A NUMERICAL STUDY OF CONVECTIVE HEAT TRANSFER IN STORED RESPIRING AGRICULTURAL PRODUCE

A THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF ENGINEERING IN MECHANICAL ENGINEERING

BY

PATRICK XIAO-PING LI

DEPARTMENT OF MECHANICAL ENGINEERING
VICTORIA UNIVERSITY OF TECHNOLOGY (FOOTSCRAY)
VICTORIA, AUSTRALIA

MAY, 1994

To my parents and my wife.

ACKNOWLEDGEMENTS

The author wishes to express his deepest appreciation and sincere thanks to his supervisors, Professor Geoffrey Lleonart, of the Department of Mechanical Engineering of Victoria University of Technology, supervisor of this study, and Associate Professor Graham Thorpe, of the Department of Civil and Building Engineering, Victoria University of Technology, co-supervisor of this project, for their guidance, inspiring assistance and encouragement, as well as for the freedom under which this research was conducted.

As an overseas student from China, the difficulties that the author met during his study were so many and so hard. He had to challenge not only the language problem but also the conflict of two cultures: a traditional young Chinese suddenly immersed in the western culture, a whole new world. Professor Lleonart showed his great open mind, patience, great efficient help and accurate guidance to lead the author out of the lost, to enter the track of study of the project. His endless encouragement and high level academic experience cover the whole processing of this project.

Although the author had a strong background in heat transfer, the knowledge in his mind was in Chinese and this had raised some difficulties in communication in academic discussion and in broadening and deepening the knowledge at the beginning of the study. Associate Professor Thorpe generously offered the author free access to his literature collections and his computer program. The constructive suggestions from and fruitful discussions with Associate Professor Thorpe have made great contributions to this study.

His years of experience in academic research made his supervision greatly efficient. Associate Professor Thorpe's kindness and optimism showed the author another example of a scholar.

The author acknowledges with pleasure the help given by Dr Osvaldo Campanella, of the Department of Food Technology, Massey University, New Zealand, in computing when the author was in New Zealand. Thanks are also due to Dr. Michael Sek for his assistance when acting co-supervisor when the principle supervisor was away overseas; to Mrs Denise Colledge, secretary of the Department, for her kindly help and maintaining the connection between the author and the department, when the author was away from Victoria University of Technology.

The author is grateful to L. P. Goh, E. Leonardi and G. de Vahl Davis of the University of New South Wales, for their program FRECON3D. A Program for the numerical solution of mixed convection in a three-dimensional rectangular cavity. The author is also grateful to the Australian International Development Assistance Bureau(AIDAB) for the scholarship under Australian Development Co-operation Scholarship Scheme (ADCSS).

Finally, the author would like to present his sincere thanks to his wife Susan Qin Su, for her constant encouragement, especially when the author was in difficulties, and her excellent typing of the manuscript. Without her understanding and support, the study might only be in his dream.

ABSTRACT

A mathematical model of natural convection in three-dimensional enclosures containing respiring fruits or vegetables is presented. The agricultural produce is treated as a porous media exhibiting transversely orthotropic permeability in which local temperature-dependent respiration heat generation occurs.

In the numerical investigations conducted, both adiabatic and isothermal boundary conditions are considered for the cooling and storage process. The influence of container size, respiration rate and permeability on natural convection is discussed. Numerical experimental results for temperature distribution, velocity distribution, true transient cooling and storage process are presented.

It was found that the style of packaging of the agricultural produce and the size of container were likely to have significant effects on the storage characteristics.

TABLE OF CONTENTS

NOMENCLATURE

Notation

A	= surface area of representative elementary volume (r.e.v.), m^2
C	= specific heat of solid, J/kg·K
C_p	= specific heat of fluid at constant pressure, J/kg·K
D_a	= Darcy Number
D_{eff}	= effective diameter of fruit and vegetables
g	= gravitational acceleration, m/s^2
$\vec{G}$	= gravity force, N
G_x, G_y, G_z	= x, y and z, components of gravity force
h	= grid distance
H_{min}	= minimum grid distance
k_{eff}	= effective thermal conductivity of porous medium, kW/m·K
L	= length of the (channels), pipes, m
n	= number of channels per unit cross-sectional area, $1/m^2$
N	= number of channels in a cross-sectional area
Nu	= Nusselt number
$Nu_{average}$	= Average Nusselt number on side walls or top or bottom of the container
$\vec{n}$	= normal unit vector
P	= pressure, N/m^2

Q	= volume flow rate, m^3/s
Q_{heat}	= internal heat source in energy equation, kW/m^3
Q_i	= flow rate in a single pipe, m^3/s
Q_{10}	= temperature quotient of respiration of fruit and vegetables
r	= effective radius, m
Ra	= Rayleigh number
R_c	= defined in equation (3.97a)
R_1 , R_2	= respiration rates, mg Co_2/kg·h
S	= cross-sectional area, m^2
S_v	= specific surface 1/m
t	= time, s
t_1 , t_2	= temperatures, °C
Δt_{inner}	= iterative time step for solving vector potential
Δt_{main}	= iterative time step for main loop
T	= Temperature, K
u, v, w	= x, y and z components of velocity
$\vec{V}_f$	= intrinsic velocity, m/s
$\hat{V}_f$	= volume of fluid inside a r.e.v., m^3
$\vec{V}_p$	= seepage velocity, m/s
$\hat{V}_p$	= volume of r.e.v., m^3
x, y, z	= coordinate
x_0, y_0, z_0	= rectangular enclosure dimensions, m

Greek Symbols

ε	= porosity
ρ	= density, kg/m^3
μ	= dynamic viscosity, kg/m·s
$\tilde{\mu}$	= effective viscosity, kg/m·s
K	= permeability tensor
α_n	= convective heat transfer at boundaries, $kW/m^2 \cdot {}^\circ C$
α, β, γ	= angles between gravity direction and x, y and z coordinate axis
ψ	= vector potential, m^2/s
ϕ	= scalar potential, m^2/s
ϕ_{water}	= water content in fruits or vegetable, per cent %
ρ_0	= reference air density at 273K, kg/m^3
β_0	= thermal coefficient of volume expansion, 1/K
κ	= permeability, m^2
θ	= dimensionless temperature
α_ψ	= false transient coefficient of vector potential
ω	= intermediate variable for Samarskii-Andreyev Implicit Alternating Direction Method
δ	= constant, 0-1
Δt	= iteration time step length

Subscripts

p	=porous media
f	= fluid
air	= air
s	= solid
x, i	= x direction or x component
y, j	= y direction or y component
z, k	= z direction or z component
h	= hot temperature
c	= cold temperature
start	= initial temperature

Superscript

'	= non-dimensional mark

CHAPTER 1

INTRODUCTION

1.1 Background

In modern society the distribution and marketing of fresh agricultural and horticultural produce is a vitally important activity and the endeavour put into it is reflected in the general standard of living of the community. Distributing and marketing of fresh fruits and vegetables encompasses storage, packaging, handling and transportation. The distribution chain for fresh produce is greatly dependent on good storage and preservation methods, which in turn, are based upon reliable knowledge of the physical and physiological processes of individual commodities. Storage conditions are not only important in long-term cold storage, but also in transport vehicles, interim storage before packaging or retail outlet storage. Regardless of the manner in which produce is stored, the principal aim of any storage operation is to extend the useful life of a commodity by preventing significant deterioration in its quality.

Unless suitable storage and preservation techniques are available to prolong shelf life, most fruits and vegetables would have to be consumed within a relatively short period of time. Advances in agriculture have generally resulted in higher yields of fruit and vegetable crops, however if seasonally harvested crops are not able to be utilised over a period considerably longer than their growing season, higher yields do not necessarily lead to increased benefits - storage is indispensable for prolonging the availability of fresh produce to consumers, extending the food processing season, damping fluctuations in supply and minimising market prices.

Fresh commodities are often highly perishable; Salunkhe (1974) estimates that 25% to 50% of all produce harvested is not consumed because of spoilage during storage or distribution. Billions of dollars are lost annually by growers, shippers, warehouse owners, processors and retailers around the world. Essentially, this is because fruits and vegetables are living organisms even after they are detached from the plant. The life processes require energy and the produce has to supply this from its own reserves and their continued respiration uses up oxygen and gives off carbon dioxide and heat. Produce sustains a continuous weight loss and constantly undergoes changes in chemical composition which may adversely affect its quality. Also this process of deterioration renders the produce more susceptible to microbial attack. The other important process common to produce in storage is water vapour loss or transpiration. Most fruits and vegetables have a water content of 75%-90%. Every percentage point of produce weigh loss may add up to millions of dollars worth of produce over a season in a large-scale distribution system.

The environmental factors affecting transpiration rates are temperature, relative humidity, air velocity and barometric pressure in the storage room or inside the storage container. High relative humidity can reduce the water loss rate but may enhance accelerated decay. To slow the deterioration, a low temperature is necessary to reduce the respiration rate. The quality of storage produce is often critically influenced by storage conditions. Table 1.1, after Hardenburg (1986), shows the recommended temperature and the suggested storage time for a variety of fruits and vegetables.

Table 1.1 Recommended temperature and approximate storage life of fresh fruits and vegetables in commercial storage (Hardenburg 1986)

Commodity	Temperature ^0C	Approximate storage life
Apple	-1-4	1-12 months
Apricots	-0.5-0	1-3 weeks
Kiwifruit	-0.5-0	3-5 months
Oranges (Fla. & Texas)	0-1	8-12 weeks
Pears	-1.5 to -0.5	2-7 months
Asparagus	0-2	2-3 weeks
Beans (green)	4-7	7-10 days
Broccoli	0	10-14 days
Brussels sprouts	0	3-5 weeks
Cabbage (Chinese)	0	2-3 months
Carrots (mature)	0	7-9 months
Celery	0	2-3 months
Corn (sweet)	0	5-8 days

Local differences in temperature within the storage rooms or in storage containers will cause different local rates of deterioration in different places. Thus non-uniform conditions will affect the overall quality of the stored produce. Sometimes unacceptable deterioration in certain areas leads to the storage having to be suspended immediately to prevent further loss or disease spread. Therefore a uniform temperature distribution in storage rooms and containers is considered highly desirable. Uniform temperature distribution is also important for fruit ripening, for example, bananas ripening prior to selling and potatoes prior to certain food processing operations. Certain commodities, notably onions and potatoes, are stored at ambient temperature rather than in cool or refrigerated storage. Optimal uniform temperature is maintained to avoid decay or chill injury. With respiration heat generated inside a closed container, there is clearly a temperature difference between the inside of the container and the environment. The density differences in air due to temperature differences will produce a buoyancy force and natural convection will occur in the container. The warm air will rise in the central area toward the top; it will then be cooled by the walls and move down near the walls to the bottom.

Natural convection occurs in some pre-cooling processes also. Normally the field heat in the produce when harvested is removed by some kind of pre-cooling operation as quickly as possible. Hydro-cooling, vacuum cooling and cold room cooling are examples of different pre-cooling processes. For some commodities which do not require, or tolerate rapid cooling to a low temperature, or those with a package not suitable for hydro-cooling, cold storage room cooling is usually the method chosen. In this case, the container or box can be considered as a relatively impervious to air flow around it and the heat transfer is mainly the result of natural convection inside the box. Forced or mixed convection may take place in some fast efficient pre-cooling situations. Some commodities which do not tolerate rapid air velocity, which causes excessive water loss, can benefit from natural convective pre-cooling.

How natural convection affects the temperature distribution inside stored packaged produce, and how the respiration heat contributes to the temperature accumulation are processes about which a deeper understanding is sought in this study. In order to improve storage techniques we have to recognise how stored commodities respond to the external environment and gain insight into the processes that govern the rate of heat and mass transfer inside storage facilities.

1.2 Literature Review

Considerable research has been carried out in the area of cold storage of agricultural and horticultural produce, but until recently, little attention has been directed toward to the influence of natural convection on the heat and mass transfer processes taking place during the cooling of produce. Moon (1987) performed a study on mixed convective hear transfer from horizontal cylinders in order to predict post harvest cooling of cylindrically shaped agricultural produce. Woods (1990) discussed at some length the physical phenomena responsible for moisture loss from commodities with high moisture contents.

Computer modelling of the thermal balance within a bin of apples was carried out to predict temperature distributions and air pattern movements by Robinson (1988). Numerical and experimental analysis of mixed convection around a sphere which was aimed at understanding the heat transfer from a single fruit was carried out by Johnson (1988) and Tang and Johnson (1992).

Interest in natural convective fluid flow and heat transfer in porous media has been influenced by a broad range of applications which include solidification of castings, acquifiers, geothermal systems, insulating materials and oil field production. Another application which has attracted a good deal of attention is the storage of nuclear waste materials, this differs from the previously mentioned examples in that it has been modelled as a natural convective process in a porous media with internal heat generation. It is proposed in this study, that an essentially analogous process takes place within impervious containers used to store agricultural produce, where convective heating or cooling of the contents occurs in conjunction with internal heat generated by the respiring produce.

In the design of cool storage facilities it has been the custom to neglect the affects of natural convection in favour of conductive heat transport when determining temperature distributions and cooling rates in closed containers. However, during cooling (or heating) natural convection must occur within closed containers if temperature gradients exist between the container walls and the produce. During the transient cooling of stored commodities that do not generate respiratory heat, natural convection processes will cease completely when steady state conditions are reached. On the other hand, for those commodities that do generate internal heat, natural convection continues, even after steady state has been achieved.

Repeated transient heating and cooling is the kind of process that might still carelessly be ignored by present day design engineers, but the lesson we learn from history is that transient heat transfer can be of overwhelming importance. Designers of food or produce storage enclosures should be well aware that such systems require comparatively little energy to keep products cool at steady state conditions. The consumption of energy to bring stored products to a low temperature constitutes the principal cost of operating such systems. Transient heat and mass transfer processes are a dominant concern in the design of storage units. Although there has been a longstanding interest in the storage of agricultural and horticultural produce, few researchers have investigated the influence in transient heat and mass transfer phenomena of natural convection in combination with respiration.

A number of studies concerned with natural convection in porous media with internal heat generation have been carried out. Gasser and Kazimi (1976) applied the method of linear stability of small disturbances to calculate the critical internal and external Rayleigh numbers that identify the onset of natural convection in porous media with internal heat generation. Buretta and Berman (1976) conducted experiments on a liquid saturated porous layer heated from below or heated from distributed sources. An experimentally obtained slope of the Nusselt versus Rayleigh number relationship was presented but details of temperature and velocity fields were not given. Experimental results have been presented by Hardee and Nilson (1977); Rhee, Dhir and Catton (1978) and Kulacki and Freeman (1979). Analytical extensions to the problem were made by Tveitereid (1977) who obtained a steady state solution in the form of hexagonal cells and two-dimensional rolls for natural convection in a horizontal porous layer heated from within.

Beukema et al (1980, 1983) reported a three-dimensional study of natural convection in a confined porous medium with internal heat generation to model the process of storing fruit and vegetables. Both numerical and experimental results were presented and discussed. In their study, they assumed that heat generation was uniformly distributed in the porous medium. The heating rate was assumed constant for each numerical experiment and results for a number of different values of constant heating rate were given. Darcy's law was applied with the permeability of the porous medium so constructed as to ensure isotropy. To solve the governing equations the Alternative Direction Implicit Procedure, Douglas and Rachford (1956), was used.

Some questions have been raised concerning the validity of the assumption that packed fruit and vegetables posses homogeneous permeability. For produce such as oranges, apples and Brussels sprouts, which are close to spherical, resistance to air flow through them may well be considered isotropic, but other produce, such as beans, cucumber, carrots or celery may better be described as having air flow resistance tensors that are transversely orthotropic. Also, the assumption that agricultural produce respires at a constant rate is not accurate. For example, Hardenburg et al (1986) pointed out that respiration rates are highly variable between species, and they are very responsive to the conditions under which produce is stored.

1.3 Objectives

In this study we wish to examine the theory, analysis and practical application of transient, multi-dimensional natural convection in a porous medium, having isotropic or orthotropic permeability, to heating or cooling of stored agricultural produce packed in closed containers of different size; with different temperature dependent respiration functions.

The key to improving the non-uniform temperature distribution and controlling loss of moisture in packed produce lies in developing an effective method of predicting the velocity and temperature fields for given boundary conditions. To date it appears that few such studies have been attempted and a successful outcome will have considerable direct application to a host of storage problems.

The specific aims of the research program are:

1. To formulate the equations governing natural convection in packed agricultural produce, including the mathematical treatment of respiratory heating sources and boundary conditions.

2. To apply a numerical analysis process appropriate to solve the form of the governing equations and to evaluate the accuracy and efficiency of the methods used.

3. To present numerical results for the heating or cooling of selected packed produce stored in differently sized closed containers under varying boundary and permeability conditions.

CHAPTER 2

RESPIRATION OF AGRICULTURAL PRODUCE AND ITS MATHEMATICAL MODELLING

Fruits and vegetables are living structures even after they are harvested from the field or removed from their stems. They keep respiring and transpiring. These metabolic activities are dependent entirely on food reserves and moisture content.

The main metabolic process taking place in harvested produce is respiration. Respiration can be described as the oxidation breakdown of the more complex materials normally present in cells such as starch, sugars and organic acids, to simpler molecules such as CO_2 and H_2O, with the concurrent production of energy, and other molecules. In most cases respiration involves taking up Oxygen (O_2) and giving off carbon dioxide (CO_2) and heat. Respiration heat directly affects the temperature of the fruits and vegetables. Respiration in fruits and vegetables is mostly under enzymic control. Usually enzymes lose their activity at temperatures above 30 °C but the temperature at which specific enzymes become inactive varies. Many are still active at 35 °C but most are inactive at 40 °C (Wills, 1977).

When fruits and vegetables are held above the temperature at which respiration ceases or becomes abnormal, the structure of the produce may be damaged. The low temperature limit for normal metabolic activity is near the freezing point of the tissue and is usually between 0 °C and -2 °C.

The rate of respiration is principally governed by temperature. The higher the temperature, the faster is the metabolic reaction rate in living produce and the greater the heat generated. Typically, for each 10 °C rise in temperature, the rate of respiration is roughly doubled or tripled. The effect of temperature on agricultural produce is usually expressed in terms of the temperature quotient, Q_{10}, which is defined as follows

$$Q_{10} = \left(\frac{R_2}{R_1}\right)^{\frac{10}{(t_2 - t_1)}} = constant \qquad (2.1)$$

where t_1 and t_2 are respiration temperatures (°C) and R_1 and R_2 are the corresponding respiration rates (mg Co_2/kg h). According to Ryall (1979), Q_{10} is not actually a constant and may take any value between 1 and 5. Values of Q_{10} are highest between 0 °C and 10 °C, commonly ranging between 2 and 3. In the temperature range 10 - 32 °C values of Q_{10} tend toward 1.

Table 2.1, from Ryall (1979), contains temperature quotient information derived from the respiration rates of various vegetables. Generally the rate of respiration increases as the temperature of the produce rises from just above its freezing point. At a certain temperature the respiration rate reaches a maximum value; beyond this temperature the respiration rate decreases sharply with temperature, until finally respiration ceases at the thermal death point - see Ryall (1979).

Table 2.1 Temperature quotients (Q_{10}) for rates of respiration of various vegetables (Ryall, 1979)

Vegetable	Temperature Range (°C)					
	0-5	5-10	10-15	15-20	20-25	25-30
Asparagus	3.3	4.2	1.2	2.3	1.5	2.0
Beans, snap		2.7	2.6	1.9	2.2	
Broccoli	5.2	4.6	3.9	2.7		
Brussels sprouts	4.9	2.7	1.5			
Cabbage	2.5	1.5	3.2	1.6		
Carrot, roots	2.0	2.3	1.5	2.8		
Celery	3.5	4.0	1.7			
Potatoes	2.2	3.4	2.5	2.1		
Tomatoes				2.6	1.5	1.2

Respiration rates of various fruits and vegetables, from Hardenburg (1986), are shown in Table 2.2. The respiration rates have been expressed as rates of carbon dioxide production (mg CO_2/kg·h) at different temperatures. Data from Tables 2.1 and 2.2 were used in equation (2.1) to calculate respiration rates for asparagus and for Brussels sprouts. The results are presented in Table 2.3 and have been compared with the measured values given in Table 2.2.

Although use of the temperature quotient allows moderately good prediction of respiration rates, it was considered that better results could be obtained by using the method of least squares to obtain a relationship between the measured respiration rates and temperatures. Thus the locally distributed heat source within the packed bed could be readily calculated from the local temperature distribution when used in the numerical model. In developing the least squares model for respiration, it was assumed that at temperatures above 35°C and below freezing point, the heat generation rates were zero. Using the respiration heat generation rates, $Q_i(T_i)$ and corresponding temperatures, T_i known from measurement, a function $Q(i) = \sum_{i=0}^{n} a_i T^i$ can be found by determining the parameters a_i. The subroutine QSOURCE in Appendix D was used in the modelling program. A schematic diagram of a typical respiration rate versus temperature function is shown in Figure 2.1, Ryall and Lipton (1979).

