partial T}{\partial r} \tag{2.1-18}$$

where:

V_N = Volume of Surface Node N

To achieve the same order of accuracy for the surface node as for the interior nodes, it is necessary to use a backward difference that involves the nodes N, N-1, and N-2. The spacial derivative is then represented by

$$\frac{\partial T}{\partial r} = \frac{T^p_{N-2} - 4T^p_{N-1} + 3T^p_N}{2\Delta r} + O(\Delta r)^2 \tag{2.1-19}$$

Substituting Eqn. 2.1-13 and Eqn. 2.1-19, neglecting higher order terms, into Eqn. 2.1-18 and solving for the temperature of node N at time p+1 results in

$$T^{p+1}_N = \frac{\Delta t}{\rho C V_N}\left[hA_N\left(T_\infty - T^p_N\right) - kA_{N-1}\left(\frac{T^p_{N-2} - 4T^p_{N-1} + 3T^p_N}{2\Delta r}\right)\right] + T^p_N \tag{2.1-20}$$

To account for the boundary conditions at the center of the product it is necessary to perform an energy balance, represented by Eqn. 2.1-21, on the center node shown in Fig. 2.4.

$$\dot{E}_{store} = -q_{cond,out} \tag{2.1-21}$$

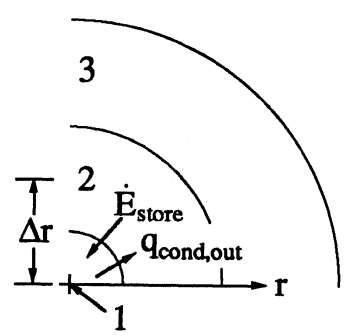

Figure 2.4 Center Node for Finite-Difference Approximation

Substituting Fourier's Law and a mathematical expression for the energy storage within the center node into Eqn. 2.1-21, results in

$$\rho C V_1 \frac{\partial T}{\partial t} = k A_1 \frac{\partial T}{\partial r} \tag{2.1-22}$$

Again, to achieve the same order of accuracy for the center node as for the surface and interior nodes, it is necessary to use a forward difference approximation of the spacial derivative that involves nodes 1, 2, and 3 which is represented by

$$\frac{\partial T}{\partial r} = \left(\frac{-3T_1^p + 4T_2^p - T_3^p}{2\Delta r} \right) + O(\Delta r)^2 \tag{2.1-23}$$

Substituting Eqn. 2.1-13 and Eqn. 2.1-23, neglecting higher order terms, into Eqn. 2.1-22 and solving for the temperature of node 1 at time p+1 results in

$$T_1^{p+1} = \frac{\Delta t}{\rho C V_1} \left[k A_1 \left(\frac{-3T_1^p + 4T_2^p - T_3^p}{2\Delta r} \right) \right] + T_1^p \tag{2.1-24}$$

Eqn. 2.1-14, Eqn. 2.1-20, and Eqn. 2.1-24 combine to form a system of finite-difference equations that can be solved for the transient temperature profiles within the product.

II.2 Product Model Modified for Mass Transfer

Mass transfer is a rather complicated subject, therefore a detailed discussion of the model modifications which account for mass transfer at the surface will be presented.

The goal of this section is to develop a method for calculating the mass transfer from the product surface based on the heat transfer results. The general approach will be to present the governing equations and then, through the use of various approximations and assumptions, simplify these equations to achieve the desired form of the solution.

II.2-1 Boundary Layer Assumptions

Fluid flow over the product is a very complicated problem. But through the use of boundary layer theory, the problem can be greatly simplified. Prandtl discovered that for most applications the influence of viscosity is confined to an extremely thin region next to the body surface [5]. In general, the flow of air over the product can be represented by Fig. 2.5.

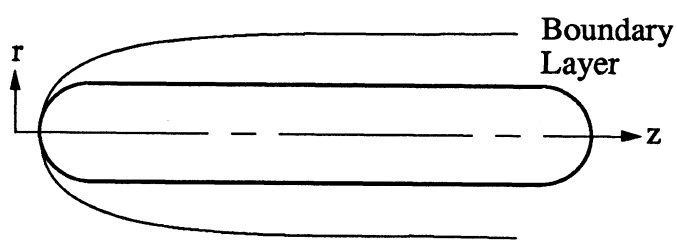

Figure 2.5 Fluid Flow Over a Hot Dog (Not to Scale)

The basic assumption of boundary layer theory is that there is a thin layer of fluid immediately adjacent to the body surface and the velocity at the surface is zero. The hydrodynamic boundary layer is defined as the region in which the fluid velocity changes from its free stream value to zero at the body surface [5]. The concept of a boundary layer can be expanded to include heat and mass transfer. The thermal bound-

ary layer is defined as the region in which the fluid temperature changes from its free stream value to the temperature at the body surface. The concentration boundary layer is defined as the region where the fluid mass concentration changes from its free stream value to the value at the body surface. The simplifications brought about by the concept of a boundary layer is that the boundary layer lends itself to a relatively simple analysis by the very fact of its thinness relative to the dimensions of the body.

Boundary layer theory therefore allows for the modeling of flow over the curved surface of the product to be approximated by flow over a flat plate. Fig. 2.6 shows the assumed product geometry and coordinate system which will be used throughout the rest of the mass transfer derivation.

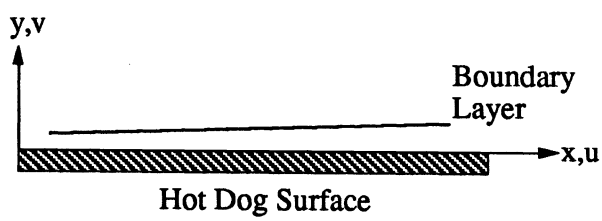

Figure 2.6 Boundary Layer Flow Over a Flat Plate

II.2-2 Mass Flux at the Product Surface

Initially the product surface can be considered totally wet. That is, enough water diffuses to the product surface such that the air above the product surface can be considered saturated. Fig. 2.7 represents the mass flux that is occurring across the product

surface and the water supplied from underneath from the product.

The 0 state is defined as a surface an infinitesimal distance inside the boundary layer where the water is in the vapor state [5]. The T surface is used as a reference state for both energy and mass transfer and can be thought of as a surface a finite distance inside the product where there are no gradients in either concentration or temperature [5]. Therefore, all mass and energy transferred across the T surface is due to the flow of fluid across the surface. It is convenient to define a reference concentration of the water within the product, $m_{w,T}$, in terms of the total mass flux, $\dot{m}''$.

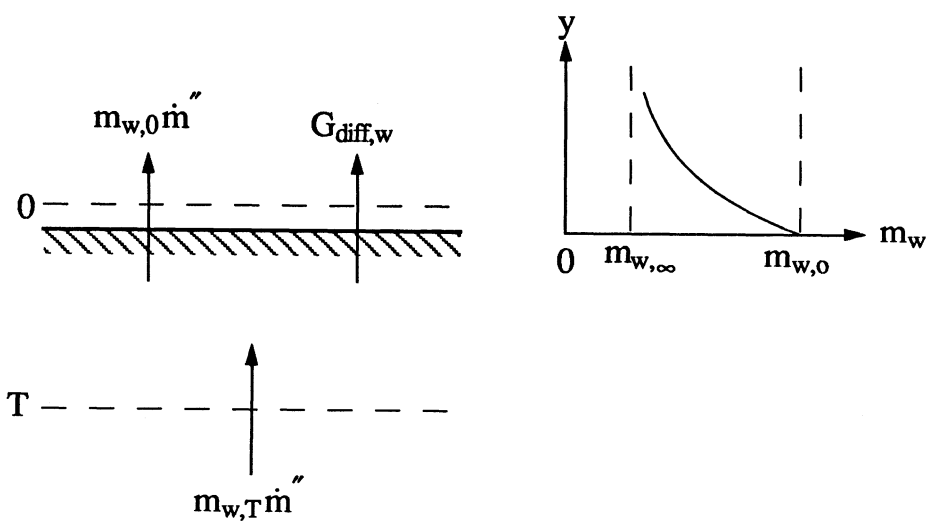

Figure 2.7 Mass Flux of Water From the Product Surface

The mass flux across the liquid-vapor interface involves two terms. The first term, $m_{w,0}\dot{m}''$, is the mass flux of water vapor from the product. It is written as the product of the total mass flux, $\dot{m}''$, and the mass concentration of water at the 0 surface, $m_{w,0}$. The second term, $G_{diff,w}$, represents the mass flux across the liquid-vapor interface due to

concentration gradients within the boundary layer. The concentration gradient arises due to the difference in the mass concentration of water at the product surface as compared to the free stream mass concentration. A typical concentration profile within the boundary layer is pictured in Fig. 2.7.

Performing a mass balance for water on the control volume shown in Fig. 2.7 results in

$$m_{w,T}\dot{m}'' = m_{w,0}\dot{m}'' + G_{diff,w} \tag{2.2-1}$$

where:

m_w = Mass concentration of water in air (kg_w/kg_{mix})

$\dot{m}''$ = Total mass flux from product surface

$G_{diff,w}$ = Rate of mass transfer of water due to diffusion within the boundary layer

Since water is the only substance being transferred from the product surface, the two mass flux terms on the right hand side of Eqn. 2.2-1 must add to equal the total mass flux of water from the product. Therefore the reference mass concentration of water at the T surface, $m_{w,T}$, must equal 1.

The rate of mass flux due to diffusion can be calculated using Fick's Law, given by Eqn. 2.2-2. Fick's Law is exact for mass diffusion of a binary mixture in the absence of other potential gradients [5].

$$G_{diff,w} = \left[-\rho D_w\left(\frac{\partial m_w}{\partial y}\right)\right] \tag{2.2-2}$$

where:

D_w = Mass diffusivity of water in air

m_w = Mass concentration of water

Substituting Eqn. 2.2-2 into Eqn. 2.2-1 and solving for the net mass flux from the product surface results in

$$\dot{m}'' = \frac{\left[\rho D_w \left(\frac{\partial m_w}{\partial y}\right)\right]_0}{m_{w,0} - m_{w,T}} \tag{2.2-3}$$

where $\dot{m}_{w,T}$ equals unity.

Eqn. 2-2.3 could be used to calculate the mass flux from the product if the mass concentration gradient at the product surface were known. The concentration gradient at the product surface can be determined by solving the system of differential equations for the boundary layer represented by Eqn. 2.2-4, Eqn. 2.2-5, and Eqn. 2.2-6.

Continuity Equation:

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \tag{2.2-4}$$

Momentum Equation:

$$\rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial v}{\partial y} - \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) + \frac{\partial p}{\partial x} = 0 \tag{2.2-5}$$

Mass Diffusion Equation:

$$\rho u \frac{\partial m_w}{\partial x} + \rho v \frac{\partial m_w}{\partial y} - \frac{\partial}{\partial y}\left(\rho D_w \frac{\partial m_w}{\partial y}\right) = 0 \qquad (2.2\text{-}6)$$

where:

u = Velocity component in the x-direction

v = Velocity component in the y-direction

p = Pressure

Eqn. 2.2-4, Eqn. 2.2-5, and Eqn. 2.2-6 have been simplified by the normal assumptions of boundary layer theory. The mass diffusion equation has been simplified for diffusion of an inert substance which, for small concentrations, is a valid assumption for diffusion of water in air [5].

II.2-3 Energy Flux at the Product Surface

Fig. 2.8 represents the energy flux that is occurring at the 0 surface inside the boundary layer and the energy flux across the reference surface T. In the energy balance it is necessary to consider both the air and water in the boundary layer because air diffuses toward the product surface and water diffuses away from the product surface.

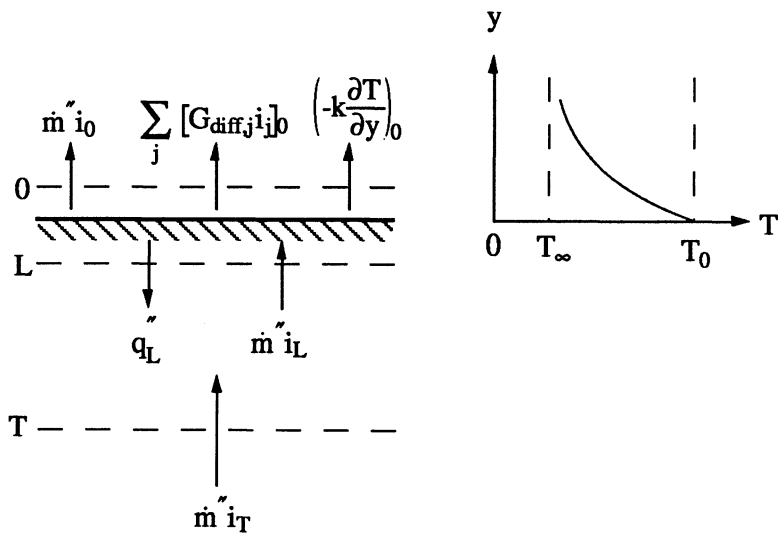

Figure 2.8 Energy Flux From Product Surface

The energy flux across the control surface at state 0, which is located in the boundary layer, involves three terms. The first term, $\dot{m}''i_0$, represents the energy flux at the 0 surface. It can be written as the product of the total mass flux, $\dot{m}''$, and the mixture enthalpy at the 0 surface, i_0. The second term, $\left(-k\frac{\partial T}{\partial y}\right)_0$, represents the energy flux due to conduction of energy to the boundary layer at the product surface. The third term, $\sum_j [G_{diff,j}i_j]_0$, represents the energy flux due to the mass flux caused by the concentration gradients within the boundary layer. A typical temperature profile within the thermal boundary layer is shown in Fig. 2.8.

The L state is defined at a surface an infinitesimal distance inside the liquid side of the liquid-vapor interface where all water is in the liquid state [5]. The T state is defined as a surface an finite distance inside the product where there are no concentration or temperature gradients therefore it can be thought of as an adiabatic surface [5]. Note

that an adiabatic surface does not need to really exist within the product since the T state, used in the heat and mass transfer derivations, is a reference state only.

There are two energy fluxes across the L surface. The first term, q_L'', represents the energy flux within the solid that is due to conduction only. The second term, $\dot{m}''i_L$, represents the energy flux due to the net mass flux across the L surface, where i_L is the liquid enthalpy.

Eqn. 2.2-7 represents an energy balance on the control volume bounded by the L and T surfaces shown in Fig. 2.8.

$$q_L'' = \dot{m}''(i_L - i_T) \tag{2.2-7}$$

where:

q_L'' = Energy flux within the solid due to conduction only

i_L = Enthalpy of water at the L surface

i_T = Reference state enthalpy at the T surface

Eqn. 2.2-7 can then be rearranged to define a reference state enthalpy, i_T, given by

$$i_T = i_L - \frac{q_L''}{\dot{m}''} \tag{2.2-8}$$

Performing an energy balance on the control volume bounded by the 0 and T surfaces shown in Fig. 2.8, substituting Fick's Law from Eqn. 2.2-2, and solving for the net mass flux of water from the product surface results in

$$\dot{m}'' = \frac{\left(k\frac{\partial T}{\partial y}\right)_0 + \sum_j \left(\rho D_j i_j \frac{\partial m_j}{\partial y}\right)_0}{(i_0 - i_T)} \qquad (2.2\text{-}9)$$

where:

i_0 = Enthalpy of the air water-vapor mixture at the 0 surface

Once again the mass flux from the product surface could be calculated using Eqn. 2.2-9 if the temperature and concentration gradients at the product surface were known. To calculate the temperature gradient at the product surface it would again be necessary to solve the system of boundary layer differential equations except now the energy equation for the boundary layer, given by Eqn. 2.2-10, must be solved also.

Energy Equation:

$$\rho u\frac{\partial i}{\partial x} + \rho v\frac{\partial i}{\partial y} - \frac{\partial}{\partial y}\left(\sum_j \rho D_j i_j \frac{\partial m_j}{\partial y}\right) - \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) = 0 \qquad (2.2\text{-}10)$$

To realize the importance of the surface energy balance, it is necessary to look at Eqn. 2.2-9 in more detail.

II.2-4 Lewis Number of Unity Assumption

The first term in the numerator of Eqn. 2.2-9 can be rewritten in a slightly different form as

$$\left(k\frac{\partial T}{\partial y}\right)_0 = \left[\frac{k}{C_p}\left(C_p\frac{\partial T}{\partial y}\right)\right]_0 = \left[\frac{k}{C_p}\sum_j\left(m_j C_{p,j}\frac{\partial T}{\partial y}\right)\right]_0 \tag{2.2-11}$$

where:

C_p = Specific heat of mixture at constant pressure

$C_{p,j}$ = Specific heat of component j at constant pressure

Eqn. 2.2-11 represents the conduction of energy within the boundary layer at the 0 state where both water vapor and air are present. C_p represents the specific heat of the mixture which, as shown in Eqn. 2.2-11, can be represented by the weighted sum of the specific heats of the individual components that make up the gas mixture. For the binary gas mixture being considered, the two components are air and water.

If water vapor and air are assumed to behave as ideal gases, then the enthalpy relation shown in Eqn. 2.2-12 can be employed for each component.

$$di_j = C_{p,j}dT \tag{2.2-12}$$

Substituting Eqn. 2.2-12 into Eqn. 2.2-11 results in

$$\left(k\frac{\partial T}{\partial y}\right)_0 = \left[\frac{k}{C_p}\sum_j\left(m_j\frac{\partial i_j}{\partial y}\right)\right]_0 \tag{2.2-13}$$

Eqn. 2.2-13 can be substituted back into Eqn. 2.2-9 which results in the modified mass flux equation represented by

$$\dot{m}'' = \frac{\left[\frac{k}{C_p}\sum_j\left(m_j\frac{\partial i_j}{\partial y}\right)\right]_0 + \sum_j\left(\rho D_j i_j\frac{\partial m_j}{\partial y}\right)_0}{(i_0 - i_T)} \tag{2.2-14}$$

At this point no simplifications have been made. But for the special case of $\frac{k}{C_p} = \rho D_j$ for each component in the boundary layer fluid, the two terms in the numerator of Eqn. 2.2-14 could be combined and the energy equation would be greatly simplified.

The ratio defined by Eqn. 2.2-15 is known as the Lewis number, Le.

$$Le = \frac{\rho D_j}{\left(\frac{k}{C_p}\right)} \tag{2.2-15}$$

For air water-vapor mixtures, the Lewis number is approximately unity [5]. Assuming a Lewis number of unity therefore allows Eqn. 2.2-14 to be rewritten as

$$\dot{m}'' = \frac{\left[\frac{k}{C_p}\sum_j \left(m_j\frac{\partial i_j}{\partial y} + i_j\frac{\partial m_j}{\partial y}\right)\right]_0}{(i_0 - i_T)} \tag{2.2-16}$$

The numerator of Eqn. 2.2-16 can be further simplified as

$$\frac{k}{C_p}\sum_j \left(m_j\frac{\partial i_j}{\partial y} + i_j\frac{\partial m_j}{\partial y}\right) = \frac{k}{C_p}\sum_j \frac{\partial}{\partial y}(m_j i_j) = \frac{k}{C_p}\frac{\partial i}{\partial y} \tag{2.2-17}$$

where i is the mixture enthalpy.

Substituting Eqn. 2.2-17 back into Eqn. 2.2-16 results in

$$\dot{m}'' = \frac{\left(\frac{k}{C_p}\frac{\partial i}{\partial y}\right)_0}{(i_0 - i_T)} \tag{2.2-18}$$

Although Eqn. 2.2-18 is a simplified form of the energy equation, it still does not allow one to calculate the mass flux without solving the boundary layer equations. The importance of the Lewis number of unity assumption, which will ultimately eliminate the need to calculate a gradient at the product surface, comes from applying the Lewis number of unity approximation to the boundary layer energy equation and comparing the resulting equation to the mass diffusion equation. These two equations will then lead to the definition of a mass transfer conductance and a mass transfer driving force as discussed in the next section.

II.2-5 Mass Transfer Conductance and Driving Force

The boundary layer energy equation, represented by Eqn. 2.2-10, can be simplified using a Lewis number of unity approximation to yield

$$\rho u\frac{\partial i}{\partial x} + \rho v\frac{\partial i}{\partial y} - \frac{\partial}{\partial y}\left(\frac{k}{C_p}\frac{\partial i}{\partial y}\right) = 0 \qquad (2.2\text{-}19)$$

Eqn. 2.2-20 is the boundary layer mass diffusion equation for the water vapor.

$$\rho u\frac{\partial m_w}{\partial x} + \rho v\frac{\partial m_w}{\partial y} - \frac{\partial}{\partial y}\left(\rho D_W\frac{\partial m_w}{\partial y}\right) = 0 \qquad (2.2\text{-}20)$$

The Lewis number of unity assumption has transformed the energy equation into the same differential equation as the mass diffusion equation. Both the energy and mass diffusion equations are therefore represented by

$$\rho u\frac{\partial P}{\partial x} + \rho v\frac{\partial P}{\partial y} - \frac{\partial}{\partial y}\left(\lambda\frac{\partial P}{\partial y}\right) = 0 \qquad (2.2\text{-}21)$$

where $P = m_w$ or i and $\lambda = \rho D_w$ or $\frac{k}{C_p}$, as appropriate.

Analogous to Eqn. 2.2-3 and Eqn. 2.2-18, the mass flux can be represented by

$$\dot{m}'' = \frac{\left(\lambda\frac{\partial P}{\partial y}\right)_0}{(P_0 - P_T)} \qquad (2.2\text{-}22)$$

The general profile of the conserved properties, i or m_w, in the boundary layer is shown in Fig 2.7 and Fig. 2.8.

A mass transfer conductance g, given by Eqn. 2.2-23, is defined in terms of the gradient of the conserved property P, at the surface and the potential difference between the value of the conserved property P, at the free stream and the surface.

$$g \equiv \frac{\left[\lambda\left(\frac{\partial P}{\partial y}\right)\right]_0}{P_\infty - P_0} \qquad (2.2\text{-}23)$$

Similar to convection heat transfer relations, the mass flux can be related to a conductance and a potential difference. Solving for the gradient from Eqn. 2.2-23 and substituting into Eqn. 2.2-22 yields

$$\dot{m}'' = g\left(\frac{P_\infty - P_0}{P_0 - P_T}\right) \qquad (2.2\text{-}24)$$

The potential is then given by

$$B = \frac{P_\infty - P_0}{P_0 - P_T} \tag{2.2-25}$$

Now that the general form of the mass transfer conductance and mass transfer driving force have been established it is possible to apply these definitions to previously derived mass flux equations.

II.2-6 Simplified Mass Flux Equations

It is assumed that the specific heat of the air water-vapor mixture is the same everywhere, which is a valid assumption for small mass concentrations. From Eqn. 2.2-24 it is apparent that the mass flux can be written in terms of a mass transfer conductance with enthalpy as the driving force.

$$\dot{m}'' = g\left(\frac{i_\infty - i_0}{i_0 - i_T}\right) \tag{2.2-26}$$

The similarity between Eqn. 2.2-21 and the boundary layer energy equation without mass transfer leads to a relation between the heat transfer coefficient and the mass transfer conductance, as shown in Eqn. 2.2-27.

$$g = \frac{h}{C_p} \tag{2.2-27}$$

A correction to the mass transfer coefficient can be made for large rates of mass transfer. If the flow of air over the product is assumed to be turbulent, then for the transpired, turbulent boundary layer with all fluid properties assumed constant, the mass transfer coefficient can be calculated using [5].

$$g = g^*\left[\frac{\ln(1 + B)}{B}\right] \tag{2.2-28}$$

where $g^* = \frac{h}{C_p}$

Eqn 2.2-26 and Eqn. 2.2-27 could be used to solve for the mass flux from the product surface without having to solve the boundary layer equations if the value of i_T were known. But i_T, defined by Eqn. 2.2-8, is a function of q_L'' which is still unknown. Therefore it is necessary to refer back to Eqn. 2.2-3. The Lewis number of unity and the fact that Eqn. 2.2-21 represents both the energy and mass diffusion equations allows one to write the mass flux in terns of a mass transfer conductance with mass concentrations as the driving force as

$$\dot{m}'' = g\left(\frac{m_{w,\infty} - m_{w,0}}{m_{w,0} - m_{w,T}}\right) \tag{2.2-29}$$

where $\dot{m}_{w,T}$ equals unity because water is the only substance being transferred.

Now that the mass flux from the product can be calculated it is possible to calculate the heat flux to the product.

II.2-7 Heat Flux to Product

To calculate the temperature profile within the product it is necessary to know how much of the energy transferred at the product surface is available to change the temperature of the product and how much energy goes into the phase change of water.

In Sec. II.2-3 the quantity q_L'' was defined as the energy flux within the solid that is due to conduction only. Therefore q_L'' is that portion of the total energy flux at the product surface which is available to cause a sensible change in the product temperature. To calculate the transient temperature profile within the product it is therefore necessary to calculate q_L''.

If Eqn. 2.2-25 is written in terms of enthalpies and solved for i_T, then the result is

$$i_T = i_0 - \frac{i_\infty - i_0}{B} \tag{2.2-30}$$

Substituting the definition of i_T, represented by Eqn. 2.2-8, into Eqn. 2.2-30 and solving for q_L'', results in

$$q_L'' = \dot{m}''\left(i_L - i_0 + \frac{i_\infty - i_0}{B}\right) \tag{2.2-31}$$

Eqn. 2.2-31 can be further simplified as

$$q_L'' = \dot{m}''(i_L - i_0) + \dot{m}''\left(\frac{i_\infty - i_0}{B}\right) \tag{2.2-32}$$

Now if Eqn. 2.2-29 is substituted into the second term in Eqn. 2.2-32 then the result is

$$q_L'' = \dot{m}''(i_L - i_0) + \frac{h}{C_p}(i_\infty - i_0) \tag{2.2-33}$$

The enthalpy of water vapor can be defined as

$$i_W = C_p(T - T_{ref}) + i_{fg} \tag{2.2-34}$$

The reference state enthalpy for Eqn. 2.2-34 has been chosen such that the enthalpy of liquid water at T_{ref} is zero. If the reference state enthalpy is chosen such that $i_{w,0} = 0$ at $T_{ref} = T_0$ then $i_L = -i_{fg,0}$ and $i_\infty = C_p(T_\infty - T_0)$. Substituting these definitions into Eqn. 2.2-33 results in

$$q_L'' = h(T_\infty - T_0) - \dot{m}'' i_{fg,0} \tag{2.2-35}$$

where $\dot{m}''$ can be calculated using Eqn. 2.2-29.

Eqn. 2.2-35 represents the conduction heat flow at the product surface. The finite-difference equation for the surface node, Eqn. 2.1-20, will have to be modified to account for the changes in the boundary condition due to mass transfer.

II.2-8 Surface Node Modified for Mass Transfer

Fig. 2-9 represents the modified boundary conditions at the surface of the product. Performing an energy balance on the surface node results in

$$\dot{E}_{store} = q_L + q_{cond,in} \tag{2.2-36}$$

where $q_L = q_L'' A_N$

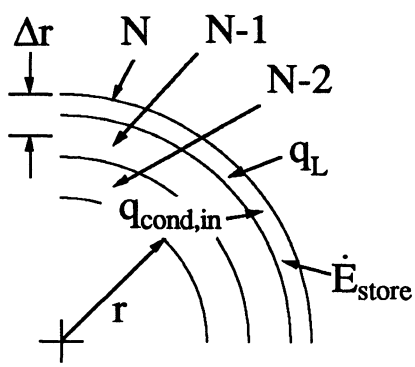

Figure 2.9 Surface Node With Mass Transfer

Substituting Eqn. 2.1-16, Eqn. 2.2-35 , and a mathematical expression for the rate of energy storage into Eqn. 2.2-36 results in

$$\rho V_N C \frac{\partial T}{\partial t} = A_N\left[h(T_\infty - T_N) - \dot{m}'' i_{fg,N}\right] - A_{N-1} k \left(\frac{\partial T}{\partial r}\right) \tag{2.2-37}$$

where $i_{fg,N} = i_{fg,0}$ since node N is at the surface of the product where surface 0 is defined.

Substituting Eqn. 2.1-13 and Eqn. 2.1-19 into Eqn. 2.2-37 and solving for the temperature of the node N at time p + 1 yields

$$T_N^{p+1} = \frac{\Delta t}{\rho C V_N}\left\{A_N\left[h\left(T_\infty - T_N^p\right) - \dot{m}'' i_{fg,N}\right] - k A_{N-1}\left(\frac{T_{N-2}^p - 4T_{N-1}^p + 3T_N^p}{2\Delta r}\right)\right\} + T_N^p \tag{2.2-38}$$

where $\dot{m}''$ is calculated using Eqn. 2.2-29.

So far the discussion of mass transfer has been limited to the mass flux from the product surface and nothing has been said about the transfer of water within the product. The next section in the chapter will briefly discuss mass diffusion within the product.

II.2-9 Mass Diffusion Within the Product

It was assumed in Sec. II.2-2 that the product surface can be considered to be totally wet. This assumes that enough water diffuses to the surface of the product such that the air at the surface is saturated. As mass transfer from the product continues, the product begins to dry out since there is a finite supply of water within the product and it takes time for the water to diffuse to the surface from the interior. The mass flux from the product surface then becomes a function of the rate at which water diffuses to the product surface.

The ratio of the resistance to mass diffusion within the product to the resistance to convection of mass across the fluid boundary is referred to as the mass transfer Biot number [1]. This ratio is analogous to the heat transfer Biot number which is defined as the ratio of the resistance to conduction within the product to the resistance to convection of heat across the fluid boundary. The similarity of these two ratios can better be understood by referring to Eqn. 2.2-39 through Eqn. 2.2-41.

$$Bi = \frac{R_{cond}}{R_{conv}} \tag{2.2-39}$$

$$Bi_{heat} = \frac{hR}{k} \tag{2.2-40}$$

$$Bi_{mass} = \frac{hR}{C_p \rho D_{AB}} \tag{2.2-41}$$

where:

D_{AB} = Mass diffusivity of component A in B

The magnitude of the mass transfer Biot number indicates whether the diffusion of mass within the product, the convection of mass from the product, or a combination of both will govern the mass flux from the product. If the Biot number is much less than one, then the mass flux from the product will be governed by the rate of convection of mass from the product surface. If the Biot number is much larger than one, then the mass flux from the product will be governed by the rate of diffusion within the product. If the Biot number is approximately equal to one, then the rate of convection of mass from the product surface is approximately equal to the rate of diffusion of mass within the product. The mass transfer Biot number for water in meat is on the order of magnitude of 100. Therefore, the mass flux from the product will be limited by the rate of diffusion of mass to the product surface.

The calculation of the mass diffusion within the product involves solving the equation of continuity for water within the product, represented by

$$\frac{1}{D_{w,hd}} \frac{\partial m_w}{\partial t} = \frac{\partial^2 m_w}{\partial r^2} + \frac{1}{r} \frac{\partial m_w}{\partial r} \tag{2.2-42}$$

where $D_{w,hd}$ = Mass diffusivity of water in hot dog meat

Eqn. 2.2-42 has been simplified for constant mass density and diffusivity. The important thing to note about Eqn. 2.2-42 is its similarity to Eqn. 2.1-3. The similarity

between these two equations indicates that the transient mass diffusion equation can be solved in the same manner as the transient conduction equation.

Even though Eqn. 2.2-42 can be solve in a similar manner to Eqn. 2.1-3, the solution of mass diffusion within the product is beyond the scope of this first study of heat and mass transfer from the product. It would be necessary to modify the product model to include mass diffusion within the product. The modification would involve calculating the humidity ratio of the air at the product surface rather than assuming the air is saturated. But more importantly, when mass diffusion within the product is included, it will introduce concentration variations within the product which will lead to the need to include variable properties. Therefore the model of the product will ignore mass diffusion within the product and it will be assumed that the only mass transfer is from a "wetted" product surface. The validity of this assumption will be tested by comparison of the results from simulations using the product model with data collected from actual experiments.

II.2-10 Mass Transfer Model Modified for Hide Formation

Even though mass diffusion within the product will be ignored, the effects of mass diffusion on the surface conditions of the product can not be ignored. As has been stated previously, as the product cooks the rate of mass transfer from the product is greatly reduced. In particular, the smoke and drying zones have a relatively high rate of mass transfer when compared to the rate of mass transfer in the dry-cook and final cook zones. Therefore as a first approximation to modeling the change in the mass transfer characteristics of the product it will be assumed that either there is mass transfer from a

"wetted" product surface or that there is no mass transfer from the product. The mass transfer model will therefore be "toggled" on or off for a zone depending on whether or not mass transfer from the product will or will not be considered.

For the smoke and drying zones it will be assumed that the product is wet and that mass transfer is important therefore the model of the product will be applied as it has been developed. That is the product model will consider both heat and mass transfer in these two zones.

For the dry-cook and final cook zones where the rate of mass transfer is not as high as in the first two zones, the model will neglect mass transfer from the product and only consider heat transfer. The modification to the product model is simply to set the mass transfer driving force, B, equal to zero when the driving force, B, is greater than zero since evaporation from the product is no longer considered. It is important to realize that by setting $B = 0$ only when $B > 0$, allows the product model to account for the heat transfer due to condensation, $B < 0$, of water on the product surface for those situations when the product enters either of these zones below the dew point temperature.

Chapter III

Heat Transfer Coefficients for Food Processing

The heat transfer coefficient is used to model both the heat and mass transfer from the product surface. This chapter will discuss the experiments used to measure the heat transfer coefficient for the baking zones of the hot dog process. The measured data will then be compared to correlations from literature. The goal will be to develop a means of estimating the heat transfer coefficient from known correlations for flow over surfaces.

III.1 Designing the Heat Transfer Coefficient Experiment

The heat transfer coefficient is a very important parameter used to model both the heat and mass transfer from the product. To establish the value of the heat transfer coefficient for each baking zone it was necessary to run experiments on the cooking process. This section will discuss the design of the experiment and a preliminary experiment used to validate the experimental method.

III.1-1 Experiment Design

The heat transfer coefficient experiment focuses on the heat transfer associated with cooking by eliminating the need to consider mass transfer. A solid metal hot dog is passed through the cooking process. The data collected using the metal hot dog can then be analyzed to establish the convective heat transfer coefficient.

The heat transfer coefficient experiments use the lumped capacitance method. The essence of the lumped capacitance method is the assumption that the temperature of the solid is spatially uniform at any instant during the transient process [3]. This assumption implies that the temperature gradients within the solid are negligible.

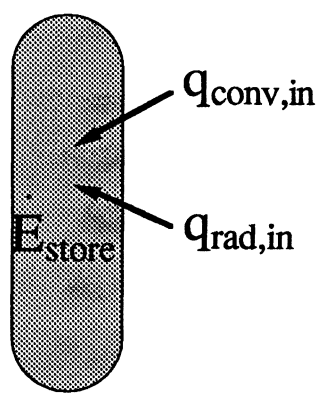

Figure 3.1 Solid Used in Lumped Capacitance Analysis

To analyze a solid that can be modeled using the lumped capacitance method, it is necessary to perform an energy balance on the solid. The energy balance will relate the rate of heat transfer at the solid surface to the rate of energy storage within the solid [3]. Performing an energy balance on the control volume pictured in Fig. 3.1 results in Eqn. 3.1-1.

$$\dot{E}_{store} = q_{conv,in} + q_{rad,in} \tag{3.1-1}$$

The rate of energy transfer at the solid surface due to radiation can be calculated using

$$q_{rad} = \varepsilon\sigma A_s\left(T_{hd}^4 - T^4\right) \tag{3.1-2}$$

where:

ε = Emissivity of solid surface

σ = Stefan-Boltzmann constant

T_{hd} = Surface temperature of the surrounding product (hot dog)

A_s = Surface area of lumped solid

T = Temperature of lumped solid

The rate of energy transfer at the solid surface due to convection can be calculated using

$$q_{conv} = hA_s\left(T_\infty - T\right) \tag{3.1-3}$$

Substituting Eqn. 3.1-2, Eqn. 3.1-3, and a mathematical expression for the rate of energy storage into Eqn. 3.1-1 results in

$$\rho VC\frac{\partial T}{\partial t} = hA_s\left(T_\infty - T\right) + \varepsilon\sigma A_s\left(T_{hd}^4 - T^4\right) \tag{3.1-4}$$

Eqn. 3.1-4 can then be solved for the convective heat transfer coefficient as

$$h = \frac{\rho VC\frac{\partial T}{\partial t} - \varepsilon\sigma A_s\left(T_{hd}^4 - T^4\right)}{A_s\left(T_\infty - T\right)} \tag{3.1-5}$$

The convective heat transfer coefficient can then be measured through experimentation. For a particular solid the volume, mass density, surface area, and specific heat can be determined. Therefore the only data that needs to be collected is the solid temperature, free stream temperature, and surrounding product surface temperature as functions of time.

III.1-2 Design Equations for a Lumped Solid

Before data can be collected it is necessary to design a solid that can be used in the heat transfer coefficient experiment which will satisfy the assumptions of the lumped capacitance method. The development of the equations used to design the lumped solid is greatly simplified if radiation from the solid surface is neglected. Therefore radiation will be ignored while deriving the design equations even though the effects of radiation are accounted for during the actual experiment.