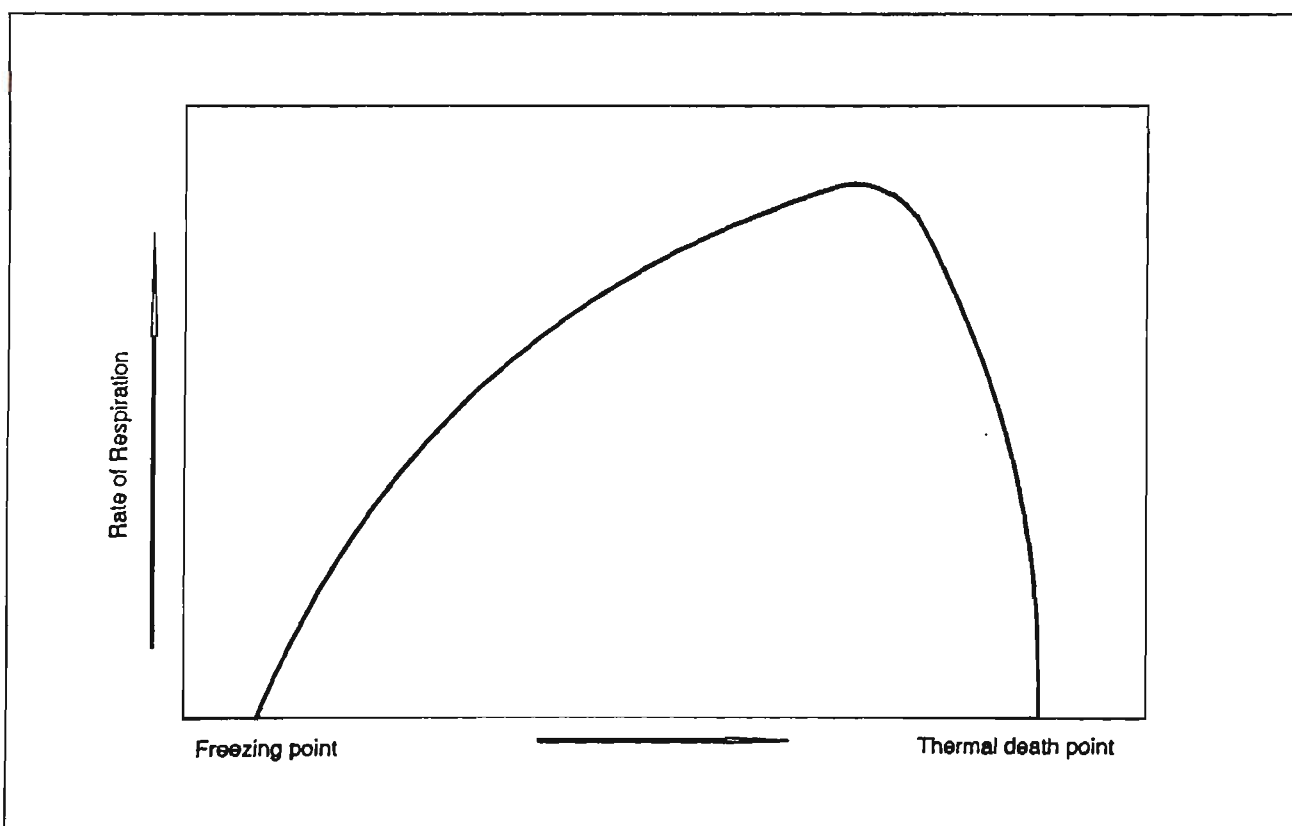


Figure 2.1 Typical respiration rate versus temperature changing.

Table 2.2 Respiration rates of fruits and vegetables, expressed as rates of carbon dioxide production (mg Co_2/kg·h), at various temperatures, from Hardenburg (1986)

	Temperature					
Commodity	0°C	4-5°C	10°C	15-16°C	20-21°C	25-27°C
Apples, summer	3-6	5-11	14-20	18-31	20-41	-
Apples, fall	2-4	5-7	7-10	9-20	15-25	-
Apricots	5-6	6-9	11-19	21-34	29-52	-
Asparagus	27-80	55-136	90-304	160-327	275-500	500-600
Beans, snap	20	35	58	93	130	193
Broccoli	19-21	32-37	75-87	161-186	278-320	-
Brussels sprouts	10-30	22-48	63-84	64-136	86-190	-
Carrots, topped	10-20	13-26	20-42	26-54	46-95	-
Carrots, bunched	18-35	25-51	32-62	55-106	87-121	-
Cauliflower	16-19	19-22	32-36	43-49	75-86	84-140
Celery	5-7	9-11	24	30-37	64	-
Kiwifruit	3	6	12	-	16-22	-
Pears, Bartlett	3-7	5-10	8-21	15-60	30-70	-
Pears, Kieffer	2	-	-	11-24	15-28	20-29
Potatoes, immature	-	12	14-21	14-31	18-45	-
Potatoes, mature	-	3-9	7-10	6-12	8-16	-
Tomatoes, mature-green	-	5-8	12-18	16-28	28-41	35-51
Tomatoes, ripening	-	-	13-16	24-29	24-44	30-52

Table 2.3 Average respiration rate(mg Co_2/kg·h): by measured, calculation of Q_{10} and least squares modelling

	Temperature	0°C	5°C	10°C	15°C
Asparagus	Measurement	53.5	95.5	197.0	243.5
	Q_{10} method	53.5	97.2	199.2	218.2
	Least squares method	53.5	95.5	196.9	243.5
Brussels sprouts	Measurement	20.0	35.0	73.5	100
	Q_{10} method	20.0	44.3	72.8	89.2
	Least squares method	20.0	35.0	73.5	100

CHAPTER 3

THEORY OF NATURAL CONVECTION IN BULK PACKED AGRICULTURAL PRODUCE

3.1 Physical Model and Some Concepts of Porous Medium

Most of the containers for fruits and vegetables storage are rectangular boxes. In this study we consider a physical system with the following characteristics: A rectangular cartesian reference system is used to define a container filled with fruits or vegetables, which are considered as the solid matrix of a porous medium saturated with air, such as shown in Figure 3.1. The respiration heats of fruit and vegetables are treated as internal heat sources which are functions of temperature. The coordinate system is also shown in Figure 3.1; the x axis is along one edge of the rectangular enclosure and $\vec{G}$ is the gravity force.

In order to describe natural convection in bulk-packed fruit or vegetables, the theories on natural convection in porous medium are introduced. The produce inside the container are considered as a porous medium. Beukema(1980,1983) showed that these theories may be applied to quantify the natural convection phenomena that occur in packed beds of agricultural produce. A porous medium is a material consisting of a solid matrix with interconnecting voids. As Dullien (1979) pointed out, a material or a structure must pass at lease one of the following two tests in order to qualify as a porous medium:

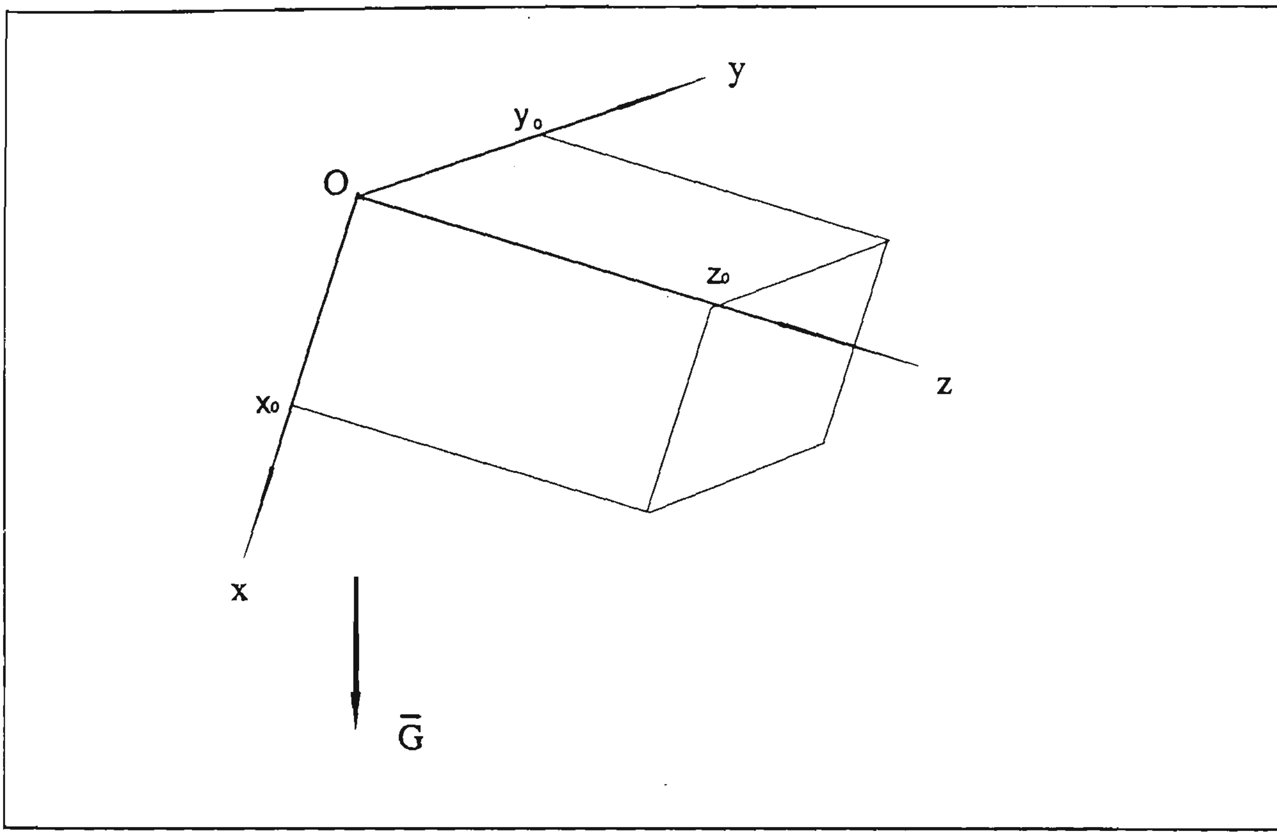


Figure 3.1 Coordinate system used in this study

(1) It must contain spaces, so-called pores or voids, free of solids, imbedded in the solid or semisolid matrix. The pores usually contain some fluid, such as air, or a mixture of different fluids.

(2) It must be permeable to a variety of fluids, ie, fluids should be able to penetrate through one face of a spectrum made of the material and emerge on the other side.

From these points of view, packed fruits and vegetables can be classed as porous medium.

Porosity ε plays an important role in a porous medium and is defined as the fraction of the total volume of the medium that is occupied by void space. Then $1-\varepsilon$ is the fraction that is occupied by solid, Nield and Bejan (1992). In order to carefully analyze systems in which natural convection is taking place, we must construct a continuum model for the porous medium based on the representative elementary volume (r.e.v.) concept. We introduce a cartesian reference system, as shown in Figure 3.2, the r.e.v. is such a control volume which is very large compared with the pore volumes but very small compared with the macroscopic (whole) flow domain.

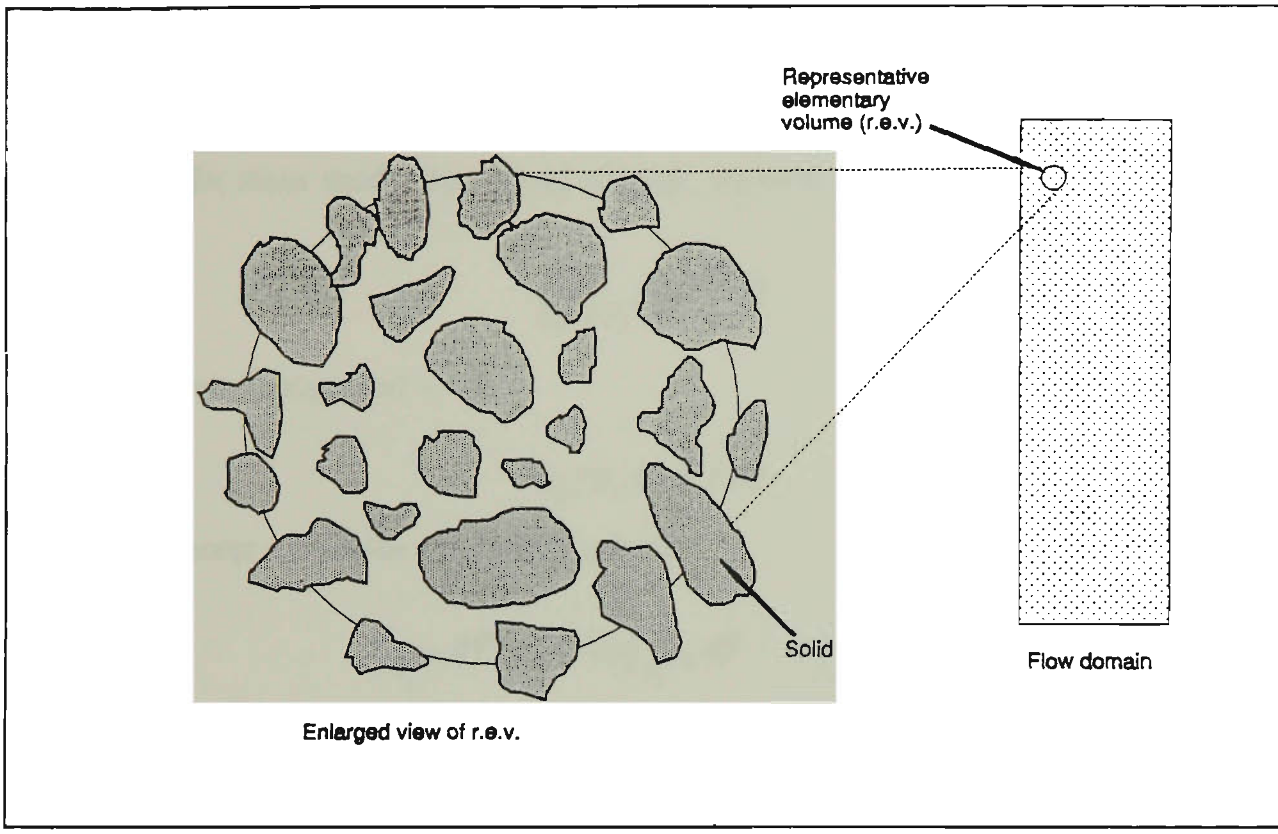


Figure 3.2 The representative elementary volume (r.e.v.): the figure illustrates.

A distinction is made between an average taken with respect to the volume $\hat{V}_p$ of r.e.v. (incorporating both solid and air) and one taken with respect to a volume element $\hat{V}_f$ consisting of fluid only. So we have two velocities according to these two different averages. One is the seepage velocity $\vec{V}_p$ and another is the intrinsic velocity $\vec{V}_f$. The Dupuit-Forchheimer relationship is

$$\vec{V}_p = \epsilon \vec{V}_f \tag{3.1}$$

In this thesis, unless specifically indicated, the velocity refers to seepage velocity, which is the average of fluid velocity over the porous medium control volume $\hat{V}_p$.

3.2 Continuity Equation

The principle of conservation of mass can be stated as:

$$(\textit{Time rate of change of the mass of the r.e.v.})=0 \tag{3.2}$$

considering the mass inside the control volume, we have

$$\frac{d}{dt}\left(\int_{\hat{V}_p} \rho_p \, d\hat{V}\right)=0 \tag{3.3}$$

ρ_p is a function of *x,y,z* and *t*, viz

$$\rho_p=\rho_p\,(x, y, z, t) \tag{3.4}$$

From the concept of porous medium, we have

$$\int_{\hat{V}_p} \rho_p d\hat{V} = (1-\epsilon)\int_{\hat{V}_s} \rho_s \, d\hat{V} + \epsilon\int_{\hat{V}_f} \rho_f d\hat{V} \tag{3.5}$$

where ρ_s is the density of solid matrix, ρ_f is the density of fluid.

Because ε and ρ_s are independent of time,

$$\frac{d}{dt}\left(\int_{\hat{V}_p} \rho_p d\hat{V} = \epsilon\frac{d}{dt}\int_{\hat{V}_f} \rho_f \, d\hat{V} = 0 \tag{3.6}$$

Application of the Reynolds transport theorem equation 3.6 leads to

$$\epsilon\frac{d}{dt}\int_{\hat{V}_p} \rho_p d\hat{V} = \epsilon\int_{\hat{V}_f} \frac{\partial \rho_f}{\partial t} d\hat{V} + \epsilon\int_{A_f} \rho_f \, \vec{V}_f \cdot \vec{n} \, dA = 0 \tag{3.7}$$

Using the divergence theorem, equation 3.7 changes to

$$\int_{\hat{V}_f} [\, \epsilon\frac{\partial \rho_f}{\partial t} + \epsilon \, \nabla\cdot(\rho_f \, \vec{V}_f)\,] \, d\hat{V}=0 \tag{3.8}$$

Since the control volume is arbitrary and ρ, V are assumed to be continuous functions with continuous derivatives, the integrand in the integral must be zero and we obtain the continuity equation:

$$\epsilon \frac{\partial \rho_f}{\partial t} + \nabla \cdot (\rho_f \vec{V}_p) = 0 \tag{3.9}$$

where $\epsilon \frac{\partial \rho_f}{\partial t}$ indicates the rate of mass increase within the control volume, and $-\nabla \cdot (\rho_f \vec{V}_p)$ denotes the net mass flux into the volume. Since mass cannot be generated or destroyed, these two terms must be equal.

To simplify this equation, it is assumed that the environment pressure, ie, atmospheric pressure remains constant and the fluid, air, is incompressible, ie, its properties remain constant except for the density variation in producing the buoyancy force. Finally equation 3.9 becomes

$$\nabla \cdot \vec{V}_p = 0 \tag{3.10}$$

3.3 Momentum Equation: Darcy's Law

The principle of momentum conservation may be stated as

(the time rate of changing of linear momentum of the r.e.v.)
=(the force acting on the r.e.v.) (3.11)

Before we discuss the momentum equation, some assumptions are introduced. The properties of the fluid are isotropic except the density which will satisfy the Boussiniq approximation.

In 1856, from his experiments on the flow through horizontally stratified beds of sand, Henri Darcy concluded the relationship between flow rate and the pressure drop. Now termed Darcy's law, it has the form in modern notation:

$$V_p = -\frac{\kappa}{\mu} \nabla p \tag{3.12}$$

where the ∇P is the pressure gradient. μ is the dynamic viscosity of the fluid. κ is the permeability. It depends on the geometry of the porous medium and in here it represents isotropic permeability.

An alternative to Darcy's law is Brinkman's equation with the Boussinesq approximation, ie, the buoyancy force is due to the density variation and the gravity acceleration, and the buoyancy force is the main factor in producing natural convection. So we have

$$\rho_f [\frac{\partial \vec{V}_f}{\partial t} + (\vec{V}_f \cdot \nabla) \vec{V}_f] = -\nabla P - \frac{\mu}{\kappa} \vec{V}_p + \tilde{\mu} \nabla^2 \vec{V}_p + \vec{G} \tag{3.13}$$

where $\tilde{\mu}$ is a effective viscosity and may have a different value than the viscosity of the fluid. Neale and Nader (1974) pointed out that they can take the same value. $\vec{G}$ is the body force.

Many authors (e.g. Rudraiah and Prabhamani, 1974; Neale and Nader 1974; Gasser and Kazimi, 1976; Walker and Homsy, 1977; Tong and Subramanian, 1985; Beckermann, et al 1987; Singh, et al 1993) used Brinkman's equation to describe natural convection in porous medium. However, Chan, Ivey and Barry (1970), found that the contribution of the viscous term $\mu \nabla^2 \vec{V}_p$ is very small even in low Rayleigh number. Furthermore Whitaker (1966,1969) has shown that the Darcy term $\frac{\mu}{\kappa} \vec{V}_p$ replaces the viscous force and therefore it is incorrect to use both terms in the momentum equation. In this study we also neglected these terms based on the following analysis:

$$\mu = \tilde{O}(2\times10^{-5}) \tag{3.14}$$

$$K = \tilde{O}(2\times10^{-6}) \tag{3.15}$$

$$\tilde{\mu} = \tilde{O}(2\times10^{-5}) \tag{3.16}$$

$$\frac{\mu}{K}\vec{V}_p = \tilde{O}(10)\vec{V}_p \tag{3.17}$$

$$\mu\nabla^2\vec{V}_p = \tilde{O}(2\times10^{-6})\,\vec{V}_p \tag{3.18}$$

Thus we have:

$$\frac{\mu}{K}\vec{V}_p > \mu\nabla^2\vec{V}_p \tag{3.19}$$

Compared to the resistance of solid matrix, the viscous resistance term is much smaller.

Beck(1972) pointed out that inclusion of the $(\vec{V}_p\cdot\nabla)\vec{V}_p$ term was inappropriate and furthermore Nield(1992) suggested to drop this term in the equation. He showed that in the case of a viscous fluid a material particle retains its momentum in the absence of applied forces when it is displaced from a point A to a neighbouring arbitrary point B. But in a porous medium with a fixed solid matrix this is not so, in general, because some solid material impedes the motion and causes a change in momentum. Thus it is not rational to include the convective term $(\vec{V}\cdot\nabla)\vec{V}$ in the momentum equation unless the porosity is very large, and in that case it can be queried whether one really has a porous medium, in the usual sense of that term, as distinct from a fluid in which there are some solid obstructions. For the same reason it does not make sense to talk about turbulence on a macroscopic scale in a natural porous medium because one can not have unimpeded eddies of arbitrary size.