The basic assumption of the lumped capacitance method has been that the temperature gradients within the solid are negligible. This implies that the resistance to conduction within the solid is much smaller than the resistance to convection at the solid surface. The Biot number, defined in Eqn. 3.1-6, is the ratio of the resistance to conduction to the resistance to convection.

$$B_i = \frac{R_{cond}}{R_{conv}} = \frac{\left(\frac{L_c}{k}\right)}{\left(\frac{1}{h}\right)} = \frac{hL_c}{k} \qquad (3.1\text{-}6)$$

where:

L_c = Characteristic length, for a cylinder $L_c = R$

If the Biot number is much less than 1, the assumption of negligible internal temperature gradients is valid. Therefore, one criterion used to design the solid used in the heat transfer coefficient experiment is that the solid must have physical properties such that it satisfies Eqn. 3.1-7 over the range of heat transfer coefficients to be measured.

$$B_i = \frac{hL_c}{k} < 0.1 \qquad (3.1\text{-}7)$$

The other design criterion is that the solid must have a "reasonable" thermal response over the time interval that data is collected. For a fixed time interval, thermal response refers to the magnitude of change in the solid temperature when exposed to a step change in free stream temperature. If the solid responds too quickly the solid will reach the free stream temperature almost immediately; therefore there will be very little time to collect useful data. If on the other hand the solid responds too slowly, the solid temperature may not change significantly over the time interval for which data is collected; therefore the heat transfer coefficient can not be calculated.

It may not be a problem in some cooking processes if the solid responds too slowly, as the time interval for data collection may not be limited. But for the hot dog cooking process each zone is only twelve minutes long. The length of time to collect data for any one zone is therefore limited to twelve minutes. It is desirable to design the solid such that its thermal response is slow enough so that there is ample time to collect useful data, yet not so slow that the solid temperature does not change during

the time interval that the solid is in a process zone.

To evaluate how a particular solid will respond thermally when exposed to a step change in free stream temperature, it is necessary to solve for the transient temperature of the solid as a function of time. Eqn. 3.1-4 can be rewritten, as shown in Eqn. 3.1-8, assuming a constant free stream temperature, neglecting radiation, introducing a temperature difference $\theta = T_\infty - T$, and recognizing that $\partial\theta/\partial t = -\partial T/\partial t$.

$$\frac{1}{\theta}\frac{\partial\theta}{\partial t} = \frac{-hA_s}{\rho VC} \tag{3.1-8}$$

Separating variables and integrating Eqn. 3.1-8 from the initial condition for which $t = 0$ and $\theta_i = T_\infty - T_i$ results in

$$\ln\left(\frac{\theta}{\theta_i}\right) = \frac{-hA_s}{\rho VC}t \tag{3.1-9}$$

Solving for the temperature ratio of the solid yields

$$\frac{\theta}{\theta_i} = \frac{T_\infty - T}{T_\infty - T_i} = \mathrm{Exp}\left(\frac{-hA_s}{\rho VC}t\right) \tag{3.1-10}$$

If everything on the right hand side of Eqn. 3.1-10 were known, it would be possible to solve for the temperature ratio θ/θ_i. The temperature ratio will be a number between 0 and 1, and serves as an indication of the thermal response of the solid. If $\theta/\theta_i = 0$ the solid has responded completely and has reached steady-state conditions, $T = T_\infty$. If $\theta/\theta_i = 1$ the solid has not responded at all thermally and is still at the initial temperature, $T = T_i$.

Eqn. 3.1-7 and Eqn. 3.1-10 were used to design a metal hot dog which was used in the heat transfer coefficient experiment.

For the hot dog process, and in general when possible, it is desirable to design the solid such that it is the same approximate size and shape as a real product so that the flow patterns around the solid will be similar to the flow patterns around a real product. Therefore the surface area of the solid is fixed. It was also assumed that the metal hot dog would be solid rather than hollow, thus fixing the volume of the metal hot dog. The other parameter that is fixed is the time interval over which useful data can be collected. For the hot dog process, each zone is approximately twelve minutes long. Therefore the metal hot dog was designed such that there was a significant thermal response over a twelve minute interval.

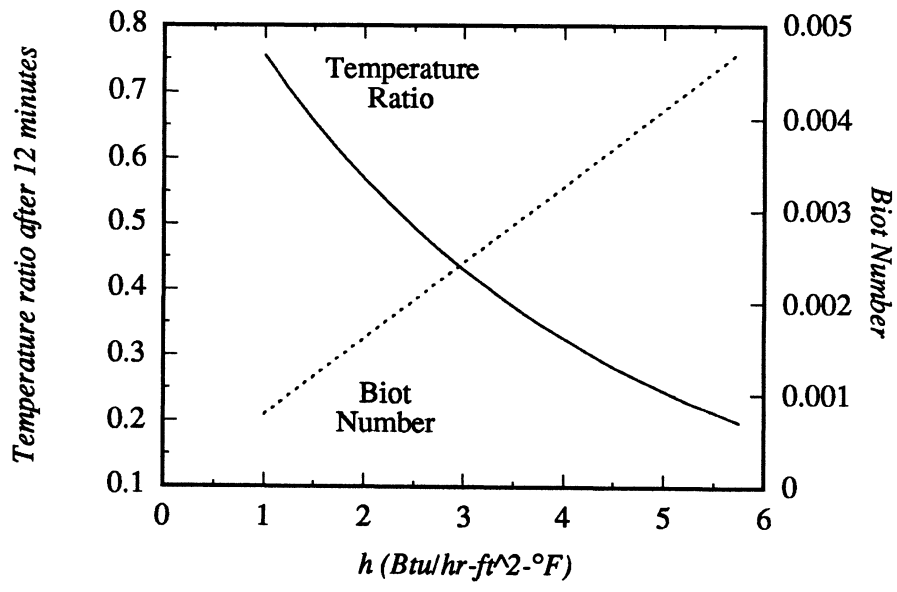

Figure 3.2 Design Graph for Aluminum Alloy 2024

With the surface area, volume, and time interval fixed it was possible to investigate, over a range of predicted heat transfer coefficients, different materials which might be used for the construction of the metal hot dog. Fig. 3.2 shows the results for aluminum alloy 2024.

Fig. 3.2 indicates that, for a twelve minute time interval, a solid with the same physical size and shape as a real hot dog but made of aluminum alloy 2024 will have a significant thermal response for a heat transfer coefficient in the range of 2.5 to 5.5 Btu/hr-ft^2-°F. The Biot number indicates that the assumption of negligible internal temperature gradients is valid over this range. It was estimated that the actual heat transfer coefficients in the convection ovens used in the hot dog process were in this same range therefore the metal hot dog was constructed using aluminum alloy 2024.

It should be noted that the procedure used to design the aluminum hot dog is only one of many ways that Eqn. 3.1-7 and Eqn. 3.1-10 can be used to help design the metal hot dog. It was also assumed that the metal hot dog would be solid. If the desired thermal response can not be obtained with a solid metal hot dog, it is possible to modify Eqn. 3.1-7 and Eqn. 3.1-10 to aid with the design of a hollow metal hot dog.

Fig. 3.3 shows the physical design of the aluminum hot dog. Two holes were drilled at one end to allow for the insertion of a temperature probe, referred to as a Data Trace, which was used to measure the aluminum hot dog temperature. The eye hooks at each end are to allow the aluminum hot dog to be tied "in-line" with other hot dogs on the stick. The aluminum hot dog was coated with high temperature flat-black paint to prevent oxidation of the surface and to establish a known surface emissivity.

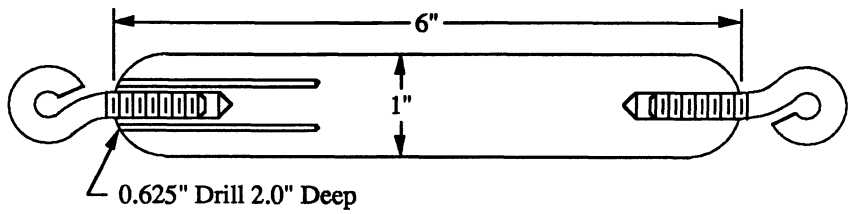

Figure 3.3 Aluminum Hot Dog Schematic (Not to Scale)

III.1-3 Uncertainty Analysis

Before Eqn. 3.1-5 can be used to analyze temperature data it is necessary to decide on a method of approximating the temperature derivative. One possible approximation is to use Taylor series expansion represented by

$$\frac{\partial T}{\partial t} = \frac{T^{p+1} - T^{p+1}}{2\Delta t} + O(\Delta t)^2 \tag{3.1-11}$$

Substituting Eqn. 3.1-11 into Eqn. 3.1-5 neglecting higher order terms, replacing the product of the volume and density with the mass, and writing the surface area in terms of measured data results in

$$h = \frac{MC\left(\frac{T^{p+1} - T^{p+1}}{2\Delta t}\right) - \varepsilon\sigma\pi DL\left(T_{hd}^{p\,4} - T^{p4}\right)}{\pi DL\left(T_{\infty}^{p} - T^{p}\right)} \tag{3.1-12}$$

where:

D = Diameter of aluminum hot dog

L = Length of aluminum hot dog

M = Mass of aluminum hot dog

Eqn. 3.1-12 is referred to as the data reduction equation. But before Eqn. 3.1-12 can be used to calculate heat transfer coefficients it is necessary to perform an uncertainty analysis. The uncertainty analysis will give an indication of the error in the calculated heat transfer coefficient due to errors in the individual measurements used in the calculation.

Most of the data collection for the heat transfer coefficient experiment is straight forward. The mass of the aluminum hot dog was obtained using a balance with an uncertainty $\delta M = 1$ gram. The length and diameter of the aluminum hot dog were measured with uncertainties of $\delta D = 1$ and $\delta L = 3$ millimeters respectively. The specific heat of aluminum alloy 2024 was obtained from literature. The resolution of the reported value indicated an uncertainty in the specific heat equal to $\delta C = 1$ J/kg-K. The uncertainty of the surface emissivity was estimated to be $\delta\varepsilon = 0.1$.

The acquisition of time-temperature data on the other hand is more difficult. It is necessary to measure the local free steam temperature, surrounding product surface temperature, and aluminum hot dog temperature as functions of time. The device used to measure and record these temperatures must be able to ride along with a stick of product and withstand the temperature extremes found within the cooking process. A Data Trace, shown in Fig. 3.4, is such a temperature recorder.

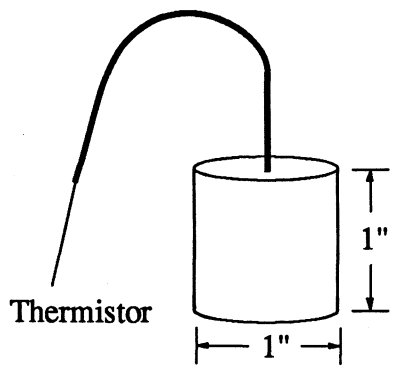

Figure 3.4 Data Trace Temperature Recorder

The Data Trace is a programmable temperature sensor. It has an internal memory which is programmed using a personal computer. The computer program is used to instruct the Data Trace at what time to start recording data and what time interval, Δtime, to use. For example, the Data Trace can be programmed to start reading data at 10:30 a.m. at 6 second intervals (Δtime = 6 sec). The time-temperature data are then stored within the internal memory of the Data Trace. The Data Trace can store up to 1000 time-temperature data points.

When all the desired temperature data have been collected, the data stored within the Data Trace's internal memory is downloaded using a personal computer. The data file generated by the Data Trace can then be opened into a Lotus spread sheet.

The error associated with the internal clock of the Data Trace is assumed to be ± 0.1 sec. Therefore the uncertainty, $\delta\Delta t$, associated with the time interval, Δt, used in the approximation of the derivative of the aluminum hot dog temperature is 0.2 sec.

The uncertainty for the temperature measurement was estimated by placing three Data Traces into a warm water bath and evaluating the spread of measured values. The data from the warm water bath experiment is shown in Fig 3.5.

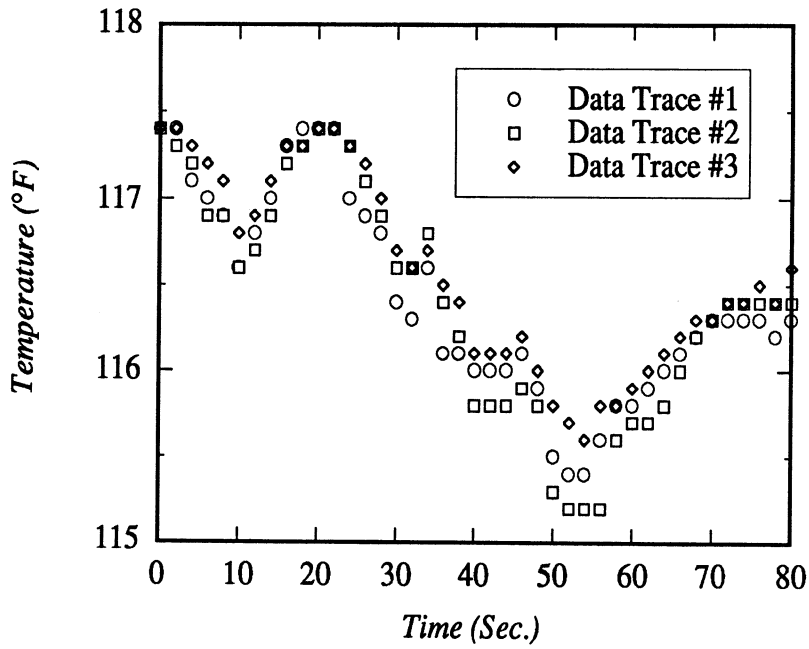

Figure 3.5 Data Trace Precision Plot

Fig. 3.5 indicates that the three Data Traces read to within 1°F of each other. Therefore the uncertainty in the temperature measurement, δT, is approximately 0.56°C. The uncertainty for the temperature change of the aluminum hot dog, $\delta\Delta T$, on the other hand is estimated by evaluating the magnitude of the noise in the temperature measurement of an individual data trance. The noise in the temperature reading is estimated to be ±0.2°F. Therefore the uncertainty $\delta\Delta T$ is equal to 0.22°C.

Table 3.1 is a summary of the estimated uncertainties for each measured variable in the experiment.

$\delta M = 0.001$ kg	$\delta C = 1.0$ J/kg-K
$\delta D = 0.001$ M	$\delta L = 0.003$ M
$\delta \varepsilon = 0.1$	$\delta \Delta t = 0.2$ Sec.
$\delta T = 0.56$ °C	$\delta \Delta T = 0.22$ °C

Table 3.1 Measurement Uncertainties

The uncertainties from Table 3.1 can be substituted into Eqn. 3.1-13 to calculate the uncertainty in the heat transfer coefficient.

$$\left(\frac{\delta h}{h}\right)^2 = \left(\frac{\partial h}{\partial M}\frac{\delta M}{h}\right)^2 + \left(\frac{\partial h}{\partial C}\frac{\delta C}{h}\right)^2 + \left(\frac{\partial h}{\partial \varepsilon}\frac{\delta \varepsilon}{h}\right)^2 + \left(\frac{\partial h}{\partial L}\frac{\delta L}{h}\right)^2 + \left(\frac{\partial h}{\partial D}\frac{\delta D}{h}\right)^2 +$$

$$\left(\frac{\partial h}{\partial T_{hd}}\frac{\delta T_{hd}}{h}\right)^2 + \left(\frac{\partial h}{\partial T_{\infty}}\frac{\delta T_{\infty}}{h}\right)^2 + \left(\frac{\partial h}{\partial \Delta T}\frac{\delta \Delta T}{h}\right)^2 + \left(\frac{\partial h}{\partial \Delta t}\frac{\delta \Delta t}{h}\right)^2 \qquad (3.1\text{-}13)$$

Notice that there are two time intervals associated with the heat transfer coefficient experiment. The first time interval, Δtime, refers to the time step used by the Data Trace when recording time-temperature data. The only limitation for selecting the time step, Δtime, is that the Data Trace can not record data faster than one a second. It must also be keep in mind that the Data Trace can only store 1000 time-temperature datum

points. Therefore if data are to be collected for the entire 96 minute cooking process, the smallest time step, Δtime, that can be used is 6.0 sec.

The other time interval, Δt, refers to the time interval used in the finite-difference approximation of the derivative of the aluminum hot dog temperature. It will be shown that as the time interval, Δt, is increased the uncertainty in the calculated heat transfer coefficient is reduced.

The characteristics of the uncertainty in the heat transfer coefficient were investigated through a preliminary experiment. The preliminary experiment used the aluminum hot dog to study natural convection from a horizontal cylinder. Thermocouples were used to measure the aluminum hot dog temperature and the free stream temperature. The aluminum hot dog was assumed to radiate to the surroundings which were assumed to be at the ambient temperature. Data were taken every second for a half hour, Δtime = 1.0 sec. Fig. 3.6 shows the results of analyzing the free convection data using a time interval, Δt, for the derivative of the aluminum hot dog temperature term of 1.0 second.

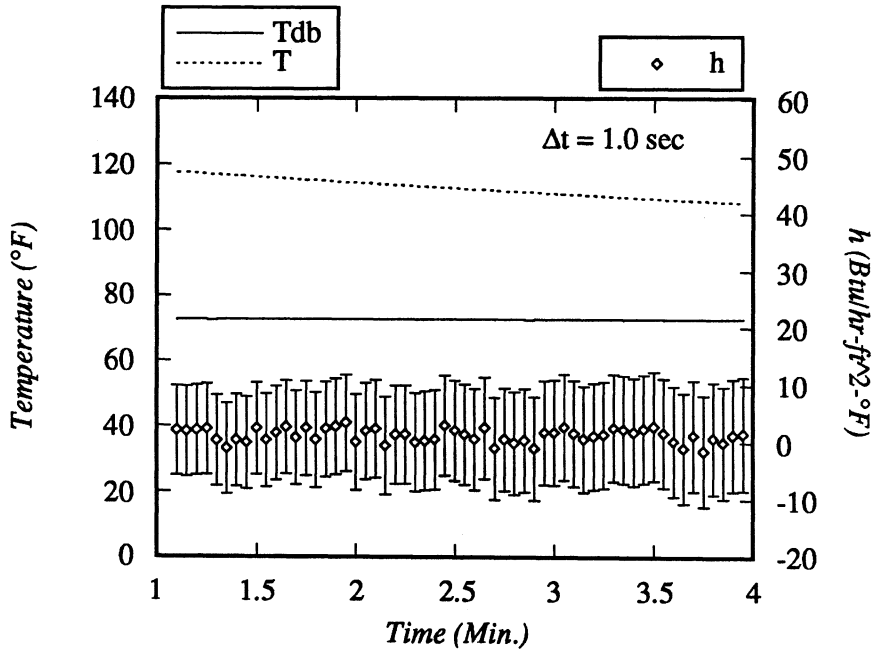

Figure 3.6 Natural Convection Heat Transfer Coefficient

Fig. 3.6 shows that there is such a large error in the heat transfer coefficient that the results are very inaccurate.

The large error associated with the heat transfer coefficient calculation is a result of the behavior of the error in the heat transfer coefficient with respect to the temperature change of the aluminum hot dog. Notice that the free convection data were collected using a 1 second time step, $\Delta time = 1.0$ sec. Notice also that when the data were analyzed using a 1 second interval, $\Delta t = 1.0$ sec, the change in the aluminum hot dog temperature, ΔT, is on the same order of magnitude as the uncertainty in the change of the aluminum hot dog temperature, $\delta\Delta T$. Therefore the error in the calculated heat transfer coefficient is large. The behavior of the error in the heat transfer coefficient,

$\delta h/h$, with respect to the error in the change of the aluminum hot dog temperature, $\delta\Delta T/\Delta T$, is shown in Fig. 3.7.

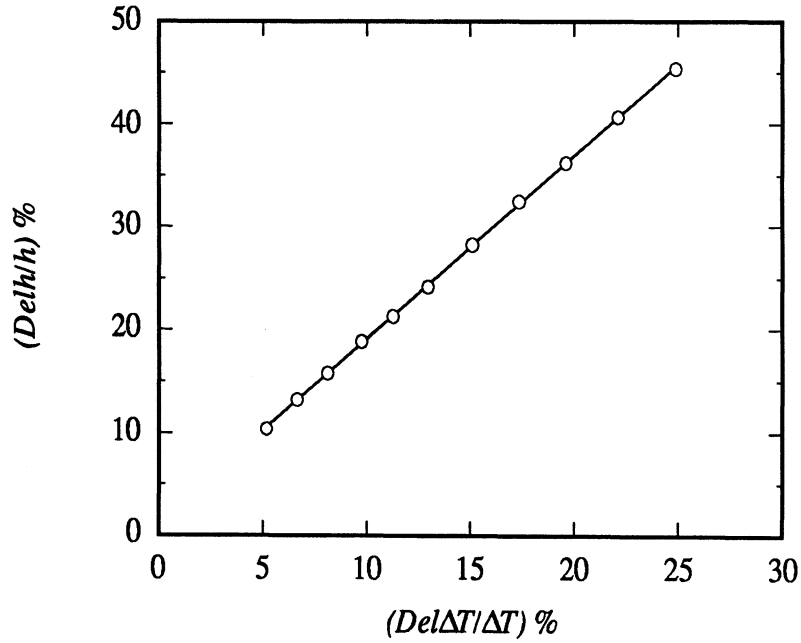

Figure 3.7 Error in Heat Transfer Coefficient

Notice that when there is a large change in the aluminum hot dog temperature, ΔT, over the time interval, Δt, then the error in the heat transfer coefficient is small. As the aluminum hot dog reaches steady-state and the change in the aluminum hot dog temperature, ΔT, approaches zero over the time interval, Δt, the error in the heat transfer coefficient becomes large. Therefore it is advantageous to select a time interval, Δt, for the approximation of the derivative of the aluminum hot dog temperature such that there is a relatively large change in the aluminum hot dog temperature over that time interval Δt.

Notice that there is a penalty for selecting a large time increment, Δt. The penalty is that as Δt is made larger the resultant heat transfer coefficient no longer applies to the immediate neighborhood for which it was calculated.

To understand the importance of this point it is necessary to think in terms of data collection in one of the zones in the cooking process. As the time increment, Δt, is increased the temperature data used to calculate the derivative of the aluminum hot dog temperature encompasses a larger physical portion of the zone. What this means is that if data were analyzed at a time increment of $\Delta t = 30.0$ sec. then the derivative of the aluminum hot dog temperature would involve temperature data recorded 30 seconds to either side of the point at which the heat transfer coefficient is desired. Since time is directly related to the position of the aluminum hot dog within the zone, the time interval $\Delta t = 30.0$ seconds translates into a distance of 2.75 ft. to either side of the point for which the heat transfer coefficient is calculated. Therefore as the time interval, Δt, is increased the calculated heat transfer coefficient becomes less valid for the exact position within the zone for which it was calculated. What this means is that as the time increment, Δt, increases it is more difficult to detect variations in the heat transfer coefficient due to position within the zone.

Now that the behavior of the uncertainty in the heat transfer coefficient has been explained it is possible to go back and re-analyze the free convection data using a time increment large enough to reduce the error associated with the change in the aluminum hot dog temperature. The results of the new analysis are shown in Fig. 3.8.

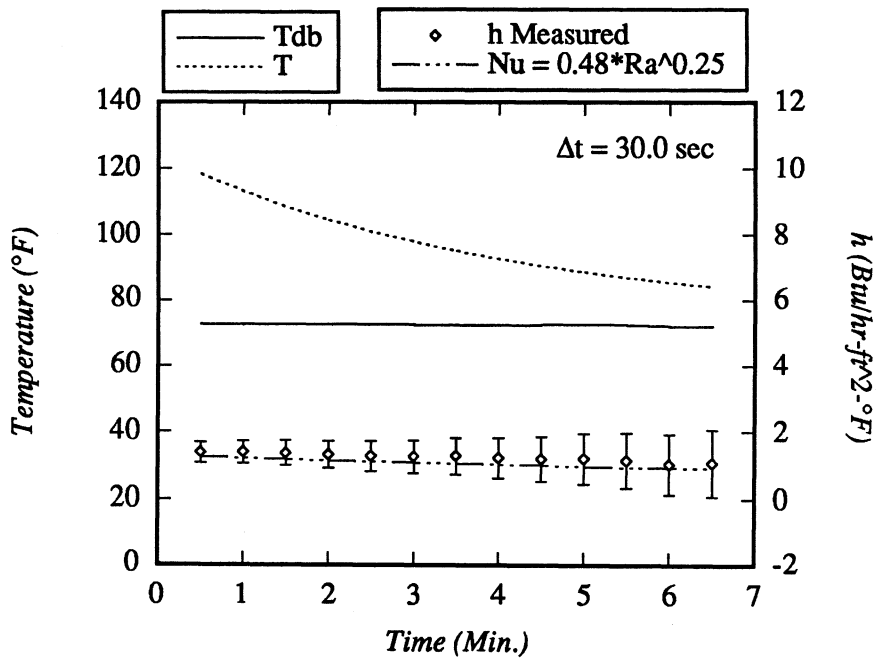

Figure 3.8 Natural Convection Data Re-analyzed

Fig. 3.8 indicates that when a larger time interval, Δt, is used the analysis procedure predicts the heat transfer coefficient for the free convection data with an acceptable level of error.

III.2 Process Heat Transfer Coefficient Experiment

The experiment design and data analysis procedure have been presented and tested. The actual process heat transfer coefficient experiment will now be discussed. First the data collection procedures and experiment set up will be discussed. The

experimental results will then be presented and compared to empirical correlations from literature for flow over a surface. Finally the modeling approached used in the simulation program will be discussed.

III.2-1 Heat Transfer Coefficient Data Acquisition

Four different Data Traces were used during the heat transfer coefficient experiment. One was used to measure and record the local dry bulb temperature. A second measured and recorded the local wet bulb temperature. A third Data Trace was used to measure and record the aluminum hot dog temperature. And finally a fourth Data Trace was used to measure and record the surface temperature of a neighboring hot dog.

The local wet bulb temperature, which tends to show a step change from zone to zone (see Fig. 3.10), was recorded because it provides a useful means of determining which zone the aluminum hot dog is in when it comes time to analyze the data. It is important to know the position of the aluminum hot dog when calculating the heat transfer coefficient so that possible trends due to variations within the zone can be determined.

All the Data Traces were programmed to start recording data at the same time using the same time step, $\Delta time = 6.0$ sec. The Data Traces were then brought down to the cooking process where they were attached to a stick of product as shown in Fig. 3.9.

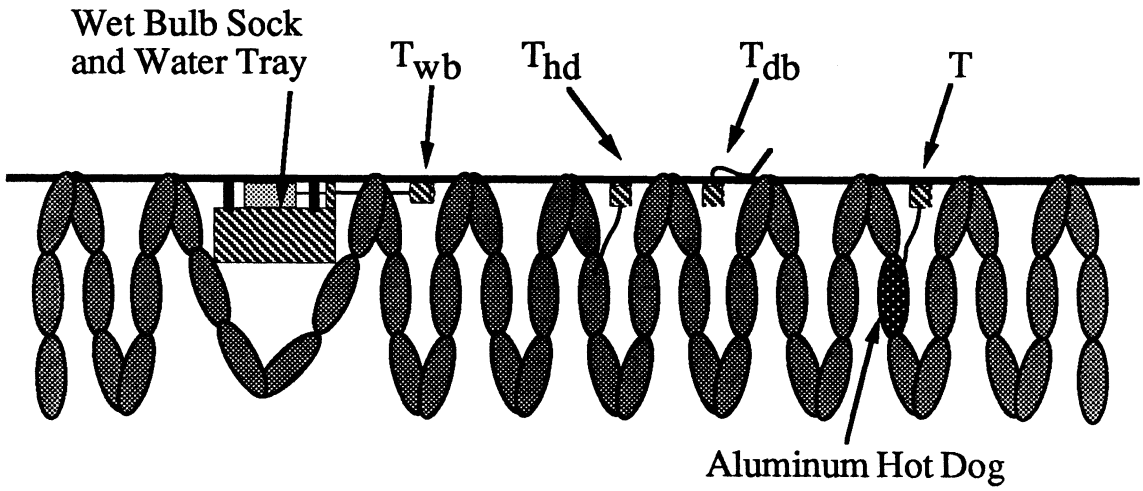

Figure 3.9 Stick of Product With Data Traces Attached

When the programmed start time was reached, the test stick of product was loaded into the cooking process where it rode along with other product during the normal cooking process. When the Data Traces came out at the end of the cooking process, approximately 96 minutes later, the data were down loaded into a Lotus spread sheet file.

Fig 3.10 shows a typical temperature profile generated from the raw data file produced by the Data Trace.

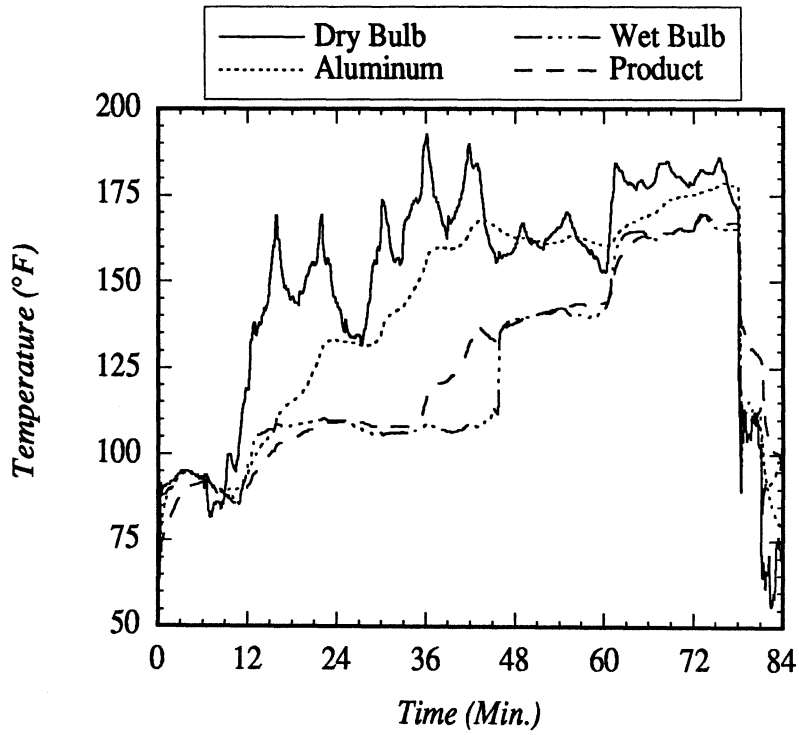

Figure 3.10 Raw Data Temperature Profile

Collecting data on a real cooking process posses some problems that warrant discussion. Because the data are being collected during normal production, there are problems that arise due to errors that occur during processing. A stick of product might get jammed in the process, thus causing the conveyance chain to stop. When the chain stops the Data Trace continues to record time-temperature data. Therefore if the delay is long, the elapsed time in the Data Trace file will no longer agree to position within the cooking process. These delays coupled with the fact that the conditions in the zone are highly dependent on the amount of product within the zone, which is constantly changing, makes it nearly impossible to obtain duplicate runs where the conditions from the two runs are exactly the same. The fact that duplicate runs can not be ob-

tained makes comparing data from different runs to determine trends in the heat transfer coefficient more difficult.

Despite the problems with measuring data on the cooking process it is still possible to extract useful information from the experiment. The next section will discuss the results of the heat transfer coefficient experiment and present the approach used to model the heat transfer coefficient using empirical correlations from literature.

III.2-2 Experimental Results

Several heat transfer coefficient experiments were run on the hot dog process. The results of these experiments will be presented and compared to heat transfer coefficients calculated using empirical correlations found in literature for flow over a flat plate and for a cylinder in cross flow.

The similarity between flow over a flat plate and flow over a hot dog has been used to model the mass transfer in Sec. II.2-1. This similarity will be used again to provide a reference heat transfer coefficient to which the experimentally measured heat transfer coefficient can be compared. An empirical correlation from literature for flow over a plate will therefore be applied to flow over the product.

The Reynold's number, based on product length, can be written as

$$Re_L = \frac{V_a L}{\nu} \approx 2.2 \times 10^4 \qquad (3.2\text{-}1)$$

where:

V_a = Average air velocity

ν = Kinematic viscosity of air

The value of the Reynold's number indicates that flow over the product is in the laminar flow regime. Therefore an empirical correlation, shown in Eqn. 3.2-2, for laminar flow over a flat plate will be applied to the product.

$$\overline{Nu} = \frac{\bar{h}L}{k} = 0.664\ Re_L^{1/2} Pr^{1/3} \qquad [5] \qquad (3.2\text{-}2)$$

where:

$\bar{h}$ = Average convective heat transfer coefficient along the plate

$\overline{Nu}$ = Average Nusselt number

The other extreme for flow over the product would be if the product were considered to behave like a cylinder in cross flow. Therefore to provide another means of comparison, an empirical correlation for a cylinder in cross flow will be presented.

The Reynold's number based on product diameter is represented by

$$Re_D = \frac{V_a D}{\nu} \approx 3.0 \times 10^3 \qquad (3.2\text{-}3)$$

The value of the Reynold's number once again indicates that the product is in the laminar flow regime therefore an empirical correlation, shown in Eqn. 3.2-4, for laminar flow over a horizontal cylinder can be used as a reference for flow over the product.

$$\overline{Nu} = \frac{\bar{h}D}{k} = 0.683\ Re_D^{0.466} Pr^{1/3} \qquad [5] \qquad (3.2\text{-}4)$$

As a means of comparison, the values of the heat transfer coefficient calculated using Eqn. 3.2-2 and Eqn. 3.2-4 will be plotted on the heat transfer coefficient graphs generated from experimental data.

Fig. 3.11 is a graph of the measured heat transfer coefficient for the smoke zone calculated from data collected during run B on Aug. 13, 1991.

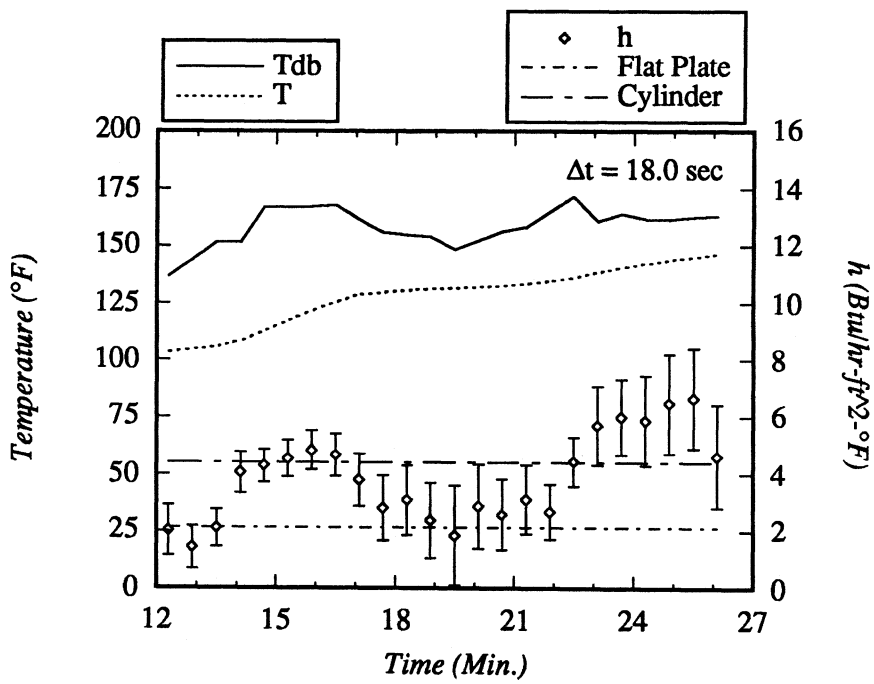

Figure 3.11 Heat Transfer Coefficient Plot for Smoke Zone

Fig. 3.11 is typical of the plots of the heat transfer coefficient for the baking zones. It shows that there is a significant variation in the heat transfer coefficient from the top of the zone to the bottom of the zone. This variation in the heat transfer coefficient is difficult to explain and is still not completely understood.

One possible explanation for the higher heat transfer coefficients at the top of the zone as compared to the bottom of the zone, might be due to the air flow patterns that develop as a result of the rows of diffusers at the top of the zone. These diffusers will tend to produce jets of high velocity air. When these jets of air blow across the product surface they will cause an increase in the rate of heat transfer from the product surface which will show up as an increase in the heat transfer coefficient. As the air moves down the zone and past the product the jets of high velocity air dissipate and the velocity profile across the zone becomes more uniform. The local velocity at the product surface will therefore decrease thus lowering the rate of heat transfer from the product surface which will lead to a decrease in the heat transfer coefficient.

If it is assumed that the flat plate correlation can be used to model the heat transfer coefficient at the bottom of the zone, which is reasonable as shown in Fig. 3.11, then the local air velocity at the top of the zone necessary to achieve a heat transfer coefficient of 5 Btu/hr-ft^2-°F, which is representative of the measured heat transfer coefficient at the top of the zone, requires a local air velocity of 26 ft/sec. The increase in the air velocity requires the diffuser cross sectional area to be approximately 20% of the cross sectional area of the zone. The actual reduction in cross sectional area is approximately 30%. Therefore the effects that the diffusers have on the local air velocity at the product surface may account for part of the variation in the heat transfer coefficient.