We can also drop the time derivative term $\rho_f \frac{\partial \vec{V}_f}{\partial t}$ because in general the transients decay rapidly. With these terms dropped we finally have the momentum equation in isotropic permeability porous medium.

$$-\nabla P - \frac{\mu}{\kappa} \vec{V}_p + \vec{G} = 0 \tag{3.20}$$

This is actually Darcy's law.

Many of the fruits and vegetable have spherical shapes and can be treated as isotropic porous medium. But many others have cylindrical shapes and can be considered as an orthotropic permeability porous medium. As shown in Figure 3.3, orthotropic permeability is a second order symmetric diagonal tensor having different permeability in component ***x, y, z*** directions.

$$\mathbf{K} = \begin{vmatrix} \kappa_x & 0 & 0 \\ 0 & \kappa_y & 0 \\ 0 & 0 & \kappa_z \end{vmatrix} \tag{3.21}$$

Permeability is related to the pore size distribution, since the distribution of the pore sizes and directions entrances and exits, and lengths of the pore walls constitutes the primary resistance.

Permeability is a function of porosity, tortuosity and connectivity. Tortuosity is defined as the relative average length of a flow path, ie, the avenge length of the flow paths to the length of the medium. Connectivity defines the arrangement and number of pore connections. Although permeability is related to these parameters, none of them can be used alone to predict the permeability. Normally the structure of a real porous medium is

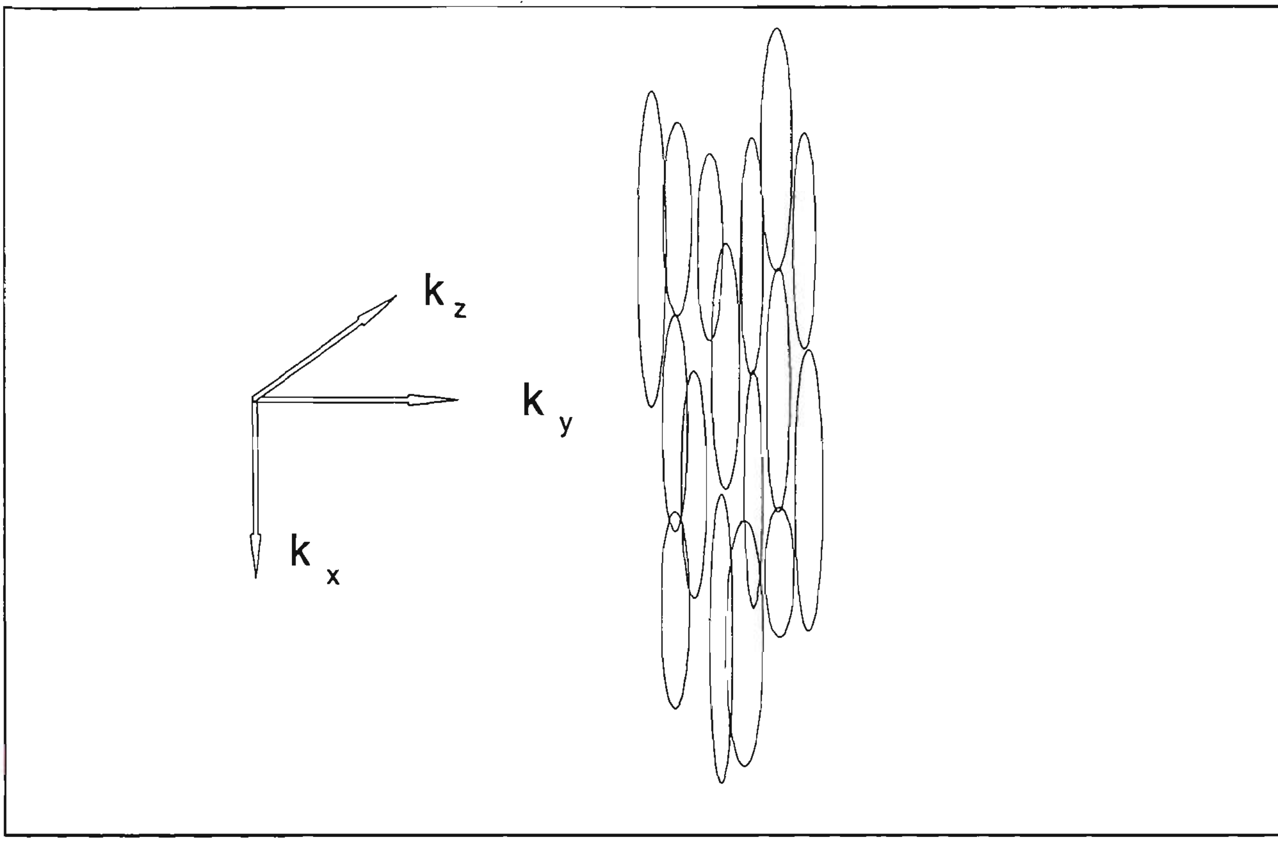


Figure 3.3 Orthotropic permeability arising as a result of alignment of the commodities in the gravitational field

complicated and it is rarely possible to predict the permeability theoretically. Experimental data are the main source of quantifying permeability.

A well-known expression of permeability for packed spherical particles is the Carman-Kozeny formula:

$$\kappa = \frac{D_{eff}^2\,\epsilon^3}{180\,(1-\epsilon)^2} \tag{3.22}$$

where D_{eff} is the effective diameter of the particles or is called volume-surface mean diameter

$$D_{eff} = \frac{6}{S_v} \tag{3.23}$$

Here S_v is the specific surface, ie, surface of solids per unit volume of solids.

Many authors have used the Carman-Kozeny formula to determine the permeability, for example, Nield (1991). However, Ergun (1952) pointed out that the permeability should have the form

$$\kappa = \frac{D_{eff}^2 \, \epsilon^3}{150(1-\epsilon)^2} \tag{3.24}$$

Plumb and Huenefeld (1981) and Beckermann al et (1987) quoted Ergun's finding but actually Beckermann al et (1987) modified the constant from 150 to 175 and used

$$\kappa = \frac{D_{eff}^2 \, \epsilon^3}{175\,(1-\epsilon)^2} \tag{3.25}$$

in their work.

In order to predict the permeability of packed apples, oranges etc. having spherical shapes the Carman-Kozeny formula was employed in this study. Because of their non-spherical shape, packaged beans, carrots, celery etc, possess permeabilities which vary in different directions. No study related to predicting orthotropic permeability was found in the literature. Nevertheless, one common pattern is discussed below.

As shown in Figure 3.3, if the commodities are placed parallel to each other and the x axis is along their axis line, we may assume that

$$K_z = K_y = aK_x \tag{3.26}$$

Where a is a proportional constant.

For simplicity, assume that the spaces between commodities are channels and the channels are connected from head to head and can be thought of as a pipe having an effective radius

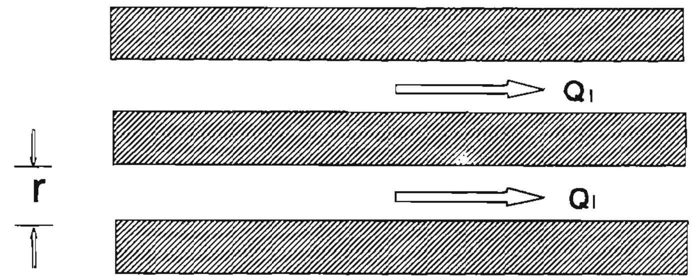


Figure 3.4 Flow model based on parallel, uniform channels

r, as shown in Figure 3.4, Applying theory of fluid flow in pipes, we have the equation for laminar flow rate in a single pipe,

$$Q_i = \frac{\pi r^4}{8\mu} \frac{\Delta P}{L} \tag{3.27}$$

where ΔP is the pressure drop.

For *N* pipes the total flow rate is given by

$$Q = NQ_i = \frac{N\pi r^4}{8\mu} \cdot \frac{\Delta P}{L} \tag{3.28}$$

applying Darcy's law in the *x* direction, we obtain

$$Q = S V_p = S \cdot \frac{\kappa_x}{\mu} \cdot \frac{\Delta P}{L} \tag{3.29}$$

where S is the cross-sectional area.

Combining the above equations and eliminating *Q*. We have

$$\kappa_x = \frac{N\pi r^4}{8S} \tag{3.30}$$

Thus

$$\kappa_x = \frac{N\pi r^4}{8S} = \frac{n\pi r^4}{8} \tag{3.31}$$

where, n = number of channels per unit cross-sectional area.

The lack of a rigorously valid method for determining the radius and the number of channels in a randomly packed porous medium is a difficulty that has often been raised. In practice however, a number of well-grounded approximations have been developed and may be used in the Darcy equation.

For orthotropic permeability porous media, the momentum equation reduces to

$$-\nabla P - \frac{\mu}{\mathbf{K}}\vec{V}_p + \vec{G} = 0 \tag{3.32}$$

3.4 Energy Equation

In order to derive the thermal energy transport equation that governs heat transfer in packed beds certain assumptions must be made. Suppose the viscous dissipation were neglected. The atmospheric pressure is constant and the work done by pressure change is also negligible. The radiative heat transfer may be ignored due to the temperature difference being small. The properties of fluid and solid are constant and isotropically homogenous.

The conservation of energy can be stated as

(*the amount of heat for temperature change incontrol volume*)
=(*the amount of heat transfer due toconvection*)
+(*the amount of heat due to conduction*)
+(*the amount of heat due to internal heat generation*) (3.33)

Expressed mathematically, we have

$$(\rho C)_p \frac{\partial T}{\partial t} = -(\rho C_p)_f \vec{V}_p \cdot \nabla T + k_{eff} \nabla^2 T + Q_{heat} \quad (3.34)$$

where

$$(\rho C)_p = (1-\varphi)(\rho C)_s + \varphi(\rho C_p)_f \quad (3.35)$$

C is specific heat of the solid,

C_p is specific heat of the fluid at constant pressure,

Q_{heat} is the heat production, W/m^3

k_{eff} is the effective thermal conductivity of porous medium.

The basic conservation equations of mass, momentum, and energy are

$$\nabla \cdot \vec{V}_p = 0 \quad (3.10)$$

$$-\nabla P - \frac{\mu}{\mathbf{K}} \vec{V}_p + \vec{G} = 0 \quad (3.32)$$

$$(\rho C)_p \frac{\partial T}{\partial t} = -(\rho C_p)_f \vec{V}_p \cdot \nabla T + k_{eff} \nabla^2 T + Q_{heat} \quad (3.34)$$

3.5 General Boundary Conditions

We consider that the walls of the enclosure are not permeable, ie, V_n=**0**, where the subscript ***n*** indicate a normal component to the walls.

The boundary conditions of momentum equation in detail are as shown below:

$$x=0: \qquad V_x=0 \tag{3.36}$$

$$x=X_0 \qquad V_x=0 \tag{3.37}$$

$$y=0: \qquad V_y=0 \tag{3.38}$$

$$y=Y_0 \qquad V_y=0 \tag{3.39}$$

$$z=0: \qquad V_z=0 \tag{3.40}$$

$$z=Z_0 \qquad V_z=0 \tag{3.41}$$

The boundary conditions for the energy equation (3.34) can reflect any of the heat transfer mechanisms listed below:

Heatflux $$\frac{\partial T}{\partial n}=Constant\neq 0 \tag{3.42}$$

Adiabatic $$\frac{\partial T}{\partial n}=0 \tag{3.43}$$

Isothermal $$T=Constant \tag{3.44}$$

Convective $$-k_{eff}\frac{\partial T}{\partial n}=\alpha_n(T-T_c) \tag{3.45}$$

Here n represents the x or y or z axis.

3.6 Vector-Potential Formulation: A Transform of Momentum Equation

The momentum equation (3.32) is written in the terms of the primitive variables, pressure and velocity. We may solve the set of equations in this form, as for example has Williams (1969) and Chorin(1968). Alternatively, we may solve them in terms of the vector potential, as has Mallinson(1973). Some benefits are obtained by using this approach. It reduces the number of equations to be solved by one and it ensures the continuity equation is automatically satisfied. The pressure term is eliminated and thus the difficulties associated with pressure boundary conditions are avoided. Aziz and Hellums(1967) found that the iterative procedure based on the primitive variable equations was less satisfactory in its convergence behaviour.

According to the continuity equation, $\nabla\cdot\vec{V}_p=0$, thus, V_p is a solenoidal; it can be shown, Hirasaki and Hellums (1970), that it is always possible to define a vector potential ψ and a scalar potential ϕ as

$$\vec{V}_p=\nabla\times\vec{\psi}-\nabla\phi \tag{3.46}$$

$$\nabla\cdot\psi=0 \tag{3.47}$$

$$\nabla^2\phi=0 \tag{3.48}$$

hence ψ is also solenoidal and ϕ satisfies the Laplace equation.

So we have the following relation

$$\begin{aligned}\nabla\times\vec{V}_p&=\nabla\times(\nabla\times\vec{\psi}-\nabla\phi)\\&=\nabla\times\nabla\times\vec{\psi}-\nabla\times\nabla\phi\\&=\nabla\times\nabla\times\vec{\psi}\\&=\nabla(\nabla\cdot\vec{\psi})-\nabla^2\vec{\psi}\\&=-\nabla^2\vec{\psi}\end{aligned} \tag{3.49}$$

By taking the curl of equation (3.32), we have

$$-\nabla\times\nabla P-\nabla\times(\frac{\mu}{\mathbf{K}}\vec{V}_p)+\nabla\times\vec{G}=0 \tag{3.50}$$

Using (3.46) and treating μ as constant, we have

$$\nabla\times(\frac{1}{\bar{\bar{\mathbf{K}}}}\nabla\times\vec{\psi})-\nabla\times(\frac{1}{\bar{\bar{\mathbf{K}}}}\nabla\phi)=\frac{1}{\mu}\nabla\times\vec{G} \tag{3.51}$$

To obtain the component equations of 3.51, firstly we expand each term. So we find:

$$\frac{1}{\bar{\bar{\mathbf{K}}}}=\bar{\bar{\mathbf{K}}}^{-1}=\begin{vmatrix}\frac{1}{\kappa_x} & 0 & 0\\ 0 & \frac{1}{\kappa_y} & 0\\ 0 & 0 & \frac{1}{\kappa_z}\end{vmatrix} \tag{3.52}$$

$$\nabla\times(\mathrm{K}^{-1}\nabla\phi)=\nabla\times\left[\begin{pmatrix}\frac{1}{\kappa_x} & 0 & 0\\ 0 & \frac{1}{\kappa_y} & 0\\ 0 & 0 & \frac{1}{\kappa_z}\end{pmatrix}\begin{pmatrix}\frac{\partial\phi}{\partial y}\\ \frac{\partial\phi}{\partial y}\\ \frac{\partial\phi}{\partial z}\end{pmatrix}(\vec{i}+\vec{j}+\vec{k})\right] \tag{3.53}$$

$$=\nabla\times\left(\frac{1}{\kappa_x}\frac{\partial\phi}{\partial x}\vec{i}+\frac{1}{\kappa_y}\frac{\partial\phi}{\partial y}\vec{j}+\frac{1}{\kappa_z}\frac{\partial\phi}{\partial z}\vec{k}\right)$$

$$=\frac{\partial^2\phi}{\partial y\partial z}\left(\frac{1}{\kappa_z}-\frac{1}{\kappa_y}\right)\vec{i}+\frac{\partial^2\phi}{\partial x\partial z}\left(\frac{1}{\kappa_x}-\frac{1}{\kappa_z}\right)\vec{j}+\frac{\partial^2\phi}{\partial x\partial y}\left(\frac{1}{\kappa_y}-\frac{1}{\kappa_x}\right)\vec{k}$$

$$\begin{aligned}\nabla\times(\mathbf{K}^{-1}\nabla\times\vec{\psi})= &\left[\frac{1}{\kappa_z}\left(\frac{\partial^2\psi_y}{\partial x\partial y}-\frac{\partial^2\psi_x}{\partial y^2}\right)+\frac{1}{\kappa_y}\left(\frac{\partial^2\psi_z}{\partial x\partial z}-\frac{\partial^2\psi_x}{\partial z^2}\right)\right]\vec{i}\\ &+\left[\frac{1}{\kappa_x}\left(\frac{\partial^2\psi_z}{\partial y\partial z}-\frac{\partial^2\psi_y}{\partial z^2}\right)+\frac{1}{\kappa_z}\left(\frac{\partial^2\psi_x}{\partial x\partial y}-\frac{\partial^2\psi_y}{\partial x^2}\right)\right]\vec{j}\\ &+\left[\frac{1}{\kappa_y}\left(\frac{\partial^2\psi_x}{\partial x\partial z}-\frac{\partial^2\psi_z}{\partial x^2}\right)+\frac{1}{\kappa_x}\left(\frac{\partial^2\psi_y}{\partial y\partial z}-\frac{\partial^2\psi_z}{\partial y^2}\right)\right]\vec{k}\end{aligned} \tag{3.54}$$

Because ψ is solenoidal, we may write

$$\begin{aligned}\nabla(\nabla\cdot\vec{\psi})= &\left(\frac{\partial^2\psi_x}{\partial x^2}+\frac{\partial^2\psi_y}{\partial x\partial y}+\frac{\partial^2\psi_z}{\partial x\partial z}\right)\vec{i}\\ &+\left(\frac{\partial^2\psi_x}{\partial x\partial y}+\frac{\partial^2\psi_y}{\partial y^2}+\frac{\partial^2\psi_z}{\partial y\partial z}\right)\vec{j}\\ &+\left(\frac{\partial^2\psi_x}{\partial x\partial z}+\frac{\partial^2\psi_y}{\partial y\partial z}+\frac{\partial^2\psi_z}{\partial z^2}\right)\vec{k}\\ =&\,\mathbf{0}\end{aligned} \tag{3.55}$$

thus it follows that

$$\frac{\partial^2\psi_y}{\partial x\partial y}=-\frac{\partial^2\psi_x}{\partial x^2}-\frac{\partial^2\psi_z}{\partial x\partial z} \tag{3.56}$$

$$\frac{\partial^2\psi_x}{\partial x\partial y}=-\frac{\partial^2\psi_y}{\partial y^2}-\frac{\partial^2\psi_z}{\partial y\partial z} \tag{3.57}$$

$$\frac{\partial^2\psi_x}{\partial x\partial z} = -\frac{\partial^2\psi_z}{\partial z^2} - \frac{\partial^2\psi_y}{\partial y\partial z} \tag{3.58}$$

Using the relationships of (3.56) to (3.58), equation (3.54) may deform to

$$\begin{aligned}\nabla\times(\mathbf{K}^{-1}\nabla\times\vec{\psi}) = & -\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_x}{\partial z^2} + (\frac{1}{\kappa_z} - \frac{1}{\kappa_y})\frac{\partial^2\psi_z}{\partial x\partial z}\right]\vec{i} \\ & -\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial y^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_y}{\partial z^2} + (\frac{1}{\kappa_z} - \frac{1}{\kappa_x})\frac{\partial^2\psi_z}{\partial y\partial z}\right]\vec{j} \\ & -\left[\frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial x^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_z}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial z^2} + (\frac{1}{\kappa_x} - \frac{1}{\kappa_y})\frac{\partial^2\psi_y}{\partial y\partial z}\right]\vec{k}\end{aligned} \tag{3.59}$$

Using equation (3.53) and (3.59) in equation (3.51), we find the ***x, y, z*** component equations:

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_x}{\partial z^2} + (\frac{1}{\kappa_z} - \frac{1}{\kappa_y})(\frac{\partial^2\psi_z}{\partial x\partial z} + \frac{\partial^2\phi}{\partial y\partial z})\right] = \frac{1}{\mu}(\frac{\partial G_z}{\partial y} - \frac{\partial G_y}{\partial z}) \tag{3.60}$$

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial y^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_y}{\partial z^2} + (\frac{1}{\kappa_z} - \frac{1}{\kappa_x})(\frac{\partial^2\psi_z}{\partial y\partial z} - \frac{\partial^2\phi}{\partial x\partial z})\right] = \frac{1}{\mu}(\frac{\partial G_x}{\partial z} - \frac{\partial G_z}{\partial x}) \tag{3.61}$$

$$-\left[\frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial x^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_z}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial z^2} + (\frac{1}{\kappa_x} - \frac{1}{\kappa_y})(\frac{\partial^2\psi_y}{\partial y\partial z} - \frac{\partial^2\phi}{\partial x\partial y})\right] = \frac{1}{\mu}(\frac{\partial G_y}{\partial x} - \frac{\partial G_x}{\partial y}) \tag{3.62}$$

where G_x, G_y, G_z are ***x, y, z*** components of the buoyancy force.