Another factor that influences the heat transfer coefficient calculation is the rate of heat transfer due to radiation to the surroundings. It has been assumed that the aluminum hot dog radiates to the surrounding product. When the aluminum hot dog is in the middle of a zone it "sees" only the surrounding product. But when the aluminum hot dog reaches the top or bottom of a pass it "sees" the top or bottom surfaces of the zone

which may be at quite a different temperature then the surrounding product. Therefore the heat transfer due to radiation calculated using the surrounding product temperature might not accurately account for the actual radiative heat transfer at these locations. Also when the aluminum hot dog is on the first or last pass within the zone it "sees" the walls of the zone which may be at quite a different temperature then the surrounding product. Since the heat transfer due to radiation is on the same order of magnitude as the heat transfer due to convection, the surfaces that the aluminum hot dog "sees" and the amount of energy transfer due to radiation to these surfaces can drastically alter the measured values of the heat transfer coefficient.

These are just two possible explanations for the variation in the heat transfer coefficient which is evident in Fig. 3.11. It is beyond the scope of this first approximation of the heat and mass transfer from the product to model the variation in the heat transfer coefficient. Therefore it will be assumed that the heat transfer coefficient is constant within a zone and can be modeled based on the average air velocity within the zone.

The data collected from six experiments run on three different days were used to calculate average heat transfer coefficients for the smoke, drying, and final cook zones. Not all the runs provided useful data on all the zones, and none of the runs provided data for the dry-cook zone because the aluminum hot dog was at the same approximate temperature as the free stream. This lead to very large uncertainties in the measured heat transfer coefficient. The average heat transfer coefficient calculated from the experimental data is plotted as a function of the average air velocity in Fig. 3.12.

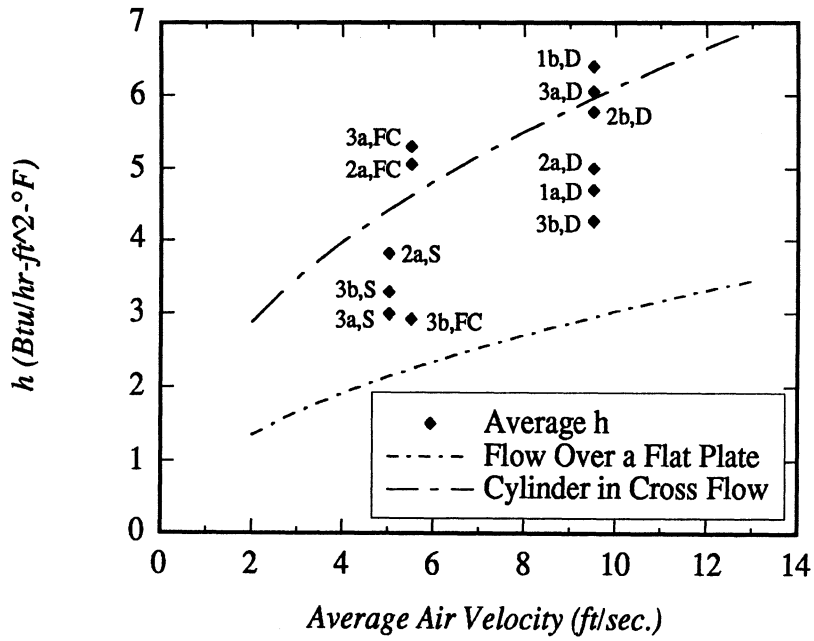

Figure 3.12 Measured h as a Function of Average Air Velocity

The data were plotted with a label that is used to indicate the date of the experiment, run number, and zone location. The code is explained as shown in Fig 3.13.

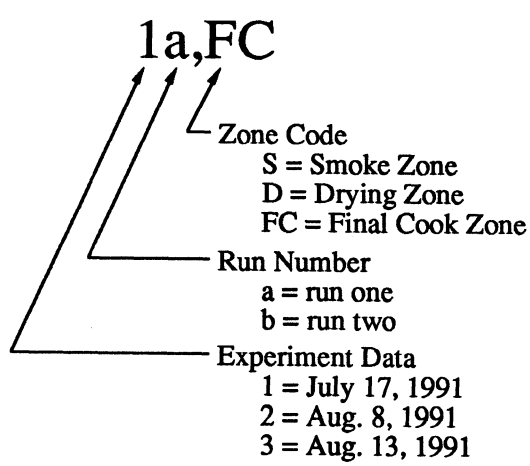

Figure 3.13 Average Heat Transfer Coefficient Labeling Code

Fig. 3.12 shows that there is a large range of variation in the measured heat transfer coefficient as a function of average air velocity. Neither the flat plate correlation nor the cylinder in cross flow correlation accurately represents the measured data. Therefore, as a first approximation, a heat transfer coefficient based on an average of both correlations will be used in the simulation program to predict the zone heat transfer coefficient based on the average air velocity in that zone.

As was stated previously, the heat transfer coefficient does not appear to be constant throughout the zone, but varies from top to bottom. As an approximation an average value will be used. It is recommended that future work should be concentrated on determining first, if the variation in heat transfer coefficient is real or just a result of experimental procedure, and secondly, modeling the variation in the heat transfer coefficient if it is indeed real.

Chapter IV

Dry Bulb Temperature Variation Models

In chapter one it was stated that the dry bulb temperature can vary from the top of the zone to the bottom of the zone by as much as 25°F. The dry bulb temperature variation arrises due to the heat and mass transfer from the air as it flows down past the product. It is important to be able to predict the variation in the dry bulb temperature because it influences the driving forces for both heat and mass transfer.

Chapter four discusses the method used to estimate the dry bulb temperature variations within a zone. The first part of the chapter discusses the modeling approach used to estimate the dry bulb temperature variation in the zones where mass transfer is negligible. The second part of the chapter presents the modeling approach used to estimate the dry bulb temperature variation in the zones where mass transfer can not be neglected.

IV.1 Sensible Heat Exchanger Model

In the dry-cook and final cook zones the rate of mass transfer from the product is reduced due to the hide formation, therefore these two zones were modeled neglecting mass transfer from the product. The dry bulb temperature variation within the zone can then be estimated using the effectiveness-NTU method for sensible heat exchanger analysis.

The flow of product within a baking zone can be thought of as shown in Fig. 4.1.

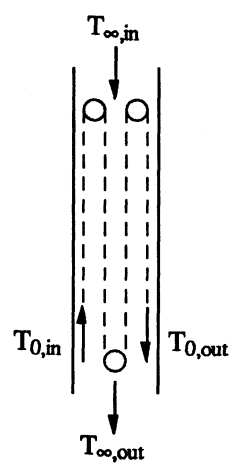

Figure 4.1 Hot Dog "Flow" Within a Baking Zone

The flow geometry in Fig. 4.1 is similar to the flow geometry for a 1 shell, 4 tube pass heat exchanger. The flow of hot dogs is analogous to the fluid flow in the tube passes and the flow of air is in the shell. Such an analogy allows one to apply the effectiveness-NTU method of heat exchanger analysis to those zones where mass transfer is neglected. The general heat exchanger analysis will be presented and then modified for the special case of hot dog flow in a convection oven.

IV.1-1 Effectiveness-NTU Heat Exchanger Analysis

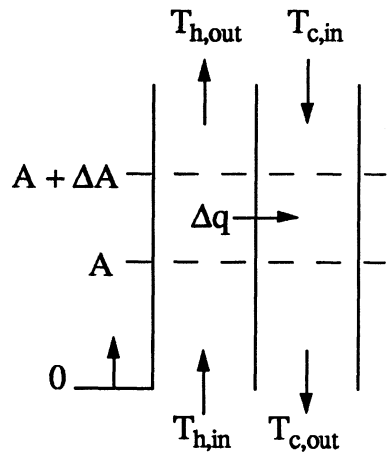

Figure 4.2 Counter Flow Heat Exchanger

Fig. 4.2 represents the flow geometry and notation for a counter flow heat exchanger. An overall energy balance on the heat exchanger relates the energy increase of the cold fluid to the energy decrease of the hot fluid. Performing the energy balance results in

$$\dot{m}_h C_{p,h} T_{h,in} + \dot{m}_c C_{p,c} T_{c,in} - \dot{m}_h C_{p,h} T_{h,out} - \dot{m}_c C_{p,c} T_{c,out} = 0 \qquad (4.1\text{-}1)$$

where:

$\dot{m}_h$ = Mass flow rate of the hot fluid

$\dot{m}_c$ = Mass flow rate of the cold fluid

$C_{p,h}$ = Constant pressure specific heat of the hot fluid

$C_{p,c}$ = Constant pressure specific heat of the cold fluid

T_h = Temperature of the hot fluid

T_c = Temperature of the cold fluid

Eqn. 4.1-1 can be rewritten, assuming constant specific heats, as

$$\dot{m}_h C_{p,h}(T_{h,in} - T_{h,out}) = \dot{m}_c C_{p,c}(T_{c,out} - T_{c,in}) \tag{4.1-2}$$

If an energy balance for the hot fluid is performed on the control volume shown in Fig. 4.2 then the result is

$$(\dot{m}_h C_{p,h} T_h)_A - (\dot{m}_h C_{p,h} T_h)_{A+\Delta A} - \Delta q = 0 \tag{4.1-3}$$

where:

q = Heat transfer from the hot fluid to the cold fluid

The mechanism equation from heat transfer allows Δq to be rewritten as

$$\Delta q = U \Delta A (T_h - T_c) \tag{4.1-4}$$

where

U = Overall heat transfer coefficient

ΔA = Surface area available for heat transfer

Substituting Eqn. 4.1-4 into Eqn. 4.1-3 and dividing by ΔA yields

$$\dot{m}_h C_{p,h} \left[\frac{(T_h)_{A+\Delta A} - (T_h)_A}{\Delta A} \right] = -U(T_h - T_c) \tag{4.1-5}$$

Taking the limit of Eqn. 4.1-5 as ΔA goes to zero results in the following differential equation for the hot fluid.

$$\dot{m}_h C_{p,h} \frac{dT_h}{dA} = -U(T_h - T_c) \tag{4.1-6}$$

Performing an energy balance on the cold fluid and going through a similar derivation as was followed for the hot fluid results in the following differential equation

$$\dot{m}_c C_{p,c} \frac{dT_c}{dA} = -U(T_h - T_c) \tag{4.1-7}$$

The governing equations for a sensible heat exchanger can be summarized as shown below. Eqn. 4.1-8 represents the overall energy balance for the heat exchanger. Eqn. 4.1-9 and Eqn. 4.1-10 represent the mechanism equations for the hot and cold fluids respectively.

$$\frac{dT_h}{dA} = \frac{\dot{m}_c C_{p,c}}{\dot{m}_h C_{p,h}} \frac{dT_c}{dA} \tag{4.1-8}$$

$$\dot{m}_h C_{p,h} \frac{dT_h}{dA} = -U(T_h - T_c) \tag{4.1-9}$$

$$\dot{m}_c C_{p,c} \frac{dT_c}{dA} = -U(T_h - T_c) \tag{4.1-10}$$

Now that the general equations governing the thermal behavior of a heat exchanger have been presented it is possible to discuss the effectiveness-NTU method used to analyze heat exchangers. The goal of the analysis is to develop a means of predicting the outlet temperatures of the hot and cold fluids given the inlet temperatures and the value of the UA product.

Before the effectiveness of a heat exchanger can be defined it is necessary to determine the maximum possible heat transfer rate q_{max}. The maximum heat transfer rate would, in principle, be achieved in a counter flow heat exchanger, shown in Fig. 4.2, of infinite length [3]. In such a heat exchanger one of the fluids would experience

the maximum possible temperature difference, $T_{h,in} - T_{c,in}$.

If the heat capacitance rate of the hot fluid, $C_h = \dot{m}_h C_{p,h}$, is greater than the heat capacitance rate of the cold fluid, $C_c = \dot{m}_c C_{p,c}$, then the cold fluid would experience the largest temperature difference and for a heat exchanger of infinite length, the cold fluid would leave the heat exchanger at the inlet temperature of the hot fluid, $T_{c,out} = T_{h,in}$. Therefore q_{max} can be represented by

$$q_{max} = C_c(T_{h,in} - T_{c,in}) \qquad C_h > C_c \tag{4.1-11}$$

If on the other hand the heat capacitance rate of the cold fluid is greater than the heat capacitance rate of the hot fluid, then the hot fluid would experience the largest temperature change and for a heat exchanger of infinite length, the hot fluid would leave the heat exchanger at the inlet temperature of the cold fluid, $T_{h,out} = T_{c,in}$. Therefore q_{max} can be represented by

$$q_{max} = C_h(T_{h,in} - T_{c,in}) \qquad C_c > C_h \tag{4.1-12}$$

It is apparent that the maximum heat transfer, q_{max}, can also be represented by

$$q_{max} = C_{min}(T_{h,in} - T_{c,in}) \tag{4.1-13}$$

where C_{min} is equal to the heat capacitance rate of the hot or cold fluid, whichever is smaller.

It is now possible to define an effectiveness, ε, for a heat exchanger as the ratio of the actual heat transfer rate to the maximum possible heat transfer rate.

$$\varepsilon = \frac{q}{q_{max}} \tag{4.1-14}$$

The actual heat transfer rate can then be calculated using

$$q = \varepsilon C_{min}(T_{h,in} - T_{c,in}) \tag{4.1-15}$$

For any heat exchanger it can be shown that

$$\varepsilon = f\left(NTU, \frac{C_{min}}{C_{max}}\right) \quad [3] \tag{4.1-16}$$

where C_{max} is defined as the larger of the two heat capacitance rates, C_h or C_c, and the number of transfer units, NTU, is defined by

$$NTU = \frac{UA}{C_{min}} \tag{4.1-17}$$

The outlet temperature of the hot fluid can then be calculated by performing an energy balance on the hot fluid in the heat exchanger shown in Fig. 4.2.

$$q = C_h(T_{h,in} - T_{h,out}) \tag{4.1-18}$$

For a given heat exchanger the UA product and inlet fluid temperatures are known. Therefore the heat transfer rate, q, can be calculated using Eqn. 4.1-15 and Eqn. 4.1-16. Eqn. 4.1-18 can then be solved for the outlet temperature of the hot fluid to give

$$T_{h,out} = T_{h,in} - \frac{q}{C_h} \tag{4.1-19}$$

In a similar manner the outlet temperature of the cold fluid can be written as

$$T_{c,out} = T_{c,in} + \frac{q}{C_c} \tag{4.1-20}$$

Now that the general framework of the effectiveness-NTU method has been presented it is necessary to discuss how these equations are applied to the flow of hot dogs in a convection oven.

IV.1-2 Modifications for Flow of Hot Dogs in an Oven

In the previous section the effectiveness-NTU approach for sensible heat exchanger analysis was presented. It has been assumed that the flow of hot dogs in a convection oven is analogous to the flow in a heat exchanger. In this section the general heat exchanger analysis equations are presented in terms of the specific case at hand. That is, the general effectiveness-NTU equations are rewritten in terms of the known parameters for the hot dog cooking process. The model used to estimate the dry bulb temperature variation within a zone is then presented.

The heat capacitance rate of the air can be calculated from the known air velocity, density, and specific heat as shown in Eqn. 4.1-21.

$$C_a = (\text{Cabinet Width})(\text{\# passes per zone})(\text{pass spacing})V_{a,\infty}\,\rho_{a,\infty}\,C_{p,a,\infty} \tag{4.1-21}$$

where:

$V_{a,\infty}$ = Free stream velocity of air water-vapor mixture

$\rho_{a,\infty}$ = Free stream mass density of air water-vapor mixture

$C_{p,a,\infty}$ = Free stream specific heat of air water-vapor mixture

Pass spacing, shown in Fig. 4.3, is defined as the horizontal distance from one pass of the conveyance chain to the next pass.

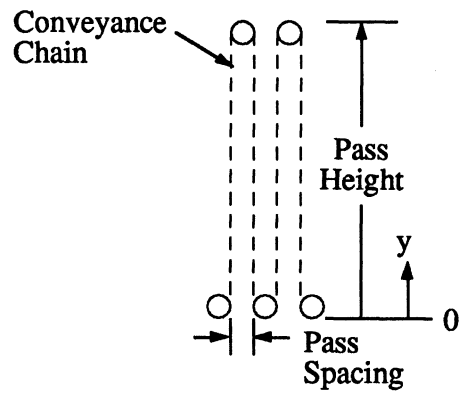

Figure 4.3 Pass Height and Pass Spacing

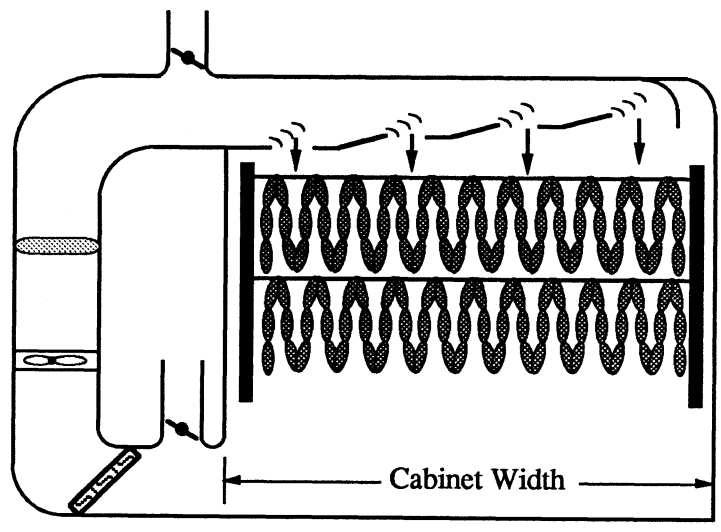

Figure 4.4 Cabinet Width

Cabinet width, as shown in Fig. 4.4, is defined as that part of the process width in which the product travels. The number of passes per zone is defined as the number of vertical passes a product makes as it travels through a zone. A baking zone in the hot dog process typically has four passes.

The product of the number of passes, cabinet width, and pass spacing can be thought of as the cross sectional area of an air duct in which the product travels. The "duct area" is used along with the average velocity and density of the air water-vapor mixture to calculate the mass flow rate of moist air in the zone.

The heat capacitance rate for the product can be calculated as

$$C_h = (\text{\# of hot dogs per stick})\left(\frac{V_{chain}}{\text{stick spacing}}\right)V_{hd}\rho_{hd}C_{p,hd} \qquad (4.1\text{-}22)$$

where:

V_{chain} = Chain speed

V_{hd} = Volume of one hot dog

ρ_{hd} = Mass density of hot dog

$C_{p,hd}$ = Constant pressure specific heat of hot dog

Chain speed refers to the velocity of the conveyance chain which, for the hot dog process, is approximately equal to 5.5 ft/min. Stick spacing is defined as the distance between sticks of product on the conveyance chain. The ratio of chain speed to stick spacing can be thought of as the stick velocity, or in other words, the rate at which sticks of product are loaded into the process. When the stick velocity is multiplied by the mass of product per stick it yields the mass flow rate of hot dogs in the zone.

The capacitance rates of both the air and hot dogs have been defined in terms of known parameters. It is now necessary to discuss the manner in which the number of transfer units can be determined.

Since there is no solid surface separating the flow of hot dogs from the flow of air the only thermal resistance between the product and the free stream is due to the convective thermal resistance at the product surface. The thermal resistance due to radiation is assumed negligible as the product will tend to "see" only other product at approximately the same temperature. Therefore the overall heat transfer coefficient can be written as

$$U = \frac{1}{R_{total} A} = \frac{1}{\left(\frac{1}{h}\right)} = h \qquad (4.1\text{-}23)$$

where:

R_{total} = Sum of individual thermal resistances

h = Convective heat transfer coefficient

To calculate the number of transfer units, NTU, it is necessary to determine the total surface area available for heat transfer within the zone. The surface area can be approximated, as shown in Eqn. 4.1-24, by the sum of the surface area of the total number of hot dogs within the zone.

$$A = 2\pi RL(\text{\# hot dogs per stick})(\text{\# of passes per zone})\left(\frac{\text{pass height}}{\text{stick spacing}}\right) \qquad (4.1\text{-}24)$$

where:

L = Length of a hot dog

R = Outer radius of a hot dog

Pass height, shown in Fig. 4.3, is defined as the distance the product travels as it moves from the bottom of the zone to the top of the zone.

As was stated previously, the effectiveness is a function of NTU and the ratio of C_{min} to C_{max} [3]. Effectiveness relations for various heat exchanger arrangements have been developed and are readily available in literature. The effectiveness relationship for a 1 shell, 4 tube pass heat exchanger is written as

$$\varepsilon = 2\left\{1 + C_r + \left(1 + C_r^2\right)^{1/2} \frac{1 + \mathrm{Exp}\left[-\mathrm{NTU}\left(1 + C_r^2\right)^{1/2}\right]}{1 - \mathrm{Exp}\left[-\mathrm{NTU}\left(1 + C_r^2\right)^{1/2}\right]}\right\}^{-1} \quad [3] \qquad (4.1\text{-}25)$$

where:

$$C_r = \frac{C_{min}}{C_{max}}$$

The outlet air temperature for a zone were mass transfer is negligible can then be approximated using

$$T_{\infty,out} = T_{\infty,in} - \frac{q}{C_a} \qquad (4.1\text{-}26)$$

The heat transfer rate, q, can be calculated using

$$q = \varepsilon C_{min}\left(T_{\infty,in} - T_{0,in}\right) \qquad (4.1\text{-}27)$$

where T_0 is the product surface temperature at state 0 as defined in Sec. II.2-2.

For a particular zone, the inlet hot dog surface temperature and inlet air temperature are known, therefore it is possible to use Eqn. 4.1-26 and Eqn. 4.1-27 to estimate the temperature at which the air exits the zone. The free stream dry bulb temperature can then be modeled as a linear function of the vertical position of the product within the zone as shown in Eqn. 4.1-28.

$$T_\infty = T_{\infty,out} + \left(\frac{T_{\infty,in} - T_{\infty,out}}{\text{pass height}}\right)y \qquad (4.1\text{-}28)$$

where y is defined as shown in Fig. 4.3.

The effectiveness-NTU analysis for sensible heat exchangers can be applied to those zones where mass transfer is negligible. For those zones where the mass transfer is significant it is necessary to apply a cooling tower model which is discussed in the next section.

IV.2 Cooling Tower Analogy

The mass transfer rate is greatly reduced in the dry-cook and final cook zones. Therefore it was assumed that these zones could be modeled neglecting mass transfer. The fact that mass transfer was negligible allowed the dry bulb temperature in these zones to be modeled using the effectiveness-NTU method of heat exchanger analysis. In the smoke and drying zones, where the rate of mass transfer is significant, the sensible heat exchanger model can not be used. Instead a cooling tower model, which parallels the heat exchanger model, will be applied to the flow of "wet" product in a

convection oven. The cooling tower analysis will be developed in terms of the hot dog process immediately rather than starting with the general case and then modifying for the hot dog process as was done for the heat exchanger model.

Fig. 4.5 shows the flow geometry and notation for a "wet" product flowing in a counter current "cooling tower".

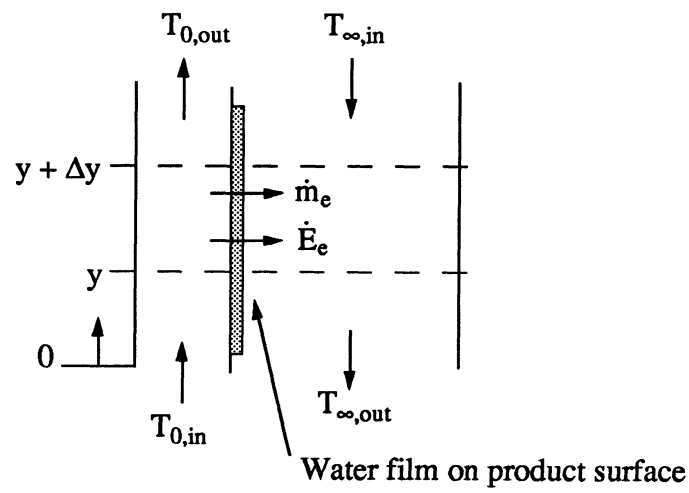

Figure 4.5 Flow of "Wet" Product in a Cook Zone

A mass balance on the control volume shown in Fig. 4.5 relates the amount of water evaporated from the product surface to the moisture content of the free stream. Performing the water mass balance results in

$$(\dot{m}_{hd})_y - (\dot{m}_{hd})_{y+\Delta y} + (\dot{m}_a\omega_\infty)_{y+\Delta y} - (\dot{m}_a\omega_\infty)_y = 0 \qquad (4.2\text{-}1)$$

where:

$\dot{m}_a$ = Mass flow rate of the air water-vapor mixture

$\dot{m}_{hd}$ = Mass flow rate of hot dogs

ω_∞ = Humidity ratio of free stream

Dividing Eqn. 4.2-1 by Δy and taking the limit as Δy goes to zero results in the following differential equation.

$$\frac{d\dot{m}_{hd}}{dy} = \dot{m}_a \frac{d\omega_\infty}{dy} \tag{4.2-2}$$

Eqn. 4.2-2 can be integrated from the initial condition at y=0 where $\dot{m}_{hd} = \dot{m}_{hd,in}$ and $\omega = \omega_{out}$, to yield the mass flow rate of product at any position in the oven.

$$\dot{m}_{hd} = \dot{m}_{hd,in} - \dot{m}_a\left(\omega_{\infty,out} - \omega_\infty\right) \tag{4.2-3}$$

Eqn. 4.2-3 shows that the mass flow rate of product is not constant but rather changes due to the evaporation of water from the product surface as it travels through the zone.

An energy balance on the control volume shown in Fig. 4.5 relates the energy change of the product to the energy change of the free stream. Performing the energy balance yields

$$\left(\dot{m}_{hd} C_{p,hd} T_0\right)_y - \left(\dot{m}_{hd} C_{p,hd} T_0\right)_{y+\Delta y} + \left(\dot{m}_a i_\infty\right)_{y+\Delta y} - \left(\dot{m}_a i_\infty\right)_y = 0 \tag{4.2-4}$$

where:

i_∞ = Enthalpy of the free stream air water-vapor mixture

Dividing Eqn. 4.2-4 by Δy, taking the limit as Δy goes to zero, and assuming a constant specific heat for the product results in the following differential equation.

$$C_{p,hd}\frac{d(\dot{m}_{hd}T_0)}{dy} = \dot{m}_a\frac{di_\infty}{dy} \tag{4.2-5}$$

If the first term in Eqn. 4.2-5 is expanded by differentiating by parts and Eqn. 4.2-3 is substituted in for the mass flow rate of hot dogs then the following differential equation relating the hot dog temperature as a function of distance to the free stream enthalpy and humidity is the result.

$$\frac{dT_0}{dy} = \frac{\dot{m}_a\left[\frac{di_\infty}{dy} - C_{p,hd}T_0\frac{d\omega_\infty}{dy}\right]}{C_{p,hd}\left[\dot{m}_{hd,in} - \dot{m}_a(\omega_{\infty,out} - \omega_\infty)\right]} \tag{4.2-6}$$

A mass balance on the free stream relates the mass transfer from the product to the humidity change of the air. Performing the mass balance on the control volume shown in Fig. 4.5 results in

$$(\dot{m}_a\omega_\infty)_{y+\Delta y} + \dot{m}_e - (\dot{m}_a\omega_\infty)_y = 0 \tag{4.2-7}$$

where $\dot{m}_e$ represents the mass flow rate of water that evaporates from the product surface.

The air at the product surface is assumed to be saturated air at the product surface temperature. Therefore the mass flow rate due to evaporation at the product surface is represented by

$$\dot{m}_e = \frac{h}{C_{p,a}}A'\Delta y(\omega_0 - \omega_\infty) \tag{4.2-8}$$

where ω_0 represents the humidity ratio of saturated air at the 0 state. The 0 state is defined as explained in section II.2-2. A' represents the total area available for heat and mass transfer per unit length of the "cooling tower". A' can be calculated using Eqn. 4.1-24 as

$$A' = \frac{2\pi RL(\text{\# hot dogs per stick})(\text{\# of passes per zone})}{(\text{stick spacing})} \tag{4.2-9}$$

Combining Eqn. 4.2-7 and Eqn. 4.2-8 and taking the limit as Δy goes to zero yields

$$\frac{d\omega_\infty}{dy} = \frac{A'h}{\dot{m}_a C_{p,a}}(\omega_0 - \omega_\infty) \tag{4.2-10}$$

An energy balance on the free stream relates the energy change of the product to the change in energy of the free stream. Performing an energy balance on the control volume shown in Fig. 4.5 yields

$$(\dot{m}_a i_\infty)_{y+\Delta y} + \dot{E}_e - (\dot{m}_a i_\infty)_y = 0 \tag{4.2-11}$$

where $\dot{E}_e$ represents the energy transfer from the product to the free stream.

Eqn. 2.2-33, rewritten below, represents an energy balance at the product surface.

$$\frac{h}{C_{p,a}}(i_0 - i_\infty) = \dot{m}''_w(i_L - i_0) - q''_L \tag{4.2-12}$$

The energy balance states that the energy flux within the product, the right hand side of Eqn. 4.2-12, must equal the energy flux at the product surface, the left hand side

of Eqn. 4.2-12. Therefore the energy transfer from the product, $\dot{E}_e$, can be represented by

$$\dot{E}_e = \frac{A'h}{C_{p,a}} \Delta y (i_0 - i_\infty) \tag{4.2-13}$$

Substituting Eqn. 4.2-13 into Eqn. 4.2-11, dividing by Δy, and taking the limit as Δy goes to zero results in

$$\frac{m_a C_{p,a}}{A'} \frac{di}{dy} = -h(i_0 - i_\infty) \tag{4.2-14}$$

The overall mass and energy balances, Eqn. 4.2-3 and Eqn. 4.2-6 respectively, and the mass and energy balances on the air side, Eqn. 4.2-10 and Eqn. 4.2-14 respectively, describe the heat and mass transfer within the "cooling tower". Various techniques have been applied to obtain a solution to these equations. The approach which will be presented here is to make use of the effectiveness model for cooling towers developed by Braun [1989]. Braun's method allows one to model a cooling tower using effectiveness-NTU relations similar to those developed for the sensible heat exchanger.

A simplifying assumption made by Merkel [1925] is to neglect the effects of water loss due to evaporation [6]. Neglecting water loss allows Eqn. 4.2-6 to be rewritten as

$$\frac{\partial T_0}{\partial y} = \frac{m_a}{m_{hd} C_{p,hd}} \frac{\partial i_\infty}{\partial y} \tag{4.2-15}$$

An effective specific heat can then be introduced. The effective specific heat allows the product surface temperature, T_0, to be replaced by the saturated air enthalpy at that temperature [6]. The effective specific heat is defined as

$$C_s = \left(\frac{\partial i_0}{\partial T_0}\right) \quad [6] \tag{4.2-16}$$

where

i_0 = Enthalpy of saturated air at the product surface temperature, T_0

The value of the effective specific heat, C_s, can be determined using a numerical approximation of the derivative as

$$C_s \approx \left(\frac{i_{0,in} - i_{0,out}}{T_{0,in} - T_{0,out}}\right) \tag{4.2-17}$$

Since C_s depends on the outlet surface temperature of the product, which is still unknown, some iteration will be necessary.

Substituting Eqn. 4.2-16 into the overall energy balance, Eqn. 4.2-15, results in

$$\frac{\partial i_0}{\partial y} = \frac{m_a C_s}{m_{hd} C_{p,hd}} \frac{\partial i_\infty}{\partial y} \tag{4.2-18}$$

The heat and mass transfer in the "cooling tower" is now represented by two equations. The first is the overall energy equation, Eqn. 4.2-18, rewritten using an effective specific heat. The other equation is the air side energy balance, Eqn. 4.2-14. These two equations are analogous to Eqn. 4.1-8 and Eqn. 4.1-9 from the heat exchanger analysis presented in Sec. IV.1-1. Therefore effectiveness-NTU relations can

be applied to the "cooling tower" [6].

The total energy transfer from the product to the free stream can be written as

$$q = \varepsilon \dot{m}_a \left(i_{0,in} - i_{\infty,in} \right) \tag{4.2-19}$$

The effectiveness, ε, is calculated using Eqn. 4.1-25 where the heat exchanger NTU and capacitance ratio is replaced by the tower NTU and capacitance ratio defined by Eqn. 4.2-20 and Eqn. 4.2-21 respectively.

$$NTU = \frac{hA'(\text{pass height})}{\dot{m}_a C_{p,a}} \tag{4.2-20}$$

$$m^* = \frac{\dot{m}_a C_s}{\dot{m}_{hd} C_{p,hd}} \tag{4.2-21}$$

The outlet air enthalpy can then be calculated from the overall mass and energy balances as

$$i_{\infty,out} = i_{\infty,in} + \frac{q}{\dot{m}_a} \tag{4.2-22}$$

The outlet product temperature can then be determined by integrating Eqn. 4.2-15 over the length of the "cooling tower" to yield

$$T_{0,out} = T_{0,in} + \frac{\dot{m}_a}{C_{p,hd} \dot{m}_{hd,out}} \left(i_{\infty,in} - i_{\infty,out} \right) \tag{4.2-23}$$

To determine the outlet air temperature it is necessary to estimate the outlet air humidity ratio. This is done by integrating Eqn. 4.2-14 to define an effective saturation

enthalpy $i_{0,eff}$ [6].

$$i_{0,eff} = i_{\infty,in} - \frac{i_{\infty,out} - i_{\infty,in}}{1 - Exp(-NTU)} \tag{4.2-24}$$

An effective saturation humidity ratio, $\omega_{0,eff}$, can then be determined from a psychometric chart using $i_{0,eff}$ and the fact that the air is saturated. The outlet humidity ratio can then be determined by assuming the evaporation is from an effective uniformly saturated surface and integrating Eqn. 4.2-10 [6].

$$\omega_{\infty,out} = \omega_{0,eff} + (\omega_{\infty,in} - \omega_{0,eff})Exp(-NTU) \tag{4.2-25}$$

With the outlet air enthalpy and humidity ratio known it is possible to determine the outlet air temperature from a psychometric chart.

The free stream dry bulb temperature can then be modeled as a linear function of the vertical position of the product within the zone as shown in Eqn. 4.2-26.

$$T_\infty = T_{\infty,out} + \left(\frac{T_{\infty,in} - T_{\infty,out}}{\text{pass height}}\right)y \tag{4.2-26}$$

where y is defined as shown in Fig. 4.3.

Two models for predicting the exit air temperature for a zone given the inlet air and product temperatures have been presented. One model is used for those zones where mass transfer is negligible. The other model is for those zones where mass transfer can not be neglected. The validity of these two models will be examined in the next chapter.

Chapter V

Model Validation

In chapter two the heat and mass transfer models used to predict the products thermal behavior were developed. Chapter three dealt with the modeling approach used to estimate the heat transfer coefficient for a zone based on the average air velocity in that zone. Chapter four presented a rather simple method of determining the dry bulb temperature at the bottom of the zone from the inlet dry bulb and product surface temperatures. These three modeling components were combined into a simulation program, see appendix, which can be used to simulate the cooking of the product. It is now time to use the simulation program to predict the thermal behavior of a hot dog as it is cooked and compare the results to actual data.

The simulation program was written such that it can be used to model the entire 96 minute cooking process. The concentration of this thesis however has been on the modeling of the four baking zones in the middle of the cooking process. Therefore the simulation program will be used to simulate the thermal behavior of the product in these zones only. It is left as a recommendation that future work should be concentrated on modeling the transition, water bath, and chilled brine zones.

V.1 Simulation of Individual Zones

To simulate the thermal behavior of the product in a baking zone it is necessary to know the product temperature as it enters the zone, the dry bulb temperature at the top of the zone, and the wet bulb temperature. Once this information is known, the simulation program can be used to predict the dry bulb temperature at the bottom of the zone and the thermal behavior of the product as it travels through the zone.

The zone set point temperatures and initial product temperature used to simulate a particular baking zone were obtained from the data collected on August 13, 1991 Run-A. The method which was followed was to view the actual temperature data and determine at what time the product entered the zone which was to be simulated. For example, if the smoke zone was to be simulated the actual data would be reviewed to determine at what time the actual product entered the smoke zone. For the data collected on Aug. 13, it was estimated that the product entered the smoke zone at approximately 13 minutes into the cooking process. Therefore the initial product temperature necessary to run the simulation program was set equal to the product temperature which was measured at 13 minutes into the process. The inlet dry bulb temperature was obtained by viewing the dry bulb temperature data for the smoke zone and estimating the inlet temperature at the top of the zone from this data. Again as an example, the dry bulb temperature data for the smoke zone was plotted and it was determined that the temperature at the top of the zone was approximately 168°F. Therefore the dry bulb set point temperature for the simulation program was set equal to 168°F. The wet bulb temperature was obtained in the same fashion as the dry bulb temperature. The wet bulb set point temperature for the simulation of the smoke zone was set equal to 108°F. This general procedure of using the measured data to determine the inlet product

temperature, dry bulb inlet temperature, and wet bulb temperature was followed for each of the baking zones.

Once all the required temperature inputs are known, it is possible to use the simulation program to estimate the thermal behavior of the product as it is cooked. Fig. 5.1 shows the results of the simulation program as compared to the actual temperature data collected on August 13, 1991 Run-A.

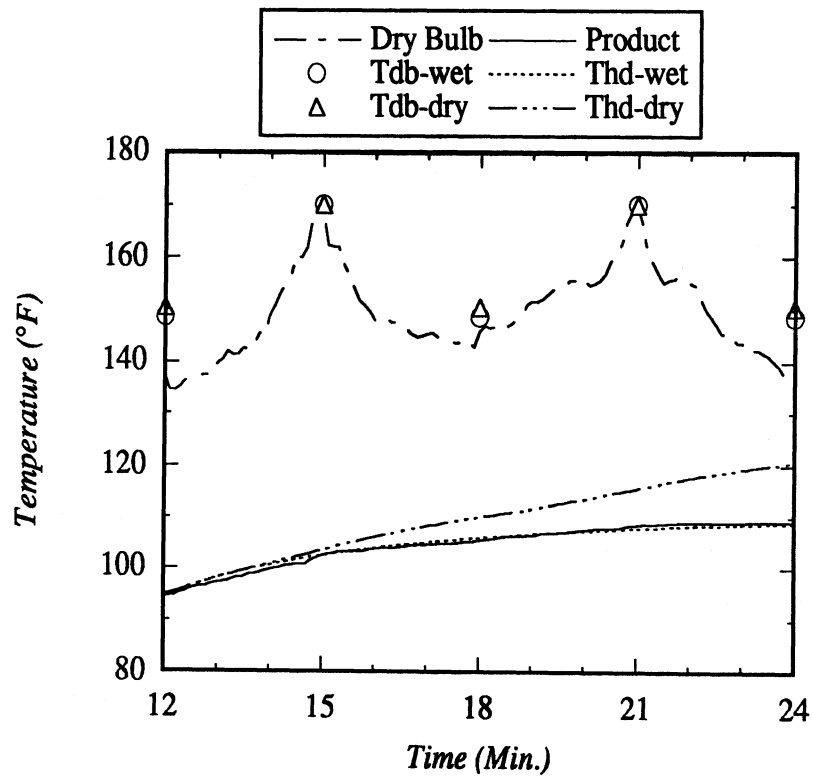

Figure 5.1 Smoke Zone Simulation Comparison

Notice that there are six different curves plotted on Fig. 5.1. Two of the curves are the actual dry bulb and product temperatures as measured on Aug. 13, 1991 Run-A.

Two of the curves are the simulated product and dry bulb temperatures, Thd-wet and Tdb-wet respectively, when the surface of the product is considered to be "wet", or in other words when mass transfer from the product is considered. Two of the curves are the simulated product and dry bulb temperatures, Thd-dry and Tdb-dry respectively, when the surface of the product is considered to be "dry" and mass transfer from the product has been neglected. This type of format was followed for all the zone temperature plots.

The most important characteristic of Fig. 5.1 is that the mass transfer model predicts the thermal behavior of the product and the dry bulb exit temperature relatively well. Note that the temperature profile simulated when mass transfer is neglected tends to over predict the temperature of the product. This result is exactly what is expected since the effects of evaporative cooling are being ignored in the model that neglects mass transfer from the product.

The fact that the simulation which includes mass transfer does a better job of modeling the product temperature then the simulation which neglects mass transfer is not surprising. As has been stated before, the rate of mass transfer is relatively high in the smoke zone. Therefore it is expected that the model which accounts for mass transfer from the product surface be able to predict the thermal behavior of the product better than the model that ignores the mass transfer.

Fig. 5.2 shows the results of the simulation of the drying zone.

Note that, at least initially, the simulation which accounts for mass transfer does a better job of modeling the product temperature than the simulation that neglects mass

transfer. About half way through the zone the actual product temperature breaks away from the wet bulb temperature. This change in the products thermal behavior has been linked to the formation of the hide. As the hide begins to form at the product surface it has been hypothesized that the rate of mass transfer decreases. Proof of this theory is based on the fact that the product temperature no longer follows the wet bulb temperature, but rather begins to increase.

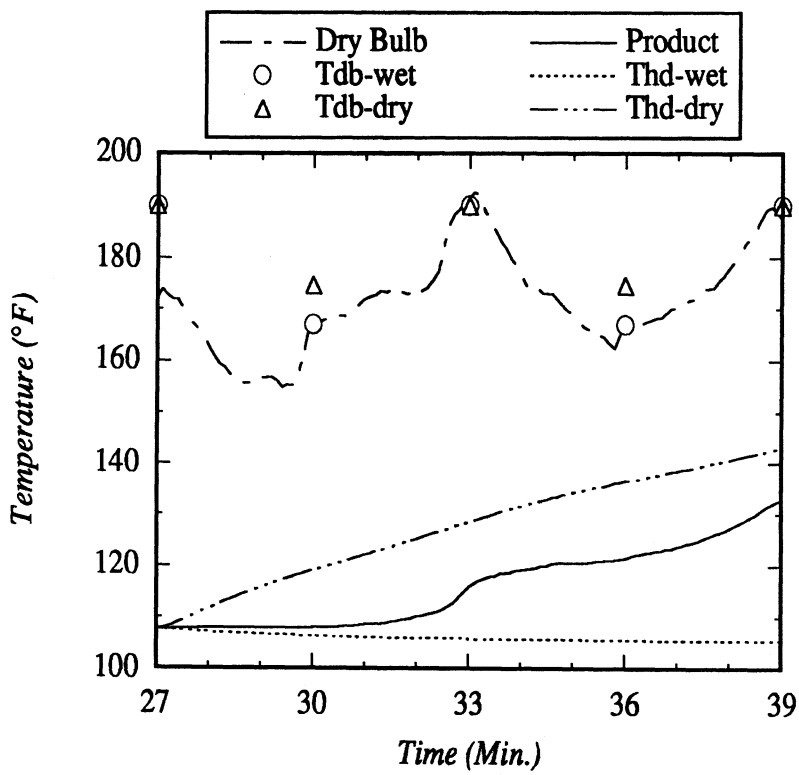

Figure 5.2 Drying Zone Simulation Comparison

As the rate of mass transfer from the product decreases, the energy that had been going into the phase change of water at the product surface is now available to cause a sensible change in the product temperature. Therefore the product temperature begins

to rise above the wet bulb temperature of the free stream.

Note also that the simulation which includes mass transfer does a much better job of predicting the exiting dry bulb temperature then the model that neglects mass transfer. It is believed that the rate of mass transfer in this zone is still rather high. Therefore the fact that the simulation which includes mass transfer does a better job of modeling the dry bulb temperature than the simulation that neglects mass transfer is not surprising.

One other interesting point about Fig. 5.2 is that the product enters the zone at a temperature slightly higher than wet bulb temperature. The expected result should be that the product should experience some evaporative cooling which will bring the product temperature back down to the wet bulb temperature. This downward trend in the product temperature is indeed visible in the simulation which accounts for mass transfer. The fact that the actual product temperature does not decrease, provides some evidence that the rate of mass transfer from the product has already begun to decrease by this point in the cooking process

Fig. 5.3 is the results from the simulation of the dry-cook zone.

Fig. 5.3 indicates that neither the simulation which accounts for mass transfer nor the simulation that neglects mass transfer does an adequate job of predicting the thermal behavior of the product in this zone. This result should have been expected. The model of the product has been developed such that the effects of mass transfer at the product surface can either be "toggled" on or off. Even though the rate of mass transfer from the product has been reduced due to the hide formation, mass transfer from the

product has not stopped completely therefore neither of the simulations can accurately model the product temperature behavior. The important thing to note about Fig. 5.3 is that the temperature of the actual product lies between the limits calculated by the simulation.

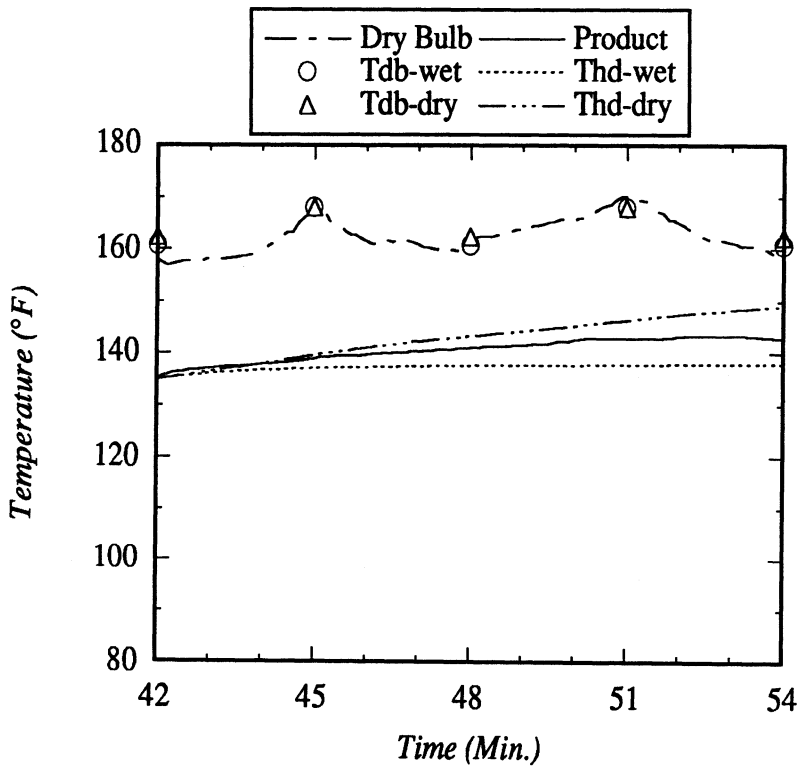

Figure 5.3 Dry-Cook Zone Simulation Comparison

It should also be noted that both simulations due a rather good job of estimating the dry bulb temperature at the bottom of the zone. The fact that both models predict about the same exiting temperature is due to the fact that the driving force for mass transfer is relatively small in this zone. That is to say that the humidity ratio at the

product surface does not differ much from the free stream humidity ratio. Therefore the effect that mass transfer has on the cooling tower model of the free stream is relatively small and the cooling tower model tends to behave in a manner similar to the heat exchanger model.

Fig. 5.4 shows the results of the simulation of the final cook zone.

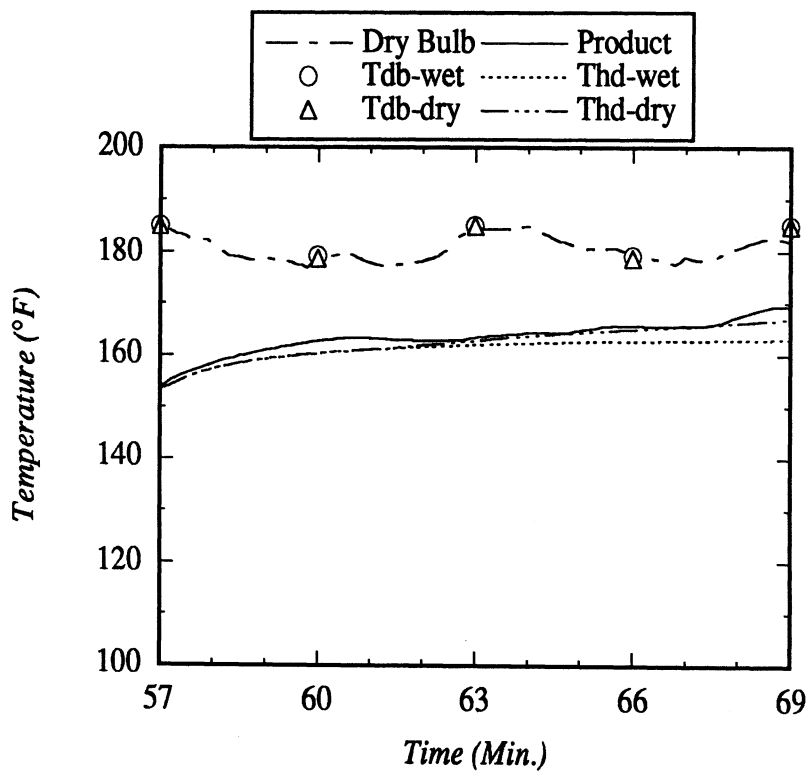

Figure 5.4 Final Cook Zone Simulation Comparison

Fig. 5.4 indicates that the simulation which neglects mass transfer does a better job of modeling the thermal behavior of the product than the simulation which accounts for mass transfer. Once again this result is expected since the rate of mass transfer from

the product surface in the final cook zone has been greatly reduced due to the formation of the hide. Note also that the driving force for mass transfer is relatively small in this zone so once again the two simulations predict approximately the same dry bulb temperature at the bottom of the zone.

In conclusion it can be said that the simulation which accounts for mass transfer does a relatively good job of predicting the thermal behavior of the product and the exiting dry bulb temperature for those zones where the rate of mass transfer is relatively high. In particular, the mass transfer simulation works best in the smoke zone. Fig. 5.5 shows an enlarged view of the product temperature predicted for the smoke zone using the simulation that accounts for mass transfer.

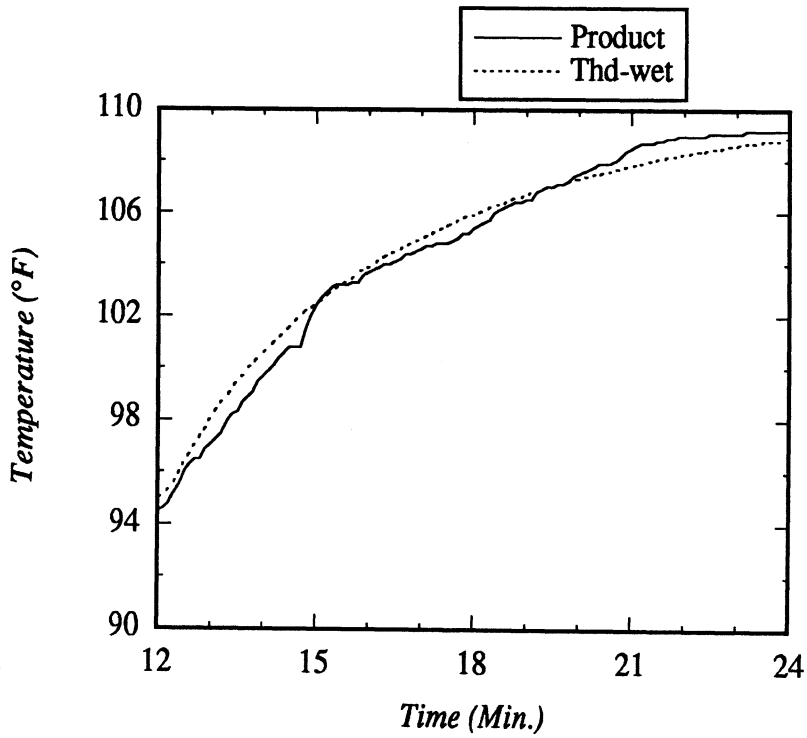

Figure 5.5 Smoke Zone Product Temperature

The simulation which neglects mass transfer does a relatively good job of predicting the thermal behavior of the product and estimating the exiting dry bulb temperature for those zones where the rate of mass transfer has been greatly reduced due to the formation of the hide. In particular, the simulation which neglects mass transfer works best in the final cook zone. Fig. 5.6 shows an enlarged view of the predicted product temperature in the final cook zone using the simulation which neglects mass transfer.

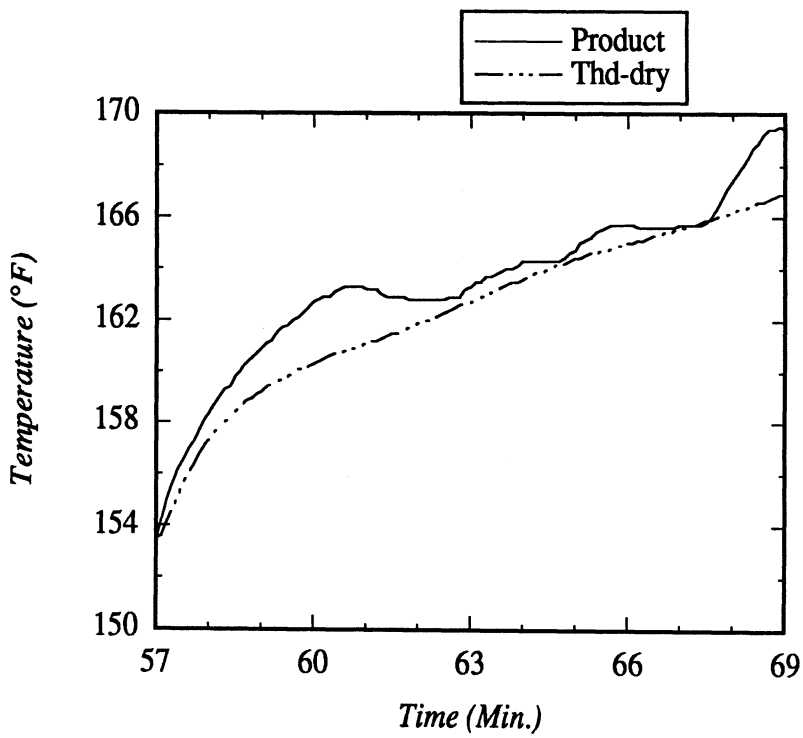

Figure 5.6 Final Cook Zone Product Temperature

An important point to note about the temperature profile in Fig. 5.6 is the initial temperature increase of the product temperature at the beginning of the zone. This temperature increase is caused by condensation of water on the product surface due to

the fact that the product enters the zone with a surface temperature which is below the dew point temperature. Even though the simulation used to predict the thermal behavior of the product temperature neglects mass transfer, it is still able to account for any condensation that might occur at the product surface.

Neither simulation is able to accurately account for the thermal behavior of the product in the drying zone nor the dry-cook zone. This is due to the fact that the simulation only allows for two modes of mass transfer from the product. That is, either mass transfer is accounted for, and then it is assumed that the product surface is totally wet, or mass transfer is neglected. There is no way of modeling a surface that is partially "wet" or in other words, there is no way to account for the fact that the rate of mass transfer gradually decreases as the product cooks and the hide forms.

V.2 Internal Temperature Profile

A major portion of the simulation program is dedicated to approximating the temperature profile within the product. Actual temperature data comparing the surface to core temperatures was not collected. Therefore the only means of verifying that the finite-difference portion of the simulation program is working correctly is to compare the results of the simulation program to the analytical solution. The analytical solution was derived in Sec. II.1-4 for the special case of a step change in the free stream temperature with no mass transfer at the product surface. The simulation program was used to predict the temperature of a 4.0 inch diameter cylinder under the same boundary conditions. The results of the simulation program are compared to the analytical solution as shown in Fig. 5.7.

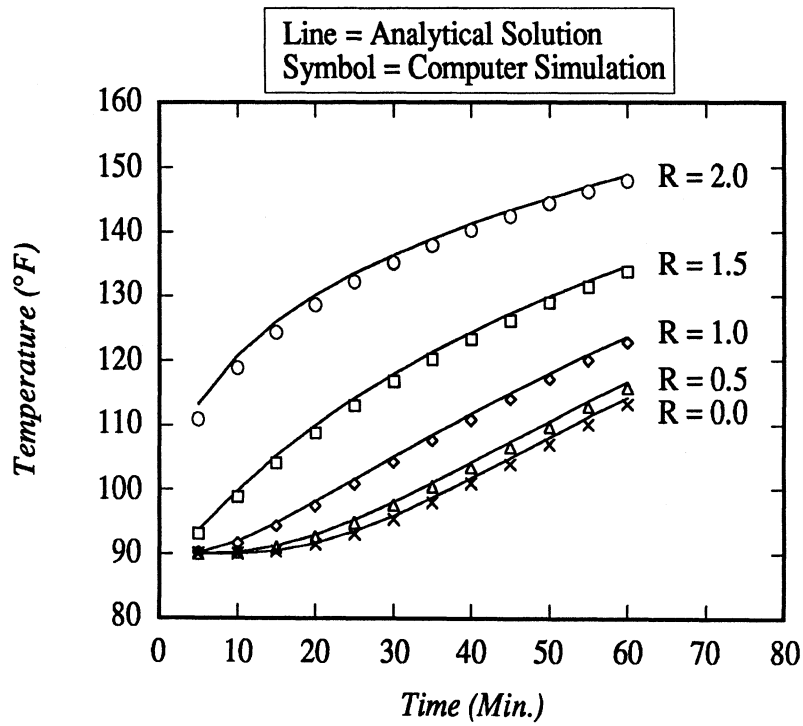

Figure 5.7 Verification of Finite-Difference Approximation

It is evident from Fig. 5.7 that the simulation program accurately models the internal temperature of the cylinder as compared to the analytical solution. Note that Fig. 5.7 is intended to show that the finite-difference portion of the simulation program is working correctly. The temperature distribution within a real product depends on the conditions at the surface of the product and the material properties within the product. Therefore the ability of the simulation program to accurately predict the core temperature of a real product depends on the ability to model the conditions at the product surface and knowing the values of the material properties of the product.

It is recommended that future experiments be conducted which collect the necessary data to verify the ability of the simulation to predict the temperature distribution within a product for those zones where the simulation is able to accurately model the surface conditions of the product. Actual surface to center temperatures should be collected for the product in the smoke and final cook zones and the resulting data should be compared to the results predicted by the simulation program for these zones.

Chapter VI

Conclusions and Recommendations

This research project and thesis have concentrated on the modeling of heat and mass transfer during cooking. Chapters II through IV discussed the modeling techniques used to simulate the cooking of hot dogs in convection ovens. The thermal aspect of the model has been compared to actual data and it has been shown that the simulation is able to predict the thermal behavior of the product in the smoke and final cook zones. The simulation program is unable to predict the thermal behavior of the product in the drying and dry-cook zones because the product model does not have the ability to account for the change in the surface boundary condition as the product begins to dry.

The simulation of the product has included modeling the temperature distribution within the product. The finite-difference portion of the simulation program used to model the temperature variation was verified by comparing the simulation results to the analytical solution. It was shown that the simulation program accurately predicts the temperature distribution for a solid cylinder when exposed to a step change in the free stream temperature.

The simulation program also provides an estimate of the dry bulb temperature at the bottom of the zone. Two models were used. One treated the flow of hot dogs like flow in a sensible heat exchanger. The other model simulated the flow of hot dogs using a cooling tower model. It was shown that these two models are able to predict the exiting dry bulb temperature rather well.

In conclusion, the simulation program provides a good building block for future simulation work. There is room for improvement in the models used in the simulation program and in the procedure used to collect experimental data. Therefore the rest of this chapter presents some recommendations regarding areas in which future research work on the heat and mass transfer in cooking processes should be concentrated.

Sec. II.2-9 discussed mass diffusion within the product. The simulation model neglects mass diffusion within the product. It is recommended that future modeling work should be concentrated on accounting for the effects of mass diffusion within the product. It was estimated that the mass transfer Biot number for the diffusion of water in meat was on the order of magnitude of 100. The fact that the Biot number is much greater than unity indicates that the overall rate of mass transfer from the product will be limited by the rate at which water can diffuse to the product surface.

The rate of mass transfer from the product decreases due to the formation of a "hide" at the product surface. It has been hypothesized that as water diffuses to the product surface it brings with it proteins which seal the surface, thus forming a "hide" which prevents any further moisture loss. Another factor that is linked to the reduced rate of mass transfer from the product is the rate of mass diffusion within the product. If water can not diffuse to the product surface fast enough to keep the product surface

"wet" then the product surface and the meat immediately below the surface will begin to dry out, thus changing the thermal response of the product. Therefore it is possible that the change in the thermal behavior of the product in the drying zone is due to the rate of mass diffusion within the product rather than the formation of a "hide" at the product surface. The theory that mass diffusion rather than "hide" formation causes the reduced rate of mass transfer from the product is based on the value of the mass transfer Biot number and an experiment Jerry Marra ran at Oscar Mayer.

Oscar Mayer was experimenting with a hot dog that had a water content higher than the water content of the hot dogs that were used during the heat transfer coefficient experiments. The temperature profile of the high water content product did not show the characteristic deviation of the product temperature from the wet bulb temperature during processing in the drying zone. Or if the product temperature did break away from the wet bulb temperature it occurred later in the drying zone.

The fact that the product did not break away from the wet bulb temperature indicates that the product surface is still wet. The rate of mass transfer from the high water content product should be at least as high as that for the lower water content product. Therefore it seems reasonable to assume that the rate at which the proteins reach the product surface and the rate at which the "hide" forms should be approximately the same for both types of hot dog. If indeed the rate of mass transfer from the product was solely governed by the clogging of the surface pores with proteins, than the two products thermal behavior should be similar.

The data collected on the high water content product does not support the theory that a hide forms at the product surface. The data rather shows that the high water

content product remains at the wet bulb temperature and mass transfer from the product continues. The theory that the rate of mass diffusion within the product governs the rate of mass transfer from the product is better able to explain the observed data. For the high water content product, more water will have to be driven off before the product begins to show the signs of drying. Therefore the product temperature will remain at the wet bulb temperature for a longer period of time until enough water is driven off to cause the meat below the surface to begin to dry out.

It is possible that the rate of mass transfer from the product is not governed by the rate at which a "hide" forms at the product surface but rater is limited by the rate at which water can diffuse to the product surface. To test the proposed theory it is recommended that future work be concentrated on modeling the diffusion of water within the product. I believe that once diffusion is accounted for, the simulation program will be able to predict the thermal behavior of the product in those zones where the product surface is neither totally wet nor totally dry.

In Sec. II.2-2 the heat transfer coefficient was modeled assuming one average value for the entire zone. The average heat transfer coefficient was approximated using the average air velocity within the zone and empirical correlations for flow over surfaces. It was stated that the collected data indicates that there is some variation in the heat transfer coefficient from the top of the zone to the bottom of the zone. Two possible explanations of this variation were presented. One explanation was linked to a variation in the free stream air velocity near the top of the zone due to the air jets produced by the diffusers. The other explanation dealt with the heat transfer from the aluminum hot dog due to radiation to the surrounding product and oven walls. It is recommended that more heat transfer coefficient experiments should be conducted to

determine if there really is a variation in the heat transfer coefficient and, if necessary, to model the variation in the heat transfer coefficient.

To eliminate the effects of radiation, heat transfer coefficient experiments should be conducted using a "shinny", low emissivity, aluminum hot dog. When the aluminum hot dog experiment was originally designed it was thought that the aluminum hot dog surface would become covered with smoke and grease and would not remain shinny. Actual experiments have indicated that the surface of the aluminum hot dog does remain clean. Therefore a shinny aluminum hot dog would reduce, if not eliminate the effects of radiation on the measurement of the heat transfer coefficient. It would be useful to run experiments with both a shinny and a black aluminum hot dog and then to compare the resulting heat transfer coefficients calculated using the data.

It is also important to determine the air velocity within the zone to determine if there is a significant difference in the local air velocity at the top of the zone as compared to the air velocity at the bottom of the zone. If there is a large variation, then the simulation program will need to be modified to account for the variation in the air velocity and the resultant effect on the heat transfer coefficient.

In Chapter IV two methods for predicting the exiting dry bulb temperature in a zone were developed. Chapter V showed that the cooling tower model was able to predict the exiting dry bulb temperature for all four baking zones. Therefore it is recommended that the cooling tower model be used to simulate the exiting dry bulb temperature for all zones, but that the curve fit used to approximate the temperature between the inlet and exit temperatures be improved. Currently the simulation program uses a linear fit between the inlet and exit temperatures. It is known from heat ex-

changer theory that the temperature of the air should decay exponentially as it flows down past the product. Therefore it is recommended that a curve fit which better models the actual data be used to approximate the dry bulb temperature as a function of position within the zone.

The last recommendation is to determine the importance of the transition zones, how they influence the thermal behavior of the product, and to include a model of the transition zone in the simulation program.

Appendices

A.1 Simulation Programs

The simulation program has been written using the models developed in the thesis. Although the program has been written such that the entire cooking process can be simulated, it has been shown that the simulation does not work for certain zones within the process. Therefore, since the program uses values calculated from the previous zone to simulate cooking in the next zone, it is recommended that the simulation program be used to simulate only one zone at a time and that it be used for the smoke, drying, dry-cook, and final cook zones only.

The simulation has been broken down into one main program and three support programs. The main program uses data files generated by the other three programs. The three support programs will be presented first then the main simulation program will be discussed.

A.1-1 House Specification Program

The program called "House.for" is used to establish the parameters which describe the physical design of the cooking process. The program is used to set the following parameters:

1. Cabinet Width
2. Pass Height
3. Pass Spacing
4. Chain Speed
5. Stick Spacing
6. Average Number of Hot Dogs per Stick

Each of these parameters must be set before the main simulation program can be run.

```
c ---------------------------------------------------------------------
c *********************************************************************
c *****************    HOUSE SPECIFICATION PROGRAM    ****************
c *********************************************************************
c ---------------------------------------------------------------------
c
c
c ---------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    ********************
c ---------------------------------------------------------------------
c
c
c    This program is used to specify those parameters that are
c    primarily dependent on the design of the house.  This includes
c    the following variables:
c
c        1.  Cabinet Width
c        2.  Pass Height
c        3.  Pass Spacing
c        4.  Velocity of conveyance chain
```

```
c         5.  Stick Spacing
c         6.  Average number of hot dogs per stick
c
c    The program then creates a data file called "house.dat" which
c    contains all the house data used in the simulation program.  The
c    data is stored in metric units as that is the form in which the
c    data is used by the simulation program.  The units are converted
c    on entry and exit of the program.
c
c
c ------------------------------------------------------------------
c ***********************    VARIABLE TABLE    **********************
c ------------------------------------------------------------------
c
c
c          width = Width of cooking zone "Cabinet" (approximately
c                  the length of a stick of hot dogs) {m}
c         height = Height of house, from top to bottom of one
c                  pass {m}
c       PassSpac = Distance between passes (pass spacing) {m}
c         Vchain = Velocity of conveyance chain {m/Sec}
c      StickSpac = Distance between sticks on the conveyance chain {m}
c        hotdogs = Average number of hot dogs per stick {integer}
c              i = counter {integer}
c           used = logical variable used to determine if data file
c                  already exists
c
c
c ------------------------------------------------------------------
c *********************    DECLARE VARIABLES    *********************
c ------------------------------------------------------------------
c
      implicit none
      real width,height,PassSpac,Vchain,StickSpac
      integer hotdogs,i
      logical used
c
c ------------------------------------------------------------------
c **********    DETERMINE IF DATA FILE ALREADY EXISTS    ************
c ------------------------------------------------------------------
c
      inquire(file='house.dat',exist=used)
c
c ------------------------------------------------------------------
c *******************    CREATE NEW DATA FILE    *******************
c ------------------------------------------------------------------
c
      if(used .EQ. .FALSE.) then
         open(unit=1,file='house.dat',status='new')
         width = 0.0
         height = 0.0
         PassSpac = 0.0
         Vchain = 0.0
         StickSpac = 0.0
         hotdogs = 0
```

```
c
c ----------------------------------------------------------------
c ***************    READ EXISTING INPUT DATA FILE    ***************
c ----------------------------------------------------------------
c
      elseif(used .EQ. .TRUE.) then
         open(unit=1,file='house.dat',status='unknown')
         rewind(unit=1)
         read(1,90) width,height,PassSpac,Vchain,StickSpac,hotdogs
         rewind(unit=1)
      endif
c
c ----------------------------------------------------------------
c ************    DETERMINE "WORKING" UNITS FROM USER    ************
c ----------------------------------------------------------------
c
      print*,'Enter the system of units you would like to work in:'
      print*
      print*,'          1.  Metric {m,KJ,Kg,C,Sec}'
      print*,'          2.  English {ft,Btu,Lbm,F,Sec}'
      print*
5     read*,i
c
c ----------------------------------------------------------------
c ******    GIVE ERROR MESSAGE IF INPUT NOT EQUAL TO 1 OR 2    ******
c ----------------------------------------------------------------
c
      if(i .LT. 1 .OR. i .GT. 2) then
         print*,'Please enter a "1" for Metric units or a "2" for
     & English units.'
         goto 5
c
c ----------------------------------------------------------------
c ****************************************************************
c ********************    WORK IN METRIC UNITS    *******************
c ****************************************************************
c ----------------------------------------------------------------
c
      elseif(i .EQ. 1 ) then
c
c ----------------------------------------------------------------
c *************    PRINT CURRENT VALUES TO THE SCREEN    ************
c ----------------------------------------------------------------
c
8        print100
         print110
         print120, width     !{m}
         print130, height     !{m}
         print140, PassSpac     !{m}
         print150, Vchain     !{m/Sec}
         print160, StickSpac     !{m}
         print170, hotdogs     !{integer}
         print*
         print*
```

```
c
c ------------------------------------------------------------------
c *****************    MAKE CHANGES TO DATA ????    *****************
c ------------------------------------------------------------------
c
         i = 7
         print*,'Type the number of the item which you would like
     & to change.'
         print*,'   (type 0 [zero] when you are through making
     & changes)'
         print200
         read*,i
c
c ------------------------------------------------------------------
c ***************    WRITE OUTPUT TO HOUSE.DAT FILE    **************
c ------------------------------------------------------------------
c
         if(i .EQ. 0) then
            rewind(unit=1)
            write(1,90) width,height,PassSpac,Vchain,StickSpac,hotdogs
            close(unit=1,status='keep')
            stop
c
c ------------------------------------------------------------------
c ********    ASK USER FOR "NEW" VALUE IF CHANGE REQUESTED    *******
c ------------------------------------------------------------------
c
         elseif(i .GE. 1 .AND. i .LE. 6) then
            goto(10, 20, 30, 40, 50, 60) i
         else
            print*,'You must enter a number between 0 and 6.'
            print*,'Press return when ready to continue.'
            pause
            goto 8
         endif
10       print*,'Enter new cabinet width [m]'
         read*,width
         goto 8
20       print*,'Enter new pass height [m]'
         read*,height
         goto 8
30       print*,'Enter new pass spacing [m]'
         read*,PassSpac
         goto 8
40       print*,'Enter new chain speed [m/sec]'
         read*,Vchain
         goto 8
50       print*,'Enter new stick spacing [m]'
         read*,StickSpac
         goto 8
60       print*,'Enter new number of hot dogs per stick [integer]'
         read*,hotdogs
         goto 8
```

```
c
c ------------------------------------------------------------------
c ******************************************************************
c *******************    WORK IN ENGLISH UNITS    *******************
c ******************************************************************
c ------------------------------------------------------------------
c
      elseif(i .EQ. 2 ) then
c
c ------------------------------------------------------------------
c ***********    CONVERT FROM METRIC TO ENGLISH UNITS    ************
c ------------------------------------------------------------------
c
         width = width/0.3048
         height = height/0.3048
         PassSpac = PassSpac/0.3048
         Vchain = Vchain*((100.0*60.0)/(2.54*12.0))
         StickSpac = StickSpac/0.3048
c
c ------------------------------------------------------------------
c *************    PRINT CURRENT VALUES TO THE SCREEN    ************
c ------------------------------------------------------------------
c
9        print100
         print110
         print121, width     !{ft}
         print131, height     !{ft}
         print141, PassSpac     !{ft}
         print151, Vchain     !{ft/Min}
         print161, StickSpac     !{ft}
         print171, hotdogs     !{integer}
         print*
         print*
c
c ------------------------------------------------------------------
c *****************    MAKE CHANGES TO DATA ????    *****************
c ------------------------------------------------------------------
c
         i = 8
         print*,'Type the number of the item which you would like
     & to change.'
         print*,'   (type 0 [zero] when you are through making
     & changes)'
         print200
         read*,i
c
c ------------------------------------------------------------------
c **************    WRITE OUTPUT TO HOUSE.DAT FILE    ***************
c ------------------------------------------------------------------
c
         if(i .EQ. 0) then
            rewind(unit=1)
            write(1,90) width*0.3048,height*0.3048,PassSpac*0.3048,
     &                  Vchain/((100.0*60.0)/(2.54*12.0)),
     &                  StickSpac*.3048,hotdogs
            close(unit=1,status='keep')
```

```
            stop
c
c ------------------------------------------------------------------
c ********    ASK USER FOR "NEW" VALUE IF CHANGE REQUESTED    *******
c ------------------------------------------------------------------
c
         elseif(i .GE. 1 .AND. i .LE. 6) then
            goto(15, 25, 35, 45, 55, 65) i
         else
            print*,'You must enter a number between 0 and 6.'
            print*,'Press return when ready to continue.'
            pause
            goto 9
         endif
15       print*,'Enter new cabinet width [ft]'
         read*, width
         goto 9
25       print*,'Enter new pass height [ft]'
         read*, height
         goto 9
35       print*,'Enter new pass spacing [ft]'
         read*, PassSpac
         goto 9
45       print*,'Enter new chain speed [ft/min]'
         read*, Vchain
         goto 9
55       print*,'Enter new stick spacing [ft]'
         read*, StickSpac
         goto 9
65       print*,'Enter new number of hot dogs per stick [integer]'
         read*, hotdogs
         goto 9
      endif
c
c ------------------------------------------------------------------
c *********************    FORMAT STATEMENTS    *********************
c ------------------------------------------------------------------
c
90    format(f8.5,f8.5,f8.5,f9.6,f8.6,i6)
100   format(/////'     HOUSE DESIGN SPECIFICATIONS     ')
110   format('_____________________________________________',/)
120   format(' 1. Cabinet width = ',f5.2,' [m]')
121   format(' 1. Cabinet width = ',f5.2,' [ft]')
130   format(' 2. Pass height = ',f5.2,' [m]')
131   format(' 2. Pass height = ',f5.2,' [ft]')
140   format(' 3. Pass spacing = ',f5.2,' [m]')
141   format(' 3. Pass spacing = ',f5.2,' [ft]')
150   format(' 4. Chain speed = ',f5.4,' [m/sec]')
151   format(' 4. Chain speed = ',f5.2,' [ft/min]')
160   format(' 5. Stick spacing = ',f5.4,' [m]')
161   format(' 5. Stick spacing = ',f5.2,' [ft]')
170   format(' 6. Number of hot dogs per stick = ',i3)
171   format(' 6. Number of hot dogs per stick = ',i3)
200   format(//////////)
      end
```

```
c
c ------------------------------------------------------------------
c ******************************************************************
c ************    END OF HOUSE SPECIFICATION PROGRAM    *************
c ******************************************************************
c ------------------------------------------------------------------
```