The general expression for the body force acting in ***x,y,z*** directions is developed when the container in such a position that none of the edges are along the gravity acceleration. Referring to Figure 3.5 we may write

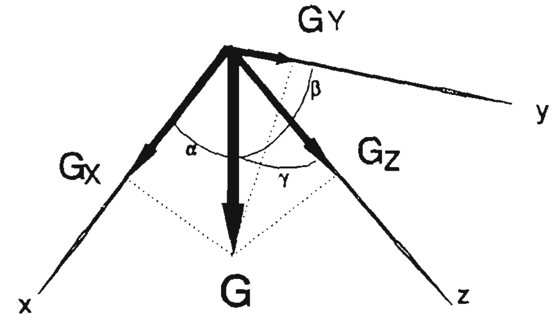


Figure 3.5 *x, y, z* components of *G*

$$G_x = G\cos\alpha \tag{3.63}$$

$$G_y = G\cos\beta \tag{3.64}$$

$$G_z = G\cos\gamma \tag{3.65}$$

$$\vec{G}_x + \vec{G}_y + \vec{G}_z = \vec{G} = g\rho_0\beta_0(T-T_0)\ (\cos\alpha\,\vec{i} + \cos\beta\,\vec{j} + \cos\vec{\gamma}) \tag{3.66}$$

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1 \tag{3.67}$$

Equations (3.60) to (3.67) may be combined to yield the following equations

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_x}{\partial z^2} + \left(\frac{1}{\kappa_z} - \frac{1}{\kappa_y}\right)\left(\frac{\partial^2\psi_z}{\partial x\partial z} + \frac{\partial^2\phi}{\partial y\partial z}\right)\right] = \frac{g\rho_0\beta_0}{\mu}\left(\cos\gamma\,\frac{\partial T}{\partial y} - \cos\beta\,\frac{\partial T}{\partial z}\right) \tag{3.68}$$

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial y^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_y}{\partial z^2} + \left(\frac{1}{\kappa_z} - \frac{1}{\kappa_x}\right)\left(\frac{\partial^2\psi_z}{\partial y\partial z} - \frac{\partial^2\phi}{\partial x\partial z}\right)\right] = \frac{g\rho_0\beta}{\mu_f}\left(\cos\alpha\,\frac{\partial T}{\partial z} - \cos\gamma\,\frac{\partial T}{\partial x}\right) \tag{3.69}$$

$$-\left[\frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial x^2} + \frac{1}{\kappa_x}\frac{\partial^2\psi_z}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial z^2} + \left(\frac{1}{\kappa_x} - \frac{1}{\kappa_y}\right)\left(\frac{\partial^2\psi_y}{\partial y\partial z} - \frac{\partial^2\phi}{\partial x\partial y}\right)\right] = \frac{g\rho_0\beta}{\mu_f}\left(\cos\beta\,\frac{\partial T}{\partial x} - \cos\alpha\,\frac{\partial T}{\partial y}\right) \tag{3.70}$$

When one of the container edges is along gravity direction, say, x axis, then $\alpha=0$ and $\beta=\gamma=90°$. So we have

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial x^2}+\frac{1}{\kappa_z}\frac{\partial^2\psi_x}{\partial y^2}+\frac{1}{\kappa_y}\frac{\partial^2\psi_x}{\partial z^2}+\left(\frac{1}{\kappa_z}-\frac{1}{\kappa_y}\right)\left(\frac{\partial^2\psi_z}{\partial x\partial z}+\frac{\partial^2\phi}{\partial y\partial z}\right)\right] = 0 \tag{3.71}$$

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial x^2}+\frac{1}{\kappa_z}\frac{\partial^2\psi_y}{\partial y^2}+\frac{1}{\kappa_x}\frac{\partial^2\psi_y}{\partial z^2}+\left(\frac{1}{\kappa_z}-\frac{1}{\kappa_x}\right)\left(\frac{\partial^2\psi_z}{\partial y\partial z}-\frac{\partial^2\phi}{\partial x\partial z}\right)\right] = \frac{g\rho_0\beta}{\mu_f}\frac{\partial T}{\partial z} \tag{3.72}$$

$$-\left[\frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial x^2}+\frac{1}{\kappa_x}\frac{\partial^2\psi_z}{\partial y^2}+\frac{1}{\kappa_y}\frac{\partial^2\psi_z}{\partial z^2}+\left(\frac{1}{\kappa_x}-\frac{1}{\kappa_y}\right)\left(\frac{\partial^2\psi_y}{\partial y\partial z}-\frac{\partial^2\phi}{\partial x\partial y}\right)\right] = -\frac{g\rho_0\beta}{\mu_f}\frac{\partial T}{\partial y} \tag{3.73}$$

3.7 Boundary Condition on ψ and ϕ

For incompressible fluid flows bounded by a solid surface, the scalar potential ϕ is specified to have the boundary condition set out below, Hirasaki and Hellums (1970)

$$\vec{n}\cdot\nabla\phi \equiv \frac{\partial\phi}{\partial n} = -\vec{n}\cdot\vec{V}_p = 0 \tag{3.74}$$

In more detail,

$$x=0 \qquad \frac{\partial\phi}{\partial x} = 0 \tag{3.75}$$

$$x=x_0 \qquad \frac{\partial\phi}{\partial x} = 0 \tag{3.76}$$

$$y=0 \qquad \frac{\partial\phi}{\partial y} = 0 \tag{3.77}$$

$$y=y_0 \qquad \frac{\partial\phi}{\partial y} = 0 \tag{3.78}$$

$$z=0 \qquad \frac{\partial\phi}{\partial z} = 0 \tag{3.79}$$

$$z = z_0 \qquad \frac{\partial \phi}{\partial z} = 0 \tag{3.80}$$

With equation 3.48 and the above conditions, it can be proved that ϕ is an arbitrary constant (see Appendix A). It means that $\frac{\partial^2 \phi}{\partial y \partial z}$, $\frac{\partial^2 \phi}{\partial x \partial z}$ *and* $\frac{\partial^2 \phi}{\partial x \partial y}$ equal 0 in equations (3.71), (3.72) and (3.73). Thus the equations are simplified as follows:

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2 \psi_x}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2 \psi_x}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2 \psi_x}{\partial z^2} + \left(\frac{1}{\kappa_z} - \frac{1}{\kappa_y}\right)\frac{\partial^2 \psi_z}{\partial x \partial z}\right] = \frac{g \rho_0 \beta}{\mu_f}\left(\cos\gamma \frac{\partial T}{\partial y} - \cos\beta \frac{\partial T}{\partial z}\right) \tag{3.81}$$

$$-\left[\frac{1}{\kappa_z}\frac{\partial^2 \psi_y}{\partial x^2} + \frac{1}{\kappa_z}\frac{\partial^2 \psi_y}{\partial y^2} + \frac{1}{\kappa_x}\frac{\partial^2 \psi_y}{\partial z^2} + \left(\frac{1}{\kappa_z} - \frac{1}{\kappa_x}\right)\frac{\partial^2 \psi_z}{\partial y \partial z}\right] = \frac{g \rho_0 \beta}{\mu_f}\left(\cos\alpha \frac{\partial T}{\partial z} - \cos\gamma \frac{\partial T}{\partial x}\right) \tag{3.82}$$

$$-\left[\frac{1}{\kappa_y}\frac{\partial^2 \psi_z}{\partial x^2} + \frac{1}{\kappa_x}\frac{\partial^2 \psi_z}{\partial y^2} + \frac{1}{\kappa_y}\frac{\partial^2 \psi_z}{\partial z^2} + \left(\frac{1}{\kappa_x} - \frac{1}{\kappa_y}\right)\frac{\partial^2 \psi_y}{\partial y \partial z}\right] = \frac{g \rho_0 \beta}{\mu_f}\left(\cos\beta \frac{\partial T}{\partial x} - \cos\alpha \frac{\partial T}{\partial y}\right) \tag{3.83}$$

Also the definition of ψ may be simplified from that given in equation (3.46) to:

$$\vec{V}_p = \nabla \times \psi \tag{3.46a}$$

Hirasaki and Hellums (1968) and Richardson and Cornish(1977) have shown the boundary conditions for ψ should satisfy following conditions:

x=0 and $\boldsymbol{x=x_0}$

$$\frac{\partial \psi_x}{\partial x} = 0 \qquad \psi_y = 0 \qquad \psi_z = 0 \tag{3.84}$$

x=0 and $\boldsymbol{x=x_0}$

$$\frac{\partial \psi_y}{\partial y} = 0 \qquad \psi_x = 0 \qquad \psi_z = 0 \tag{3.85}$$

x=0 and $\boldsymbol{x=x_0}$

$$\frac{\partial \psi_z}{\partial z} = 0 \qquad \psi_x = 0 \qquad \psi_y = 0 \tag{3.86}$$

Considering equation (3.81) and the boundary condition (3.84), in the case of $\kappa_z=\kappa_y$ the equation changes to

$$\nabla^2 \psi_x = 0 \tag{3.87}$$

and thus ψ_x is zero and the number of equation to be solved has been reduced by one. When $\kappa_z \neq \kappa_y$ or the x axis is not along the gravity direction, ψ_x is not zero and this equation must be solved numerically.

3.8 Non-Dimensional Equations

The solution of equations (3.34), (3.81), (3.82) and (3.83) and their boundary conditions is dependant on the following physical properties of the fluid and the solid matrix of the porous medium - ρ_f, ρ_s, $(C_P)_f$, C_s, k_{eff}, μ, $\bar{\mathbf{K}}$, T_{start} *and* T_c. Each must be specified before a numerical solution can be obtained and the solution must be repeated entirely if any of them are varied.

The number of parameters can be reduced, and the range of applicability of a solution can be extended if the equations and the boundary conditions are recast in terms of non-dimensional variables and parameters.

Non-dimensionalisation can be performed by using the following non-dimensional variables

$$x' = \frac{x}{x_0} \tag{3.88}$$

$$y' = \frac{y}{x_0} \tag{3.89}$$

$$z' = \frac{z}{x_0} \tag{3.90}$$

$$\nabla' = x_0 \nabla \tag{3.91}$$

$$\nabla'^2 = x_0^2 \nabla^2 \tag{3.92}$$

$$\theta = \frac{T - T_c}{T_h - T_c} \tag{3.93}$$

$$\psi' = \frac{(\rho C_p)_f}{k_{eff}} \psi \tag{3.94}$$

$$V_p' = \frac{(\rho C_p)_f x_0}{k_{eff}} V_p \tag{3.95}$$

$$t' = \frac{k_{eff}}{(\rho C_p)_f x_0^2} t \tag{3.96}$$

Re-casting equation (3.34) leads to

$$\frac{\partial\theta}{\partial t'} = -R_c \vec{V}'_p \cdot \nabla'\theta + R_c \nabla'^2\theta + \frac{R_c Q_{heat} x_0^2}{(T_h - T_c)\, k_{eff}} \tag{3.97}$$

where

$$R_c = \frac{(\rho C_p)_f}{(\rho C)_p} \tag{3.97a}$$

equations (3.81), (3.82) and (3.83) change to

$$\frac{\partial^2 \psi'_x}{\partial x'^2} + \frac{\partial^2 \psi'_x}{\partial y'^2} + \frac{Da_z}{Da_y}\frac{\partial^2 \psi'_x}{\partial z'^2} + (1 - \frac{Da_z}{Da_y})\frac{\partial^2 \psi'_z}{\partial x' \partial z'} = 0 \tag{3.98}$$

$$\frac{\partial^2 \psi'_y}{\partial x'^2} + \frac{\partial^2 \psi'_y}{\partial y'^2} + \frac{Da_z}{Da_x}\frac{\partial^2 \psi'_y}{\partial z'^2} + (1 - \frac{Da_z}{Da_x})\frac{\partial^2 \psi'_z}{\partial y' \partial z'} + Ra\, Da_z \cos\alpha \frac{\partial\theta}{\partial z'} = 0 \tag{3.99}$$

$$\frac{\partial^2 \psi'_z}{\partial x'^2} + \frac{Da_y}{Da_x}\frac{\partial^2 \psi'_z}{\partial y'^2} + \frac{\partial^2 \psi'_z}{\partial z'^2} + (\frac{Da_y}{Da_x} - 1)\frac{\partial^2 \psi'_y}{\partial y' \partial z'} - Ra\, Da_y \cos\alpha \frac{\partial\theta}{\partial y'} = 0 \tag{3.100}$$

Where Da_x, Da_y, Da_z are Darcy numbers in the ***x, y, z*** direction

$$Da_x = \frac{\kappa_x}{x_0^2} \tag{3.101}$$

$$Da_y = \frac{\kappa_y}{x_0^2} \tag{3.102}$$

$$Da_z = \frac{\kappa_z}{x_0^2} \tag{3.103}$$

and Ra is the Rayleigh number defined below

$$Ra = \frac{g\,\beta_0 x_0^3 (T_h - T_c)\,\rho_0 (\rho_0 C_p)_f}{\mu\, k_{eff}} \tag{3.104}$$

In general, α≠0, and the container is not on a horizontal plane, none of the edges is along the gravity direction and we have a general non-dimensional format for the vector potentials:

$$\frac{\partial^2 \psi'_x}{\partial x'^2} + \frac{\partial^2 \psi'_x}{\partial y'^2} + \frac{Da_z}{Da_y}\frac{\partial^2 \psi'_x}{\partial z'^2} + (1 - \frac{Da_z}{Da_y})\frac{\partial^2 \psi'_z}{\partial x' \partial z'} + Ra\,Da_z(\cos\gamma \frac{\partial \theta}{\partial y'} - \cos\beta \frac{\partial \theta}{\partial z'}) = 0 \tag{3.105}$$

$$\frac{\partial^2 \psi'_y}{\partial x'^2} + \frac{\partial^2 \psi'_y}{\partial y'^2} + \frac{Da_z}{Da_x}\frac{\partial^2 \psi'_y}{\partial z'^2} + (1 - \frac{Da_z}{Da_x})\frac{\partial^2 \psi'_z}{\partial y' \partial z'} + Ra\,Da_z(\cos\alpha \frac{\partial \theta}{\partial z'} - \cos\gamma \frac{\partial \theta}{\partial x'}) = 0 \tag{3.106}$$

$$\frac{\partial^2 \psi'_z}{\partial x'^2} + \frac{Da_y}{Da_x}\frac{\partial^2 \psi'_z}{\partial y'^2} + \frac{\partial^2 \psi'_z}{\partial z'^2} + (\frac{Da_y}{Da_x} - 1)\frac{\partial^2 \psi'_y}{\partial y' \partial z'} - Ra\,Da_y(\cos\beta \frac{\partial \theta}{\partial x'} - \cos\alpha \frac{\partial \theta}{\partial y'}) = 0 \tag{3.107}$$

3.9 Nusselt Number

As to the definition of Nusselt number, Nusselt number is equal to the dimensionless temperature gradient at the surface of the container. It provides of measure of the convection heat transfer occurring at the surface.

The local Nu numbers were defined as follows on the six surfaces of the container:

$$x' = 0: \qquad Nu = -\left.\frac{\partial\theta}{\partial x'}\right|_{x'=0} \tag{3.108}$$

$$x' = 1: \qquad Nu = \left.\frac{\partial\theta}{\partial x'}\right|_{x'=1} \tag{3.109}$$

$$y' = 0: \qquad Nu = -\left.\frac{\partial\theta}{\partial y'}\right|_{y'=0} \tag{3.110}$$

$$y' = 1: \qquad Nu = \left.\frac{\partial\theta}{\partial y'}\right|_{y'=1} \tag{3.111}$$

$$z' = 0: \qquad Nu = -\left.\frac{\partial\theta}{\partial z'}\right|_{z'=0} \tag{3.112}$$

$$z' = 1: \qquad Nu = \left.\frac{\partial\theta}{\partial z'}\right|_{z'=1} \tag{3.113}$$

The average Nusselt numbers on side-walls and top and bottom were defined as follows: on top $x' = 0$ and bottom $x' = x_0'$

$$Nu_{average} = \frac{1}{y'_0 z'_0} \int_0^{y'_0} \int_0^{z'_0} Nu(y', z')\, dy' d' z' \tag{3.114}$$

on side walls $y' = 0$ and $y' = y_0'$

$$Nu_{average} = \frac{1}{x'_0 z'_0} \int_0^{x'_0} \int_0^{z'_0} Nu(x', z')\, dx' d' z' \tag{3.115}$$

on side walls $z' = 0$ and $z' = z_0'$

$$Nu_{average} = \frac{1}{x'_0 y'_0} \int_0^{x'_0} \int_0^{y'_0} Nu(x', y')\, dx' d' y' \tag{3.116}$$

CHAPTER 4

NUMERICAL METHODS

4.1 Introduction

Equations (3.97), (3.105), (3.106) and (3.107) must be solved numerically. They are coupled, second-order partial differential equations, and to date it has not proved possible to obtain their exact mathematical solutions. In this chapter the numerical methods used to solve them are described.

If ψ_x, ψ_y and ψ_z are known then we can obtain velocity field via equation (3.46a). By using the known velocity field and solving equation (3.97), temperature field can be found. This is one complete iterative step in the main loop or called outer iteration. We can also revise the solution procedure by solving temperature field first and them update velocity field, but the result is the same. When one iterative step is completed, the new values of vector potentials and temperature can then be used as the starting state for the next iteration step. The iteration loop continue until the desire iteration number is reached or the solution is convergent.

Inside the main loop, two important procedures are defined as solving the vector potential field to obtain velocity field and solving temperature field. The former is more complicated than the latter.

The component equations of vector potential, equations (3.105), (3.106) and (3.107) are elliptic and coupled. The iterative loop for solving the vector potential is called the inner iterative loop. Inside the inner loop, firstly ψ_x and ψ_y are advanced by using the old ψ_z value, then the inner iteration advanced for solving ψ_z. Updated values of ψ_x, ψ_y and ψ_z can be used in the next iteration. The iteration will continue until the solutions of ψ_x, ψ_y and ψ_z cease to change significantly. We term these results the false steady state of vector potential within one outer iteration step. Because actually they are the real transient vector potential field which represents the real transient velocity field, however from the mathematical viewpoint we can imagine that we are solving a set of coupled elliptic partial differential equations which have an initial state and a final steady state. The initial state is known from the updated value of the outer iteration loop. Then the inner iteration is carried out until the final state is reached. This is the so-called steady state.

A third-level iteration loop is used for solving a single component of vector potential. We applied the false transient technique, after Mallinson and de Vahl Davis (1973), to improve the rate of convergence. The above multi-iteration scheme was employed to determine the solution of equations (3.105) to (3.107). The third-level iteration was used for solving the single component equation of vector potential. The inner iteration was used for updating the new vector potential field in which its three component equations are coupled, and the outer loop was used to obtain transient velocity field and temperature fields. A non-uniform central differential approximation was applied to obtain the differential equations, which were solved by the Samarskii-Androyev alterative direction implicit scheme.

4.2 The Approximation of Derivatives by Finite Differences

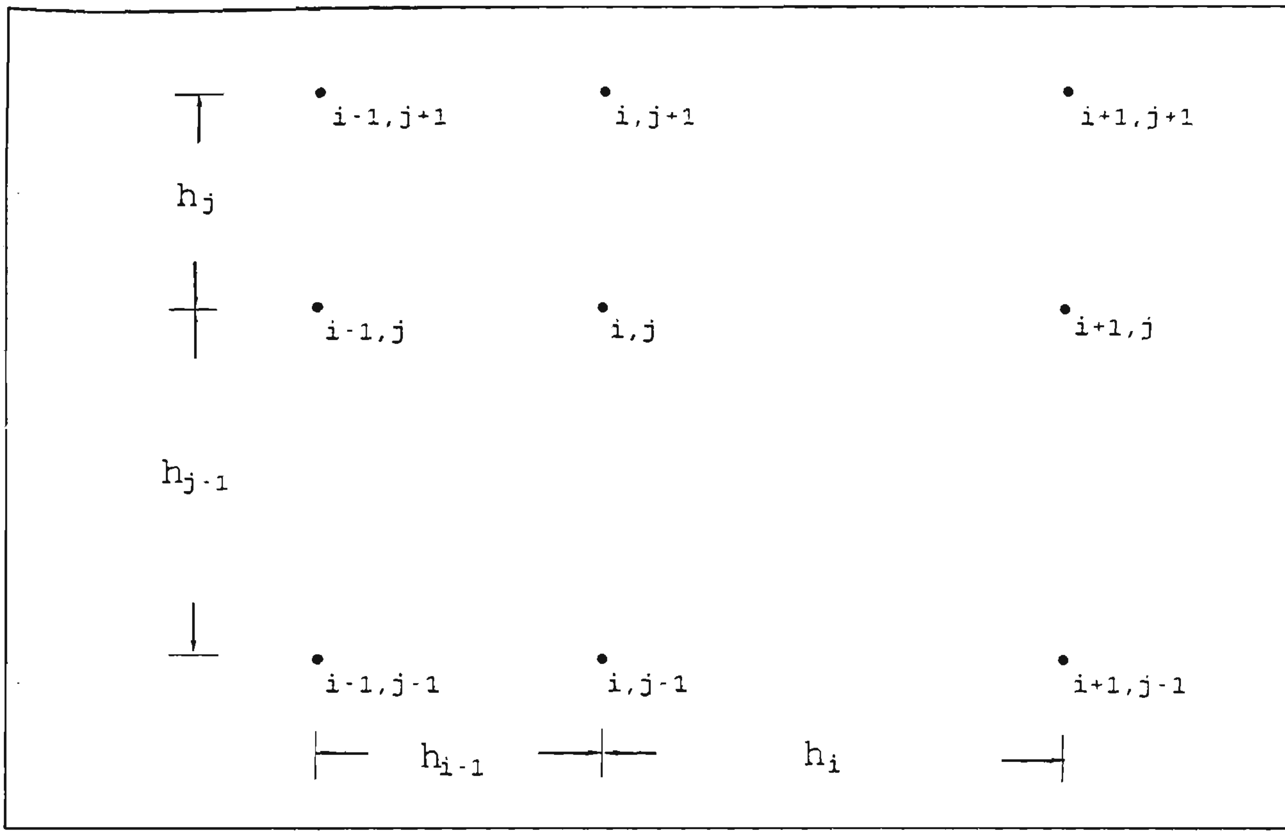


Figure 4.1 Arrangement of grid points

A non-uniform mesh was used and is shown in Figure 4.1, h_i represents the distance between ***(i-1)th*** point and ***ith*** point along the x axis, as were h_j, h_k respectively. x_i was the ***ith*** point position in x direction, so we have

$$h_i = x_i - x_{i-1} \tag{4.1}$$

$$h_j = y_j - y_{j-1} \tag{4.2}$$

$$h_k = z_k - z_{k-1} \tag{4.3}$$

A notation to indicate the value of θ or ψ at the ***pth*** time step and the position in the system was adopted as follows,

$$\theta = \theta^{n}_{i,j,k} \tag{4.4}$$

where ***p, i, j,*** and ***k*** were default values and are not shown, hence

$$\theta^{p+1}_{i-1,j,k+1} \quad or \quad \theta^{p+1}_{i-1,k+1} \tag{4.5}$$

denotes the dimensionless temperature at the position in space x_i-h_i , y_j , z_k+h_i and at time t+Δt. The indices i, j, and k as used here will always denote position in space and the superscript p will denote time.