A.1-2 Product Specification Program

The program "Product.for" is used to specify the physical parameters of the meat product to be cooked in the simulation program. The program is used to specify the following product parameters:

1. Units to be Used in Output Files (E => English or M => Meteric)
2. Product Name
3. Outer Radius of Product
4. Length of Product
5. Number of Nodes Used in Finite-Difference Approximation
6. Thermal Conductivity of Meat
7. Mass Density of Meat
8. Specific Heat of Meat
8. Initial Product Temperature

Each of these parameters must be set before the main simulation program can be run.

```
c ------------------------------------------------------------------
c ******************************************************************
c ****************    PRODUCT SPECIFICATION PROGRAM    **************
c ******************************************************************
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    *******************
c ------------------------------------------------------------------
c
c
c    This program is used to set the initial product specifications.
c    This includes the following information:
c
c       1.  Name of Product
c       2.  Outer radius of product
c       3.  Length of product
c       4.  Number of nodes used in radial direction (less than 20)
c       5.  Thermal Conductivity
c       6.  Mass density
c       7.  Specific Heat
c       8.  Initial product temperature
c
c    The program allows the user to enter these initial values in
c    either English or Metric (SI) units. The program creates a
c    file called "prod.spc" that contains the information stored
c    in metric units.  This file is then used by the simulation
c    program.  Note that the units selected in this program will
c    determine the units used in the output printed by the
c    simulation program.  For example if you elect to input the
c    created by the product data in English units then the output
c    data files simulation program will also be in English Units.
c
c
c ------------------------------------------------------------------
c ************************    VARIABLE TABLE    ********************
c ------------------------------------------------------------------
c
c
c              Ro = Outer Radius of Product {m}
c               L = Length of Product {m}
c               k = Thermal Conductivity {W/m^2-C}
c             rho = Mass Density {Kg/m^3}
c              Cp = Specific Heat {KJ/Kg-K}
c              Ti = Initial Temperature {C}
c           nodes = number of nodes
c               i = counter
c            prod = name of meat product
c           units = flag to indicate unit choice (M => Metric (SI)
c                   or E => English)
c            used = logical variable used to determine if data file
c                   already exists
c
```

```
c
c ------------------------------------------------------------------
c *********************    DECLARE VARIABLES    *********************
c ------------------------------------------------------------------
c
      implicit none
      real Ro,L,k,rho,Cp,Ti
      integer nodes,i
      character prod(20)*4,units*1
      logical used
c
c ------------------------------------------------------------------
c **********    DETERMINE IF DATA FILE ALREADY EXISTS    ************
c ------------------------------------------------------------------
c
      inquire(file='prod.dat',exist=used)
c
c ------------------------------------------------------------------
c *******************    CREATE NEW DATA FILE    *******************
c ------------------------------------------------------------------
c
      if(used .EQ. .FALSE.) then
         open(unit=1,file='prod.dat',status='new')
         units = 'E'
         prod(1) = 'new'
         do 3 i = 2,20
            prod(i) = '    '
3        continue
         Ro = 0.0
         L = 0.0
         nodes = 0
         k = 0.0
         Cp = 0.0
         Ti = -17.77777778
c
c ------------------------------------------------------------------
c ***************    READ EXISTING INPUT DATA FILE    ***************
c ------------------------------------------------------------------
c
      elseif(used .EQ. .TRUE.) then
         open(unit=1,file='prod.dat',status='old')
         rewind(unit=1)
         read(1,91) units
         read(1,95,END = 999) (prod(i), i=1,20)
         read(1,92) Ro
         read(1,92) L
         read(1,93) nodes
         read(1,96) k
         read(1,97) rho
         read(1,96) Cp
         read(1,97) Ti
      endif
c
c ------------------------------------------------------------------
c ************    DETERMINE "WORKING" UNITS FROM USER    ************
c ------------------------------------------------------------------
```

```
      print*,'Enter the system of units you would like to work in:'
      print*
      print*,'          1.  Metric {m,KJ,Kg,C,Sec}'
      print*,'          2.  English {ft,Btu,Lbm,F,Sec}'
      print*
5     read*,i
c
c ----------------------------------------------------------------
c ***********    ERROR MESSAGE IF INPUT IS NOT A 1 OR 2    **********
c ----------------------------------------------------------------
c
      if(i .LT. 1 .OR. i .GT. 2) then
         print*,'Please enter a "1" for Metric units or a "2" for
     & English units.'
         goto 5
c
c ----------------------------------------------------------------
c ****************************************************************
c *******************    WORK IN METRIC UNITS    *******************
c ****************************************************************
c ----------------------------------------------------------------
c
      elseif(i .EQ. 1 ) then
         units = 'M'
         Ro = 1000.0*Ro    !Convert from meters to millimeters
         L = 100.0*L    !Convert from meters to centimeters
c
c ----------------------------------------------------------------
c *************    PRINT CURRENT VALUES TO THE SCREEN    ************
c ----------------------------------------------------------------
c
8        print180
         print100
         print110
         print115, (prod(i), i=1,20)
         print120, Ro  !{mm}
         print121, L  !{cm}
         print130, nodes  !{integer}
         print140, k  !{W/m-C}
         print150, rho  !{Kg/m^3}
         print160, Cp  !{KJ/Kg-K}
         print170, Ti  !{C}
         print*
         print*
c
c ----------------------------------------------------------------
c ****************    MAKE CHANGES TO DATA ????    ****************
c ----------------------------------------------------------------
c
         i = 8
         print*,'Type the number of the item which you would like
     & to change.'
         print*,'   (type 0 [zero] when you are through making
     & changes)'
         print200
         read*,i
```

```
c
c ------------------------------------------------------------------
c ***************    WRITE OUTPUT TO PROD.DAT FILE    ***************
c ------------------------------------------------------------------
c
         if(i .EQ. 0) then
            rewind(unit=1)
            write(1,91) units
            write(1,95) (prod(i), i=1,20)
            write(1,92) Ro/1000.0
            write(1,92) L/100.0
            write(1,93) nodes
            write(1,96) k
            write(1,97) rho
            write(1,96) Cp
            write(1,97) Ti
            stop
c
c ------------------------------------------------------------------
c ********    ASK USER FOR "NEW" VALUE IF CHANGE REQUESTED    *******
c ------------------------------------------------------------------
c
         elseif(i .GE. 1 .AND. i .LE. 8) then
            goto(10, 20, 30, 40, 50, 60, 70, 80) i
         else
            print*,'You must enter a number between 0 and 8.'
            print*,'Press return when you are ready to continue.'
            pause
            goto 8
         endif
10       print*,'Enter new product name'
         read95,(prod(i), i=1,20)
         goto 8
20       print*,'Enter new outer radius [mm]'
         read*,Ro
         goto 8
30       print*,'Enter new sausage length [cm]'
         read*,L
         goto 8
40       print*,'Enter new number of nodes [<= 20]'
         read*,nodes
         nodes = abs(nodes)
         if(nodes .GT. 20) then
            print*,'Please enter a value of less than 20!'
            goto 30
         endif
         goto 8
50       print*,'Enter new thermal conductivity [W/m-K]'
         read*,k
         goto 8
60       print*,'Enter new mass density [Kg/m^3]'
         read*,rho
         goto 8
70       print*,'Enter new specific heat value [KJ/Kg-K]'
         read*,Cp
         goto 8
```

```
80        print*,'Enter new initial temperature of sausage [C]'
          read*,Ti
          goto 8
c
c ------------------------------------------------------------------
c ******************************************************************
c *******************    WORK IN ENGLISH UNITS    *******************
c ******************************************************************
c ------------------------------------------------------------------
c
      elseif(i .EQ. 2 ) then
          units = 'E'
c
c ------------------------------------------------------------------
c ***********    CONVERT FROM METRIC TO ENGLISH UNITS    ************
c ------------------------------------------------------------------
c
          Ro = (Ro*100.0)/2.54
          L = (L*100.0)/2.54
          k = k/1.7296
          rho = rho/16.018463
          Cp = Cp/4.184
          Ti = 1.8*Ti + 32.0
c
c ------------------------------------------------------------------
c *************    PRINT CURRENT VALUES TO THE SCREEN    ************
c ------------------------------------------------------------------
c
9         print180
          print100
          print110
          print115, (prod(i), i=1,20)
          print125, Ro  !{inch}
          print126, L  !{inch}
          print130, nodes  !{integer}
          print145, k  !{Btu/hr-ft-F}
          print155, rho  !{lbm/ft^3}
          print165, Cp  !{Btu/lbm-F}
          print175, Ti  !{F}
          print*
          print*
c
c ------------------------------------------------------------------
c *****************    MAKE CHANGES TO DATA ????    *****************
c ------------------------------------------------------------------
c
          i = 8
          print*,'Type the number of the item which you would like
     & to change.'
          print*,'   (type 0 [zero] when you are through making
     & changes)'
          print200
          read*,i
```

```
c
c ----------------------------------------------------------------
c ***************    WRITE OUTPUT TO PROD.DAT FILE    ***************
c ----------------------------------------------------------------
c
         if(i .EQ. 0) then
            rewind(unit=1)
            write(1,91) units
            write(1,95) (prod(i), i=1,20)
            write(1,92) (Ro*2.54)/100.0
            write(1,92) (L*2.54)/100.0
            write(1,93) nodes
            write(1,96) k*1.7296
            write(1,97) rho*16.018463
            write(1,96) Cp*4.184
            write(1,97) (Ti - 32.0)/1.8
            stop
c
c ----------------------------------------------------------------
c ********    ASK USER FOR "NEW" VALUE IF CHANGE REQUESTED    *******
c ----------------------------------------------------------------
c
         elseif(i .GE. 1 .AND. i .LE. 8) then
            goto(15, 25, 35, 45, 55, 65, 75, 85) i
         else
            print*,'You must enter a number between 0 and 8.'
            print*,'Press return when you are ready to continue.'
            pause
            goto 9
         endif
15       print*,'Enter new product name'
         read95,(prod(i), i=1,20)
         goto 9
25       print*,'Enter new outer radius [inch]'
         read*,Ro
         goto 9
35       print*,'Enter new sausage length [inch]'
         read*,L
         goto 9
45       print*,'Enter new number of nodes [<= 20]'
         read*,nodes
         nodes = abs(nodes)
         if(nodes .GT. 20) then
            print*,'Please enter a value of less than 20!'
            goto 35
         endif
         goto 9
55       print*,'Enter new thermal conductivity [Btu/hr-ft-F]'
         read*,k
         goto 9
65       print*,'Enter new mass density [lbm/ft^3]'
         read*,rho
         goto 9
75       print*,'Enter new specific heat value [Btu/lbm-F]'
         read*,Cp
         goto 9
```

```
85        print*,'Enter new initial temperature of sausage [F]'
          read*,Ti
          goto 9
       endif
c
c ---------------------------------------------------------------------
c *********************    FORMAT STATEMENTS    *********************
c ---------------------------------------------------------------------
c
91     format(a1)
92     format(f7.4)
93     format(i2)
95     format(20a4)
96     format(f8.4)
97     format(f8.3)
100    format(t12,'INITIAL SPECIFICATIONS OF MEAT PRODUCT')
110    format(t12,'_____________________________________',/)
115    format(' 1. Meat Product Name = ',20a4)
120    format(' 2. Outer radius of sausage = ',f6.3,' [mm]')
121    format(' 3. Length of sausage = ',f6.3,' [cm]')
125    format(' 2. Outer radius of sausage = ',f6.3,' [inch]')
126    format(' 3. Length of sausage = ',f6.3,' [inch]')
130    format(' 4. Number of nodes = ',i2)
140    format(' 5. Initial thermal conductivity = ',f7.3,
     &        ' [W/m-K]')
145    format(' 5. Initial thermal conductivity = ',f7.3,
     &        ' [Btu/hr-ft-F]')
150    format(' 6. Initial mass density = ',f7.2,' [Kg/m^3]')
155    format(' 6. Initial mass density = ',f7.2,' [lbm/ft^3]')
160    format(' 7. Initial specific heat = ',f7.3,' [KJ/Kg-K]')
165    format(' 7. Initial specific heat = ',f7.3,' [Btu/lbm-R]')
170    format(' 8. Initial temperature of sausage ',f7.2,' [C]')
175    format(' 8. Initial temperature of sausage ',f7.2,' [F]')
180    format(/////)
200    format(////////)
999    continue
       end
c
c ---------------------------------------------------------------------
c *********************************************************************
c **********    END OF PRODUCT SPECIFICATION PROGRAM    *************
c *********************************************************************
c ---------------------------------------------------------------------
```

A.1-3 Process Specification Program

The program "Process.for" is used to specify the zones which make up the cook-

ing process. The program allows the user to specify a zone and the conditions within that zone. The program is used to specify the following parameters:

1. Total Number of Zones in Data File
2. Zone Name
3. Fluid Type (air, water, or brine)
4. Number of Passes per Zone
5. Average Fluid Velocity
6. Inlet Dry Bulb Temperature
7. Inlet Wet Bulb Temperature
8. Mode of Mass Transfer

   yes => consider mass transfer

   no => neglect mass transfer)

Each of these parameters must be set before the main simulation program can be run.

```
c ---------------------------------------------------------------------
c *********************************************************************
c ****************    PROCESS SPECIFICATION PROGRAM    **************
c *********************************************************************
c ---------------------------------------------------------------------
c
c
c ---------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    ********************
c ---------------------------------------------------------------------
c
c
c    This program is used to edit the process zones.  This includes
c    entering the following information:
c
c       1. Zone name
c       2. Fluid type (water, brine, air)
```

```
c        3. Number of passes in the zone
c        4. Average free stream air velocity
c        5. Entering free stream dry bulb temperature
c        6. Average free stream wet bulb temperature
c        7. Consider mass transfer from product (yes/no)
c
c     The program allows the user to enter the above information in
c     either English or Metric (SI) units.  The program creates a
c     file called 'proc.spc' that contains the above information
c     stored in metric units.
c
c
c ------------------------------------------------------------------
c ***********************   VARIABLE TABLE    ***********************
c ------------------------------------------------------------------
c
c            Vset = Array of zone average free stream air
c                   velocities
c             Tdb = Array of zone inlet dry bulb temperatures
c             Twb = Array of zone wet bulb temperatures
c             Val = Choice of zone number that user wants to edit
c           zones = Number of existing zones
c            pass = Array of zone passes
c           units = Integer used to determine units choice
c               i = counter
c               j = counter
c               k = counter
c           fluid = Array of fluid types for each zone (air,
c                   water, or brine.
c        MassTran = Array of mass transfer modes for each zone
c                   yes => account for mass transfer from product
c                   no => do not account for mass transfer
c            zone = Array of zone names
c            hold = used to temporarily store new zone name
c            flag = used to flag units choice
c                   M => Metric (SI) units
c                   E => English units
c            used = logical variable used to determine if data file
c                   already exists
c
c
c ------------------------------------------------------------------
c *********************    DECLARE VARIABLES    *********************
c ------------------------------------------------------------------
c
      implicit none
      real Vset,Tdb,Twb,Val
      integer zones,pass,units,i,j,k
      character fluid(20)*8,MassTran(20)*8
      character zone(80)*4,hold(4)*4,flag*1
      logical used
      dimension Vset(20),Tdb(20),Twb(20),pass(20)
c
c ------------------------------------------------------------------
c **********    DETERMINE IF DATA FILE ALREADY EXISTS    ************
c ------------------------------------------------------------------
```

```
c
      inquire(file='proc.dat',exist=used)
c
c ------------------------------------------------------------------
c *******************    CREATE NEW DATA FILE    *******************
c ------------------------------------------------------------------
c
      if(used .EQ. .FALSE.) then
         open(unit=1,file='proc.dat',status='new')
         zones = 1
         zone(1) = 'new'
         zone(2) = '    '
         zone(3) = '    '
         zone(4) = '    '
         fluid(1) = 'air'
         pass(1) = 0
         Vset(1) = 0.0
         Tdb(1) = -17.7777778
         Twb(1) = -17.7777778
         MassTran(1) = 'no'
c
c ------------------------------------------------------------------
c ***************    READ EXISTING INPUT DATA FILE    ***************
c ------------------------------------------------------------------
c
      elseif(used .EQ. .TRUE.) then
         open(unit=1,file='proc.dat',status='old')
         rewind(unit=1)
         read(1,93) zones
         do 10 i = 1,zones
            read(1,95,end = 999) (zone(j), j=4*(i-1)+1,4*i)
            read(1,97) fluid(i),pass(i),Vset(i),Tdb(i),Twb(i),
     &                 MassTran(i)
10       continue
      endif
c
c ------------------------------------------------------------------
c *******    DELETE EXISTING DATA FILE AND CREATE A NEW ONE    ******
c ------------------------------------------------------------------
c
      close(unit=1,status='delete')
      open(unit=1,file='proc.dat',status='new')
      rewind(unit=1)
c
c ------------------------------------------------------------------
c ***************    PRINT INSTRUCTIONS TO SCREEN    ***************
c ------------------------------------------------------------------
c
      print*,'Edit Zone;   type the zone number of zone to be edited.'
      print*,'Add Zone;    type a zone number one grater than last'
      print*,'             existing zone.'
      print*,'Insert Zone; type a decimal for the zone number.'
      print*,'             (Ex. 1.5 to insert between zones 1 and 2)'
      print*,'Remove Zone; type zone number to be removed and when'
      print*,'             asked for zone name type "remove" (all in'
      print*,'             lower case!)'
```

```
      print*,'End Editing; type "0" [zero] for zone number to edit.'
      print180
c
c ----------------------------------------------------------------
c ************    DETERMINE "WORKING" UNITS FROM USER    ************
c ----------------------------------------------------------------
c
      print*,'Enter the system of units you would like to work in:'
      print*
      print*,'          1.  Metric {m,KJ,Kg,C,Sec}'
      print*,'          2.  English {Ft,Btu,Lbm,F,Sec}'
      print*
15    read*,units
      flag = 'M'
      if(units .LT. 1 .OR. units .GT. 2) then
         print*,'Please enter a "1" for Metric units or a "2" for
     & English units.'
         goto 15
      elseif(units .EQ. 2 ) then
c
c ----------------------------------------------------------------
c ***********    CONVERT FROM METRIC TO ENGLISH UNITS    ************
c ----------------------------------------------------------------
c
      flag = 'E'
      do 32 i = 1,zones
         Vset(i) = Vset(i)/0.3048
         Tdb(i) = 1.8*Tdb(i) + 32.0
         Twb(i) = 1.8*Twb(i) + 32.0
32    continue
      endif

c
c ----------------------------------------------------------------
c *************    PRINT CURRENT VALUES TO THE SCREEN    ************
c ----------------------------------------------------------------
c
20    print180
      print100
      if(flag .EQ. 'M') then
         print105
      else
         print106
      endif
      print110
      do 25 i = 1,zones
         print120,i,(zone(j), j=4*(i-1)+1,4*i),fluid(i),pass(i),
     &   Vset(i),Tdb(i),Twb(i),MassTran(i)
25    continue
      print*
      print*
c
c ----------------------------------------------------------------
c *****************    MAKE CHANGES TO DATA ????    *****************
c ----------------------------------------------------------------
c
```

```
      i = 0
      print*,'Type the number of the zone which you would like
     & to edit.'
      print*,'   (type 0 [zero] when you are through making changes)'
      print200
      read*,Val
      i = int(Val)
c
c ------------------------------------------------------------------
c *********    SHIFT DATA SO THAT A ZONE MAY BE INSERTED    *********
c ------------------------------------------------------------------
c
      if(i .NE. Val) then
         zones = zones + 1
         i = i+1
         do 27 j = zones,i+1,-1
            do 28 k = 4*(j-1)+1,4*j
               zone(k) = zone(k-4)
28          continue
            fluid(j) = fluid(j-1)
            pass(j) = pass(j-1)
            Vset(j) = Vset(j-1)
            Tdb(j) = Tdb(j-1)
            Twb(j) = Twb(j-1)
            MassTran(j) = MassTran(j-1)
27       continue
         do 29 k = 4*(i-1)+1,4*i
            zone(k) = '    '
29       continue
         zone(4*(i-1)+1) = 'new'
         pass(i) = 1
         Vset(i) = 0.0
         Tdb(i) = 0.0
         Twb(i) = 0.0
         MassTran(i) = 'no'
         goto 31
c
c ------------------------------------------------------------------
c *******    ASK USER FOR "NEW" VALUES IF CHANGE REQUESTED    *******
c ------------------------------------------------------------------
c
      elseif(i .GE. 1 .AND. i .LE. zones+1) then
c
c ------------------------------------------------------------------
c ***********    SHIFT DATA IF NEW ZONE IS TO BE ADDED    ***********
c ------------------------------------------------------------------
c
         if(i .EQ. zones + 1) then
            zones = zones + 1
            do 50 k = 4*(i-1)+1,4*i
               zone(k) = '    '
50          continue
            zone(4*(i-1)+1) = 'new'
            fluid(i) = 'air'
            pass(i) = 1
            Vset(i) = 0.0
```

```
            Tdb(i) = 0.0
            Twb(i) = 0.0
            MassTran(i) = 'no'
         endif
c
c ----------------------------------------------------------------
c ****************    GET NEW ZONE NAME FROM USER    ****************
c ----------------------------------------------------------------
c
31       print*,'Enter Zone Name [',(zone(j), j=4*(i-1)+1,4*i),']'
         read(*,95,end = 999) (hold(j), j=1,4)
         if(hold(1) .NE. '    ' .AND. hold(1) .NE. 'remo') then
            do 33 j = 1,4
               zone(4*(i-1)+j) = hold(j)
33          continue
c
c ----------------------------------------------------------------
c ***********    SHIFT DATA IF A ZONE IS TO BE REMOVED    ***********
c ----------------------------------------------------------------
c
         elseif(hold(1) .EQ. 'remo') then
            zones = zones - 1
            do 35 j = i,zones
               do 37 k = 1,4
                  zone(4*(j-1)+k) = zone(4*(j)+k)
37             continue
               fluid(j) = fluid(j+1)
               pass(j) = pass(j+1)
               Vset(j) = Vset(j+1)
               Tdb(j) = Tdb(j+1)
               Twb(j) = Twb(j+1)
               MassTran(j) = MassTran(j+1)
35          continue
         goto 20
         endif
c
c ----------------------------------------------------------------
c **********    GET NO. PASSES, VEL. AND TEMP. FROM USER    *********
c ----------------------------------------------------------------
c
36       print*,'Enter New Fluid Type [',fluid(i),'].'
         read*,fluid(i)
         if(fluid(i) .NE. 'air' .AND. fluid(i) .NE. 'brine'
     &      .AND. fluid(i) .NE. 'water') then
            print*,'Fluid type must be "air", "brine", or "water".'
            goto 36
         endif
         print*,'Enter Number of Passes [',pass(i),'].'
         read*,pass(i)
         print*,'Enter Free Stream Velocity [',Vset(i),'].'
         read*,Vset(i)
         print*,'Enter Free Stream Dry Bulb Temperature ['
     &          ,Tdb(i),'].'
         read*,Tdb(i)
         print*,'Enter Free Stream Wet Bulb Temperature ['
     &          ,Twb(i),'].'
```

```
         read*,Twb(i)
38       print*,'Consider Mass Transfer From Product (yes/no) ['
     &          ,MassTran(i),'].'
         read*,MassTran(i)
         if(MassTran(i) .NE. 'yes' .AND. MassTran(i) .NE. 'no') then
            print*,'Answer must be "yes" or "no".'
            goto 38
         endif
         goto 20
c
c ----------------------------------------------------------------
c ***************    WRITE OUTPUT TO PROC.DAT FILE    ***************
c ----------------------------------------------------------------
c
      elseif(i .EQ. 0) then
         if(flag .EQ. 'E') then
           do 40 j = 1,zones
              Vset(j) = Vset(j)*0.3048
              Tdb(j) = (Tdb(j) - 32.0)/1.8
              Twb(j) = (Twb(j) - 32.0)/1.8
40         continue
         endif
         write(1,93) zones
         do 30 j = 1,zones
            write(1,95) (zone(k), k=4*(j-1)+1,4*j)
            write(1,97) fluid(j),pass(j),Vset(j),Tdb(j),
     &                  Twb(j),MassTran(j)
30       continue
         stop
c
c ----------------------------------------------------------------
c ***********    ERROR MESSAGE IN ZONE NUMBER INCORRECT    **********
c ----------------------------------------------------------------
c
      else
         print*,'You must enter a number between 0 and '
     &          ,zones+1
         print*,'Press enter when you are ready to continue.'
       pause
         goto20
      endif
c
c ----------------------------------------------------------------
c ********************    FORMAT STATEMENTS    ********************
c ----------------------------------------------------------------
c
      close (unit=1,status='keep')
93    format(i5)
95    format(4a4)
97    format(a8,i5,f8.4,f8.3,f8.3,a12)
100   format(t17,'INITIAL SPECIFICATIONS OF PROCESS PARAMETERS',/)
105   format(t2,'No.  Zone  Name     Fluid    Pass   Vset  (m/s)
     &  Tdb (C)  Twb (C)    Mass Tran.')
106   format(t2,'No.  Zone  Name     Fluid    Pass   Vset (ft/s)
     &  Tdb (F)  Twb (F)    Mass Tran.')
110   format(t2,'_____________________________________________
```

```
     &_____________________________________________')
120   format(t2,i2,t7,4a4,t23,a8,t32,i2,t39,f7.3,t51,f6.1,t60,
     &       f6.1,t75,a8)
180   format(/////)
200   format(/////)
999   continue
      end
c
c ---------------------------------------------------------------------
c *********************************************************************
c **********    END OF PROCESS SPECIFICATION PROGRAM    *************
c *********************************************************************
c ---------------------------------------------------------------------
```