The space point (x_i , y_j , z_k) , also called the grid-point (i, j, k) was surrounded by neighbouring grid points shown in Figure 4.1. Assuming that the function θ, ψ possesses a sufficient number of partial derivatives. The value of them, for example, θ, at the two points, say, (x_i, y_j, z_k) and (x_{i+1}, y_j, z_k) are related by Taylor's expansion:

$$\theta_{i+1,j,k} = \theta_{i,j,k} + h_{i+1}\left.\frac{\partial\theta}{\partial x}\right|_{i,j,k} + \frac{1}{2!}h^2_{i+1}\left.\frac{\partial^2\theta}{\partial x^2}\right|_{i,j,k} + O(h^2_{i+1}) \tag{4.6}$$

Dropping the remainder term $O(h_i^2)$ and expanding in Taylor's series for θ_{i-1} and θ_{i+1} about the central value $\theta_{i,j,k}$, we obtain

$$\theta_{i-1} = \theta - h_i\left.\frac{\partial\theta}{\partial x}\right|_{i,j,k} + \frac{1}{2}h_i^2\left.\frac{\partial^2\theta}{\partial x^2}\right|_{i,j,k} \tag{4.7}$$

$$\theta_{i+1} = \theta + h_{i+1}\left.\frac{\partial\theta}{\partial x}\right|_{i,j,k} + \frac{1}{2}h^2_{i+1}\left.\frac{\partial^2\theta}{\partial x^2}\right|_{i,j,k} \tag{4.8}$$

From these two relations it is easy to show that

$$\left.\frac{\partial\theta}{\partial x}\right|_{i,j,k} = -\frac{h^2_{i+1}}{h_i h_{i+1}(h_i+h_{i+1})}\theta_{i-1} + \frac{h^2_{i+1}-h_i^2}{h_i h_{i+1}(h_i+h_{i+1})}\theta + \frac{h_i^2}{h_i h_{i+1}(h_i+h_{i+1})}\theta_{i+1} \tag{4.9}$$

setting

$$CX1(i) = -\frac{h_{i+1}^2}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.10a}$$

$$CX2(i) = \frac{h_{i+1}^2 - h_i^2}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.10b}$$

$$CX3(i) = \frac{h_i^2}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.10c}$$

we have

$$\left.\frac{\partial\theta}{\partial x}\right|_{i,j,k} = CX1(i)\theta_{i-1} + CX2(i)\theta + CX3(i)\theta_{i+1} \tag{4.10}$$

Similarly, we find

$$\left.\frac{\partial^2\theta}{\partial x^2}\right|_{i,j,k} = CX4(i)\theta_{i-1} + CX5(i)\theta + CX6(i)\theta_{i+1} \tag{4.11}$$

where:

$$CX4(i) = \frac{2h_{i+1}}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.11a}$$

$$CX5(i) = -\frac{2(h_i + h_{i+1})}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.11b}$$

$$CX6(i) = \frac{2h_i}{h_i h_{i+1} (h_i + h_{i+1})} \tag{4.11c}$$

Applying the same procedure to the y and z directions we obtain

$$\left.\frac{\partial\theta}{\partial y}\right|_{i,j,k} = CY1(j)\theta_{j-1} + CY2(j)\theta + CY3(j)\theta_{j+1} \tag{4.12}$$

$$CY1(j) = -\frac{h_{j+1}^2}{h_j h_{j+1} (h_j + h_{j+1})} \tag{4.12a}$$

$$CY2(j) = \frac{h_{j+1}^2 - h_j^2}{h_j h_{j+1}(h_j + h_{j+1})} \tag{4.12b}$$

$$CY3(j) = \frac{h_j^2}{h_j h_{j+1}(h_j + h_{j+1})} \tag{4.12c}$$

$$\left.\frac{\partial^2\theta}{\partial y^2}\right|_{i,j,k} = CY4(j)\theta_{j-1} + CY5(j)\theta + CY6(j)\theta_{j+1} \tag{4.13}$$

$$CY4(j) = \frac{2h_{j+1}}{h_j h_{j+1}(h_j + h_{j+1})} \tag{4.13a}$$

$$CY5(j) = -\frac{2(h_j + h_{j+1})}{h_j h_{j+1}(h_j + h_{j+1})} \tag{4.13b}$$

$$CY6(j) = \frac{2h_j}{h_j h_{j+1}(h_j + h_{j+1})} \tag{4.13c}$$

$$\left.\frac{\partial\theta}{\partial z}\right|_{i,j,k} = CZ1(k)\theta_{k-1} + CZ2(k)\theta + CZ3(k)\theta_{k+1} \tag{4.14}$$

$$CZ1(k) = -\frac{h_{k+1}^2}{h_k h_{k+1}(h_k + h_{k+1})} \tag{4.14a}$$

$$CZ2(k) = \frac{h_{k+1}^2 - h_k^2}{h_k h_{k+1}(h_k + h_{k+1})} \tag{4.14b}$$

$$CZ3(k) = \frac{h_k^2}{h_k h_{k+1}(h_k + h_{k+1})} \tag{4.14c}$$

$$\left.\frac{\partial^2\theta}{\partial z^2}\right|_{i,j,k} = CZ4(k)\theta_{k-1} + CZ5(k)\theta + CZ6(k)\theta_{k+1} \tag{4.15}$$

$$CZ4(k) = \frac{2h_{k+1}}{h_k h_{k+1}(h_k + h_{k+1})} \tag{4.15a}$$

$$CZ5(k) = -\frac{2(h_k + h_{k+1})}{h_k h_{k+1}(h_k + h_{k+1})} \tag{4.15b}$$

$$CZ6(k) = \frac{2\,h_k}{h_k\,h_{k+1}\,(h_k + h_{k+1})} \tag{4.15c}$$

There are second-order partial differences in the equations (3.98), (3.99) and (3.100). To derive their finite difference approximations, we applied Taylor's expansion on the grid points as shown in Figure 4.1, we obtain

$$\psi_{i+1,\,j+1} = \psi_{i+1,\,j} + h_j \frac{\partial\psi}{\partial y}\bigg|_{i+1,\,j} + \frac{h_j^2}{2}\frac{\partial^2\psi}{\partial y^2}\bigg|_{i+1,\,j} \tag{4.16}$$

$$\psi_{i+1,\,j-1} = \psi_{i+1,\,j} - h_{j-1} \frac{\partial\psi}{\partial y}\bigg|_{i+1,\,j} + \frac{h_{j-1}^2}{2}\frac{\partial^2\psi}{\partial y^2}\bigg|_{i+1,\,j} \tag{4.17}$$

$$\psi_{i-1,\,j+1} = \psi_{i-1,\,j} + h_j \frac{\partial\psi}{\partial y}\bigg|_{i-1,\,j} + \frac{h_j^2}{2}\frac{\partial^2\psi}{\partial y^2}\bigg|_{i-1,\,j} \tag{4.18}$$

$$\psi_{i-1,\,j-1} = \psi_{i-1,\,j} - h_{j-1} \frac{\partial\psi}{\partial y}\bigg|_{i-1,\,j} + \frac{h_{j-1}^2}{2}\frac{\partial^2\psi}{\partial y^2}\bigg|_{i-1,\,j} \tag{4.19}$$

$Eq.\,(4.16) * h_{j-1}^2 - Eq.\,(4.17) * h_j^2$, we have

$$\psi_{i+1,j+1} h_{j-1}^2 - \psi_{i+1,j-1} h_j^2 = (h_{j-1}^2 - h_j^2)\,\psi_{i+1,j} + (h_j\,h_{j-1}^2 + h_j^2\,h_{j-1})\frac{\partial\psi}{\partial y}\bigg|_{i+1,j} \tag{4.20}$$

$Eq.\,(4.18) * h_{j-1}^2 - Eq.\,(4.19) * h_j^2$, we have

$$\psi_{i-1,j+1}\, h_{j-1}^2 - \psi_{i-1,j-1}\, h_j^2 = (h_{j-1}^2 - h_j^2)\,\psi_{i-1,j} + (h_j\,h_{j-1}^2 + h_j^2\,h_{j-1})\frac{\partial\psi}{\partial y}\bigg|_{i-1,j} \tag{4.21}$$

Using Taylor's expansion, we obtain

$$\frac{\partial\psi}{\partial y}\bigg|_{i+1,j} = \frac{\partial\psi}{\partial y} + h_i \frac{\partial^2\psi}{\partial x\,\partial y} + \frac{h_i^2}{2}\frac{\partial^3\psi}{\partial y\,\partial x^2} \tag{4.22}$$

$$\frac{\partial\psi}{\partial y}\bigg|_{i-1,j} = \frac{\partial\psi}{\partial y} - h_{i-1} \frac{\partial^2\psi}{\partial x\,\partial y} + \frac{h_{i-1}^2}{2}\frac{\partial^3\psi}{\partial y\,\partial x^2} \tag{4.23}$$

By using equations (4.20), (4.21), (4.22) and (4.23), and eliminating $\dfrac{\partial^3 \psi}{\partial y\, \partial x^2}$, we get

$$h_{i-1}^2(\psi_{i+1,j+1} h_{j-1}^2 - \psi_{i+1,j-1} h_j^2) - h_i^2(\psi_{i-1,j+1} h_{j-1}^2 - \psi_{i-1,j-1} h_j^2)$$
$$= (h_{j-1}^2 - h_j^2)\, h_{i-1}^2\, \psi_{i+1,j} - (h_{j-1}^2 - h_j^2)\, h_i^2\, \psi_{i-1,j}$$
$$+ (h_{i-1}^2 - h_i^2)(h_j h_{j-1}^2 + h_j^2 h_{j-1}) \frac{\partial \psi}{\partial y} \tag{4.24}$$
$$+ (h_j h_{j-1}^2 + h_j h_{j-1}^2)(h_{i-1}^2 h_i + h_i^2 h_{i-1}) \frac{\partial^2 \psi}{\partial x\, \partial y}$$

By using equation (4.12) to find the differential expression for $\dfrac{\partial \psi}{\partial y}$, finally we obtain the differential expression as:

$$\frac{\partial^2 \psi}{\partial x\, \partial y} = CXM1YM1(i,j)\, \psi_{i-1,j-1} + CXYM1(i,j)\, \psi_{i,j-1} + CXP1YM1(i,j)\, \psi_{i+1,j-1}$$
$$+ CXM1Y(i,j)\quad \psi_{i-1,j} + CXY\quad \psi_{i,j} + CXP1Y(i,j)\, \psi_{i+1,j}$$
$$+ CXM1YP1(i,j)\, \psi_{i-1,j+1} + CXYP1(i,j)\, \psi_{i,j+1} + CXP1YP1(i,j)\, \psi_{i+1,j+1} \tag{4.25}$$

where

$$CXM1YM1(i,j) = \frac{h_i^2\, h_j^2}{DENOM1} \tag{4.25a}$$

$$CXYM1(i,j) = \frac{DENOM2 * h_j^2}{DENOM1 * DENOM3} \tag{4.25b}$$

$$CXP1YM1(i,j) = -\frac{h_{i-1}^2\, h_j^2}{DENOM1} \tag{4.25c}$$

$$CXM1Y(i,j) = \frac{h_i^2(h_{j-1}^2 - h_j^2)}{DENOM1} \tag{4.25d}$$

$$CXY(i,j) = -\frac{DENOM2 * (h_j^2 - h_{j-1}^2)}{DENOM1 * DENOM3} \tag{4.25e}$$

$$CXP1Y(i,j) = -\frac{h_{i-1}^2 (h_{j-1}^2 - h_j^2)}{DENOM1} \tag{4.25f}$$

$$CXM1YP1(i,j) = -\frac{h_i^2 h_{j-1}^2}{DENOM1} \tag{4.25g}$$

$$CXYP1(i,j) = -\frac{DENOM2 * h_{j-1}^2}{DENOM1 * DENOM3} \tag{4.25h}$$

$$CXP1YP1(i,j) = \frac{h_{i-1}^2 h_{j-1}^2}{DENOM1} \tag{4.25i}$$

$$DENOM1 = (h_{j-1}^2 h_j + h_{j-1} h_j^2)(h_{i-1}^2 h_i + h_{i-1} h_i^2) \tag{4.25j}$$

$$DENOM2 = (h_{j-1}^2 h_j + h_{j-1} h_j^2)(h_{i-1}^2 - h_i^2) \tag{4.25k}$$

$$DENOM3 = h_{j-1} h_j (h_{j-1} + h_j) \tag{4.25m}$$

Similarly we can obtain differential expressions for $\frac{\partial^2 \psi}{\partial x \partial z}, \frac{\partial^2 \psi}{\partial y \partial z}$.

4.3 Finite Differences Approximations on Boundary

We can not use the above differences equations to determine the values on the boundary because they are derived for a central point which relies on values on both sides of the point. The values of the point outside the boundary are unknown so the central point can not be determined by using this method.

4.3.1 Finite Difference Formulas for the First-Order Derivatives on Boundaries for Energy Equation

There were only first-order derivatives in the boundary conditions, refer to the boundary conditions for energy equation.

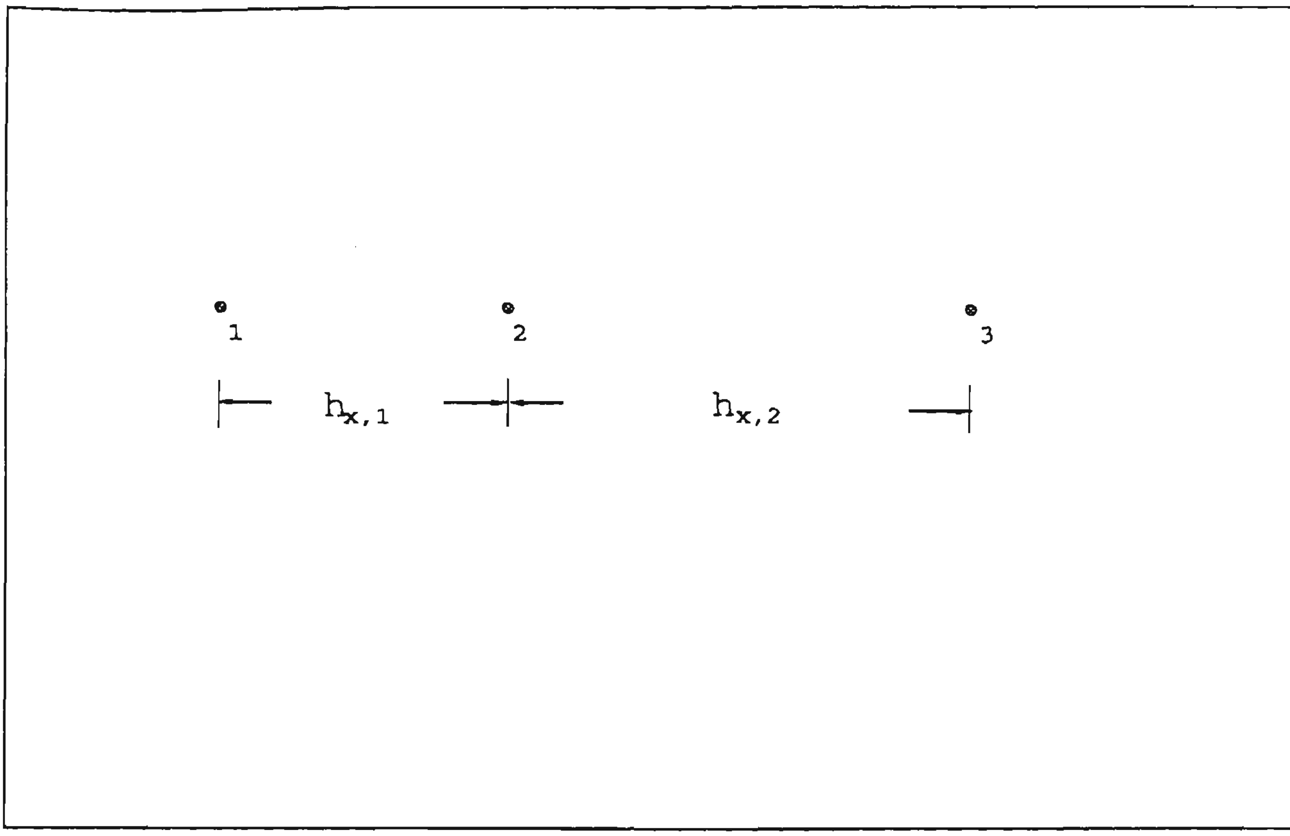


Figure 4.2 Arrangement of boundary grid points

For the low limit, ie $x=0$, as shown in Figure 4.2, we have

$$\theta_{2,j,k} = \theta_{1,j,k} + h_{x,1}\left.\frac{\partial\theta}{\partial x}\right|_{1,j,k} + \frac{h_{x,1}^2}{2}\left.\frac{\partial^2\theta}{\partial x^2}\right|_{1,j,k} \tag{4.26}$$

$$\theta_{3,j,k} = \theta_{1,j,k} + (h_{x,1}+h_{x,2})\left.\frac{\partial\theta}{\partial x}\right|_{1,j,k} + \frac{(h_{x,1}+h_{x,2})^2}{2}\left.\frac{\partial^2\theta}{\partial x^2}\right|_{1,j,k} \tag{4.27}$$

by eliminating the second-order derivative, we obtain

$$\left.\frac{\partial\theta}{\partial x}\right|_{1,j,k} = CX1TL\,\theta_{1,j,k} + CX2TL\,\theta_{2,j,k} + CX3TL\,\theta_{3,j,k} \tag{4.28}$$

where

$$CX1TL = -\frac{(h_{x,1}+h_{x,2})^2 - h_{x,1}^2}{h_{x,1}(h_{x,1}+h_{x,2})^2 - h_{x,1}^2(h_{x,1}+h_{x,2})} \tag{4.28a}$$

$$CX2TL = \frac{(h_{x,1}+h_{x,2})^2}{h_{x,1}(h_{x,1}+h_{x,2})^2 - h_{x,1}^2(h_{x,1}+h_{x,2})} \tag{4.28b}$$

$$CX3TL = -\frac{h_{x,1}^2}{h_{x,1}(h_{x,1}+h_{x,2})^2 - h_{x,1}^2(h_{x,1}+h_{x,2})} \tag{4.28c}$$