A.1-4 Main Simulation Program

The simulation program called "Simul.for" is used to simulate the cooking of the product using the oven design as specified in the "house.dat" file, the product parameters as specified in the "prod.dat" file, and the process parameters as specified in the "proc.dat" file. The simulation program will print out an estimated critical time step and then ask the user to enter a user time step. The user time step should be smaller than the critical time step to insure that the simulation will converge. The program will also ask the user for a output increment. The output increment is used to shorten the output data files. For example, if the critical time step for a certian zone is 0.135 seconds, the user may wish to enter a user time step of 0.1 seconds. But it would be wasteful to print out to a data file the simulated values every 0.1 second. Therefore the user can enter an output increment of 300 and then the program will only print to the data file at 30.0 second intervals.

```
c ------------------------------------------------------------------
c ******************************************************************
c *********************    SIMULATION PROGRAM    *******************
c ******************************************************************
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    *******************
c ------------------------------------------------------------------
c
c
c    NOTE: THAT THIS PROGRAM MUST BE LINKED SEPARATELY USING THE
c    MS LINK PROGRAM.  THE PROCEDURE IS TO RUN LINK THEN TYPE
c    THE PROGRAM NAME 'SIMUL' WITHOUT THE .FOR ENDING.  THEN SET
c    THE BUFFER SIZE TO Z 00050.  THEN TYPE g.  WHEN RUNNING THE
c    LINK PROGRAM IT WILL LOOK SOMETHING LIKE THIS:
c
c    f simul
c    z 00050
c    g
c
c
c    This program is designed to simulate the cooking of a product
c    in a convection oven.  It is designed to be able to simulate
c    the whole process from water bath through chill brine zones.
c
c    This program uses three data input files.
c
c    The data file called 'House.dat' is created using the program
c    'house.for'.  The data file contains the following information
c    pertaining to the house design.
c
c        1.  Cabinet Width
c        2.  Pass Height
c        3.  Pass Spacing
c        4.  Velocity of conveyance chain
c        5.  Stick Spacing
c        6.  Average number of hot dogs per stick
c
c     An example of the data file is as shown below.
c     Note column titles are not included in actual data file
c     and that the data is always stored in metric units.
c
c    Cabinet   Pass    Pass    Chain   Stick    No. of
c     Width   Height  Spacing  Veloc.  Spacing  Hot dogs
c
c    2.28600 5.02920  .34290  .027940 .508010   238
c
c    The data file called 'Prod.dat' is created using the program
c    'Product.for'.  The data file contains the following information
c    pertaining to the type of product that is to be simulated using
c    the simulation program.
c
c        1.  Name of Product
c        2.  Outer radius of product
```

```
c         3.  Length of product
c         4.  Number of nodes used in radial direction (less than 20)
c         5.  Thermal Conductivity
c         6.  Mass density
c         7.  Specific Heat
c         8.  Initial product temperature
c
c    An example of the dat file is as shown below.
c    Note that the captions in { } do not appear in the actual
c    data file, also note that all data is stored in metric units.
c
c          E   {flag used to distinguish English/Metric units}
c          Alkar Light Pork    {Product Name}
c          .0127    {Product radius}
c          .1524    {Product Length}
c          5        {Number on nodes}
c          .6054    {Thermal conductivity of meat}
c         993.145   {Mass density of meat}
c         3.5773    {Specific heat of meat}
c         32.222    {Initial temperature of product}
c
c    The data file called 'Proc.dat' is created using the program
c    'Process.for'.  The data file contains the following information
c    pertaining to the zones that make up the cooking process which
c    is to be simulated.
c
c         1. Zone name
c         2. Fluid type (water, brine, air)
c         3. Number of passes in the zone
c         4. Average free stream air velocity
c         5. Entering free stream dry bulb temperature
c         6. Average free stream wet bulb temperature
c         7. Consider mass transfer from product (yes/no)
c
c    An example data file is as shown below.
c
c          1
c     Final Cook
c     air          4  1.5651  79.444  43.333    yes
c
c    The simulation program then uses the information in the three
c    data files to simulate cooking of the product.  The program will
c    ask the user for a time step and increment used to write the
c    the data to the output file.  The program then simulates the
c    cooking of the product and creates two data files.  One data
c    file called 'output.user' presents the data in a manner that
c    is easy for the user to read.  It contains a copy of the
c    information contained in the three data files along with
c    the temperature profiles to the free stream and product.  The
c    other data file 'data.graph' contains just the temperature data
c    of the free stream and product in a comma separated variable
c    format that can be read in to a graphing program.
c
c
c ------------------------------------------------------------------
c ***********************   VARIABLE TABLE    ***********************
c ------------------------------------------------------------------
```

```
c
c
c          width = Width of one zone (house width) {m}
c         height = Height of house (distance from top to bottom of
c                  one pass {m}
c       PassSpac = Pass spacing {m}
c         Vchain = Velocity of conveyance chain {m/sec}
c      StickSpac = Distance between sticks of product on conveyance
c                  chain {m}
c             Ro = Outer radius of product {m}
c              L = Length of product {m}
c              k = Thermal conductivity of product {W/m-K}
c            rho = Mass density of product{Kg/m^3}
c             Cp = Specific heat of product {KJ/Kg-K}
c             Ti = Initial temperature of product {C}
c            Vol = Volume of product {m^3}
c             SA = Surface area of product {m^2}
c           mass = Mass of product {Kg}
c           Vset = Array of set point air velocities
c                  for each zone {m/sec}
c            Tdb = Array of set point dry bulb temperatures
c                  for each zone {C}
c            Twb = Array of set point wet bulb temperatures
c                  for each zone {C}
c              T = Array used to store product internal
c                  temperatures as the program steps through
c                  time {C}
c              R = Radius of node at which finite-difference
c                   calculation is currently at {m}
c             EM = Emissivity of product surface {dimensionless}
c             SB = Stefan-Boltzmann Constant {W/m^2-K^4}
c             pi = 3.14159 {dimensionless}
c            Hfg = Heat of vaporization of water {KJ/Kg}
c            gen = Energy generation term {W/m^3}
c           Pinf = Atmospheric pressure {KPa}
c              B = Mass transfer driving force {dimensionless}
c              g = Mass transfer conductance {Kg/m^2-sec}
c          Mh2oO = Mass fraction at product surface
c         Mh2oinf = Mass fraction at free stream conditions
c           Mdot = Mass flux {Kg/m^2-sec}
c           Loss = Percent weight loss of product due
c                  to mass transfer {%}
c            C11 = Used to calculate critical time step
c            S11 = Used to calculate critical time step
c            S12 = Used to calculate critical time step
c            S13 = Used to calculate critical time step
c         crtime = Critical time step {sec}
c             dR = Delta radius used in finite-difference
c                  calculations
c                  {m}
c          dtime = Time step used in finite-difference calculations
c                  {sec}
c          start = Time at beginning of zone {sec}
c           stop = Time at end of zone {sec}
c          theta = Time used in finite-difference calculations {sec}
c             Re = Reynolds number {dimensionless}
c              C = Constant used in Nusselt number approximation
```

```
c                    {dimensionless}
c                 m = Constant used in Nusselt number approximation
c                    {dimensionless}
c             Nubar = Average Nusselt number {dimensionless}
c                hr = Heat transfer coefficient for radiation {W/m^2-K}
c             hbarC = Heat transfer coefficient calculated using a
c                    cylinder in cross flow correlation {W/m^2-K}
c            hbarFP = Heat transfer coefficient calculated using a
c                    correlation for flow over a flat plate {W/m^2-K}
c              hbar = Average convective heat transfer coefficient
c                    calculated using hbarC and hbarFP {W/m^2-K}
c             Prair = Prandtl number of air as a function of
c                    temperature {dimensionless}
c              kair = Thermal conductivity of air as a function
c                    of temperature {W/m^2-K}
c             Cpair = Specific heat of air {J/Kg-K}
c             muair = Kinematic viscosity of air as a function
c                    of temperature {m^2/sec}
c             Prh2o = Prandtl number of water as a function of
c                    temperature {dimensionless}
c              kh2o = Thermal conductivity of water as a function
c                    of temperature {W/m^2-K}
c            Mdothd = Mass flow rate of hotdogs {Kg/sec}
c           Mdotair = Mass flow of air {Kg/sec}
c               Chd = Heat capacity rate of a hot dog {W/K}
c              Cair = Heat capacity rate of air {W/K}
c              Cmin = Minimum heat capacity rate (hot dog or air) {W/K}
c              Cmax = Maximum heat capacity rate (hot dog or air) {W/K}
c                Cr = Cmin/Cmax {dimensionless}
c              Area = Approximate heat transfer area use in heat
c                    exchanger model {m^2}
c               Ntu = Number of transfer units, used in heat exchanger
c                    model approximation {dimensionless}
c              Qmax = Maximum heat flux (used in heat exchanger modeling
c                    of hot dog and air temperature approx. {W/m^2}
c                 Q = Approximate heat flux corrected for efficiency
c                    of heat exchanger {W/m^2}
c               Eff = Calculated efficiency of heat exchanger model
c                    {dimensionless}
c                C1 = Constant used in heat exchanger calculations
c                    {dimensionless}
c                C2 = Constant used in heat exchanger calculations
c                    {dimensionless}
c                Ta = Local air temperature in the zone {C}
c               Tao = Leaving zone air temperature (bottom of zone) {C}
c               Tai = Entering zone air temperature (top of zone) {C}
c               dTa = Used for a linear approximation of the free stream
c                    dry bulb temperature distribution in zone {C}
c            psydat = Array used in psych subroutine
c            Wairin = Humidity ratio of air at top of zone
c             Wh2oO = Humidity ratio of saturated air at T = Tsurface
c            Rhoain = Density of air at inlet of zone {Kg/m^3}
c             Taout = Temperature of air at bottom of zone {C}
c             slope = Used in linear curve fit of Tdb as a function
c                    of position within zone {C/m}
c             Thdin = Surface temperature of product at inlet to
c                    zone {C}
```

```
c              i1 = Enthalpy of air at Tdb + dTdb used to estimate
c                   specific heat of air at Tdb {KJ/Kg}
c              i2 = Enthalpy of air at Tdb - dTdb used to estimate
c                   specific heat of air at Tdb {KJ/Kg}
c              W1 = Humidity ratio of air at inlet
c              AA = Constant used in efficiency calculation
c              BB = Constant used in efficiency calculation
c           Tstep = Used when seeking the iterative solution for
c                   Tseff
c          iairin = Enthalpy of air at top of zone {KJ/Kg}
c           ihdin = Enthalpy of saturated air at top of zone where
c                   T = Tsurface {KJ/Kg}
c              Cs = Effective specific heat {dimensionless}
c              T2 = Used to iterate when finding Thdout {C}
c           Tseff = Effective saturated air temperature used when
c                   iterating to fine Wairout {C}
c          Thdout = Estimated surface temperature of product at bottom
c                   of zone {C}
c         iairout = Enthalpy of saturated air at bottom of zone where
c                   T = Tsurface of product {KJ/Kg}
c           iseff = Enthalpy of effective surface at bottom of zone
c                   {KJ/Kg}
c           Wseff = Humidity ratio of saturated air at enthalpy of
c                   effect surface {dimensionless}
c           Mstar = Capacitance ratio used in cooling tower model
c         Wairout = Humidity ratio of air at bottom of zone
c           zones = Number of zones in process {integer}
c           nodes = Number of nodes used in finite-difference
c                   approximation {integer}
c           times = Number of intervals in the finite-difference
c                   approximation of the time space {integer}
c            pass = Array of the number of passes in each zone
c               i = counter used to step program through zones
c               j = counter
c               p = counter used to step program through time space
c                   in finite-difference approximation
c               n = counter
c             old = integer used in Temperature array to indicate
c                   column of old temperatures
c             new = integer used in Temperature array to indicate
c                   column of new temperatures
c         passLeng = Number of time steps in needed to complete one
c                   pass {integer}
c               w = Counter used to keep track of passes and flow
c                   direction
c           count = counter used to keep track of printing intervals
c            step = interval size used for writing data to output
c                   files
c          status = Used in pysch subroutine
c             inc = Increment used to step through Tdb function
c         hotdogs = Average number of hot dogs per stick
c           units = Character variable used to indicate E => English
c                    or M => Metric units
c            prod = Array used to store product name
c            zone = Array used to store zone names
c           fluid = Array used to store fluid type for each zone
```

```
c                    "air", "brine", or "water"
c          flowdir = Flag used to tell when product is flowing
c                    parallel with the air stream (Parallel Flow) or
c                    when the product is flowing opposite the air
c                    stream (Counter Flow)
c         MassTran = Flag used to determine mode of mass transfer
c                    yes => account for mass transfer from product
c                           product surface
c                    no => do not account for mass transfer from
c                          product surface
c
c
c ---------------------------------------------------------------------
c *********************    DECLARE VARIABLES    *********************
c ---------------------------------------------------------------------
c
      implicit none
      real width,height,PassSpac,Vchain,StickSpac
      real Ro,L,k,rho,Cp,Ti,Vol,SA,mass
      real Vset,Tdb,Twb,T,R
      real Em,SB,pi,Hfg,gen,Pinf
      real B,g,Mh2oO,Mh2oinf,Mdot,Loss
      real C11,S11,S12,S13,crtime
      real dR,dtime,start,stop,theta
      real Re,C,m,Nubar,hr,hbarC,hbarFP,hbar
      real Prair,kair,Cpair,muair
      real Prh2o,kh2o
      real Mdothd,Mdotair
      real Chd,Cair,Cmin,Cmax,Cr,Area,Ntu
      real Qmax,Q,Eff,C1,C2,Ta,Tao,Tai,dTa
      real psydat,Wairin,Wh2oO,Rhoain,Taout,slope
      real Thdin,i1,i2,W1,AA,BB,Tstep
      real iairin,ihdin,Cs,T2,Tseff
      real Thdout,iairout,iseff,Wseff,Mstar,Wairout
      integer zones,nodes,times,pass,i,j,p,n,old,new
      integer passLeng,w,count,step,status,inc,hotdogs
      character units*1,prod(20)*4,zone(80)*4,fluid(20)*8
      character flowdir*16,MassTran(20)*8
      external Prair,kair,muair
      dimension Vset(20),Tdb(20),Twb(20),pass(20),T(20,2),psydat(9)
c
c ---------------------------------------------------------------------
c ***************    READ HOUSE SPECIFICATION FILE    ***************
c ---------------------------------------------------------------------
c
      open(unit=3,file='house.dat',status='old')
      rewind(unit=3)
      read(3,801) width,height,PassSpac,Vchain,StickSpac,hotdogs
      close(unit=3,status='keep')
c
c ---------------------------------------------------------------------
c **************    READ PRODUCT SPECIFICATION FILE    **************
c ---------------------------------------------------------------------
c
      open(unit=3,file='prod.dat',status='old')
      rewind(unit=3)
```

```
      read(3,921) units
      read(3,900,END = 999) (prod(i), i=1,20)
      read(3,922) Ro
      read(3,922) L
      read(3,923) nodes
      read(3,924) k
      read(3,927) rho
      read(3,924) Cp
      read(3,927) Ti
      close(unit=3,status='keep')
c
c ------------------------------------------------------------------
c *********    INITIALIZE ARRAY OF TEMPERATURES WITH Ti    **********
c ------------------------------------------------------------------
c
      old = 1
      new = 2
      do 5 j = 1,nodes
         T(j,old) = Ti
         T(j,new) = Ti
5     continue
c
c ------------------------------------------------------------------
c **************    READ PROCESS SPECIFICATION FILE    **************
c ------------------------------------------------------------------
c
      open(unit=4,file='proc.dat',status='old')
      rewind(unit=4)
      read(4,941) zones
      do 10 i = 1,zones
         read(4,905,end = 999) (zone(j), j=4*(i-1) + 1,4*i)
         read(4,943) fluid(i),pass(i),Vset(i),Tdb(i),Twb(i),
     &              MassTran(i)
10    continue
      close(unit=4,status='keep')
c
c ------------------------------------------------------------------
c *****************    WRITE DATA TO OUTPUT FILE    *****************
c ------------------------------------------------------------------
c
      open(unit=5,file='output.user',status='new')
      open(unit=8,file='data.graph',status='new')
      rewind(unit=5)
      rewind(unit=8)
c
c ------------------------------------------------------------------
c ***************    WRITE OUTPUT IN METRIC UNITS    ****************
c ------------------------------------------------------------------
c
      if (units .EQ. 'M') then
         write(5,913)
         write(5,914)
         write(5,910) (prod(i), i=1,20)
         write(5,915) Ro*100.0
         write(5,917) L*100.0
         write(5,920) nodes
```

```
         write(5,925) k
         write(5,935) rho
         write(5,930) Cp
         write(5,938) Ti
         write(5,995)
         write(5,805)
         write(5,810)
         write(5,820) width
         write(5,830) height
         write(5,840) PassSpac
         write(5,850) Vchain
         write(5,860) StickSpac
         write(5,870) hotdogs
         write(5,995)
         write(5,940)
         write(5,945)
         write(5,955)
         do 20 i = 1,zones
            write(5,960) i,(zone(j), j=4*(i-1)+1,4*i),fluid(i)
     &                ,pass(i),Vset(i),Tdb(i),Twb(i),MassTran(i)
20       continue
c
c ----------------------------------------------------------------
c ***************    WRITE OUTPUT IN ENGLISH UNITS    ***************
c ----------------------------------------------------------------
c
      else
         write(5,913)
         write(5,914)
         write(5,910) (prod(i), i=1,20)
         write(5,916) (Ro*100.0)/2.54
         write(5,918) (L*100.0)/2.54
         write(5,920) nodes
         write(5,926) k/1.7296
         write(5,936) rho/16.018463
         write(5,931) Cp/4.184
         write(5,937) Ti*1.8 + 32.0
         write(5,995)
         write(5,805)
         write(5,810)
         write(5,821) width/0.3048
         write(5,831) height/0.3048
         write(5,841) PassSpac/0.3048
         write(5,851) Vchain*((100.0*60.0)/(2.54*12.0))
         write(5,861) StickSpac/0.3048
         write(5,871) hotdogs
         write(5,995)
         write(5,940)
         write(5,950)
         write(5,955)
         do 30 i = 1,zones
            write(5,960) i,(zone(j), j=4*(i-1)+1,4*i),fluid(i),
     &      pass(i),Vset(i)/0.3048,Tdb(i)*1.8 + 32.0,
     &      Twb(i)*1.8 + 32.0,MassTran(i)
30       continue
      endif
```

```
      write(5,995)
c
c ------------------------------------------------------------------
c *******************    PROCESS/MATH CONSTANTS    *****************
c ------------------------------------------------------------------
c
      Cp = Cp*1000.0   !Note: Convert from KJ to J
      pi = 4.0*Atan(1.0)
      Pinf = 101.325     !Note Atmospheric Pressure in KPa
      SB = 5.67e-8       !Note Stefan-Boltzmann Constant
      Hfg = (2501.4 - 2.31905*T(1,old))*1000.0
c
c     !Note Hfg is the Heat of Vaporization of Water calculated at
c           surface temperature of product
c
c
c ------------------------------------------------------------------
c *************    SET INITIAL % WEIGHT LOSS TO ZERO    *************
c ------------------------------------------------------------------
c
      Loss = 0.0
c
c ------------------------------------------------------------------
c **************    PRODUCT SUR. AREA, VOL. AND MASS    *************
c ------------------------------------------------------------------
c
      SA = pi*(2.0*Ro)*L
      Vol = pi*Ro*Ro*(L-2.0*Ro) + (4.0/3.0)*pi*Ro*Ro*Ro
      mass = rho*Vol
c
c
c ------------------------------------------------------------------
c ******************************************************************
c ****************    BEGIN MAIN CALCULATIONS LOOP    ***************
c ******************************************************************
c ------------------------------------------------------------------
c
c
      theta = 0.0   !Note: Assume process starts at time zero
      flowdir = 'Counter Flow'
      inc = 1
      w = 0
      do 800 i = 1,zones
c
c ------------------------------------------------------------------
c ******    PRINT MESSAGE TO SCREEN INDICATING CURRENT ZONE    ******
c ------------------------------------------------------------------
c
         print906, (zone(j), j=4*(i-1)+1,4*i)
c
c ------------------------------------------------------------------
c ***************    WRITE ZONE NAME TO OUTPUT FILE    **************
c ------------------------------------------------------------------
c
         write(5,995)
```

```
         write(5,905) (zone(j), j=4*(i-1)+1,4*i)
c
c ------------------------------------------------------------------
c ------------------------------------------------------------------
c ***********    HEAT TRANSFER COEFFICIENT CALCULATION    ***********
c ------------------------------------------------------------------
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c *******************    FLAT PLATE CORRELATION    ******************
c ------------------------------------------------------------------
c
c    NOTE THAT THE HEAT TRANSFER COEFFICIENT IS CALCULATED USING
c         AN EMPIRICAL CORRELATION FROM INCROPERA AND DE WITT FOR
c         FLOW OVER A FLAT PLATE.
c
         Re = (Vset(i)*L)/muair(Tdb(i))
         C = 0.664
         m = 0.5
         if(fluid(i) .EQ. 'air') then
            Nubar = C*(Re**m)*Prair(Tdb(i))**(1.0/3.0)
            hbarFP = Nubar*kair(Tdb(i))/(L)
         elseif(fluid(i) .EQ. 'water') then
            Nubar = C*(Re**m)*Prh2o(Tdb(i))**(1.0/3.0)
            hbarFP = Nubar*kh2o(Tdb(i))/(L)
         elseif(fluid(i) .EQ. 'brine') then
            Nubar = C*(Re**m)*Prh2o(0.0)**(1.0/3.0)
c
c   !NOTE: The brine temperature is assumed to be 0.0 C.
c
            hbarFP = Nubar*kh2o(0.0)/(L)
         endif
c
c ------------------------------------------------------------------
c *************    CYLINDER IN CROSS FLOW CORRELATION    ************
c ------------------------------------------------------------------
c
c    NOTE THAT THE HEAT TRANSFER COEFFICIENT IS CALCULATED USING
c         AN EMPIRICAL CORRELATION FROM INCROPERA AND DE WITT FOR
c         A CYLINDER IN CROSS FLOW.
c
         Re = (Vset(i)*2.0*Ro)/muair(Tdb(i))
         if(Re .GE. 4.0 .AND. Re .LE. 40.0) then
            C = 0.911
            m = 0.385
         elseif(Re .GT. 40.0 .AND. Re .LE. 4000.0) then
            C = 0.683
            m = 0.466
         elseif(Re .GT. 4000.0 .AND. Re .LE. 40000.0) then
            C = 0.193
            m = 0.618
         elseif(Re .GT. 40000.0 .AND. Re .LE. 400000.0) then
            C = 0.027
            m = 0.805
         endif
```

```
         if(fluid(i) .EQ. 'air') then
            Nubar = C*(Re**m)*Prair(Tdb(i))**(1.0/3.0)
            hbarC = Nubar*kair(Tdb(i))/(2.0*Ro)
         elseif(fluid(i) .EQ. 'water') then
            Nubar = C*(Re**m)*Prh2o(Tdb(i))**(1.0/3.0)
            hbarC = Nubar*kh2o(Tdb(i))/(2.0*Ro)
         elseif(fluid(i) .EQ. 'brine') then
            Nubar = C*(Re**m)*Prh2o(0.0)**(1.0/3.0)
c
c   !NOTE: The brine temperature is assumed to be 0.0 C.
c
            hbarC = Nubar*kh2o(0.0)/(2.0*Ro)
         endif
c
c ----------------------------------------------------------------
c *********    AVERAGE HTC BASED ON BOTH FLAT PLATE AND    **********
c *********      CYLINDER IN CROSS FLOW CORRELATIONS       **********
c ----------------------------------------------------------------
c
         hbar = (hbarFP + hbarC)/2.0
c
c ----------------------------------------------------------------
c **********    WRITE HEAT TRANSFER COEF. TO DATA FILE    ***********
c ----------------------------------------------------------------
c
         if (units .EQ. 'M') then
            print*,'HTC = ',hbar,' W/m^2-K'
            write(5,*) 'Convective Heat Transfer Coef. = ',
     &                 hbar,' W/m^2-K'
         else
            print*,'HTC = ',hbar/5.67796,' Btu/hr-ft^2-F'
            write(5,*) 'Convective Heat Transfer Coef. = ',
     &                 hbar/5.67796,' Btu/hr-ft^2-F'
         endif
c
c ----------------------------------------------------------------
c ***    CALCULATE SPECIFIC HEAT OF AIR USING PSYCH SUBROUTINE    ***
c ----------------------------------------------------------------
c
         psydat(1) = 1.0
         psydat(2) = Tdb(i)
         psydat(3) = Twb(i)
         call psych(0,0,1,1,0,psydat,0,status)
         W1 = psydat(6)
         psydat(2) = Tdb(i) - 0.5
         call psych(0,0,1,4,0,psydat,0,status)
         i1 = psydat(7)
         psydat(2) = Tdb(i) + 0.5
         psydat(6) = W1
         call psych(0,0,1,4,0,psydat,0,status)
         i2 = psydat(7)
         Cpair = (i2 - i1)*1000.0
c
c ----------------------------------------------------------------
c ************    DETERMINE MASS TRANSFER COEFFICIENT    ************
c ----------------------------------------------------------------
```

```
c
        g = hbar/Cpair
c
c ----------------------------------------------------------------
c ****************    RADIATION HEAT TRANSFER TERM    ***************
c ----------------------------------------------------------------
c
        Em = 0.0  !Note: Assume NO radiation to surrounding product
c
c ----------------------------------------------------------------
c ******************    ENERGY GENERATION TERM    ******************
c ----------------------------------------------------------------
c
        gen = 0.0   !Note: Assume NO energy generation
c
c ----------------------------------------------------------------
c ******************    CALCULATE DELTA RADIUS    ******************
c ----------------------------------------------------------------
c
        dR = Ro/(float(nodes) - 1.0)
c
c ----------------------------------------------------------------
c *****    CALCULATE FREE STREAM AND SURFACE HUMIDITY RATIOS    *****
c ----------------------------------------------------------------
c
        psydat(1) = 1.0   !Note pressure in Atmospheres
        psydat(2) = Tdb(i)
        psydat(3) = Twb(i)
        call psych(0,0,1,1,0,psydat,0,status)
        Wairin = psydat(6)
        iairin = psydat(7)*1000.0 !Note enthalpy in J/Kg
        Rhoain = psydat(8)
        Mh2oinf = 1.0 - (1.0/(1.0 + Wairin))
        psydat(2) = T(1,old)
        psydat(4) = 1.0
        call psych(0,0,1,2,0,psydat,0,status)
        Wh2o0 = psydat(6)
        Mh2o0 = 1.0 - (1.0/(1.0 + Wh2o0))
c
c ----------------------------------------------------------------
c **********    FIND MASS FLOW RATES OF AIR AND HOTDOGS    **********
c ----------------------------------------------------------------
c
        Mdothd = mass*float(hotdogs)*(Vchain/StickSpac)
        Mdotair = width*float(pass(i))*PassSpac*Vset(i)*Rhoain
c
c ----------------------------------------------------------------
c ****************    CALCULATE CRITICAL TIME STEP    ***************
c ----------------------------------------------------------------
c
        hr = Em*SB*((Tdb(i)+273.0)**2 + (T(1,old)+273.0)**2)
     &          *(Tdb(i)+273.0 + T(1,old)+273.0)
        B = (Mh2oinf - Mh2o0)/(Mh2o0 - 1.0)
        Mdot = g*B
        C11 = Ro*dR - 0.25*dR*dR
        S11 = (2.0*Ro*(hbar + hr - Abs(Mdot)*Hfg/(Tdb(i)+273.0))
```

```
     &          + 3.0*k*(Ro - 0.5*dR)/dR)/(rho*Cp)
          S12 = -4.0*k*(Ro - 0.5*dR)/(rho*Cp*dR)
          S13 = +k*(Ro - 0.5*dR)/(rho*Cp*dR)
          crtime = 0.5/((abs(S11) + abs(S12) + abs(S13))/C11)
c
c ------------------------------------------------------------------
c *************    ASK USER FOR CALCULATION TIME STEP    ************
c ------------------------------------------------------------------
c
          print965, crtime
          write(5,965) crtime
          dtime = crtime
40        print*
          print*,'Enter user time step (sec.)';
          read*, dtime
          print*,'Enter output interval (integer)'
          read*,step
          print*
          print*,'Calculations in Progress.'
          print*
c
c ------------------------------------------------------------------
c **************    CALCULATE TIME INTERVAL FOR ZONE    *************
c ------------------------------------------------------------------
c
          if (i .EQ. 1) then
             start = 0.0
          else
             start = stop
          endif
          stop = start + Float(pass(i))*(height/Vchain)
c
c ------------------------------------------------------------------
c ********    FIND NO. OF TIME 'NODES' AND MODIFIED DTIME    ********
c ------------------------------------------------------------------
c
          times = int((stop - start)/dtime)
          dtime = (stop - start)/float(times)
          write(5,966) dtime
          passLeng = times/pass(i)
c
c
c ------------------------------------------------------------------
c ------------------------------------------------------------------
c ****************    DETERMINE ZONE Tdb VARIATION    ***************
c ------------------------------------------------------------------
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c *************    Tdb FOR WATER OR BRINE FLUID ZONES    ************
c ------------------------------------------------------------------
c
4         if(fluid(i) .NE. 'air') then
             Taout = Tdb(i)
```

```
            slope = 0.0
c
c ------------------------------------------------------------------
c ******************    Tdb FOR TRANSITION ZONES    *****************
c ------------------------------------------------------------------
c
         elseif(fluid(i) .EQ. 'air') then
            if(zone(4*(i-1) + 1) .EQ. 'tran') then
               if(flowdir .EQ. 'Counter Flow') then
                  slope = (Tdb(i+1) - Taout)/float(passLeng)
               elseif(flowdir .EQ. 'Parallel Flow') then
                  Taout = Tdb(i)
                  slope = (Tdb(i-1) - Tdb(i))/float(passLeng)
               endif
c
c ------------------------------------------------------------------
c **********    Tdb USING SENSIBLE HEAT EXCHANGER MODEL    **********
c ------------------------------------------------------------------
c
            elseif(MassTran(i) .EQ. 'no') then
               Chd = Cp*Mdothd
               Cair = Cpair*Mdotair
               Cmin = Min1(Chd,Cair)
               Cmax = Max1(Chd,Cair)
               Cr = Cmin/Cmax
               Area = 2.0*Ro*pi*L*float(hotdogs)*float(pass(i))
     &                *(height/StickSpac)
               Ntu = (hbar*Area)/Cmin
               C1 = sqrt(1.0 + Cr*Cr)
               Eff = 2.0*(1.0 + Cr + C1*((1.0 + Exp(-Ntu*C1))/
     &               (1.0 - Exp(-Ntu*C1))))**(-1.0)
               Q = Cmin*Eff*(T(1,new) - Tdb(i))
               Taout = Tdb(i) + Q/(Mdotair*Cpair)
               slope = (Tdb(i) - Taout)/float(passLeng)
c
c ------------------------------------------------------------------
c ***************    Tdb USING COOLING TOWER MODEL    ***************
c ------------------------------------------------------------------
c
            elseif(MassTran(i) .EQ. 'yes') then
               Tstep = 20.0
               Thdin = T(1,new)
               psydat(2) = Thdin
               psydat(4) = 1.0
               call psych(0,0,1,2,0,psydat,0,status)
               ihdin = psydat(7)*1000.0 !Note enthalpy in J/Kg
               Tseff = Tdb(i)
               T2 = Thdin + 16.667 !Note initial guess value
80             psydat(2) = T2
               psydat(4) = 1.0
               call psych(0,0,1,2,0,psydat,0,status)
               i2 = psydat(7)*1000.0 !Note enthalpy in J/Kg
               Cs = (ihdin - i2)/(Thdin - T2)
               Area = 2.0*Ro*pi*L*float(hotdogs)*float(pass(i))
     &                *(height/StickSpac)
               Ntu = (hbar*Area)/(Mdotair*Cpair)
```

```
                Mstar = (Mdotair*Cs)/(Mdothd*Cp)
                C1 = sqrt(1.0 + Mstar*Mstar)
                AA = 1.0 + Exp(-Ntu*C1)
                BB = 1.0 - Exp(-Ntu*C1)
                Eff = 2.0*(1 + Mstar + C1*(AA/BB))**(-1.0)
                Q = Eff*Mdotair*(ihdin - iairin)
                iairout = iairin + Q/Mdotair
                Thdout = Thdin - Q/(Mdothd*Cp)
                if(abs((Thdout - T2)/Thdout) .GT. 0.01) then
                   T2 = Thdout
                   goto 80
                endif
                iseff = iairin + (iairout - iairin)/(1.0 - Exp(-Ntu))
70              psydat(2) = Tseff
                psydat(7) = iseff/1000.0
                call psych(0,0,1,5,0,psydat,0,status)
                if (1.0 - psydat(4) .GT. 0.001) then
73                 Tseff = Tseff - Tstep
                   psydat(2) = Tseff
                   psydat(4) = 1.0
                   call psych(0,0,1,2,0,psydat,0,status)
                   if (psydat(7)*1000.0 .LT. iseff) then
                      Tseff = Tseff + Tstep
                      Tstep = Tstep/2.0
                      goto 73
                   endif
                   goto 70
                endif
                Wseff = psydat(6)
                Wairout = Wseff + (Wairin - Wseff)*Exp(-Ntu)
                psydat(6) = Wairout
                psydat(7) = iairout/1000.0
                call psych(0,0,1,6,0,psydat,0,status)
                Taout = psydat(2)
                slope = (Tdb(i) - Taout)/float(passLeng)
             endif
          endif
c
c ------------------------------------------------------------------
c ************    WRITE COLUMN HEADINGS TO OUTPUT FILE    ***********
c ------------------------------------------------------------------
c
          count = 0
          Ta = Taout + slope*float(w)
          if (units .EQ. 'M') then
             write(5,967)
             write(5,968) ((Ro - float(n)*dR)*100.0, n = 0,nodes-1)
             write(5,975) theta,Ta,Twb(i),Loss,(T(n,old), n = 1,nodes)
             if (i .EQ. 1) then
                write(8,971) ((Ro - float(n)*dR)*100.0, n = 0,nodes-1)
                write(8,976) theta,Ta,Twb(i),Loss,(T(n,old),
     &                       n = 1,nodes)
             endif
          else
             write(5,969)
             write(5,970) (((Ro - float(n)*dR)*100.0/2.54), n = 0
```

```
     &                    ,nodes-1)
            write(5,975) theta,(Ta*1.8 + 32.0),(Twb(i)*1.8
     &                   + 32.0),Loss,((T(n,old)*1.8 + 32.0),
     &                   n = 1,nodes)
            if (i .EQ. 1) then
               write(8,971) (((Ro - float(n)*dR)*100.0/2.54), n = 0
     &                      ,nodes-1)
               write(8,976) theta,(Ta*1.8 + 32.0),(Twb(i)*1.8
     &                   + 32.0),Loss,((T(n,old)*1.8 + 32.0),
     &                   n = 1,nodes)
            endif
         endif
c
c ----------------------------------------------------------------
c ****************    BEGIN STEPPING THROUGH ZONE    ****************
c ----------------------------------------------------------------
c
         do 60 p = 1,times,1
            w = w + inc
c
c ----------------------------------------------------------------
c ******************    DETERMINE FLOW DIRECTION    *****************
c ----------------------------------------------------------------
c
            if(w .EQ. passLeng) then
               flowdir = 'Parallel Flow'
               inc = -1
            elseif(w .EQ. 0) then
               flowdir = 'Counter Flow'
               inc = 1
            endif
            Ta = Taout + slope*float(w)
c
c ----------------------------------------------------------------
c **********    CALCULATE MASS TRANSFER DRIVING FORCE-B    **********
c ----------------------------------------------------------------
c
            psydat(1) = 1.0
            psydat(2) = Ta
            psydat(3) = Twb(i)
            call psych(0,0,1,1,0,psydat,0,status)
            Wairin = psydat(6)
            Mh2oinf = 1.0 - (1.0/(1.0 + Wairin))
            psydat(2) = T(1,old)
            psydat(4) = 1.0
            call psych(0,0,1,2,0,psydat,0,status)
            Wh2o0 = psydat(6)
            Mh2o0 = 1.0 - (1.0/(1.0 + Wh2o0))
            B = (Mh2oinf - Mh2o0)/(Mh2o0 - 1.0)
            if(fluid(i) .EQ. 'air') then
               if(MassTran(i) .EQ. 'yes') then
                  Mdot = g*log(1+B)
                  mass = mass + Mdot*SA*dtime
                  Loss = Loss + 100.0*Mdot*SA*dtime/mass
               elseif(MassTran(i) .EQ. 'no') then
                  if(B .LT. 0) then
```

```
                  Mdot = g*log(1+B)
               elseif(B .GE. 0) then
                  Mdot = 0.0
               endif
            endif
         elseif(fluid(i) .NE. 'air') then
            Mdot = 0.0
         endif
c
c ---------------------------------------------------------------------
c *************    CALCULATE TEMPERATURE DISTRIBUTION    ************
c ---------------------------------------------------------------------
c
         Hfg = (2501.4 - 2.31905*T(1,old))*1000.0
c
c     !Note Hfg is the Heat of Vaporization of Water calculated at
c        surface temperature of product
c

         T(1,new) = (dtime/(rho*Cp))*(((2.0*Ro)/(Ro*dR - 0.25*
     &      dR*dR))*(Em*SB*((Ta+273.2)**4.0 - (T(1,old)+273.2)
     &      **4.0) + hbar*(Ta - T(1,old)) - Mdot*Hfg) - ((k*(Ro
     &       - 0.5*dR))/(Ro*dR - 0.25*dR*dR))*((T(3,old) - 4.0*
     &      T(2,old) + 3.0*T(1,old))/dR) + gen) + T(1,old)
         do 50 j = 2,nodes-1,1
            R = Ro - float(j-1)*dR
            T(j,new) = (dtime/(rho*Cp))*((k*(T(j+1,old)
     &                 - 2.0*T(j,old) + T(j-1,old))/(dR*dR))
     &                 + (k*(T(j-1,old) - T(j+1,old))/(2.0*dR*R))
     &                 + gen) + T(j,old)
50       continue
         T(nodes,new) = (dtime/(rho*Cp))*(gen + 2.0*k*
     &                  (-3.0*T(nodes,old) + 4.0*T(nodes-1,old)
     &                  - T(nodes-2,old))/(dR*dR)) + T(nodes,old)
c
c ---------------------------------------------------------------------
c ****************    WRITE DATA TO OUTPUT FILES    ***************
c ---------------------------------------------------------------------
c
      count = count + 1
      if (count .EQ. step) then
         if (units .EQ. 'M') then
            write(8,976) theta + float(p)*dtime/60.0,Ta,Twb(i),
     &                   Loss,(T(n,new),n = 1,nodes)
            write(5,975) theta + float(p)*dtime/60.0,Ta,Twb(i),
     &                   Loss,(T(n,new),n = 1,nodes)
          else
            write(8,976) theta + float(p)*dtime/60.0,(Ta*1.8 +
     &                 32.0),(Twb(i)*1.8 + 32.0),Loss,
     &                 ((T(n,new)*1.8 + 32.0), n = 1,nodes)
            write(5,975) theta + float(p)*dtime/60.0,(Ta*1.8 +
     &                 32.0),(Twb(i)*1.8 + 32.0),Loss,((T(n,new)
     &                 *1.8+ 32.0), n = 1,nodes)
         endif
         count = 0
```

```
         endif
c
c ----------------------------------------------------------------
c ************    SWITCH NEW AND OLD VALUES FOR ARRAYS    ***********
c ----------------------------------------------------------------
c
            if (new .EQ. 2) then
               new = 1
               old = 2
            else
               new = 2
               old = 1
            endif
60       continue
c
c ----------------------------------------------------------------
c ***************    INCREASE TIME BY ONE INTERVAL    ***************
c ----------------------------------------------------------------
c
         theta = theta + float(times)*dtime/60.0
800   continue
c
c ----------------------------------------------------------------
c *********************    CLOSE OUTPUT FILE    *********************
c ----------------------------------------------------------------
c
      close(unit=5,status='keep')
      close(unit=8,status='keep')
c
c ----------------------------------------------------------------
c *********************    FORMAT STATEMENTS    *********************
c ----------------------------------------------------------------
c
801    format(f8.5,f8.5,f8.5,f9.6,f8.6,i6)
805   format('     HOUSE DESIGN SPECIFICATIONS     ')
810   format('_____________________________________')
820   format('Cabinet width = ',f5.2,' [m]')
821   format('Cabinet width = ',f5.2,' [ft]')
830   format('Pass height = ',f5.2,' [m]')
831   format('Pass height = ',f5.2,' [ft]')
840   format('Pass spacing = ',f5.2,' [m]')
841   format('Pass spacing = ',f5.2,' [ft]')
850   format('Chain speed = ',f5.4,' [m/sec]')
851   format('Chain speed = ',f5.2,' [ft/min]')
860   format('Stick spacing = ',f5.4,' [m]')
861   format('Stick spacing = ',f5.2,' [ft]')
870   format('Number of hot dogs per stick = ',i3)
871   format('Number of hot dogs per stick = ',i3)
900   format(20a4)
905   format(4a4)
906   format('Calculating: ',4a4)
910   format('Product Name = ',20a4)
913   format('                PRODUCT SUMMARY')
914   format('_______________________________________________')
915   format('Outer Radius = ',f6.3,' [cm]')
916   format('Outer Radius = ',f6.3,' [inch]')
```

```
917   format('Length of Sausage = ',f6.3,' [cm]')
918   format('Length of Sausage = ',f6.3,' [inch]')
920   format('Number of Nodes = ',i3)
921   format(a1)
922   format(f7.4)
923   format(i2)
924   format(f8.4)
927   format(f8.3)
925   format('Thermal Conductivity = ',f6.3,' [W/m-C]')
926   format('Thermal Conductivity = ',f7.3,' [Btu/ft-hr-F]')
930   format('Specific Heat = ',f6.3,' [KJ/Kg-K]')
931   format('Specific Heat = ',f6.3,' [Btu/lbm-R]')
935   format('Mass Density = ',f6.2,' [Kg/m^3]')
936   format('Mass Density = ',f6.2,' [lbm/ft^3]')
937   format('Initial Temperature = ',f7.2,' [F]')
938   format('Initial Temperature = ',f7.2,' [C]')
940   format(t35,'PROCESS SUMMARY',/)
941   format(i5)
943   format(a8,i5,f8.4,f8.3,f8.3,a12)
945   format(t2,'No.  Zone  Name     Fluid    Pass   Vset  [m/s]
     &  Tdb [C]  Twb [C]    MassTran')
950   format(t2,'No.  Zone  Name     Fluid    Pass   Vset [ft/s]
     &  Tdb [F]  Twb [F]    MassTran')
955   format(t2,'----------------------------------------
     &-------------------------------------')
960   format(t2,i2,t7,4a4,t23,a8,t32,i2,t39,f7.3,t51,f6.1,t60,
     &      f6.1,t73,a8)
965   format('Critical time step = ',f7.4,' seconds.')
966   format('Actual time step used = ',f7.4,' seconds.')
967   format(/,'   Time   ','   Tdb  ','   Twb  ',' Weight ',
     &       '   Radius (cm)')
968   format(' Minutes  ','    C   ','    C   ',' Loss % ',
     &t36,f6.3,t44,f6.3,t52,f6.3,t60,f6.3,t68,f6.3,t76,f6.3,t84,
     &f6.3,t92,f6.3,t100,f6.3,t108,f6.3,t116,f6.3,t124,f6.3,t132,
     &f6.3,t140,f6.3,t148,f6.3,t156,f6.3,t164,f6.3)
969   format(/,'   Time   ','   Tdb  ','   Twb  ',' Weight ',
     &       '   Radius (inch)')
970   format(' Minutes  ','    F   ','    F   ',' Loss % ',
     &t36,f6.3,t44,f6.3,t52,f6.3,t60,f6.3,t68,f6.3,t76,f6.3,t84,
     &f6.3,t92,f6.3,t100,f6.3,t108,f6.3,t116,f6.3,t124,f6.3,t132,
     &f6.3,t140,f6.3,t148,f6.3,t156,f6.3,t164,f6.3)
971   format('Time (Min.)',',',',Tdb',',',Twb',',',%Loss'
     &  ,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3
     &  ,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3
     &  ,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3,',',f6.3
     &  ,',',f6.3)
975   format(f7.3,t12,f6.1,t20,f6.1,t28,f6.2,t36,f6.1,t44,f6.1,
     &t52,f6.1,t60,f6.1,t68,f6.1,t76,f6.1,t84,f6.1,t92,f6.1,t100,f6.1
     &t108,f6.1,t116,f6.1,t124,f6.1,t132,f6.1,t140,f6.1,t148,f6.1,
     &t156,f6.1,t164,f6.1)
976   format(f7.3,',',f6.1,',',f6.1,',',f7.2,',',f6.1,',',f6.1
     &  ,',',f6.1,',',f6.1,',',f6.1,',',f6.1,',',f6.1,',',f6.1
     &  ,',',f6.1,',',f6.1,',',f6.1,',',f6.1,',',f6.1,',',f6.1
     &  ,',',f6.1,',',f6.1,',',f6.1,',',f6.1,',',f6.1)
995   format(///)
999   continue
      end
```

```
c ------------------------------------------------------------------
c ******************************************************************
c ***********    END OF VARIABLE Tdb HOT DOG SIMULATION    **********
c ******************************************************************
c ------------------------------------------------------------------
c
c
c
c
c
c ------------------------------------------------------------------
c *****************    VARIABLE FLUID PROPERTIES    *****************
c ------------------------------------------------------------------
c
c
c
c *********    VARIABLE AIR KINEMATIC VISCOSITY FUNCTION    *********
c
      real function muair(temp)
      real temp,t
      t = temp + 273.2
      muair = -1.3018e-5 + 9.6105e-8*t
      return
      end
c
c *********    VARIABLE AIR THERMAL CONDUCTIVITY FUNCTION    ********
c
      real function kair(temp)
      real temp,t
      t = temp + 273.2
      kair = 4.4064e-3 + 7.2241e-5*t
      return
      end
c
c ************    VARIABLE AIR PRANDTL NUMBER FUNCTION    ***********
c
      real function Prair(temp)
      real temp,t
      t = temp + 273.2
      Prair = 0.85841412 - 8.0150294e-4*t + 1.0364703e-6*t*t
      return
      end
c
c **********    VARIABLE WATER PRANDTL NUMBER FUNCTION    ***********
c
      real function Prh2o(temp)
      real temp,M0,M1,M2,M3,M4,M5
      M0 = 13.436714401
      M1 = -0.50060019146
      M2 = 0.011704256979
      M3 = -0.00016545789809
      M4 = 1.2452979121e-06
      M5 = -3.7737493848e-09
      Prh2o = M0 + M1*temp + M2*temp**2 + M3*temp**3 + M4*temp**4
     &        + M5*temp**5
      return
      end
```

```
c
c *******    VARIABLE WATER THERMAL CONDUCTIVITY FUNCTION    ********
c
      real function kh2o(temp)
      real temp
      kh2o = 0.56292582747 + 0.0019712853789*temp
     &       - 8.3483321882e-06*temp**2
      return
      end
```