Similarly we can obtain derivatives for the high limit in the x direction and those for y, z axis as follows:

$$\left.\frac{\partial\theta}{\partial x}\right|_{NX,j,k} = CX1TH\,\theta_{NX,j,k} + CX2TH\,\theta_{NX-1,j,k} + CX3TH\,\theta_{NX-2,j,k} \tag{4.29}$$

where

$$CX1TH = \frac{(h_{x,NX-1}+h_{x,NX-2})^2 - h_{x,NX-1}^2}{h_{x,NX-1}(h_{x,NX-1}+h_{x,NX-2})^2 - h_{x,NX-1}^2(h_{x,NX-1}+h_{x,NX-2})} \tag{4.29a}$$

$$CX2TH = -\frac{(h_{x,NX-1}+h_{x,NX-2})^2}{h_{x,NX-1}(h_{x,NX-1}+h_{x,NX-2})^2 - h_{x,NX-1}^2(h_{x,NX-1}+h_{x,NX-2})} \tag{4.29b}$$

$$CX3TH = \frac{h_{x,NX-1}^2}{h_{x,NX-1}(h_{x,NX-1}+h_{x,NX-2})^2 - h_{x,NX-1}^2(h_{x,NX-1}+h_{x,NX-2})} \tag{4.29c}$$

$$\left.\frac{\partial\theta}{\partial y}\right|_{i,1,k} = CY1TL\,\theta_{i,1,k} + CY2TL\,\theta_{i,2,k} + CY3TL\,\theta_{i,3,k} \tag{4.30}$$

where

$$CY1TL = -\frac{(h_{y,1}+h_{y,2})^2 - h_{y,1}^2}{h_{y,1}(h_{y,1}+h_{y,2})^2 - h_{y,1}^2(h_{y,1}+h_{y,2})} \tag{4.30a}$$

$$CY2TL = \frac{(h_{y,1}+h_{y,2})^2}{h_{y,1}(h_{y,1}+h_{y,2})^2 - h_{y,1}^2(h_{y,1}+h_{y,2})} \tag{4.30b}$$

$$CY3TL = -\frac{h_{y,1}^2}{h_{y,1}(h_{y,1}+h_{y,2})^2 - h_{y,1}^2(h_{y,1}+h_{y,2})} \tag{4.30c}$$

$$\left.\frac{\partial\theta}{\partial y}\right|_{i,NY,k} = CY1TH\,\theta_{i,NY,k} + CY2TH\,\theta_{i,NY-1,k} + CY3TH\,\theta_{i,NY-2,k} \tag{4.31}$$

where

$$CY1TH = \frac{(h_{y,NY-1}+h_{y,NY-2})^2 - h_{y,NY-1}^2}{h_{y,NY-1}(h_{y,NY-1}+h_{y,NY-2})^2 - h_{y,NY-1}^2(h_{y,NY-1}+h_{y,NY-2})} \tag{4.31a}$$

$$CY2TH = -\frac{(h_{y,NY-1}+h_{y,NY-2})^2}{h_{y,NY-1}(h_{y,NY-1}+h_{y,NY-2})^2 - h_{y,NY-1}^2(h_{y,NY-1}+h_{y,NY-2})} \tag{4.31b}$$

$$CY3TH = \frac{h_{y,NY-1}^2}{h_{y,NY-1}(h_{y,NY-1}+h_{y,NY-2})^2 - h_{y,NY-1}^2(h_{y,NY-1}+h_{y,NY-2})} \tag{4.31c}$$

$$\left.\frac{\partial\theta}{\partial z}\right|_{i,j,1} = CZ1TL\,\theta_{i,j,1} + CZ2TL\,\theta_{i,j,2} + CZ3TL\,\theta_{i,j,3} \tag{4.32}$$

where

$$CZ1TL = -\frac{(h_{z,1}+h_{z,2})^2 - h_{z,1}^2}{h_{z,1}(h_{z,1}+h_{z,2})^2 - h_{z,1}^2(h_{z,1}+h_{z,2})} \tag{4.32a}$$

$$CZ2TL = \frac{(h_{z,1}+h_{z,2})^2}{h_{z,1}(h_{z,1}+h_{z,2})^2 - h_{z,1}^2(h_{z,1}+h_{z,2})} \tag{4.32b}$$

$$CZ3TL = -\frac{h_{z,1}^2}{h_{z,1}(h_{z,1}+h_{z,2})^2 - h_{z,1}^2(h_{z,1}+h_{z,2})} \tag{4.32c}$$

$$\left.\frac{\partial\theta}{\partial z}\right|_{i,j,NZ} = CZ1TH\,\theta_{i,j,NZ} + CZ2TH\,\theta_{i,j,NZ-1} + CZ3TH\,\theta_{i,j,NZ-2} \tag{4.33}$$

where

$$CZ1TH = \frac{(h_{z,NZ-1}+h_{z,NZ-2})^2 - h_{z,NZ-1}^2}{h_{z,NZ-1}(h_{z,NZ-1}+h_{z,NZ-2})^2 - h_{z,NZ-1}^2(h_{z,NZ-1}+h_{z,NZ-2})} \tag{4.33a}$$

$$CZ2TH = -\frac{(h_{z,NZ-1}+h_{z,NZ-2})^2}{h_{z,NZ-1}(h_{z,NZ-1}+h_{z,NZ-2})^2 - h_{z,NZ-1}^2(h_{z,NZ-1}+h_{z,NZ-2})} \quad (4.33b)$$

$$CZ3TH = \frac{h_{z,NZ-1}^2}{h_{z,NZ-1}(h_{z,NZ-1}+h_{z,NZ-2})^2 - h_{z,NZ-1}^2(h_{z,NZ-1}+h_{z,NZ-2})} \quad (4.33c)$$

4.3.2 Finite Difference Formulas for the Second-Order Derivatives on Boundaries for ψ

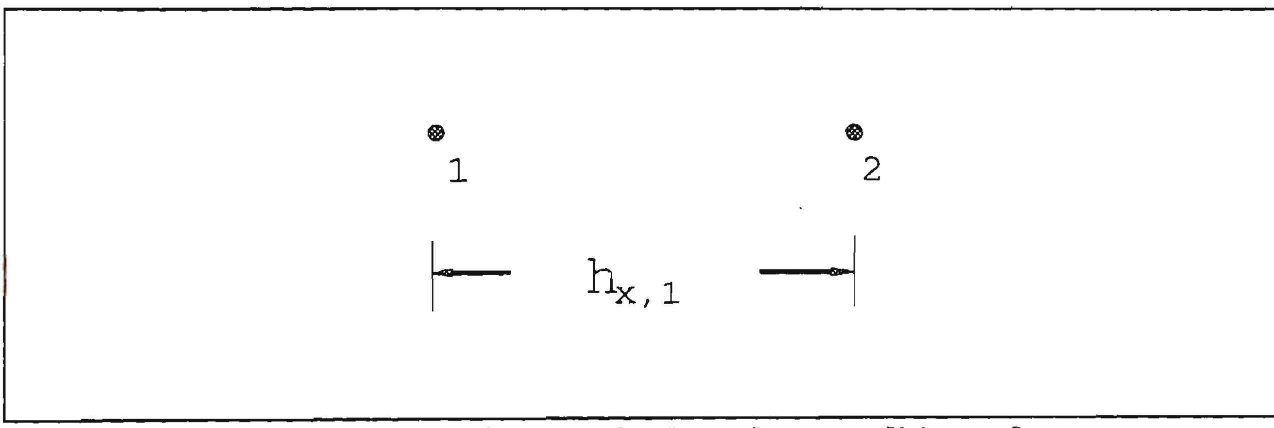


Figure 4.3 Arrangement of grid points for boundary conditions of ψ

The finite difference formulas on boundaries for ψ are quite simple because at the boundary the first derivatives vanish. Using Taylor's series, we have the following relation for the lower limit boundary as illustrated in Figure 4.3.

$$\psi_{x,2} = \psi_{x,1} + \frac{h_{x,1}^2}{2}\frac{\partial^2\psi}{\partial x^2}\bigg|_{1,j,k} \quad (4.34)$$

We define

$$\frac{\partial^2\psi}{\partial x^2}\bigg|_{1,j,k} = CX5PL\,\psi_{x,1} + CX6PL\,\psi_{x,2} \quad (4.35)$$

where

$$CX5PL = -\frac{2}{h_{x,1}^2} \quad (4.35a)$$

$$CX6PL = \frac{2}{h_{x,1}^2} \quad (4.35b)$$

Similarly we can find the approximations for the higher limit side for the x direction and those for y and z axes.

$$\left.\frac{\partial^2\psi}{\partial x^2}\right|_{NX,j,k} = CX4PH\,\psi_{x,NX-1} + CX5PH\,\psi_{x,NX-2} \tag{4.36}$$

where

$$CX4PH = \frac{2}{h^2_{x,NX-1}} \tag{4.36a}$$

$$CX5PH = -\frac{2}{h^2_{x,NX-1}} \tag{4.36b}$$

$$\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,1,k} = CY5PL\,\psi_{y,1} + CY6PL\,\psi_{y,2} \tag{4.37}$$

where

$$CY5PL = -\frac{2}{h^2_{y,1}} \tag{4.37a}$$

$$CY6PL = \frac{2}{h^2_{y,1}} \tag{4.37b}$$

$$\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,NY,k} = CY4PH\,\psi_{y,NY-1} + CY5PH\,\psi_{y,NY-2} \tag{4.38}$$

where

$$CY4PH = \frac{2}{h^2_{y,NY-1}} \tag{4.38a}$$

$$CY5PH = -\frac{2}{h^2_{y,NY-1}} \tag{4.38b}$$

$$\left.\frac{\partial^2\psi}{\partial z^2}\right|_{i,j,1} = CZ5PL\,\psi_{z,1} + CZ6PL\,\psi_{z,2} \tag{4.39}$$

where

$$CZ5PL = -\frac{2}{h^2_{z,1}} \tag{4.39a}$$

$$CZ6PL = \frac{2}{h^2_{z,1}} \tag{4.39b}$$

$$\left.\frac{\partial^2\psi}{\partial z^2}\right|_{i,j,NZ} = CZ4PH\,\psi_{z,NZ-1} + CZ5PH\,\psi_{z,NZ-2} \tag{4.40}$$

where

$$CZ4PH = \frac{2}{h^2_{z,NZ-1}} \tag{4.40a}$$

$$CZ5PH = -\frac{2}{h^2_{z,NZ-1}} \tag{4.40b}$$

4.4 The False Transient Equations of Vector Potential

The method of false transient makes two simple changes to each vector potential component equation: a fictitious transient term is inserted in the left side of the equations and the time derivatives are given modified coefficients. Equations (3.105), (3.106) and (3.107) become:

$$\frac{1}{\alpha_\psi}\frac{\partial\psi'_x}{\partial t'} = \frac{\partial^2\psi'_x}{\partial x'^2} + \frac{\partial^2\psi'_x}{\partial y'^2} + \frac{Da_z}{Da_y}\frac{\partial^2\psi'_x}{\partial z'^2} + (1-\frac{Da_z}{Da_y})\frac{\partial^2\psi'_z}{\partial x'\partial z'} + RaDa_z(\cos\gamma\frac{\partial\theta}{\partial y'} - \cos bata\frac{\partial\theta}{\partial z'}) \tag{4.41}$$

$$\frac{1}{\alpha_\psi}\frac{\partial\psi'_y}{\partial t'} = \frac{\partial^2\psi'_y}{\partial x'^2} + \frac{\partial^2\psi'_y}{\partial y'^2} + \frac{Da_z}{Da_x}\frac{\partial^2\psi'_y}{\partial z'^2} + (1-\frac{Da_z}{Da_x})\frac{\partial^2\psi'_z}{\partial y'\partial z'} + RaDa_z(\cos\alpha\frac{\partial\theta}{\partial z'} - \cos\gamma\frac{\partial\theta}{\partial x'}) \tag{4.42}$$

$$\frac{1}{\alpha_\psi}\frac{\partial\psi'_z}{\partial t'} = \frac{\partial^2\psi'_z}{\partial x'^2} + \frac{Da_y}{Da_x}\frac{\partial^2\psi'_z}{\partial y'^2} + \frac{\partial^2\psi'_z}{\partial z'^2} + (\frac{Da_y}{Da_x}-1)\frac{\partial^2\psi'_y}{\partial y'\partial z'} + RaDa_y(\cos\beta\frac{\partial\theta}{\partial x'} - \cos\alpha\frac{\partial\theta}{\partial y'}) \tag{4.43}$$

If such a " steady state" solution exists, clearly the left hand side of equations (4.41), (4.42) and (4.43) will be zero and the results for them are identified as being the same as solved from equations (3.105), (3.106), (3.107).

The introduction of false transient coefficients α_ψ enhance the rate of convergence by setting different value of α_ψ, therefore the steady state ψ can be obtained more efficiently.

4.5 The Iteration Scheme for a Single Equation - Samarskii-Andreyev Implicit Alternating Direction Method.

Equations (3.93), (4.41), (4.42) and (4.43) have four independent variables x,y,z and time t. In this study an implicit alternating-direction methods - the Samarskii- Andreyev (1963) Scheme was employed. It is also called the locally-one dimensional method because each equation is split into three one-dimensional equations to be solved at each stage of the solution.

4.5.1 Solution Procedure of Vector Potential

4.5.1.1 Solution of X Component ψ_x

$$\frac{1}{\alpha_\psi}\frac{\partial \psi_x'}{\partial t'} = \frac{\partial^2 \psi'_x}{\partial x'^2} + \frac{\partial^2 \psi'_x}{\partial y'^2} + \frac{Da_z}{Da_y}\frac{\partial^2 \psi'_x}{\partial z'^2} + (1-\frac{Da_z}{Da_y})\frac{\partial^2 \psi'_z}{\partial x' \partial z'} + RaDa_z(\cos\gamma \frac{\partial \theta}{\partial y'} - \cos\beta \frac{\partial \theta}{\partial z'}) \tag{4.41}$$

We define intermediate variable ω as

$$\omega = \frac{1}{\alpha_\psi}\frac{\partial \psi_x'}{\partial t'} \tag{4.44}$$

and

$$\psi_x' = \alpha_\psi \delta \Delta t' \omega + \psi_x'^P \tag{4.45}$$

implicit in x direction:

$$\omega^* = \frac{\partial^2}{\partial x'^2}(\alpha_\psi \delta \Delta t' \omega^* + \psi_x'^P) + \frac{\partial^2 \psi_x'}{\partial y'} + \frac{Da_z}{Da_y}\frac{\partial^2 \psi_x'}{\partial z'}$$
$$+ (1-\frac{Da_z}{Da_y})\frac{\partial^2 \psi_z'}{\partial x' \partial z'} + Ra\, Da_z(\cos\gamma \frac{\partial \theta}{\partial y'} - \cos\beta \frac{\partial \theta}{\partial z'}) \tag{4.46}$$

Using differential approximations, the above expression can be expanded to:

$$
\begin{aligned}
\omega^{*} - \alpha_{\psi}\delta\Delta t'(CX4(i)\,\omega^{*}_{i-1} &+ CX5(i)\,\omega^{*} + CX6(i)\,\omega^{*}_{i+1}) \\
&= CX4(i)\,\psi'_{x,\,i-1} + CX5(i)\,\psi'_{x} + CX6(i)\,\psi'_{x,\,i+1} \\
&+ CY4(j)\,\psi'_{x,\,j-1} + CY5(j)\psi'_{x} + CY6(j)\psi'_{x,\,j+1} \\
&+ \frac{Da_z}{Da_y}\,(\,CZ4(k)\,\psi'_{x,\,k-1} + CZ5(k)\,\psi'_{x} + CZ6(k)\,\psi'_{x,\,k+1}\,) \\
&+ (\,1 - \frac{Da_z}{Da_y}\,)\,(\,CXM1ZM1(i,\,k)\,\psi'_{z,\,i-1,\,k-1} + CXZM1(i,\,k)\,\psi'_{z,\,i,\,k-1} \\
&+ CXP1ZM1\;(i,\,k)\,\psi'_{z,\,i+1,\,k-1} + CXM1Z(i,\,k)\,\psi'_{z,\,i-1,\,k} + CXZ(i,\,k)\,\psi'_{z,\,i,\,k} \\
&+ CXP1Z(i,\,k)\,\psi'_{z,\,i+1,\,k} + CXM1ZP1(i,\,k)\,\psi'_{z,i-1,k+1} \\
&+ CXZP1(i,\,k)\,\psi'_{z,i,k+1} + CXP1ZP1(i,k)\,\psi'_{z,i+1,k+1} \\
&+ RaDa_z[\cos\gamma\,(CY1(j)\theta_{j-1} + CY2(j)\theta + CY3(j)\theta_{j+1}) \\
&- \cos\beta\,(CZ1(k)\theta_{k-1} + CZ2(k)\theta + CZ3(k)\theta_{k+1})]
\end{aligned}
\tag{4.47}
$$

Now we define:

$$A(I) = -\alpha_{\psi}\delta\Delta t'\;CX4(i) \tag{4.48}$$

$$B(I) = -\alpha_{\psi}\delta\Delta t'\;CX5(i) + 1 \tag{4.49}$$

$$C(I) = -\alpha_{\psi}\delta\Delta t'\;CX6(i) \tag{4.50}$$

$$D(I) = RHS\ of\ above\ equation \tag{4.51}$$

Simplifying the expression yields:

$$A(I)\omega^{*}_{i-1} + B(I)\omega^{*}_{i} + C(I)\omega_{i+1} = D(I) \tag{4.52}$$

We may expand equation (4.52) as

$$
\begin{aligned}
&B(1)\omega^{*}_{1} + C(1)\omega^{*}_{2} && = D(1) \\
&A(2)\omega^{*}_{1} + B(2)\omega^{*}_{2} + C(2)\omega^{*}_{3} && = D(2) \\
&\quad A(3)\omega^{*}_{2} + B(3)\omega^{*}_{3} + C(3)\omega^{*}_{4} && = D(3) \\
&\qquad\qquad \cdots\cdots\cdots\cdots \\
&\qquad A(NX-1)\omega^{*}_{NX-2} + B(NX-1)\omega^{*}_{NX-1} + (NX-1)\omega^{*}_{NX} = D(NX-1) \\
&\qquad\qquad A(NX)\omega^{*}_{NX-1} + B(NX)\omega^{*}_{NX} = D(NX)
\end{aligned}
\tag{4.53}
$$

This is a tri-diagonal equation set and can be easily solved using the Thomas solution. (see Appendix B)

The second step is to implicitly advance the iteration in y direction:

$$\omega_j^{**} - \delta \alpha_\psi \Delta t' \left(CY4(j)\, \omega_{j-1}^{**} + CY5(j) \omega_j^{**} + CY6(j) \omega_{j+1}^{**} \right) = \omega^* \tag{4.54}$$

let

$$A(J) = -\alpha_\psi \delta \Delta t'\, CY4(j) \tag{4.55}$$

$$B(J) = -\alpha_\psi \delta \Delta t'\, CY5(j) + 1 \tag{4.56}$$

$$C(J) = -\alpha_\psi \delta \Delta t'\, CY6(j) \tag{4.57}$$

$$D(J) = \omega^* \tag{4.58}$$

Thus we have:

$$A(J)\, \omega_{j-1}^{**} + B(J)\, \omega_j^{**} + C(J)\, \omega_{j+1}^{**} = D(J) \tag{4.59}$$

Again this is a set of tri-diagonal equations and can be solved by the Thomas method.

The third step is to implicitly advance in the z direction:

$$\omega_k^{***} - \delta \alpha_\psi \Delta t' \frac{Da_z}{Da_y} \left(CZ4(k)\, \omega_{k-1}^{***} + CZ5(k)\, \omega_k^{***} + CZ6(k)\, \omega_{k+1}^{***} \right) = \omega^{**} \tag{4.60}$$

$$A(K) = -\alpha_\psi \delta \Delta t' \frac{Da_z}{Da_y}\, CZ4(k) \tag{4.61}$$

$$B(K) = -\alpha_\psi \delta \Delta t' \frac{Da_z}{Da_y}\, CZ5(k) + 1 \tag{4.62}$$

$$C(K) = -\alpha_\psi \delta \Delta t' \frac{Da_z}{Da_y}\, CZ6(k) \tag{4.63}$$

$$D(K) = \omega^{**} \tag{4.64}$$

By using the Thomas method, we obtain the final intermediate value of ω^{***}, the vector potential can be updated by:

$$\psi_x^{p+1} = \psi_x^p + \alpha_\psi \Delta t' \omega^{***} \tag{4.65}$$

4.5.1.2 Solutions of Y Component: ψ_y

$$\frac{1}{\alpha_\psi}\frac{\partial\psi_y'}{\partial t'} = \frac{\partial^2\psi'_y}{\partial x'^2} + \frac{\partial^2\psi'_y}{\partial y'^2} + \frac{Da_z}{Da_x}\frac{\partial^2\psi'_y}{\partial z'^2} + (1-\frac{Da_z}{Da_x})\frac{\partial^2\psi'_z}{\partial y'\partial z'} + RaDa_z(\cos\alpha\frac{\partial\theta}{\partial z'} - \cos\gamma\frac{\partial\theta}{\partial x'}) \tag{4.42}$$

Similar to ψ_x, we also define an intermediate variable ω as

$$\omega = \frac{1}{\alpha_\psi}\frac{\partial\psi_y'}{\partial t'} \tag{4.66}$$

and

$$\psi_y' = \alpha_\psi\,\delta\Delta t'\omega + \psi_y'^{p} \tag{4.67}$$

implicit in x direction:

$$\begin{aligned}\omega^* = {} & \frac{\partial^2}{\partial x'^2}(\alpha_\psi\delta\Delta t'\omega^* + \psi_y'^{p}) + \frac{\partial^2\psi_y'}{\partial y'} + \frac{Da_z}{Da_x}\frac{\partial^2\psi_y'}{\partial z'} + \\ & (1-\frac{Da_z}{Da_x})\frac{\partial^2\psi_z'}{\partial y'\partial z'} + Ra\,Da_z(\cos\alpha\frac{\partial\theta}{\partial z'} - \cos\gamma\frac{\partial\theta}{\partial x'})\end{aligned} \tag{4.68}$$

Using differential approximations, the above expression can be expanded to:

$$\begin{aligned}
& \omega^* - \alpha_\psi\delta\Delta t'(CX4(i)\,\omega^*_{i-1} + CX5(i)\,\omega^* + CX6(i)\,\omega^*_{i+1}) \\
& = CX4(i)\,\psi'_{y,\,i-1} + CX5(i)\,\psi'_y + CX6(i)\,\psi'_{y,\,i+1} \\
& + CY4(j)\,\psi'_{y,\,j-1} + CY5(j)\psi'_y + CY6(j)\psi'_{y,\,j+1} \\
& + \frac{Da_z}{Da_x}(CZ4(k)\,\psi'_{y,\,k-1} + CZ5(k)\,\psi'_y + CZ6(k)\,\psi'_{y,\,k+1}) \\
& + (1-\frac{Da_z}{Da_x})(CYM1ZM1(j,k)\,\psi'_{z,\,j-1,\,k-1} + CYZM1(j,k)\,\psi'_{z,\,j,\,k-1} \\
& + CYP1ZM1(j,k)\,\psi'_{z,\,j+1,\,k-1} + CYM1Z(j,k)\,\psi'_{z,\,j-1,\,k} + CYZ(j,k)\,\psi'_{z,\,j,\,k} \\
& + CYP1Z(j,k)\,\psi'_{z,\,j+1,\,k} + CYM1ZP1(j,k)\,\psi'_{z,j-1,k+1} \\
& + CYZP1(j,k)\,\psi'_{z,j,k+1} + CYP1ZP1(j,k)\,\psi'_{z,j+1,k+1} \\
& + RaDa_z[\cos\alpha\,(CZ1(k)\theta_{k-1} + CZ2(k)\theta + CZ3(k)\theta_{k+1}) \\
& - \cos\gamma\,(CX1(i)\theta_{i-1} + CX2(i)\theta + CX3(i)\theta_{i+1})]
\end{aligned} \tag{4.69}$$

Now we define:

$$A(I) = -\alpha_\psi\,\delta\Delta t'\;CX4(i) \tag{4.70}$$

$$B(I) = -\alpha_{\psi}\delta\Delta t'\ CX5(i) + 1 \tag{4.71}$$

$$C(I) = -\alpha_{\psi}\delta\Delta t'\ CX6(i) \tag{4.72}$$

$$D(I) = RHS\ of\ above\ equation \tag{4.73}$$

simplifying the expression we obtain

$$A(I)\omega^{*}_{i-1} + B(I)\omega^{*}_{i} + C(I)\omega_{i+1} = D(I) \tag{4.74}$$

Again this is a tri-diagonal equation set easily solved by the Thomas method.