An external subroutine required to run the simulation program calculates psychometeric properties. The subroutine "psych.for" was copied from the TRNSYS program and is reprinted here.

```
      SUBROUTINE PSYCH(TIME,INFO,IUNITS,MODE,WBMD,PSYDAT,EMODE,STATUS)
C*****LAST MODIFIED 1/25/90 -- REU
      COMMON /LUNITS/LUR,LUW,IFORM,LUK
      INTEGER WBMD,WBMODE,EMODE,STATUS
      DIMENSION INFO(10),PSYDAT(9)
      DATA RA/287.055/, PAOLD/0./
C
C  MODES 1-5:
C  THESE MODES TAKE AS INPUT PATM (IN ATMOSPHERES), A DRY BULB
C  TEMP., AND ONE OTHER PROPERTY: WET BULB TEMP.,
C  REL.HUMIDITY(FRACTION),
C  DEW PT.TEMP.,HUMIDITY RATIO, OR ENTHALPY, DEPENDING ON MODE.
C  OUTPUTS ARE HUMIDITY RATIO (OR REL.HUMIDITY IN MODE 4), WET BULB
C  TEMP.,
C  ENTHALPY (OR REL.HUMIDITY IN MODE 5), MIXTURE DENSITY, AND DRY
C  AIR DENSITY.
C
C  MODE 6:
C  MODE 6 TAKES AS AN INPUT PATM, HUMIDITY RATIO AND ENTHALPY, AND
C  RETURNS ALL OF THE OTHER PROPERTIES.
C
C  THE WET BULB TEMPERATURE IS ONLY CALCULATED IF WBMODE EQUALS ONE.
C
C  TEMPERATURES ARE IN CELSIUS (IUNITS=1) OR FAHRENHEIT (IUNITS=2).
C  ENTHALPY IS IN KJ/KG (IUNITS=1) OR BTU/LBM (IUNITS=2), AND
C  DENSITY IS IN KG/M**3 (IUNITS=1) OR LBM/FT**3 (IUNITS=2).
C  THE REFERENCE STATES FOR ENTHALPIES ARE:
C         HAIR=0.0 AT 0. DEG C AND 0. DEG F
C         HW(LIQUID)=0.0 AT 0. DEG C AND 32. DEG F
C
C  THE INFO ARRAY CONTAINS THE TRNSYS UNIT AND TYPE NUMBER FROM THE
C  COMPONENT WHICH CALLS THE PSYCH ROUTINE.
C  THE PSYDAT ARRAY CONTAINS THE MOIST AIR PROPERTIES.
C
```

```
C  ASSIGN LOCAL VARIABLES:
      WBMODE = WBMD
      PATM = PSYDAT(1)
      TDB =  PSYDAT(2)
      TWB =  PSYDAT(3)
      TWBOLD=-9.999E20
      RH =   PSYDAT(4)
      TDP =  PSYDAT(5)
      W =    PSYDAT(6)
      H =    PSYDAT(7)
      STATUS = 0
C
C  UNIT CONVERSIONS
C
      IF(IUNITS .EQ. 2) THEN
        TDB = (TDB - 32.)/1.8
        TWB = (TWB - 32.)/1.8
        TDP = (TDP - 32.)/1.8
        H   = (H - 7.687)/0.43002
      ENDIF
C-- CHECK THAT THE TOTAL PRESSURE IS WITHIN THE IDEAL GAS RANGE.
      IF (PATM .GT. 5.0) THEN
         CALL PSYERR(TIME,INFO,EMODE,1,STATUS)
      END IF
      IF (PATM .LE. 0.0) THEN
         WRITE(LUW,250) INFO(1),INFO(2)
         STOP
      END IF
C
C   FOR MODE 6, CHECK THAT THE ENTHALPY IS GREATER THAN THE
C   SATURATION ENTHALPY (MINIMUM) FOR THE GIVEN HUMIDITY RATIO
C   AND THAT THE HUMIDITY RATIO IS GREATER THAN 0.
C
      IF (MODE .EQ. 6) THEN
            IF (W .LT. 0.) THEN
               CALL PSYERR(TIME,INFO,EMODE,3,STATUS)
               W = 0.
               GO TO 99
            ELSE IF (W .EQ. 0.) THEN
               GO TO 99
            END IF
         CALL DEWPT(INFO,PATM,W,TDP,STATUS,EMODE)
         HDP = TDP + W*(2501. + 1.805*TDP)
         HMIN = HDP
           IF (H .LT. HMIN) THEN
               CALL PSYERR(TIME,INFO,EMODE,2,STATUS)
               H = HMIN
           END IF
99       TDB = (H - 2501.*W)/(1. + 1.805*W)
       END IF
C
C  FIND SATURATION PRESSURE OF WATER AT WET BULB, DRY BULB, OR
C  DEW POINT TEMPERATURE.
C
      CALL SAT(INFO,TDB,PSATDB,STATUS,EMODE)
```

```
      GOTO (1,2,3,2,2,2) ,MODE
C-- CHECK FOR IMPOSSIBLE WET BULB TEMPERATURES AND CORRECT THEM
C   IF POSSIBLE.
1     IF (TWB .GT. TDB .AND. PSATDB .GE. PATM) THEN
         WRITE(LUW,1750) INFO(1),INFO(2)
         STOP
      ELSE IF (TWB .GT. TDB) THEN
         CALL PSYERR(TIME,INFO,EMODE,4,STATUS)
         TWB = TDB
         PSAT = PSATDB
         GOTO 5
      END IF
      CALL SAT (INFO,TWB,PSAT,STATUS,EMODE)
C-- ERROR: IF PSATWB IS GREATER THAN PATM.
         IF (PSAT .GE. PATM) THEN
             WRITE(LUW,1750) INFO(1),INFO(2)
             STOP
         END IF
      GOTO 5
2     PSAT = PSATDB
      GOTO 5
C-- CHECK FOR IMPOSSIBLE DEW POINT TEMPERATURES AND
C   CORRECT THEM IF POSSIBLE.
3     IF (TDP .GT. TDB .AND. PSATDB .GE. PATM) THEN
         WRITE(LUW,2250) INFO(1),INFO(2)
         STOP
      ELSE IF (TDP .GT. TDB) THEN
         CALL PSYERR(TIME,INFO,EMODE,5,STATUS)
         TDP=TDB
      END IF
      CALL SAT(INFO,TDP,PSAT,STATUS,EMODE)
C-- ERROR: IF PSATDP IS GREATER THAN PATM.
      IF (PSAT .GE. PATM) THEN
         WRITE(LUW,2250) INFO(1),INFO(2)
         STOP
      END IF
5     CONTINUE
C
C   CALCULATE HUMIDITY RATIO AND WET BULB TEMPERATURE
C
      GO TO (10,20,30,40,50,60), MODE
C
C  MODE 1 -- DRY BULB AND WET BULB SUPPLIED
C
10    IF (TWB .LE. 0.) THEN
        P = PSAT - 5.704E-4*(TDB-TWB)*PATM
        W = .62198 * P/(PATM-P)
      ELSE
        WSAT = .62198 * PSAT/(PATM-PSAT)
        W = WSAT - (TDB-TWB)*(0.24 + .441*WSAT)/(597.31
     .       + 0.441*TDB - TWB)
      ENDIF
      IF (W .LT. 0.0) THEN
         CALL PSYERR(TIME,INFO,EMODE,6,STATUS)
         W = 0.0
         H = TDB
```

```
         TWBOLD = TWB
         WBMODE = 1
         PSAT = PSATDB
         GO TO 60
      END IF
      H = TDB + W*(2501. + 1.805*TDB)
      GO TO 100
C
C  MODE 2 -- DRY BULB AND RELATIVE HUMIDITY SUPPLIED
C
20    IF (RH .LT. 0.) THEN
          CALL PSYERR(TIME,INFO,EMODE,7,STATUS)
          RH=0.0
      ELSE IF (PSAT .GE. PATM) THEN
          RHMAX = PATM/PSAT
          IF (RH .GE. (.99*RHMAX)) THEN
             WRITE(LUW,3750) INFO(1),INFO(2)
             STOP
          END IF
      ELSE IF (RH .GT. 1.) THEN
          CALL PSYERR(TIME,INFO,EMODE,8,STATUS)
          RH=1.0
      END IF
      W = .62198 * PSAT*RH/(PATM-PSAT*RH)
      GO TO 40
C
C  MODE 3 -- DRY BULB AND DEW POINT SUPPLIED
C
30    W = .62198 * PSAT/(PATM-PSAT)
C
C  FIND ENTHALPY FOR MODES 2 - 4
C
40    IF (PSATDB .LT. PATM) THEN
         WMAX = .62198 * PSATDB/(PATM-PSATDB)
         IF (W .GT. WMAX) THEN
             CALL PSYERR(TIME,INFO,EMODE,9,STATUS)
             W = WMAX
         END IF
      END IF
      IF (W .LT. 0.0) THEN
          CALL PSYERR(TIME,INFO,EMODE,3,STATUS)
          W = 0.
      END IF
      H = TDB + W*(2501. + 1.805*TDB)
      GO TO 60
C
C  MODE 5 -- DRY BULB AND ENTHALPY SUPPLIED
C
50    IF (PSATDB .LT. PATM) THEN
         WMAX = .62198 * PSAT/(PATM-PSAT)
         HMAX = TDB + WMAX*(2501. + 1.805*TDB)
         IF (H .GT. HMAX) THEN
             CALL PSYERR(TIME,INFO,EMODE,10,STATUS)
             H = HMAX
         END IF
      END IF
```

```
      HMIN = TDB
      IF (H .LT. HMIN) THEN
          CALL PSYERR(TIME,INFO,EMODE,11,STATUS)
          H = HMIN
      END IF
        W = (H-TDB)/(2501.+1.805*TDB)
C
C  FIND WET BULB TEMPERATURE FOR MODES 2 - 6 IF WBMODE EQUALS 1.
C
60    IF (WBMODE .NE. 1) THEN
         TWB=TDB
         GO TO 100
      END IF
      DPRESS = ABS(1.-PATM)
C  THE FOLLOWING CORRELATION IS FOR 1 ATMOSPHERES TOTAL PRESSURE.
C  IF OUTSIDE THE CORRELATION RANGE, THE CORRELATION IS USED FOR
C  THE INITIAL GUESS IN THE ITERATIVE METHOD.
      IF (H .GT. 0. .AND. H .LT. 2000.) THEN
          Y = ALOG(H*.43002+7.687)
          IF (H .LE. 9.473) THEN
             TWB=-17.4422+1.9356*Y+.7556*Y**2+.5406*Y**3
          ELSE IF (H .GT. 9.473) THEN
             TWB=-.6008-22.04556*Y+11.4356*Y**2-.97667*Y**3
          END IF
       ELSE
          TWB = 9.99999E25
       END IF
C  USE A NEWTON'S ITERATIVE METHOD TO FIND THE WET BULB
C  TEMPERATURE.
       IF (DPRESS.GT..001 .OR. H.LE.0. .OR. H.GT.275.) THEN
          ITEST=0
          IF (ABS(PATM-PAOLD) .GT. 1.0E-10) THEN
          CALL BOIL(PATM,TBOIL)
          PAOLD = PATM
          END IF
C-- INITIAL GUESS
          TWBNEW=AMIN1(TWB,(TBOIL-0.1),TDB)
70        IF (TWBNEW .GE. (TBOIL-0.09)) TWBNEW=TBOIL-0.1
          CALL SAT(INFO,TWBNEW,PSAT,STATUS,EMODE)
          WSSTAR=.62198*PSAT/(PATM-PSAT)
          IF(MODE .EQ. 5) THEN
            W = (H-TDB)/(2501.+1.805*TDB)
          END IF
          WNEW=((2501.-2.381*TWBNEW)*WSSTAR-(TDB-TWBNEW))/
     .          (2501.+1.805*TDB-4.186*TWBNEW)
          ERR = W - WNEW
          IF (ABS(ERR) .LE. (.01*W)) GO TO 75
          IF (W .EQ. 0.) THEN
             IF (ABS(ERR) .LE. .0001) GO TO 75
          END IF
          ITEST = ITEST + 1
          IF (ITEST .GE. 25) GO TO 75
C--FIND THE SLOPE OF THE ERROR FUNCTION
          TSLOPE = 0.999*TWBNEW
          IF (TWBNEW .EQ. 0.) TSLOPE = -.005
          CALL SAT(INFO,TSLOPE,PSLOPE,STATUS,EMODE)
```

```
          WSSLP=.62198*PSLOPE/(PATM-PSLOPE)
          WSLOPE=((2501.-2.381*TSLOPE)*WSSLP-(TDB-TSLOPE))/
     .          (2501.+1.805*TDB-4.186*TSLOPE)
          ERRSLP = W - WSLOPE
          DERRDT = (ERRSLP - ERR)/(TSLOPE - TWBNEW)
          TWBNEW = TWBNEW - ERR/DERRDT
          GO TO 70
75        TWB = TWBNEW
      END IF
      IF (TWB .LT. TWBOLD) TWB = TWBOLD
100   CONTINUE
C
C  FIND RELATIVE HUMIDITY, DEW POINT, MIXTURE DENSITY,
C  AND DRY AIR DENSITY
C
      PV  = PATM*W/(.62198+W)
      IF(MODE .NE. 2) RH = PV/PSATDB
      IF(MODE .NE. 3 .AND. PV .GT. 0.) THEN
        CALL DEWPT(INFO,PATM,W,TDP,STATUS,EMODE)
      ELSE IF (MODE .NE. 3 .AND. PV .LE. 0) THEN
C  FOR DRY AIR, THERE IS NO DEW POINT TEMPERATURE
        TDP=-9.99999E25
      ENDIF
      SPCVOL = RA*(TDB+273.15)/(PATM*101325)*(1+1.6078*W)
      RHOWA = 1/SPCVOL
      RHOWM = RHOWA*(1+W)
C
C  CONVERT OUTPUTS TO APPROPRIATE UNITS
C
      IF(IUNITS .EQ. 2) THEN
        H   = H*0.43002+7.68
        TDB = 1.8*TDB + 32.
        TWB = 1.8*TWB + 32.
        IF (TDP .GT. -9.99999E24) THEN
         TDP = 1.8*TDP + 32.
        END IF
        RHOWM = RHOWM/16.02
        RHOWA = RHOWA/16.02
      ENDIF
C SET OUTPUTS
      IF(MODE .EQ. 6) PSYDAT(2) = TDB
      PSYDAT(3) = TWB
      PSYDAT(4) = RH
      PSYDAT(5) = TDP
      PSYDAT(6) = W
      PSYDAT(7) = H
      PSYDAT(8) = RHOWM
      PSYDAT(9) = RHOWA
C
      RETURN
C
250   FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
     .1X,'***',/6X,'THE PSYCHROMETRICS
     . SUBROUTINE WAS CALLED WITH AN ATMOSPHERIC PRESSURE',/6X,
     .'LESS THAN OR EQUAL TO 0.')
1750  FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
```

```
     .1X,'***',/6X,'THE PSYCHROMETRICS
     . SUBROUTINE WAS CALLED WITH A WET BULB TEMPERATURE',/6X,
     .'ABOVE THE MAXIMUM FOR THE GIVEN ATMOSPHERIC PRESSURE.')
2250  FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
     .1X,'***',/6X,'THE PSYCHROMETRICS
     . SUBROUTINE WAS CALLED WITH A DEW POINT TEMPERATURE',/6X,
     .'ABOVE THE MAXIMUM FOR THE GIVEN ATMOSPHERIC PRESSURE.')
3750  FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
     .1X,'***',/6X,'THE PSYCHROMETRICS
     . SUBROUTINE WAS CALLED WITH A RELATIVE HUMIDITY',/6X,
     .'GREATER THAN POSSIBLE AT THE GIVEN DRY BULB TEMPERATURE.')
6000  FORMAT(/30X,'*** ERROR ***',/6X,'THE SIMULATION WAS HALTED
     . BECAUSE THERE WERE',1X,I3,1X,'WARNINGS FROM THE',/6X,
     .'PSYCHROMETRICS SUBROUTINE.  CHECK FOR PROPER USE OF
     . THIS SUBROUTINE.')
      END
C
C*********************************************************************
C
C  SUBROUTINE FOR FINDING SATURATION PRESSURE OF WATER AT A GIVEN
C  TEMPERATURE
C
      SUBROUTINE SAT(INFO,TIN,PSAT,STATUS,EMODE)
      DIMENSION INFO(10)
      INTEGER EMODE,STATUS
      COMMON/LUNITS/LUR,LUW,IFORM,LUK
C
C  THE FOLLOWING CORRELATION FOR THE SATURATION PRESSURE OF
C  WATER VAPOR (IN PASCALS) AS A FUNCTION OF TEMPERATURE IS
C  TAKEN FROM THE 1985 ASHRAE FUNDAMENTALS HANDBOOK (SI).
C
      DATA C1/-5674.5359/,C2/6.3925247/,C3/-0.9677843E-2/
      DATA C4/0.62215701E-6/,C5/0.20747825E-8/,C6/-0.9484024E-12/
      DATA C7/4.1635019/,C8/-5800.2206/,C9/1.3914993/
      DATA C10/-0.048640239/,C11/0.41764768E-4/,C12/-0.14452093E-7/
      DATA C13/6.5459673/
C
      T = TIN + 273.15
      IF (T .LE. 0.) THEN
         WRITE(LUW,200) INFO(1),INFO(2)
         STOP
      END IF
C
C  SATURATION PRESSURE OVER ICE (-100 C TO 0 C)
C
      IF (T .LT. 273.15) THEN
         PSAT=EXP(C1/T+C2+C3*T+C4*T**2+C5*T**3+C6*T**4+C7*ALOG(T))
C
C  SATURATION PRESSURE OVER LIQUID WATER (0 C TO 200 C)
C
      ELSE IF (T .GE. 273.15) THEN
         PSAT=EXP(C8/T+C9+C10*T+C11*T**2+C12*T**3+C13*ALOG(T))
      END IF
C
C  TEMPERATURE OUT OF THE RANGE USED FOR THE CORRELATION
C
```

```
      IF (T .LT. 173.15 .OR. T .GT. 473.15) THEN
           CALL PSYERR(TIME,INFO,EMODE,12,STATUS)
      END IF
C  CONVERT PRESSURE FROM PASCALS TO ATMOSPHERES
      PSAT = PSAT/101325
      RETURN
C
200   FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
     .1X,'***',/6X,'THE "SAT" SUBROUTINE OF THE PSYCHROMETRICS
     . SUBROUTINE WAS CALLED',/6X,'WITH A TEMPERATURE BELOW -273 C.')
      END
C
C*****************************************************************
C
C  SUBROUTINE FOR FINDING THE DEW POINT TEMPERATURE GIVEN
C  THE HUMIDITY RATIO.  THE CORRELATION IS FROM THE 1981
C  ASHRAE FUNDAMETALS HANDBOOK.  THE DEW POINT TEMPERATURE
C  IS IN DEGREES C AND ATMOSPHERIC PRESSURE IN PASCALS.
C
      SUBROUTINE DEWPT(INFO,PATM,W,TDP,STATUS,EMODE)
      DIMENSION INFO(10)
      INTEGER EMODE,STATUS
      COMMON /LUNITS/ LUR,LUW,IFORM,LUK
C
      PV  = PATM*W/(.62198+W)
      Y = ALOG(1.013E05*PV)
      TDP = -35.957 - 1.8726*Y + 1.1689*Y*Y
      IF(TDP .LT. 0.) TDP = -60.45 + 7.0322*Y + 0.3700*Y*Y
C
      IF (TDP.GT.70. .OR. TDP.LT.-60.) THEN
         CALL PSYERR(TIME,INFO,EMODE,13,STATUS)
      END IF
      RETURN
      END
C
C*****************************************************************
C
C SUBROUTINE FOR FINDING THE BOILING TEMPERATURE OF WATER GIVEN
C THE TOTAL PRESSURE.  A NEWTON'S METHOD IS USED WITH THE
C SATURATED WATER VAPOR PRESSURE CORRELATION FROM THE 1985 ASHRAE
C FUNDAMENTALS HANDBOOK.
C
      SUBROUTINE BOIL(PATM,TBOIL)
C
      DATA C1/-5674.5359/,C2/6.3925247/,C3/-0.9677843E-2/
      DATA C4/0.62215701E-6/,C5/0.20747825E-8/,C6/-0.9484024E-12/
      DATA C7/4.1635019/,C8/-5800.2206/,C9/1.3914993/
      DATA C10/-0.048640239/,C11/0.41764768E-4/,C12/-0.14452093E-7/
      DATA C13/6.5459673/
C
      PBOIL = PATM*101325
      ITEST = 0
      IF (PBOIL .LT. 611.21) GO TO 100
C--USING ASHRAE CORRELATION FOR 0 C TO 200 C.
C  FIRST GUESS TBOIL EQUALS 100 C.
      T1 = 100 + 273.15
```

```
10      ZZ=C8/T1+C9+C10*T1+C11*T1**2+C12*T1**3+C13*ALOG(T1)
        P1=EXP(ZZ)
        ERR = PBOIL - P1
        IF (ABS(ERR) .LE. (.01*PBOIL)) THEN
            TBOIL = T1 - 273.15
            RETURN
        END IF
        ITEST = ITEST +1
        IF (ITEST .GT. 100) STOP
        DERRDT=-(P1)*(-C8/T1**2+C10+2*C11*T1+3*C12*T1**2+C13/T1)
        TOLD = T1
        T1 = T1 - ERR/DERRDT
        GO TO 10
C
100     CONTINUE
C--USING ASHRAE CORRELATION FOR -100 C TO 0 C.
C  FIRST GUESS TBOIL EQUALS 0 C.
        T1 = 273.15
110     ZZ=C1/T1+C2+C3*T1+C4*T1**2+C5*T1**3+C6*T1**4+C7*ALOG(T1)
        P1=EXP(ZZ)
        ERR = PBOIL - P1
        IF (ABS(ERR) .LE. (.01*PBOIL)) THEN
            TBOIL = T1 - 273.15
            RETURN
        END IF
        ITEST = ITEST +1
        DERRDT=-P1*(-C1/T1**2+C3+2*C4*T1+3*C5*T1**2+4*C6*T1**3+C7/T1)
        TOLD = T1
        T1 = T1 - ERR/DERRDT
        GO TO 110
        END
C
C
*************************************************************************
C
C  SUBROUTINE FOR HANDLING OUTPUT OF WARNING MESSAGES
C
        SUBROUTINE PSYERR(TIME,INFO,EMODE,ENUM,STATUS)
C
C  EMODE   IS EITHER 0, 1, OR 2.  IF EQUAL TO:
C          0 - DONOT PRINT WARNING MESSAGES, RETURN WARNING NUMBER
C              (STATUS)
C          1 - PRINT ONLY ONE WARNING FOR THE ENTIRE SIMULATION
C          2 - PRINT ONE WARNING EACH TIMESTEP
C       WARNINGS ARE PRINTED FOR EACH WARNING CONDITION AND EACH UNIT
C       (UP TO IUMAX) CALLING PSYCH
C
        INTEGER ENUM,STATUS,EMODE,UNIT
        LOGICAL MAXU
C
C  IUMAX   IS THE MAXIMUM NUMBER OF UNIT NUMBERS THAT WILL BE STORED.
C          IF MORE THAN IUMAX UNITS CALL PSYCH WITH THE SAME WARNING
C          CONDITION THEN THE UNIT NUMBERS ARE NO LONGER USED IN
C          DETERMINING IF WARNINGS HAVE ALREADY BEEN PRINTED
C  IWMAX   IS THE NUMBER OF POSSIBLE WARNING CONDITIONS
C          (ENUM MUST BE BETWEEN 1 AND IWMAX)
```

```
C
      PARAMETER (IUMAX=7,IWMAX=13,INIT=(IUMAX+1)*IWMAX)
      DIMENSION INFO(10),WTIME(IWMAX,IUMAX+1),UNIT(IWMAX,IUMAX),
     .          MAXU(IWMAX),IUNITS(IWMAX)
C
      COMMON /SIM/ TIMEO,TFINAL,DELT
      COMMON /LUNITS/LUR,LUW,IFORM,LUK
      DATA WTIME/INIT*1.0E6/, MAXU/IWMAX*.FALSE./, IUNITS/IWMAX*0/
C
C CHECK FOR ALLOWABLE EMODE (0,1,2)
C
      IF (EMODE .LT. 0 .OR. EMODE .GT. 2) THEN
         WRITE(LUW,100) INFO(1),INFO(2)
100      FORMAT(/20X,'*** ERROR IN UNIT',1X,I2,1X,'TYPE',1X,I2,
     .          1X,'***',/6X,'THE PSYCHROMETRICS SUBROUTINE WAS
CALLED',
     .          ' WITH AN UNALLOWABLE "EMODE."')
         STOP
      END IF
C
C  SET STATUS
C
      STATUS = ENUM
C
C  TRANSFER ACCORDING TO EMODE
      LINE = EMODE + 1
      GOTO (50,60,60) LINE
C
C                        *** EMODE EQUALS 0 ***
C
50    RETURN
C
C                      *** EMODE EQUALS 1 OR 2 ***
C
C *** CHECK TO SEE IF UNIT NUMBER HAS BEEN WARNED BEFORE
C
60    IF (IUNITS(ENUM) .EQ. 0) THEN
C
C  ** FIRST UNIT TO HAVE A WARNING

         IUNITS(ENUM)=1
         UNIT(ENUM,1)=INFO(1)
         ISTORE = 1
         GO TO 75
      END IF
C
C  ** CHECK IF STORAGE MAXIMUM PREVIOUSLY REACHED
C     IF SO, GO TO TIMER SECTION
C
      IF (MAXU(ENUM)) THEN
         ISTORE = IUMAX + 1
         GO TO 75
      END IF
C
C  ** CHECK STORED UNIT NUMBERS. IF FOUND, GO TO TIMER SECTION
C
```

```
      DO 70 I=1,IUNITS(ENUM)
         IF (INFO(1) .EQ. UNIT(ENUM,I)) THEN
            ISTORE = I
            GO TO 75
         END IF
70    CONTINUE
C
C  ** STORE NEW UNIT NUMBER IF STORAGE AVAILABLE
C
C   * RAN OUT OF UNIT STORAGE
C
      IF (IUNITS(ENUM) .EQ. IUMAX) THEN
         MAXU(ENUM) = .TRUE.
         ISTORE = IUNITS(ENUM) + 1
      ELSE
C
C   * STORE NEW UNIT NUMBER
C
         IUNITS(ENUM) = IUNITS(ENUM) + 1
         ISTORE = IUNITS(ENUM)
         UNIT(ENUM,ISTORE) = INFO(1)
      END IF
C
C  *** CHECK WARNING TIMES
C
C  ** CHECK TO SEE IF SPECIFIED WARNING HAS EVER BEEN PRINTED FOR THIS
C     UNIT AND SET WTIME TO CURRENT TIME IF WARNING WILL BE PRINTED
C
75    IF (TIME .LT. (WTIME(ENUM,ISTORE)-DELT/2.)) THEN
C
C   * NEVER PRINTED
C
         WTIME(ENUM,ISTORE) = TIME
C
C  ** CHECK TO SEE IF WARNING HAS BEEN PRINTED FOR THIS UNIT DURING
C     THE PRESENT TIMESTEP
C
      ELSE IF (ABS(TIME-WTIME(ENUM,ISTORE)) .LT. DELT/2.) THEN
C
C   * ALREADY PRINTED THIS TIMESTEP SO RETURN WITHOUT PRINTING
C
         RETURN
C
C   * WARNING PRINTED DURING PREVIOUS TIMESTEP. IF EMODE=1 DONOT PRINT
C
      ELSE
         IF (EMODE .EQ. 1) RETURN
         WTIME(ENUM,ISTORE) = TIME
      END IF
C
C *** PRINT SECTION ***
C
C  ** EMODE EQUALS 1
C
      IF (EMODE .EQ. 1) THEN
         IF (MAXU(ENUM)) THEN
```

```
            WRITE(LUW,200) IUMAX
200         FORMAT(/20X,'*** WARNING ***',/6X,'MORE THAN ',I2,'
UNITS',
     .              ' HAVE CALLED THE PSYCHROMETRICS',/6X,'ROUTINE
ONE',
     .              ' OR MORE TIMES DURING THE SIMULATION WITH')
         ELSE
            WRITE(LUW,300) INFO(1),INFO(2),INFO(1)
300         FORMAT(/10X,'*** WARNING IN UNIT ',I2,' TYPE ',I2,' ***',
     .          /6X,'UNIT NUMBER ',I2,' HAS CALLED THE
PSYCHROMETRICS',
     .          /6X,'ROUTINE ONE OR MORE TIMES DURING THE SIMULATION',
     .          ' WITH')
         END IF
      ELSE
C
C  ** EMODE EQUALS 2
C
         IF (MAXU(ENUM)) THEN
            WRITE(LUW,400) TIME
400         FORMAT(/10X,'*** WARNING AT TIME ',F10.4,' ***',/6X,
     .              'ONE OR MORE UNITS HAVE CALLED THE PSYCH ROUTINE',
     .              ' WITH')
         ELSE
            WRITE(LUW,500) INFO(1),INFO(2),TIME,INFO(1)
500         FORMAT(/4X,'*** WARNING IN UNIT ',I2,' TYPE ',I2,
     .           ' AT TIME ', F10.4,' ***',/6X,'UNIT NUMBER ',
     .           I2,' HAS CALLED THE ',
     .          'PSYCH ROUTINE WITH')
         END IF
      END IF
C
C  WARNING CONDITIONS
C
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13), ENUM
C
1     WRITE(LUW,1001)
1001  FORMAT(6X,'A TOTAL PRESSURE GREATER THAN 5 ATMOSPHERES.  BE-
WARE',
     .      /6X,'THAT THE IDEAL GAS RELATIONS USED IN THE PSYCHRO-',
     .      /6X,'METRICS ARE NOT ACCURATE AT HIGHER PRESSURES.')
      RETURN
2     WRITE(LUW,1002)
1002  FORMAT(6X,'AN ENTHALPY LESS THAN POSSIBLE FOR THE GIVEN HUMID-
ITY',
     .      /6X,'RATIO.  THE AIR WAS ASSUMED TO BE SATURATED AT THE',
     .      /6X,'GIVEN HUMIDITY RATIO AND THE ENTHALPY SET FOR THIS',
     .      /6X,'CONDITION.')
      RETURN
3     WRITE(LUW,1003)
1003  FORMAT(6X,'A HUMIDITY RATIO LESS THAN 0.0.  DRY AIR WAS AS-
SUMED',
     .      /6X,'AND THE HUMIDITY RATIO SET TO 0.0.')
      RETURN
4     WRITE(LUW,1004)
1004  FORMAT(6X,'A WET BULB TEMPERATURE GREATER THAN THE DRY BULB',
```

```
     .      /6X,'TEMPERATURE.  SATURATED AIR WAS ASSUMED AND THE WET',
     .      /6X,'BULB TEMPERATURE SET TO THE DRY BULB TEMPERATURE.')
      RETURN
5     WRITE(LUW,1005)
1005  FORMAT(6X,'A DEW POINT TEMPERATURE GREATER THEN THE DRY BULB',
     .      /6X,'TEMPERATURE.  THE DEW POINT WAS SET EQUAL TO THE
DRY',
     .      /6X,'BULB TEMPERATURE.')
      RETURN
6     WRITE(LUW,1006)
1006  FORMAT(6X,'A WET BULB TEMPERATURE BELOW THAT FOR DRY AIR AT
THE',
     .      /6X,'GIVEN DRY BULB TEMPERATURE.  DRY AIR WAS ASSUMED
AND',
     .      /6X,'A NEW WET BULB TEMPERATURE FOUND.')
      RETURN
7     WRITE(LUW,1007)
1007  FORMAT(6X,'A RELATIVE HUMIDITY LESS THAN 0.  THE RELATIVE',
     .      /6X,'HUMIDITY WAS SET TO 0.0.')
      RETURN
8     WRITE(LUW,1008)
1008  FORMAT(6X,'A RELATIVE HUMIDITY GREATER THEN 1.0.  THE RELATIVE',
     .      /6X,'HUMIDITY WAS SET TO 1.0.')
      RETURN
9     WRITE(LUW,1009)
1009  FORMAT(6X,'A HUMIDITY RATIO ABOVE THE SATURATION HUMIDITY RA-
TIO',
     .      /6X,'FOR THE GIVEN DRY BULB TEMPERATURE.  THE AIR WAS',
     .      /6X,'ASSUMED SATURATED AT THE GIVEN DRY BULB TEMPERATURE',
     .      /6X,'AND THE HUMIDITY RATIO SET FOR THIS CONDITION.')
      RETURN
10    WRITE(LUW,1010)
1010  FORMAT(6X,'AN ENTHALPY GREATER THEN THAT FOR SATURATED AIR AT',
     .      /6X,'THE GIVEN DRY BULB TEMPERATURE.  THE AIR WAS AS-
SUMED',
     .      /6X,'SATURATED AND THE ENTHALPY SET FOR THIS CONDITION.')
      RETURN
11    WRITE(LUW,1011)
1011  FORMAT(6X,'AN ENTHALPY LESS THEN THAT FOR DRY AIR AT THE',
     .      /6X,'GIVEN DRY BULB TEMPERATURE.  DRY AIR WAS ASSUMED',
     .      /6X,'AND THE ENTHALPY SET FOR THIS CONDITION.')
      RETURN
12    WRITE(LUW,1012)
1012  FORMAT(6X,'CONDITIONS OUTSIDE THE INTENDED RANGE OF THE WATER',
     .      /6X,'VAPOR SATURATION PRESSURE CORRELATION.  THE',
     .      /6X,'CORRELATION, FROM THE 1985 ASHRAE FUNDAMENTALS',
     .      /6X,'HANDBOOK, IS FOR TEMPERATURE RANGE OF -100 C',
     .      /6X,'TO 200 C.')
      RETURN
13    WRITE(LUW,1013)
1013  FORMAT(6X,'CONDITIONS OUTSIDE THE INTENDED RANGE OF THE DEW',
     .      /6X,'POINT TEMPERATURE CORRELATION.  THEN CORRELATION',
     .      /6X,'FROM THE 1981 ASHRAE FUNDAMENTALS HANDBOOK IS FOR
THE',
```

```
.       /6X,'TEMPERATURE RANGE OF -60 C TO 70 C.')
 RETURN
 END
```

Typical data files generated by the simulation program are shown below. The file "output.user" is a file in a format which is easy for the user to read using the MS Fortran editor. The other data file "data.graph." is in a format which can be easily opened using a graphing program. The data in the graphing file is strored as comma seperated variables.

Sample data from output.user file:

```
                PRODUCT SUMMARY
-------------------------------------------
Product Name = Alkar Light Pork
Outer Radius =    .500 [inch]
Length of Sausage =  6.000 [inch]
Number of Nodes =    5
Thermal Conductivity =     .350 [Btu/ft-hr-F]
Mass Density =  62.00 [lbm/ft^3]
Specific Heat =    .855 [Btu/lbm-R]
Initial Temperature =   95.00 [F]


     HOUSE DESIGN SPECIFICATIONS
-----------------------------------
Cabinet width =  7.50 [ft]
Pass height = 16.50 [ft]
Pass spacing =  1.12 [ft]
Chain speed =  5.50 [ft/min]
Stick spacing =  1.67 [ft]
Number of hot dogs per stick = 238
```

PROCESS SUMMARY

No.	Zone	Name	Fluid	Pass	Vset [ft/s]	Tdb [F]	Twb [F]	MassTran
1	Smoke		air	4	5.021	170.0	110.0	yes

Smoke
Convective Heat Transfer Coef. = 3.180 Btu/hr-ft^2-F
Critical time step = 3.8618 seconds.
Actual time step used = 3.0000 seconds.