The second step is to implicitly advance the iteration in the y direction:

$$\omega^{**}_{j} - \delta\alpha_{\psi}\Delta t'\ (CY4(j)\,\omega^{**}_{j-1} + CY5(j)\omega^{**}_{j} + CY6(j)\omega^{**}_{j+1}) = \omega^{*} \tag{4.75}$$

$$A(J) = -\alpha_{\psi}\delta\Delta t'\ CY4(j) \tag{4.76}$$

$$B(J) = -\alpha_{\psi}\delta\Delta t'\ CY5(j) + 1 \tag{4.77}$$

$$C(J) = -\alpha_{\psi}\delta\Delta t'\ CY6(j) \tag{4.78}$$

$$D(J) = \omega^{*} \tag{4.79}$$

Thus we have

$$A(J)\,\omega^{**}_{j-1} + B(J)\,\omega^{**}_{j} + C(J)\,\omega^{**}_{j+1} = D(J) \tag{4.80}$$

Again this is a set of tri-diagonal equations and can be solved by the Thomas method.

The third step is to implicitly advance in z direction:

$$\omega^{***}_{k} - \delta\alpha_{\psi}\Delta t'\frac{Da_z}{Da_x}\ (CZ4(k)\,\omega^{***}_{k-1} + CZ5(k)\,\omega^{***}_{k} + CZ6(k)\,\omega^{***}_{k+1}) = \omega^{**} \tag{4.81}$$

$$A(K) = -\alpha_{\psi}\delta\Delta t'\frac{Da_z}{Da_x}\ CZ4(k) \tag{4.82}$$

$$B(K) = -\alpha_{\psi}\delta\Delta t'\frac{Da_z}{Da_x}\ CZ5(k) + 1 \tag{4.83}$$

$$C(K) = -\alpha_{\psi}\delta\Delta t'\frac{Da_z}{Da_x}\ CZ6(k) \tag{4.84}$$

$$D(K) = \omega^{**} \tag{4.85}$$

By using the Thomas method, we obtain the final intermediate value of ω^{***}, the vector potential can be updated by:

$$\psi_y^{p+1} = \psi_y^p + \alpha_\psi \Delta t' \omega^{***} \tag{4.86}$$

4.5.1.3. Solution of Z Component ψ_z

$$\frac{1}{\alpha_\psi}\frac{\partial \psi_z'}{\partial t'} = \frac{\partial^2 \psi'_z}{\partial x'^2} + \frac{Da_y}{Da_x}\frac{\partial^2 \psi'_z}{\partial y'^2} + \frac{\partial^2 \psi'_z}{\partial z'^2} + (\frac{Da_y}{Da_x} - 1)\frac{\partial^2 \psi'_y}{\partial y' \partial z'} + RaDa_y(\cos\beta \frac{\partial \theta}{\partial x'} - \cos\alpha \frac{\partial \theta}{\partial y'}) \tag{4.43}$$

We define intermediate variable ω as

$$\omega = \frac{1}{\alpha_\psi}\frac{\partial \psi_z'}{\partial t'} \tag{4.87}$$

and

$$\psi_z' = \alpha_\psi \delta \Delta t' \omega + \psi_z'^p \tag{4.88}$$

implicit in x direction:

$$\begin{aligned} \omega^* = {} & \frac{\partial^2}{\partial x'^2}(\alpha_\psi \delta \Delta t' \omega^* + \psi_z'^p) + \frac{Da_y}{Da_x}\frac{\partial^2 \psi_z'}{\partial y'} + \frac{\partial^2 \psi_z'}{\partial z'} \\ & + (\frac{Da_y}{Da_x} - 1)\frac{\partial^2 \psi_y'}{\partial y' \partial z'} + Ra\, Da_y(\cos\beta \frac{\partial \theta}{\partial x'} - \cos\alpha \frac{\partial \theta}{\partial y'}) \end{aligned} \tag{4.89}$$

Using differential approximations, the above expression can be expanded to:

$$\begin{aligned} & \omega^* - \alpha_\psi \delta \Delta t'(CX4(i)\,\omega^*_{i-1} + CX5(i)\,\omega^* + CX6(i)\,\omega^*_{i+1}) \\ & = CX4(i)\,\psi'_{z,\,i-1} + CX5(i)\,\psi_z' + CX6(i)\,\psi'_{z,\,i+1} \\ & + \frac{Da_y}{Da_x}(CY4(j)\,\psi'_{z,\,j-1} + CY5(j)\psi_z' + CY6(j)\psi_{z,\,j+1}) \\ & + CZ4(k)\,\psi'_{z,\,k-1} + CZ5(k)\,\psi_z' + CZ6(k)\,\psi'_{z,\,k+1}) \\ & + (\frac{Da_y}{Da_x} - 1)(CYM1ZM1(j,k)\,\psi'_{z,\,j-1,\,k-1} + CYZM1(j,k)\,\psi'_{z,\,j,\,k-1} \\ & + CYP1ZM1(j,k)\,\psi'_{z,\,j+1,\,k-1} + CYM1Z(j,k)\,\psi'_{z,\,j-1,\,k} + CYZ(j,k)\,\psi'_{z,\,j,\,k} \\ & + CYP1Z(j,k)\,\psi'_{z,\,j+1,\,k} + CYM1ZP1(j,k)\,\psi'_{z,j-1,k+1} \\ & + CYZP1(j,k)\,\psi'_{z,j,k+1} + CYP1ZP1(j,k)\,\psi'_{z,j+1,k+1} \\ & + RaDa_y[\cos\beta\,(CX1(i)\,\theta_{i-1} + CX2(i)\,\theta + CX3(i)\,\theta_{i+1}) \\ & - \cos\alpha\,(CY1(j)\,\theta_{j-1} + CY2(j)\,\theta + CY3(j)\,\theta_{j+1})] \end{aligned} \tag{4.90}$$

Now we define:

$$A(I) = -\alpha_{\psi}\delta\Delta t'\ CX4(i) \tag{4.91}$$

$$B(I) = -\alpha_{\psi}\delta\Delta t'\ CX5(i) + 1 \tag{4.92}$$

$$C(I) = -\alpha_{\psi}\delta\Delta t'\ CX6(i) \tag{4.93}$$

$$D(I) = RHS\ of\ above\ equation \tag{4.94}$$

simplifying the expression gives

$$A(I)\omega^{*}_{i-1} + B(I)\omega^{*}_{i} + C(I)\omega_{i+1} = D(I) \tag{4.95}$$

This is a tri-diagonal equation set and can be solved as before.

The second step is to implicitly advance the iteration in the y direction:

$$\omega^{**}_{j} - \delta\alpha_{\psi}\Delta t' \frac{Da_y}{Da_x}\left(CY4(j)\omega^{**}_{j-1} + CY5(j)\omega^{**}_{j} + CY6(j)\omega^{**}_{j+1} \right) = \omega^{*} \tag{4.96}$$

$$A(J) = -\alpha_{\psi}\delta\Delta t' \frac{Da_y}{Da_x} CY4(j) \tag{4.97}$$

$$B(J) = -\alpha_{\psi}\delta\Delta t' \frac{Da_y}{Da_x} CY5(j) + 1 \tag{4.98}$$

$$C(J) = -\alpha_{\psi}\delta\Delta t' \frac{Da_y}{Da_x} CY6(j) \tag{4.99}$$

$$D(J) = \omega^{*} \tag{4.100}$$

Thus we have

$$A(J)\omega^{**}_{j-1} + B(J)\omega^{**}_{j} + C(J)\omega^{**}_{j+1} = D(J) \tag{4.101}$$

Again this is a set of tri-diagonal equations and can be solved by Thomas method.

The third step is to implicitly advance the iteration in the z direction:

$$\omega^{***}_{k} - \delta\alpha_{\psi}\Delta t'\left(CZ4(k)\omega^{***}_{k-1} + CZ5(k)\omega^{***}_{k} + CZ6(k)\omega^{***}_{k+1}\right) = \omega^{**} \tag{4.102}$$

$$A(K) = -\alpha_{\psi}\delta\Delta t'\ CZ4(k) \tag{4.103}$$

$$B(K) = -\alpha_{\psi}\delta\Delta t'\ CZ5(k) + 1 \tag{4.104}$$

$$C(K) = -\alpha_{\psi}\delta\Delta t'\ CZ6(k) \tag{4.105}$$

$$D(K) = \omega^{**} \tag{4.106}$$

By using the Thomas method, we obtain the final intermediate value of ω^{***}, the vector potential can be updated by using:

$$\psi_z^{p+1} = \psi_z^p + \alpha_\psi \Delta t' \omega^{***} \tag{4.107}$$

4.5.2. Solution Procedure of Temperature

The solution procedure is simple and very similar to those of vector potential.

$$\frac{\partial \theta}{\partial t'} = -R_c \vec{V}_p' \cdot \nabla' \theta + R_c \nabla'^2 \theta + \frac{R_c Q_{heat} x_0^2}{(T_h - T_c) k_{eff}} \tag{3.97}$$

because

$$\begin{aligned} \vec{V}_p \cdot \nabla' \theta &= \vec{V}_p \cdot \nabla' \theta + \theta \nabla' \cdot \vec{V}_p \\ &= \nabla' \cdot (\vec{V}_p \theta) \end{aligned} \tag{4.108}$$

Equation (3.97) may deform to:

$$\frac{\partial \theta}{\partial t'} = -R_c \nabla' \cdot (\vec{V}_p \theta) + R_c \nabla'^2 \theta + \frac{R_c Q_{heat} x_0^2}{(T_h - T_c) k_{eff}} \tag{4.109}$$

We define

$$\omega = \frac{\partial \theta}{\partial t'} \quad (4.110), \text{ and} \qquad \theta = \delta \Delta t' \omega + \theta^p \quad (4.111)$$

we may have

$$\begin{aligned} \omega = \frac{\partial \theta}{\partial t'} &= -\delta \Delta t' R_c \left[\frac{\partial(\omega u)}{\partial x} + \frac{\partial(\omega v)}{\partial y} + \frac{\partial(\omega w)}{\partial z}\right] \\ &- R_c \left[\frac{\partial(\theta u)}{\partial x} + \frac{\partial(\theta v}{\partial y} + \frac{\partial(\theta w)}{\partial z}\right] \\ &+ \Delta t' R_c \left[\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} + \frac{\partial^2 \omega}{\partial z^2}\right] \\ &+ R_c \left[\frac{\partial^2 \theta^p}{\partial x^2} + \frac{\partial^2 \theta^p}{\partial y^2} + \frac{\partial^2 \theta^p}{\partial z^2}\right] \\ &+ \frac{x_0^2 R_c Q_{heat}}{(T_h - T_c) k_{eff}} \end{aligned} \tag{4.112}$$

implicit in x direction:

$$\begin{aligned} &\omega^* - R_c \delta \Delta t' (CX4(i) \omega_{i-1}^* + CX5(i) \omega^* + CX6(i) \omega_{i+1}^*) \\ &+ R_c \delta \Delta t' [CX1(i)(u_{i-1} \ \omega_{i-1}^*) + CX2(i)(u \omega^*) + CX3(i)(u_{i+1} \omega_{i+1}^*] \\ &= R_c [CX4(i) \theta_{i-1} + CX5(i) \theta + CX6(i) \theta_{i+1}] \\ &R_c [CY4(j) \theta_{j-1} + CY5(j) \theta + CY6(j) \theta_{j+1}] \\ &+ R_c [CZ4(k) \theta_{k-1} + CZ5(k) \theta + CZ6(k) \theta_{k+1}] \\ &- R_c [CX1(i)(\theta_{i-1} v_{j-1} 0 + CY2(j)(\theta v) + CY3(j)(\theta_{j+1} v_{j+1})] \\ &- R_c [CZ1(k)(\theta_{k-1} w_{k-1}) + CZ2(k) \theta w + CZ3(j)(\theta_{k+1} w_{k+1})] \\ &+ CC0Q \end{aligned} \tag{4.113}$$

let

$$A(I) = Rc\,\delta\,\Delta t'\,CX1(i)\,u_{i-1} - Rc\,\delta\,\Delta t'\,CX4(i) \tag{4.114}$$

$$B(I) = Rc\,\delta\,\Delta t'\,CX2(i)\,u_i - Rc\,\delta\,\Delta t'\,CX5(i) + 1 \tag{4.115}$$

$$C(I) = Rc\,\delta\,\Delta t'\,CX3(i)\,u_{i+1} - Rc\,\delta\,\Delta t'\,CX6(i) \tag{4.116}$$

$$D(I) = RHS\ of\ above\ equation \tag{4.117}$$

By solving these equations we obtain ω^* and then forward the iteration implicitly in the y direction, we have

$$\begin{aligned}\omega^{**} - R_c\delta\Delta t'\,(CY4(j)\omega^{**}_{j-1} + CY5(j)\omega^{**} + CY6(j)\omega^{**}_{j+1})\\ + R_c\delta\Delta t'\,[CY1(j)(u_{j-1}\ \omega^{**}_{j-1}) + CY2(j)(u\omega^{**}) + CY3(j)(u_{j+1}\,\omega^{**}_{j+1}]\\ = \omega^*\end{aligned} \tag{4.118}$$

$$A(J) = R_c\,\delta\,\Delta t'\,CY1(j)\,v_{j-1} - R_c\,\delta\,\Delta t'\,CY4(j) \tag{4.119}$$

$$B(J) = R_c\,\delta\,\Delta t'\,CY2(j)\,v_j - R_c\,\delta\,\Delta t'\,CY5(j) + 1 \tag{4.120}$$

$$C(J) = R_c\,\delta\,\Delta t'\,CY3(j)\,v_{j+1} - R_c\,\delta\,\Delta t'\,CY6(j) \tag{4.121}$$

$$D(J) = \omega^* \tag{4.122}$$

Similarly implicit in z,

$$\begin{aligned}\omega^{***} - R_c\delta\Delta t'\,(CZ4(k)\omega^{***}_{k-1} + CZ5(k)\omega^{***} + CZ6(k)\omega^{***}_{k+1})\\ + R_c\delta\Delta t'\,[CZ1(k)(w_{k-1}\ \omega^{***}_{k-1}) + CZ2(k)(w\omega^{***}) + CZ3(k)(w_{k+1}\,\omega^{***}_{k+1}]\\ = \omega^{**}\end{aligned} \tag{4.123}$$

$$A(K) = R_c\,\delta\,\Delta t'\,CZ1(k)\,w_{k-1} - R_c\,\delta\,\Delta t'\,CZ4(k) \tag{4.124}$$

$$B(K) = R_c\,\delta\,\Delta t'\,CZ2(k)\,w_k - R_c\,\delta\,\Delta t'\,CZ5(k) + 1 \tag{4.125}$$

$$C(K) = R_c\,\delta\,\Delta t'\,CZ3(k)\,w_{k+1} - R_c\,\delta\,\Delta t'\,CZ6(k) \tag{4.126}$$

$$D(K) = \omega^{**} \tag{4.127}$$

updating θ using

$$\theta^{p+1} = \theta^p + \Delta t'\,\omega^{***} \tag{4.128}$$

4.6 Stability and Consistency

Next we discuss the convergency of the numerical procedure. As described by Carnahan et al (1969) the term *convergency* is understood to mean that the exact solution of the finite-difference problem (in the absence of round-off error) tends to the solution of the partial differential equation(PDE) as the grid spacings in time and distance tend to zero. There are two important concepts closely associated with the convergence of a particular finite-difference procedure, namely, those of *consistency* and *stability*.

The term *stability* denotes a property of the particular finite-difference equation(s) used as the time increment is made vanishingly small. It means that there is an upper limit (as $\Delta t \rightarrow 0$) to the extent to which any piece of information, whether present in the initial conditions, or brought in via the boundary conditions, or arising from any sort of error in the calculations, can be amplified in the computations. The term *consistency*, applied to a certain finite-difference procedure, means that the procedure may in fact approximate the solution of the PDE under study, and not the solution of some other PDE. Consistency is often taken for granted.

It can be shown that, for a single linear equation, the Samarskii-Andreyev (1963) method is unconditionally stable. This scheme was stable when applied to a natural convection problem, but unfortunately it is impossible to prove or disprove the stability of it when applied to coupled, non-linear equations, Mallinson and de Vahl Davis (1973). Mallinson and de Vahl Davis (1973) also demonstrated that instability can still be introduced by the coupling between equations of a system, but the time step Δt and false transient parameters α_ψ can be adjusted to control the instability and therefore a solution was still obtained. In this study we observed that if Ra>10^9, a fine mesh and smaller time step was needed to make the program stable, although much more computing time was used.

4.7 Solution Procedure and Program Explanations

The arrangement of the solution procedure is discussed in this section. The numerical solutions of the governing equations were found by embedding their problem-specific discretized forms in FRECON3D, a computer program developed by Goh et al (1988). Flow charts of program AGRI_3D.FOR for this study are illustrated in Figure 4.4 and 4.5. To begin, the initial condition, boundary conditions and control variables are input by a subroutine DATIN then all the fields of vector potential, velocity, temperature and work space are initialised by subroutine INITA. Component equations of vector potential are solved by P1SOL, P2SOL and P3SOL subroutines, temperatures are advanced by TSOL and TBC, velocity fields are solved in VBC and VELOC.

ITERAT is the iteration control subroutine. It redirects the advance routine from the inner iteration to the main iteration or from the main iteration to the inner iteration. The outer iteration loop or main loop, is started from the advance routine for solving the vector potential field. Initially the ψ_z field is treated as constant in subroutines P1SOL and P2SOL, the new ψ_y is then used in subroutine P3SOL to obtain ψ_z. Every subroutine in

P1SOL, P2SOL and P3SOL includes its own third-level iteration loop to approach steady state. These processes continue until ψ_x, ψ_y and ψ_z do not change significantly, they consist of the inner iteration loops and within the inner loops, temperature is held constant.

When updated values of ψ_x, ψ_y and ψ_z are obtained, the process will jump from the inner loop and return to the main loop. Velocity field is then updated by subroutine TSOL and TBC. Once the temperature field is updated, the main iteration loop turns to the next cycle by applying the updated θ, ψ_x, ψ_y and ψ_z fields as starting points. When the maximum iteration number is reached or the solution is convergent or indicates divergence, the program will jump from the iteration loops and writes the results by calling subroutines OUTPUT and WRITER.

Program AGRI_3D.FOR is shown in Appendix D.