Time Minutes	Tdb F	Twb F	Weight Loss %	Radius (inch) .500	.375	.250	.125	.000
.000	148.5	110.0	.00	95.0	95.0	95.0	95.0	95.0
1.000	155.7	110.0	-.10	102.1	98.1	96.1	95.3	95.0
2.000	162.8	110.0	-.08	104.2	100.6	98.1	96.7	96.0
3.000	170.0	110.0	.01	105.5	102.5	100.1	98.6	97.8
4.000	162.8	110.0	.12	106.5	103.9	101.8	100.4	99.7
5.000	155.7	110.0	.23	107.1	105.0	103.3	102.1	101.4
6.000	148.5	110.0	.32	107.7	105.9	104.5	103.5	102.9
7.000	155.7	110.0	.43	108.1	106.7	105.5	104.6	104.2
8.000	162.8	110.0	.58	108.5	107.3	106.3	105.6	105.2
9.000	170.0	110.0	.77	108.9	107.8	107.0	106.4	106.1
10.000	162.8	110.0	.97	109.1	108.3	107.6	107.1	106.8
11.000	155.7	110.0	1.14	109.3	108.6	108.0	107.6	107.4
12.000	148.5	110.0	1.29	109.4	108.9	108.4	108.1	107.9

Sample data from data.graph file:

```
Time (Min.),Tdb,Twb,%Loss,  .500,  .375,  .250,  .125,  .000,
   .000, 148.5, 110.0,    .00,  95.0,  95.0,  95.0,  95.0,  95.0,
  1.000, 155.7, 110.0,   -.10, 102.1,  98.1,  96.1,  95.3,  95.0,
  2.000, 162.8, 110.0,   -.08, 104.2, 100.6,  98.1,  96.7,  96.0,
  3.000, 170.0, 110.0,    .01, 105.5, 102.5, 100.1,  98.6,  97.8,
  4.000, 162.8, 110.0,    .12, 106.5, 103.9, 101.8, 100.4,  99.7,
  5.000, 155.7, 110.0,    .23, 107.1, 105.0, 103.3, 102.1, 101.4,
  6.000, 148.5, 110.0,    .32, 107.7, 105.9, 104.5, 103.5, 102.9,
  7.000, 155.7, 110.0,    .43, 108.1, 106.7, 105.5, 104.6, 104.2,
  8.000, 162.8, 110.0,    .58, 108.5, 107.3, 106.3, 105.6, 105.2,
  9.000, 170.0, 110.0,    .77, 108.9, 107.8, 107.0, 106.4, 106.1,
 10.000, 162.8, 110.0,    .97, 109.1, 108.3, 107.6, 107.1, 106.8,
 11.000, 155.7, 110.0,   1.14, 109.3, 108.6, 108.0, 107.6, 107.4,
 12.000, 148.5, 110.0,   1.29, 109.4, 108.9, 108.4, 108.1, 107.9,
```

A.2 Analytical Solution

The analytical solution was presented in Sec. II.1-4. The analytical solution was solved using an EES program and a fortran program. The fortran code, written by Doug Reindl, is used to find the roots of the eigenfunction which will satisfy the eigencondition. The importance of the root finding routine is that it has the ability to approximate the bessel functions which are found in the eigenfunction.

A.2-1 Bessel Function Root Finder (Written by Doug Reindl)

The necessary input to use the program is the outer radius of the sausage and the value of the Biot number for the problem which is to be solved.

```
      implicit none
c ==================================================================
c  This program will solve the related equations associated with    =
c  the pure conduction solution of energy from an isothermal inner  =
c  cylinder and an insulated outer cylinder of radius r2 (inner     =
c  has a radius of r1).  Most subroutines/functions are general,    =
c  however, function:f is specific and computes the eigenvalues     =
c  associated with the specific eigencondition for this problem.    =
c  Also, function:C is the cylinder function specific for this prob.=
c                                                                   =
c   5/29/91                                                         =
c ==================================================================
      integer i,j,nroot,k,nsteps
      double precision r0,lambda(500),gamma
      double precision jbesse,pi,Bi
      common/dimensions/r0,Bi
c      pi = 4.0d0*atan(1.0d0)
c
c  open appropriate files
c
      open (unit=12, file='roots.dat', status='unknown')
      rewind 12
c
      write(*,*) 'Input the cylinder radius, r0 (inch)'
```

```
      read(*,*) r0
      write(*,*) 'Input the Biot number'
      read(*,*) Bi
c
      write(*,*) 'Finding roots'
      call rfind(nroot,lambda)
      write(*,*) nroot, ' roots were found'
c
 85   continue
c
      end  ! ============  end of main program  ==============

c
c ************************************************************
c  Subroutine:  Rfind                                        *
c  -----------  -----                                        *
c    This subroutine will find roots of the eignencondition, *
c    f as given by,                                          *
c                                                            *
c    This function computes the value of an arbitrary        *
c      J0(lam*r1) Y1(lam*r2) - J1(lam*r2) Y0(lam*r1)         *
c                                                            *
c ============================================================
c
      subroutine rfind(nroot, lambda)
      integer info(5), i, nval, nroot, errchk
      double precision xin(5), xout(5), f, r0, errtol, unit
      double precision lambda(500), delta, Bi
      parameter(errtol=1.E-8,nval=5000,delta=0.2,unit=1.)
      common/dimensions/r0,Bi
      external f
c ------------------------------------------------------------
c  Nomenclature:
c  -------------
c
c     delta    =  step size in the root search
c     errtol   =  error tolerance for bisection root finding
c     errchk   =  errchk=0 for valid soln, errchk=1 for invalid soln
c     f        =  objective function
c     i        =  loop variable
c     info     =  info array from bisection routine
c     lambda   =  value of the root to f
c     nroot    =  number of roots found
c     nval     =  search for function sign changes a max of nval times
c     r0       =  cyl diameter
c     unit     =  unity
c     xin      =  input values for bisection routine
c     xout     =  output values from bisection routine
c
c
c  ---------------------------------------------------------------
c
c set error tolerance for finding roots of eigencondition
c
      xin(3) = errtol
c
```

```
c  initialize eigencondition root count
c
      nroot = 1
      do 50 i=1, nval
c
c  compute upper and lower guess values of function to see
c  when the sign changes indicating a root.
c
        xin(1) = delta * real(i)
        xin(2) = delta * real(i+1)
        if (Dsign(unit,f(xin(1))).ne.Dsign(unit,f(xin(2)))) then
c
c  upon locating a sign change, try to find the root by bisection
method
c
           call bisect (xin, xout, f, info)
           errchk = info(3)
c          write(*,*) '* iterations required = ', info(2)
c
c  Check to see if output from bisection subroutine is valid
c  (errchk = 1: denotes invalid roots)
c  (errchk = 0: denotes valid roots)
c
           if(errchk.eq.0) then
c
c  Output roots
c
             lambda(nroot) = xout(1)
             write(12,998) lambda(nroot)
             write(*,999) nroot, lambda(nroot)
c  update root count
             nroot = nroot + 1
c
           endif
c
c  Do not exceed the bounds of the array lambda()
c
           if (nroot .eq. 10000) go to 51
        endif

 50   continue
 51   continue
 998  format(f12.6)
 999  format(i3,t8,f12.6)
      return
      end  ! ------------  End of subroutine rfind ---------------

c
c ************************************************************
c  Function:  JBessel                                        *
c  ---------  -------                                        *
c    This function computes the value of an arbitrary        *
c    integer orders of the Bessel function of the first kind *
c    at some value x.                                        *
c                                                            *
c      for positive integer orders k,                        *
```

```
c                                                                      *
c                              inf                                     *
c                              ---                                     *
c                      x^k     \            (-1)^n x^2n                *
c        J(k,x) =     ---     /    --------------------------         *
c                     2^k     --- 2^2n gamma(n+1) gamma(n+k+1)         *
c                             n=0                                      *
c                                                                      *
c       for negative integer orders,                                   *
c                                                                      *
c        J(-k,x) = (-1)^k * J(k,x)                                     *
c                                                                      *
c The routine also will use simplified expressions or                  *
c asympototic estimates for large values of the arguments              *
c                                                                      *
c       J(k,x) = (2/pi*x)^1/2 cos(x-pi/4-n*pi/2)                       *
c                                                                      *
c  This routine uses: function gamma                                   *
c ***********************************************************************
c
      double precision function jbessel(k,x)
      integer k, n, nterms
      double precision x, kernel, incr, sign, sum, gamma, errtol, diff
      double precision pi, f0, q0, f1, q1
      logical asymp
      parameter(nterms=500, errtol=1E-7,diff=1E-6,pi=3.141592)
      common/asymptotic/asymp
c
c  Nomenclature:
c  -------------
c    diff    =  tolerance to switch to asymtotic est.
c    errtol  =  error tolerance on bessel sum
c    k       =  the integer order of the bessel function
c    kernel  =  current value of the summation kernel
c    nterms  =  maximum number of terms to keep in summation
c    x       =  point at which to evaluate the bessel fcn.
c    n       =  summation index
c    sign    =  either plus 1 or minus 1 to control sign
c    sum     =  current value of the bessel sum
c ------------------------------------------------------------
c
c ------------------------------------
c       go to 500  ! use series approx  -
c ------------------------------------
c
c       kernel = 0.
c       sum  = 0.
c       sign = 1.
c
c
c         do 100 n=0, nterms
c           kernel = ( sign * (x/2.)**(2*n) ) / ( gamma(n+1)
c     &            * gamma(n+abs(k)+1) )
c         sum = sum + kernel
c 22      format(I3,4(1x,E15.8))
c         if (abs(kernel) .le. errtol) go to 101   ! end if kernel is
```

```
small
c         sign = -sign
c 100     continue
c 101     continue
c
c       jbessel = x**abs(k) * sum / (2.**abs(k))
c       if (k .lt. 0) jbessel = (-1)**k * jbessel
c
c
 500  continue

C ------------  Series Approx  --------------------------
c
c Alternatively, we could use the series approximations as
c given in Abramowitz and Stegun.
c
c  do J(0,x) bessel first
c
      if (k.eq.0) then
        if ( (x.gt.(-3.)) .and. (x.le.3.) ) then
         jbessel = 1. - 2.2499997 * (x/3.)**2 + 1.2656208 * (x/3.)**4
     &      - 0.3163866 * (x/3.)**6 + 0.0444479 * (x/3.)**8
     &      - 0.0039444 * (x/3.)**10 + 0.0002100 * (x/3.)**12
        else
         f0 = 0.79788456 - 0.00000077 * (3./x) - 0.00552740*(3./x)**2
     &     - 0.00009512 * (3./x)**3 + 0.00137237 * (3./x)**4
     &     - 0.00072805 * (3./x)**5 + 0.00014476 * (3./x)**6
c
         q0 = x - 0.78539816 - 0.04166397 * (3./x) - 0.00003954 *
     &    (3./x)**2 + 0.00262573 * (3./x)**3 - 0.00054125 * (3./x)**4
     &   - 0.00029333 * (3./x)**5 + 0.00013558 * (3./x)**6
c
         jbessel = 1./Dsqrt(x) * f0 * Dcos(q0)
        endif  ! end if (x.le.3)...
c
c  now do J(1,x)
c
      else
       if (k.eq.1) then
        if ( (x.gt.(-3.)) .and. (x.le.3.) ) then
          jbessel = x * (0.5 - 0.56249985 * (x/3.)**2 + 0.21093573
     &        * (x/3.)**4 - 0.03954289 * (x/3.)**6 + 0.00443319
     &        * (x/3.)**8 - 0.00031761 * (x/3.)**10 + 0.00001109
     &        * (x/3.)**12 )
        else
          f1 = 0.79788456+0.00000156 * (3./x) + 0.01659667*(3./x)**2
     &     + 0.00017105 * (3./x)**3 - 0.00249511 * (3./x)**4
     &     + 0.00113653 * (3./x)**5 - 0.00020033 * (3./x)**6
c
         q1 = x - 2.35619449 + 0.12499612 * (3./x) + 0.0000565 *
     &    (3./x)**2 - 0.00637879 * (3./x)**3 + 0.00074348 * (3./x)**4
     &   + 0.00079824 * (3./x)**5 - 0.00029166 * (3./x)**6
c
         jbessel = 1./Dsqrt(x) * f1 * Dcos(q1)
c
        endif  ! end if (x.le.3)...
```

```
       endif  ! end  if (k.eq.1)
      endif   ! end  if (k.eq.0)
c
      return
      end  !----------- End of function JBessel --------------
c
c ************************************************************
c  Function:  Gamma                                          *
c  ---------  -----                                          *
c    This function will compute the values of the gamma fcn  *
c    with integer arguments.  Where the gamma function is    *
c    related to the factorial by                             *
c                                                            *
c          gamma(n+1) = n!                                   *
c                                                            *
c ************************************************************
c
      double precision function gamma(n)
c
      integer n, k
c
      gamma = 0.
      if (n .eq. 1) then
        gamma = 1.
      else
        do 100 k= (n-1), 1, -1
          if (k .eq. (n-1)) then
            gamma = dfloat(k)
          else
            gamma = gamma * dfloat(k)
          endif
c
 100    continue
      endif
c
      return
      end        ! ------------ end of gamma function --------------
---
c
c ************************************************************
c  Function:  Factorial                                      *
c    This function computes the factorial of the arg, n.     *
c ************************************************************
c
      double precision function fact(n)
C
      integer n, k
c
      fact = 0.
      if (n .eq. 0) then
        fact = 1.
      else
        do 100 k= n, 1, -1
          if (k .eq. n) then
            fact = real(k)
```

```
          else
            fact = fact * real(k)
          endif
c
 100    continue
      endif
c
      end        ! ------------  end of function fact ---------------
-
c
c ==================================================================
c
      subroutine bisect (xin, xout, f, info)
      integer i, n, num, info(5), maxiter
      double precision  xin(5), xout(5), tol, a, b
      double precision signfa, signfb, signfp, f, unity, fa, fb
      double precision p, aerr, rerr, diff, lt, ld
      external f
c
c douglas reindl,  cs412,  spring '89
c ------------------------------------------------------------------
c  this subroutine is called by main.  it uses the bisection method to
c   find roots of a function "f".
c
c   xin(1) = first guess, a, of root.
c   xin(2) = second guess, b, of the root.
c   xin(3) = tolerance for convergence.
c   xout(1) = solution (p) or root of function.
c   xout(2) = value of the function evaluated at p.
c   xout(3) = absolute error in the estimate of the root.
c   xout(4) = relative error in the estimate of the root.
c   info(1) = flag for initial guesses, = 1 if both guesses are
c             positive.
c   info(2) = n, number of iteration required for solution.
c   info(3) = 0 for valid roots, 1 for invalid roots
c   f       = objective function
c   maxiter = maximum # of iterations allowed to find soln.
c   signfa  = sign of the function, f, evaluated at a.
c   signfb  = sign of the function, f, evaluated at b.
c   signfp  = sign of the function, f, evaluated at p.
c ==================================================================
c
c      write(*,*)
c      write(*,*) 'solving with the bisection method'
c      write(*,*)
c
      a   = xin(1)
      b   = xin(2)
      tol = xin(3)
      unity = 1.
      maxiter = 100
c
c  initialize info(3) for valid roots
c
      info(3) = 0
c
```

```
c  checking the sign of the objective function at the two given
c  guesses
c   (signs must be opposite at "a" and "b".)
c
      signfa = dsign( unity, f(a) )
      signfb = dsign( unity, f(b) )
      if (signfa.eq.signfb) then
         info(1) = 1
         write(*,*) 'bad guess, signfa = signfb'
         return
      endif
c
c  estimating the number of iterations required for solution
c
      diff = b - a
      lt = log10(tol)
      if (diff.gt.0.0) then
        ld = log10(diff)
        num = ( ld - lt ) / log10(2.) - 0.5
      endif
c
c  for first step
c
      n = 1
      p = 0.5 * (a + b)
c
c begin bisection
c
      do 500 i=1, maxiter
c
        signfp = dsign( unity, f(p) )
        if ( signfp.eq.signfa ) then
             a = p
             signfa = signfp
        else
             b = p
             signfb = signfp
        endif
c
c  computing errors
c
        aerr = abs(b - a)/2
        rerr = abs(b - a) / abs(p)
        if ((aerr.lt.tol).and.(abs(f(p)).lt.tol)) go to 501
c
c  preparing for next step
c
        n = n + 1
        p = 0.5 * (a + b)
c
 500  continue
c
c  end of bisection routine
c
c ************* End of Iteration Loop  ***************
c
```

```
c
c   Check to see if function is really converged or
c   just ran out of iterations.
c
      if (n.ge.maxiter) then
        if ((aerr.gt.tol).or.(abs(f(p)).gt.tol)) info(3) = 1
c
c No solution found
c
        return
      endif

 501  continue
c
c  output of solution
c
      xout(1) = p
      xout(2) = f(p)
      xout(3) = aerr
      xout(4) = rerr
      info(2) = n
c
c
 1000 continue

c
      return
      end


c
c ************************************************************
c  Function:  f                                              *
c    This is the objective function for finding the zeros    *
c ************************************************************
c
      double precision function f(lambda)
c
      double precision r0, JBessel, YBessel, lambda, Bi
      common/dimensions/r0,Bi
c
      f = Bi * JBessel(0,lambda*r0/12.)/JBessel(1,lambda*r0/12.)
     &       - lambda * r0 / 12.
c
      return
      end      ! ---------  end of function f  ---------------------
```

A.2-2 EES Code Used to Solve Analytical Solution

```
{THIS PROGRAM SOLVES THE EXACT SOLUTION FOR CONDUCTION IN A
SAUSAGE

L[1] - L[20]    First 20 eigen values found using eigenvalue program  [1/ft]
     R          Radius at which problem is to be evaluated  [inch]
     Ro         Outer radius of sausage  [inch]
     h          Convection heat transfer coefficient  [Btu/hr-ft^2-F]
     k          Thermal capacitance  [Btu/hr-ft-F]
     c          Specific heat  [Btu/lb-F]
   rho          Mass density  [lb/ft^2]
 alpha          Thermal diffusivity  [ft^2/hr]
     T          Temperature at specified time and radius
  Tsur          Temperature of surrounding fluid stream  [F]
    To          Initial temperature of sausage  [F]
    Bi          Biot number
 theta          Time at which problem it to be evaluated  [min]}

{PROBLEM PARAMETERS}
R = 2.0 {inch}
Ro = 2.0 {inch}
Tsur = 180.0 {F}
To = 90.0 {F}
h = 4.0 {Btu/hr-ft^2-F}
k = 0.35 {Btu/hr-ft-F}
rho = 62.0 {lbm/ft^3}
c = 0.855 {Btu/lb-F}

{EIGEN VALUES  1/ft}
DUPLICATE j=1,20
L[j] =lookup(j,2)
END

{PROBLEM CALCULATIONS}
alpha = k/(rho*c) {ft^2/hr}
Bi = h*(Ro/12)/k
T = Tsur + 2*Bi*(To-Tsur)*sum((bessel_j0(L[i]*R/12)*exp(-alpha*(theta/
60)*L[i]^2))/(((L[i]*Ro/12)^2 + Bi^2)*bessel_j0(L[i]*Ro/12)),i=1,20) {F}
```

A.3 Heat Transfer Coefficient Experiment Programs

The heat transfer coefficient experiment discussed in Chapter III used the following programs to help design the aluminum hot dog and to analize the results of the experiments.

A.3-1 EES Program Used to Design the Aluminum Hot Dog

```
{Program Description:
This program is designed to aid with the
design of a metal hot dog used in the
convective heat transfer coefficient experiments.
Program inputs are the hot dog
diameter and length. In a table the user puts
in a range of estimated heat transfer
coefficients and the length of time for the
experiment. The program calculates a fraction
which indicates how close the metal hot dog
temperature is to the free stream temperature.
For example: if the fraction equals 1.0 then the
metal hot dog temperature has not changes and is
still at the initial temperature, on the other hand
if the fraction equals 0.0 then the metal hot dog
has reached the free stream temperature.}

{Hot Dog Parameters}

D = 1.0/12 {Ft Outer Diameter of Hot Dog}
L = 6.0/12 {Ft Length of Hot Dog}

{Material Properties Metric Units}

{Copper Rho = 8933.0 Cp = 385.0  k = 401.0}
{AISI 302 Stainless Steel Rho = 7900.0 Cp = 477.0 k = 14.9}
```

{Plain Carbon Steel Rho = 7854.0 Cp = 434.0 k = 60.5}
{Aluminum Alloy 2024-T6 Rho = 2770.0 Cp = 875.0 k = 177.0}
{Commercial Bronze Rho = 8800.0 Cp = 420.0 k = 52.0}
{Nickel Rho = 8900.0 Cp = 444.0 k = 90.7}

{Convert from Metric Units to English Units}

Rho = 2770.0*0.062428 {lbm/Ft^3}
Cp = 875.0*2.3886E-4 {Btu/lbm-F}
k = 177.0*0.57782 {Btu/hr-Ft-F}
{Main Program}

As = pi*D*L
Vol = ((pi*D*D)/4)*(L-D) + (pi/6)*D*D*D
Bi = (hbar*D)/k
percent = exp(((-hbar*As)/(Rho*Cp*Vol))*(Time/60))

A.3-2 Program Used to Sort Data Trace Data

```
c ------------------------------------------------------------------
c ******************************************************************
c **************    DATA TRACE DATA FILE SORT PROGRAM    ************
c ******************************************************************
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    *******************
c ------------------------------------------------------------------
c
c
c     This program is designed to read an input file (name to be
c     specified by the user) which contains the time (seconds),
c     dry bulb temperature (F), aluminum hot dog temperature (F),
c     and the real hot dog temperature (F).  This type of
c     data file is typically from the data trace used to measure
c     and record the temperature data recorded during an
c     experiment.  A typical input file is shown below:
c
c         time   Tdb     Tal     Thd
c          10   130.0   110.0   95.3
```

```
c          20   135.0   115.0   103.2
c          30   140.5   123.0   111.3
c          40   145.3   135.5   117.8
c         etc.   etc.    etc.    etc.
c
c     The program then asks the user for an interval size.  This
c     is an option to use only part of the data.  For example, if
c     the data is taken every second, but the user wants to use
c     data averaged over a 5 second interval, then the user would
c     put in a value of 5 for the interval.  The program
c     will then average the data points five at a time.  The
c     program then asks the user for a start and end.  These values
c     are used to allow the user to select certain sections of the
c     complete data file. For example, if a data file contains data
c     for the whole cooking process, but the user wants to isolate
c     the second drying zone, then the user can put in the node
c     numbers corresponding to the start and end of that particular
c     zone. The program then creates a file called raw.dat which
c     contains the selected portion of data to be used in the
c     Heat Transfer Coefficient calculation program.
c
c
c ---------------------------------------------------------------
c **********************    VARIABLE TABLE    **********************
c ---------------------------------------------------------------
c
c
c          Tdb = dry bulb temperature
c          Tal = aluminum hot dog temperature
c          Thd = hot dog temperature
c         Tavg = average temperature over interval
c          sum = sum of temperatures
c         name = input file name
c     interval = number of nodes read (Ex. read every
c                3rd = "interval" node)
c        start = number of node to start writing at
c          end = number of node to stop writing at
c            i = node number read from input file
c            j = counter
c            k = counter
c         junk = used to read titles from input data file
c
c
c
c ---------------------------------------------------------------
c *********    OPEN INPUT FILES AND ASK USER FOR VALUES    **********
c ---------------------------------------------------------------
c
      implicit none
      real Tdb,Tal,Thd,Tavg,sum
      integer i,j,k,interval,start,end
      character name*16,junk*8
      print*,'Enter original file name'
      read*,name
      open (unit=12,file=name,status = 'old')
      open (unit=15,file='raw.dat',status='new')
```

```
      rewind(unit=12)
      rewind(unit=15)
      print*,'Enter interval size'
      read*,interval
      print*,'Enter initial node number'
      read*,start
      print*,'Enter total number of nodes desired'
      read*,end
c
c ----------------------------------------------------------------
c ********    STEP THROUGH INPUT FILE FOR SELECTED VALUES    ********
c ----------------------------------------------------------------
c
      read(12,*) junk,junk,junk
      k = 0
      sum = 0.0
      do 10 j = 0,start+end*interval
         read(12,*) i,Tdb,Tal,Thd
       if(j .GE. start) then
          k = k + 1
          sum = sum + Tal
          if (k .EQ. interval) then
             k = 0
             Tavg = sum/real(interval)
               write(15,90) i,Tdb,Tavg,Thd
             sum = 0.0
          endif
       endif
10    continue
      write(15,*)
c
c ----------------------------------------------------------------
c ******************    CLOSE INPUT/OUTPUT FILES    *****************
c ----------------------------------------------------------------
c
      close(unit=15,status='keep')
      close(unit=12,status='keep')
c
c ----------------------------------------------------------------
c *********************    FORMAT STATEMENTS    *********************
c ----------------------------------------------------------------
c
90    format (i5,',',f9.2,',',f9.2,',',f9.2)
      end
c
c ----------------------------------------------------------------
c ****************************************************************
c ************    END OF DATA TRACE FILE SORT PROGRAM    ************
c ****************************************************************
c ----------------------------------------------------------------
```

A.3-3 Heat Transfer Coefficient Calculation Program

```
c ------------------------------------------------------------------
c ******************************************************************
c *************    CONVECTIVE HEAT TRANSFER CALCULATION    ***********
c ******************************************************************
c ------------------------------------------------------------------
c
c
c ------------------------------------------------------------------
c ********************    PROGRAM DESCRIPTION    ********************
c ------------------------------------------------------------------
c
c
c    This program is used to calculate the heat transfer coefficient
c    from a set of data collected using the "Aluminum" hot dog.  It
c    requires a data file called "raw.dat" which contains four
c    columns.  The data appears as shown below.
c
c            Time(sec.)     Tdb(F)     Tal(F)    Thd(F)
c
c    The program also requires, in the code, for the user to input
c    the physical size of the metal hot dog.  Currently it is set up
c    for the aluminum hot dogs that were designed at the University
c    of Wisconsin-Madison.  The program also has the ability to
c    account for radiation to the surroundings which is assumed to be
c    at the same temperature as the surrounding product.
c
c    The program will then ask the user for the total number of
c    data points in the file called 'raw.dat'.  The program will
c    then calculate the heat transfer coefficient using a central
c    difference to estimate the temperature derivative of the
c    aluminum hot dog temperature.
c
c
c    The program creates a output data file called 'HTC.dat" which
c    contains the the following information.
c
c     Time   Tdb   Tal   Thd     HTC-rad          HTC-conv      error
c    (Sec.)  (F)   (F)   (F)  (Btu/hr-ft^2-R) (Btu/hr-ft^2-R)
c
c
c
c ------------------------------------------------------------------
c ***********************    VARIABLE TABLE    **********************
c ------------------------------------------------------------------
c
c       Tdb = array of dry-bulb temperatures (F)
c      TdbC = array of dry-bulb temperatures (C)
c      TdbK = dry bulb temperature (K)
c       Tal = array of aluminum hot dog temperatures (F)
c      TalC = array of aluminum hot dog temperatures (C)
```

```
c       TalK = aluminum hot dog temperature (K)
c        Thd = array of surface temperatures of surrounding
c              hot dogs (F)
c       ThdC = array of surface temperatures of surrounding
c              hot dogs (C)
c       ThdK = surface temperature of surrounding hot dogs (K)
c          D = diameter of aluminum hot dog (M)
c          L = length of aluminum hot dog (M)
c          C = specific heat of aluminum (J/Kg-K)
c          M = mass of aluminum hot dog (Kg)
c         Em = surface emissivity
c         DT = temperature difference (K)
c      Dtime = time difference (Sec.)
c       dTdt = derivative of temperature wrt time
c       DelD = uncertainty in measurement of diameter of aluminum
c              hot dog (m)
c       DelL = uncertainty in measurement of length of aluminum
c              hot dog (m)
c       DelC = uncertainty in specific heat of aluminum (J/Kg-K)
c       DelM = uncertainty in measurement of mass of aluminum
c              aluminum hot dog (Kg)
c      DelEm = uncertainty in estimation of surface emmisivity of
c              aluminum hot dog
c      DelDT = estimate of noise in data trace temperature
c              sensor (C)
c   DelDtime = estimate of uncertainty in time measurement of
c              data trace (Sec.)
c       DelT = uncertainty in any temperature measurement
c       Delh = uncertainty in the measured heat transfer coefficient
c       dhdD = derivative of the heat transfer coef. wrt diameter
c       dhdL = derivative of the heat transfer coef. wrt length
c       dhdC = derivative of the heat transfer coef. wrt
c              specific heat
c       dhdM = derivative of the heat transfer coef. wrt mass
c      dhdEM = derivative of the heat transfer coef. wrt emissivity
c       dhDT = derivative of the heat transfer coef. wrt the
c              temperature difference DT
c    dhDtime = derivative of the heat transfer coef. wrt the
c              time difference Dtime
c     dhDThd = derivative of the heat transfer coef. wrt the
c              temperature of a hot dog
c     dhDTdb = derivative of the heat transfer coef. wrt the
c              dry bulb temperature
c          A = surface area of aluminum hot dog (m^2)
c         SB = stefan-boltzmanN constant (W/m^2-K^4)
c         pi = 3.141592
c       time = array of times at which data was collected (Sec.)
c       qrad = heat transfer rate due to radiation (W)
c       hrad = radiative heat transfer coefficient (W/m^2-K)
c          h = convective heat transfer coefficient (W/m^2-K)
c       dTdt = estimate of the derivative of the aluminum hot
c              dog temperature with respect to time
c      i,j = counters
c      nodes = total number of data points in "raw.dat" file
c
c
```

```
c
c ------------------------------------------------------------------
c *********************    DECLAIR VARIABLES    *********************
c ------------------------------------------------------------------
c
      implicit none
      double precision Tdb,TdbC,TdbK,Tal,TalC,TalK,Thd,ThdC,ThdK
      double precision D,L,C,M,Em,DT,Dtime,dTdt
      double precision DelD,DelL,DelC,DelM,DelEm,DelDT,DelDtime
      double precision DelT,Delh
      double precision dhdD,dhdL,dhdC,dhdM,dhdEm,dhDT,dhDtime
      double precision dhdThd,dhdTdb
      double precision A,SB,pi,time,qrad,hrad,h,dTdt
      integer nodes,i,j
      dimension time(0:100),Tdb(0:100),Tal(0:100),Thd(0:100)
      dimension TdbC(0:100),TalC(0:100),ThdC(0:100)
c
c ------------------------------------------------------------------
c ******************    METAL HOT DOG PARAMETERS    *****************
c ------------------------------------------------------------------
c
      D = 0.023d0 !M
      DelD = 0.001d0 !M
      L = 0.1524d0 !M
      DelL = 0.003d0 !M
      C = 875.0d0 !J/Kg-K
      DelC = 1.0d0 !J/Kg-K
      M = 0.127d0 !Kg
      DelM = 0.001d0 !Kg
      Em = 0.9d0
      DelEm = 0.1d0
      DelT = 0.56d0 !C
      DelDT = 0.22d0 !C
      DelDtime = 0.2d0 !Sec
c
c ------------------------------------------------------------------
c *********************    PHYSICAL CONSTANTS    *********************
c ------------------------------------------------------------------
c
      pi = 4.0d0*datan(1.0d0)
      SB = 5.67d-8 !W/M^2-K^4
c
c ------------------------------------------------------------------
c *****************    METAL HOT DOG SURFACE AREA    *****************
c ------------------------------------------------------------------
c
      A = pi*D*(L - D) + pi*D*D
c
c ------------------------------------------------------------------
c **************    OPEN INPUT AND OUTPUT DATA FILES    *************
c ------------------------------------------------------------------
c
      open (unit=12,file='raw.dat',status='old')
      open (unit=15,file='HTC.dat',status='new')
      rewind (unit=12)
      rewind (unit=15)
```

```
c
c ------------------------------------------------------------------
c *****************    ASK USER FOR INFORMATION    *****************
c ------------------------------------------------------------------
c
      print*,'Enter total number of data points in raw.dat file.'
      read*,nodes
      write(15,900)
c
c ------------------------------------------------------------------
c *************    CONVERT FROM FAHRENHEIT TO CELSIUS    ************
c ------------------------------------------------------------------
c
      do 100 i = 1,int(nodes/3.0)
      do 90 j = 1,3,1
         read(12,*) time(j),Tdb(j),Tal(j),Thd(j)
         TdbC(j) = (Tdb(j) - 32.0d0)/1.8d0
         TalC(j) = (Tal(j) - 32.0d0)/1.8d0
         ThdC(j) = (Thd(j) - 32.0d0)/1.8d0

90    continue
c
c ------------------------------------------------------------------
c ****    ESTIMATE TEMPERATURE DERIVATIVE OF ALUMINUM HOT DOG    ****
c ****       TEMPERATURE USING A SIMPLE CENTERAL DIFFERENCE      ****
c ------------------------------------------------------------------
c
         j = 3/2 + 1
         TdbK = TdbC(j) + 273.2d0
         TalK = TalC(j) + 273.2d0
         DT = TalC(j+1) - TalC(j-1)
         Dtime = time(j+1) - time(j-1)
         dTdt = DT/Dtime
         ThdK = ThdC(j) + 273.2d0
         qrad = A*Em*SB*(ThdK**4 - TalK**4)
         hrad = Em*SB*(ThdK+TalK)*(ThdK**2 + TalK**2)
         h = (M*C*dTdt - qrad)/(A*(TdbK - TalK))
c
c ------------------------------------------------------------------
c **********    CALCULATE UNCERTAINTY IN HTC CALCULATION    *********
c ------------------------------------------------------------------
c
         dhdM = C*dTdt/(pi*D*L*(TdbK-TalK))
         dhdC = M*dTdt/(pi*D*L*(TdbK-TalK))
         dhdEm = (-SB*(ThdK**4 - TalK**4))/(TdbK - TalK)
         dhdD = ((-Em*SB*(ThdK**4 - TalK**4))/(D*(TdbK - TalK)))
     &           - ((C*M*dTdt - pi*D*L*Em*SB*(ThdK**4 - TalK**4))
     &           /(pi*L*D*D*(TdbK - TalK)))
         dhdL = ((-Em*SB*(ThdK**4 - TalK**4))/(L*(TdbK - TalK)))
     &           - ((C*M*dTdt - pi*D*L*Em*SB*(ThdK**4 - TalK**4))
     &           /(pi*D*L*L*(TdbK - TalK)))
         dhdThd = (-4.0d0*Em*SB*ThdK**3)/(TdbK - TalK)
         dhdTdb = (-C*M*dTdt + pi*D*L*Em*SB*(ThdK**4 - TalK**4))
     &            /(pi*D*L*(TdbK - TalK)**2)
         dhDT = C*M/(D*Dtime*L*pi*(TdbK-TalK))
         dhDtime = -C*M*DT/(D*Dtime*Dtime*L*pi*(TdbK - TalK))
```

```
          Delh = Sqrt((dhdM*DelM)**2 + (dhdC*DelC)**2 +
     &           (dhdEm*DelEm)**2 +(dhdD*DelD)**2 + (dhdL*DelL)**2 +
     &           (dhdThd*DelT)**2 +(dhdTdb*DelT)**2 + (dhDT*DelDT)**2
     &           + (dhDtime*DelDtime)**2)
c
c ------------------------------------------------------------------
c *****************    WRITE DATA TO OUTPUT FILE    *****************
c ------------------------------------------------------------------
c
          write(15,910) time(j)/60.0,Tdb(j),Tal(j),Thd(j),
     &                  hrad/5.6783d0,h/5.6783d0,(Delh/5.6783d0)
100    continue
c
c ------------------------------------------------------------------
c *********************    FORMAT STATEMENTS    *********************
c ------------------------------------------------------------------
c
900    format('Time,Tdb,Tal,Thd,HTC-rad,HTC-conv, error')
910    format(f8.2,',',t10,f8.4,',',t20,f8.4,',',t30,f8.4,',',
     &       t40,f8.4,',',t50,f8.4,',',t60,f8.4)
920    format(t3,'A,B')
       end
c
c ------------------------------------------------------------------
c ***************    END OF HTC CALCULATION PROGRAM    **************
c ------------------------------------------------------------------
```

References

[1] Sweat, Vincent E., "Thermal Properties of Foods", Saravacos George D., "Mass Transfer Properties of Foods", Engineering Properties of Foods, Edited by M. A. Rao and S. S. H. Rizui, New York, 1986.

[2] E. Holtz, C. Skjoldebrand, A. Bognar, and J. Piekarski, "Modeling the Baking Process of Meat Products Using Convection Ovens", Thermal Processing and Quality of Foods, Elsevier: Applied Science Publishers, 1984.

[3] Incropera, Frank P. and De Witt David P., Introduction to Heat Transfer, John Wiley and Sons, New York, 1985

[4] Ledward D. A., "Meat", Effects of Heating on Foodstuffs, Edited by R. J. Priestley, London: Applied Science Publishers, 1979.

[5] Kays, W. M. and Crawford, M. E., Convective Heat and Mass Transfer, McGraw-Hill Publishing Company, New York, 1987.

[6] Braun, James E., "Methodologies for the Design and Control of Central Cooling Plants", Ph.D. Thesis in Mechanical Engineering, University of Wisconsin-Madison, 1989.

Bibliography

ASHRAE, Handbook of Fundamentals, American Society of Heating, Refrigeration, and Air Conditioning Engineers, Inc., Atlanta, GA, 1989

Bird, Byron R., Stewart, Warren E., and Lightfoot, Edwin N., Transport Phenomena, John Wiley and Sons, New York, 1960.

Cheney, Ward and Kincaid, David, Numerical Mathematics and Computing, Brooks / Cole Publishing Company, Pacific Grove, California, 1980.

Myers, Glen E., Analytical Methods in Conduction Heat Transfer, Genium Publishing Corporation, Schenectady, New York, 1987.