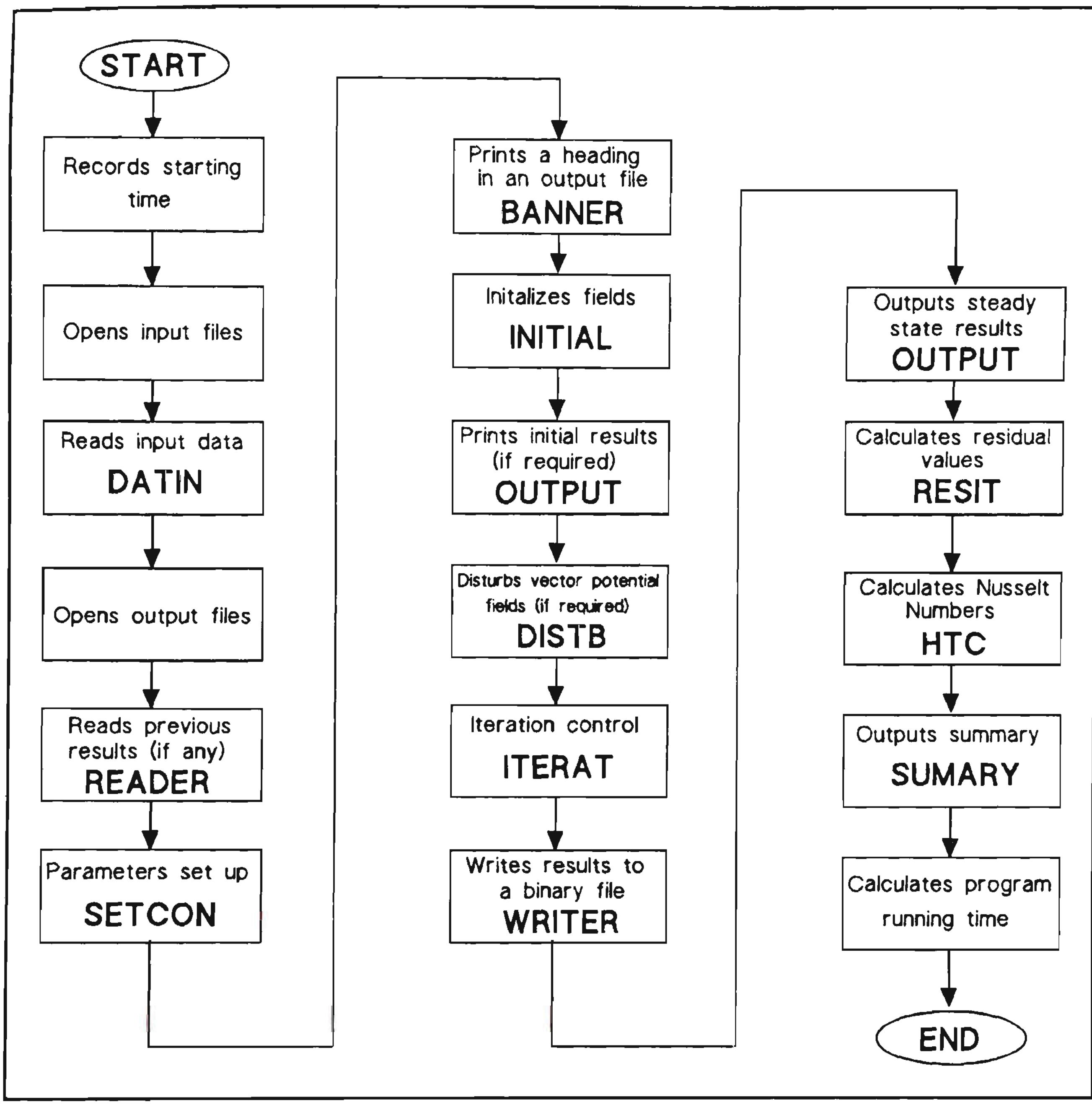


Figure 4.4 Program flow chart: main iteration loop.

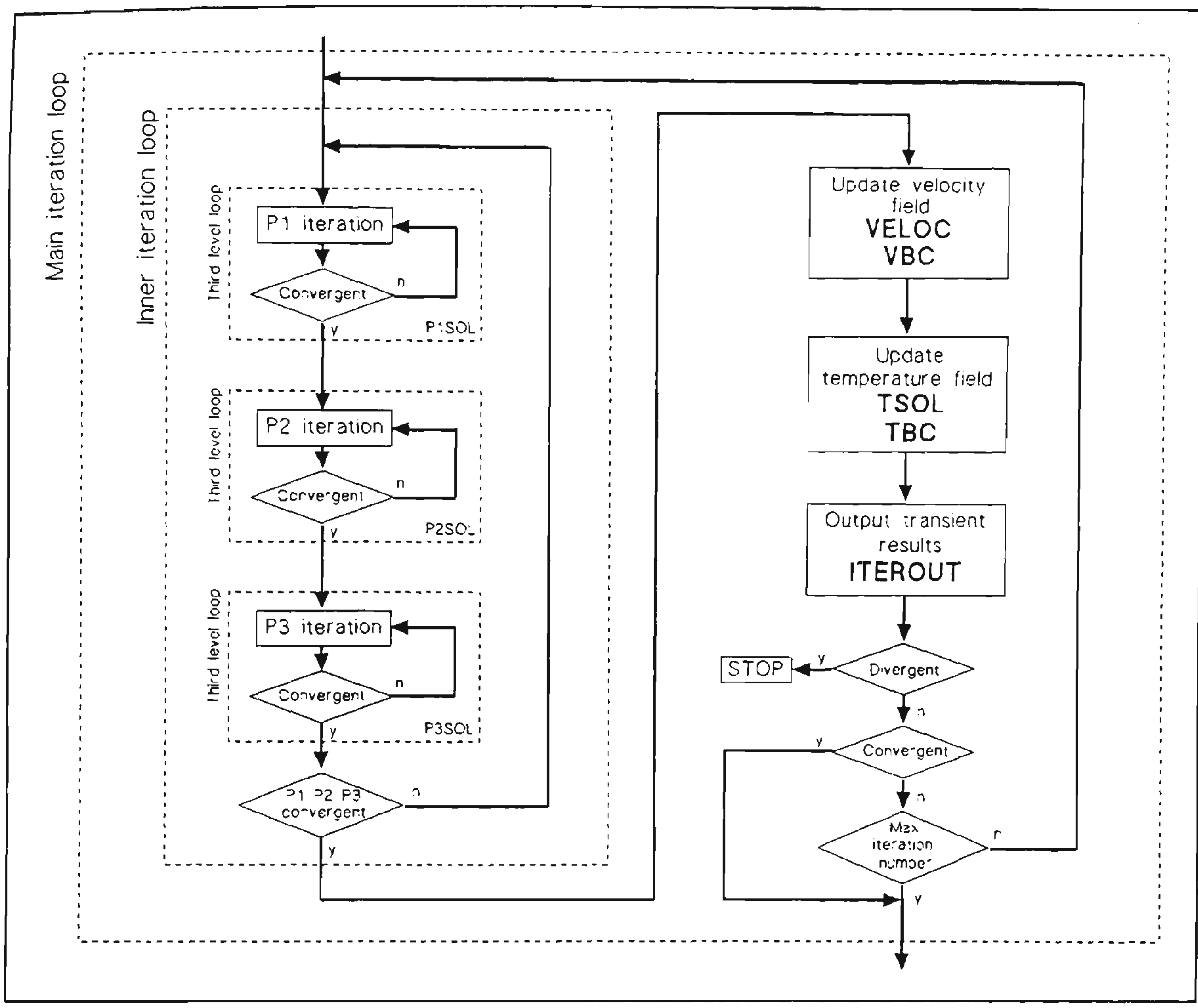


Figure 4.5 Program flow chart: inner loop.

CHAPTER 5

NUMERICAL EXPERIMENTS AND COMPARISONS WITH PRIOR WORK

5.1 Numerical Experiments

In obtaining a physically sensible solution to a set of differential equations by numerical integration, not only does correct formulation of the governing equations and the initial and boundary conditions affect the results, but, as is well known, the numerical solution procedure itself may have a significant influence on the final results. In this chapter, the effects on the solution of the mathematical model developed in this study for different time steps, grid sizes and grid patterns are investigated. As well, convergence criteria for the vector potential and temperature are examined along with the effect of varying the Rayleigh and Darcy numbers. Many numerical experiments of packaged apples in a level rectangular container had been performed to examine the effects of these factors on solutions of a range of Ra and Da numbers. Finally, the output of the program is compared with the results of Beukema's (1983) numerical study in which he characterised three-dimensional heat transfer in packed agricultural produce by assuming isotropic permeability and a constant rate of respiratory heat generation.

5.1.1 Effect of Criterion for Convergence on Steady State Potential

As described in chapter 4, before the new temperature and velocity fields are obtained, updated vector potential should be iterated until it reaches its steady state. The criterion for convergence of the vector potentials was called PHIERR and when the condition

$$\frac{\sum_i \sum_j \sum_k |\psi^{p+1} - \psi^p|}{\sum_i \sum_j \sum_k |\psi^{p+1}|} \leq PHIERR \tag{5.1}$$

was satisfied the vector potential field was iterated to a steady state. In the above inequality, superscript ^{p} represents the ***pth*** cycle of the third-level iteration loop. As shown in Table 5.1, when the product of Da and Ra is large, and the iteration time step is not small, PHIERR plays an important role in determining the final state of the vector potential.

When a uniform mesh of 11x11x11 grid is used, the differences between P2min and P3max at PHIERR = 10 and PHIERR = 0.001 were large (nearly 40%), thus PHIERR = 10 appears too large for the vector potential field to reach a steady state. Theoretically, the smaller PHIERR is, the smaller difference there is between ψ^p and ψ^{p+1}, but we can observe that ψ_2 and ψ_3 do not change significantly when PHIERR is less than 0.001. The difference between P2mins when PHIERR = 0.001 and PHIERR = 0.00001 is only 1%. So PHIERR = 0.01 is small enough to obtain reasonably accurate results for vector potential.

The iteration number in the third level iterative loop increases significantly as PHIERR decreases. When PHIERR changes from 0.001 to 0.00001, the accuracy of the vector potential field can be improved by 1%, but the third level iteration number needed is 51 times greater. This causes the program to take many more hours to run.

To balance the accuracy and economy of computation time, PHIERR should be chosen carefully. In most circumstances, PHIERR = 0.001 insures that the vector potential field converges to a satisfactory steady state while only a small number of inner iterations is needed.

A small time step for the main iteration loop can reduce the influence of PHIERR as shown in Table 5.2. It was found that when Δt was less than a critical value for a particular mesh size, there was no significant difference evident when choosing large and small values of PHIERR for the P2 and P3 iterations. Probably because the main loop time step was small, the temperature difference between one step was also small and the difference between vector potentials was also small, thus enabling the third level loop to readily reach steady state.

A further interesting result is revealed on examining Table 5.2. When the advancement time step is small, the decrease in PHIERR does not make the iteration number inside the third level loop increase rapidly. It takes 38 cycles to reach steady state when PHIERR = 10 and only 43 cycles when PHIERR = 0.001 which is 10^4 times smaller.

Table 5.1 Effect of PHIERR ($Ra*Da=100$, $Ra=6.59\times10^{6}$, $Da=1.518\times10^{-5}$)

Mesh Size	Time Step	PHIERR	P2min	P2max	P3min	P3max	Iter_num
11	1.	10.	-23.88	23.88	-23.37	-8.89	2
11	1.	1.	-29.39	29.39	-28.66	-11.23	3
11	1.	0.1	-36.03	36.03	-33.65	-14.83	6
11	1.	0.01	-37.95	37.95	-32.23	-17.62	13
11	1.	0.001	-38.52	38.52	-27.83	-20.03	51
11	1.	0.0001	-39.11	39.11	-27.06	-19.76	227
11	1.	0.00001	-38.92	38.92	-27.23	-19.62	791
11	1.	0.000001	-38.95	38.95	-27.37	-19.84	3115
11	1.	0.0000001	-38.94	38.94	-27.34	-19.95	8716
21	1.	10.	-15.34	15.34	-14.98	-3.08	2
21	1.	1.	-18.83	18.83	-18.36	-3.91	3
21	1.	0.1	-24.29	24.29	-22.98	-5.79	7
21	1.	0.01	-25.38	25.38	-22.43	-7.44	15
21	1.	0.001	-25.44	25.44	-18.65	-9.98	53
21	1.	0.0001	-25.75	25.75	-15.60	-10.94	210
21	1.	0.00001	-25.63	25.63	-14.56	-10.27	940
21	1.	0.000001	-25.62	25.62	-14.34	-9.69	3875
21	1.	0.0000001	-25.63	25.63	-14.37	-9.54	14425

Note:
1. ***x*** component of vector potential P1=0;
2. P2 and P3 represent ***y*** and ***z*** components of vector potential; '***min***' and '***max***' represent the minimum and maximum values.
3. ***Iter_num*** represent the iteration number needed to reach 'steady' state within the inner loop.

Table 5.2 Effect of PHIERR on solution of vector potential when using small time step.

Mesh Size	Time Step	PHIERR	P2min	P2max	P3min	P3max	Iter_num
5	0.1000	10.0000000	-49.1247559	49.1247559	-44.1120949	49.1247635	38
5	0.1000	1.0000000	-49.1247787	49.1247826	-44.1121254	49.1247826	39
5	0.1000	0.1000000	-49.1247749	49.1247749	-44.1121254	-36.8230019	40
5	0.1000	0.0100000	-49.1247787	49.1247711	-44.1121292	49.1247711	41
5	0.1000	0.0010000	-49.1247711	49.1247673	-44.1121254	49.1247711	43
5	0.1000	0.0001000	-49.1244926	49.1244926	-44.1148148	49.1244926	85
5	0.1000	0.0000100	-49.1230354	49.1230316	-44.1148415	49.1230316	164
5	0.1000	0.0000010	-49.1226425	49.1226425	-44.1150780	49.1226425	317
5	0.1000	0.0000001	-49.1225891	49.1225853	-44.1151352	49.1225853	584

Note:

1. ***x*** component of vector potential P1=0;
2. P2 and P3 represent ***y*** and ***z*** components of vector potential; '***min***' and '***max***' represent the minimum and maximum values.
3. ***Iter_num*** represent the iteration number needed to reach 'steady' state within the inner loop.

5.1.2. Effect of Iteration Time Step

Special attention must be paid to specifying iteration time steps, Δt_{main} and Δt_{inner}. In the inner loop and third level iterative loop, Δt_{inner} is specified to be large enough to reduce the running-time yet small enough to avoid instability. In the main loop, Δt_{main} is specified to be large enough to reduce the running-time yet small enough to obtain a satisfactorily accurate true transient solution. The following relations were used:

$$\Delta t_{main} = Relx \cdot h_{min}^2 \tag{5.2}$$

where Δt_{main} is the time step for the main iteration loop, ***Relx*** is the relaxation parameter to improve the solution economy and accuracy, h_{min} is the minimum grid distance. And

$$\Delta t_{inner} = \alpha_{\psi} \Delta t_{main} \tag{5.3}$$

where Δt_{inner} is the iteration time step for iterative advancement in solving the vector potential, α_{ψ} is the false transient coefficient of the vector potential.

In this study it was observed that when Ra was small, a large iteration time step could be used, but when Ra was large, a small time step improved the convergence and stability of the program. We could always find ***Relx*** and α_{ψ} to obtain a satisfied combination of Δt_{main} and Δt_{inner} for different commodities in different initial and boundary conditions. It is found that the choosing of ***Relx*** and α_{ψ} is not unique. When Δt_{main} is large, a small Δt_{inner} may be used to balance the stability. When Δt_{main} is small, a large Δt_{inner} can be used, thus improving the convergence speed in the inner loop. Figure 5.1 shows an example of choosing two sets of different Δt_{main} and Δt_{inner} to obtain the same results.

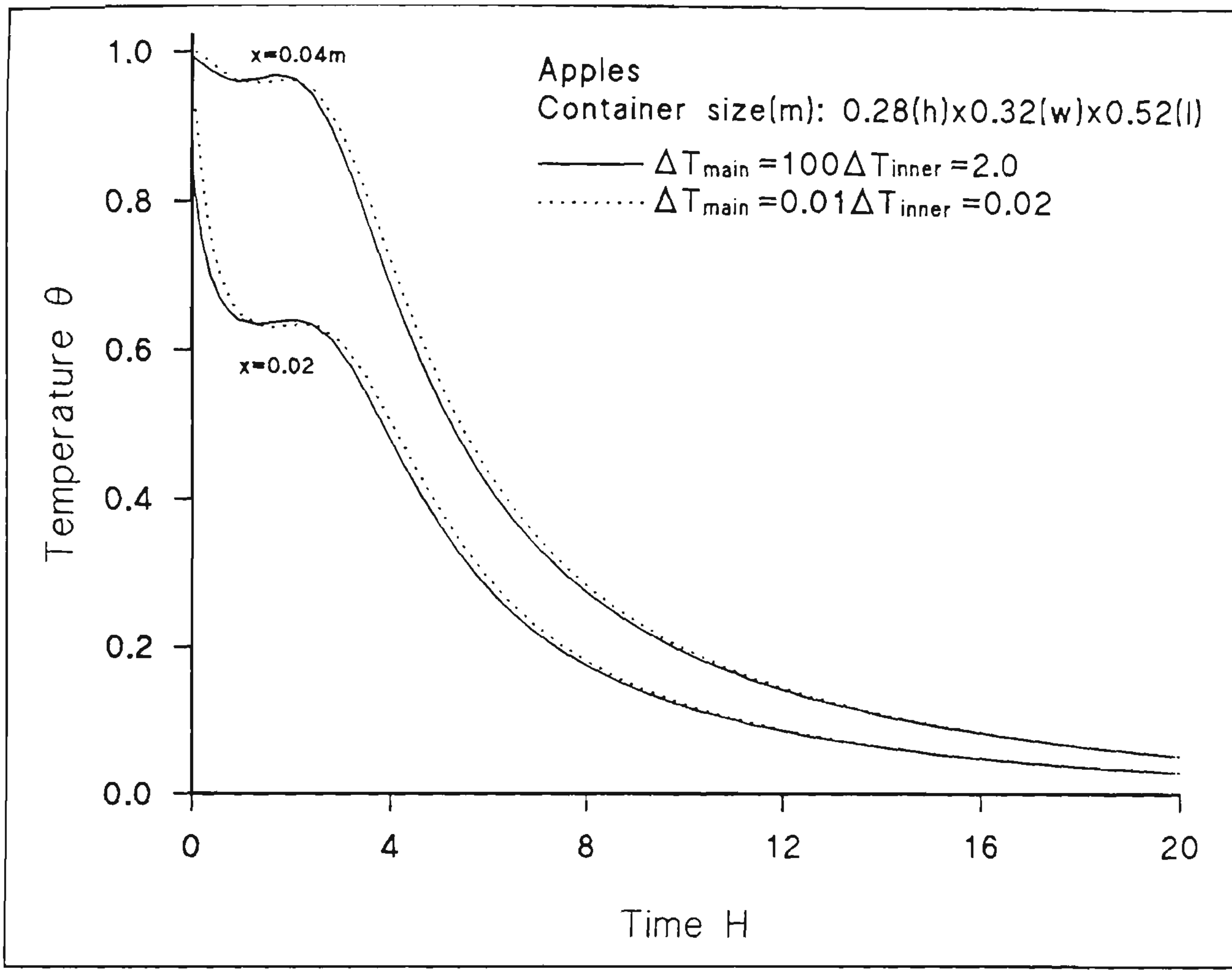


Figure 5.1 Effect of Iteration Time Step on Solution

Table 5.3 Mesh size effect on solution, Da*Ra=10

Mesh size	Time step	PHIERR	Temperature at central	Cooling time
5	0.001	0.001	1.001165	1.184726834
11	0.001	0.001	1.003337	1.184726834
21	0.001	0.001	1.003337	1.184726834
31	0.001	0.001	1.003337	1.184726834
41	0.001	0.001	1.003337	1.184726834
51	0.001	0.001	1.003337	1.184726834

5.1.3 Effect of Grid Size on Uniform Mesh

When using a uniform mesh, the mesh size plays a very similar role in influencing the stability to that of the time step. Also the mesh size has some influence in choosing the iteration time steps. As we can see in equations 5.2 and 5.3, mesh size is relative to time steps.

Numerical results showed that different Ra and Da numbers required different mesh sizes to obtain the best compromise between accuracy and economy. A fine mesh caused higher time consumption but improved the convergence when ***Ra*** was very large. Table 5.3 shows the comparison of different mesh size effects on predicting temperature at the central point of a 0.28×0.52×0.32m container of apples, after cooling 1.184 hours at 30 °C initial temperature. It shows that a mesh size of 11×11×11 can achieve very accurate results.In this study it is found that the Samarskii-Androyev alterative direction implicit scheme has a very good stability and convergence characteristics when applied to natural convection. We can always find a satisfactory mesh size and time steps to obtain an economic and accurate solution.

5.1.4 Non-Uniform Grid Size

A non-uniform mesh can sometimes improve both the accuracy and time economy of a solution process. The design of a non-uniform mesh is usually based on a prior 'guess' or prior results of a simple uniform mesh. A non-uniform mesh was designed to perform a numerical investigation on the effect of different mesh types on the program. A narrow grid size near the walls and a wide grid size in the central regions were used to form the non-uniform mesh pattern defined below

x direction(mm): 20 × 20 × 28 × 36 × 36 × 36 × 36 × 28 × 20 × 20
y direction(mm): 20 × 52 × 52 × 52 × 68 × 68 × 68 × 68 × 52 × 52 × 52 × 20
z direction(mm): 20 × 20 × 32 × 44 × 44 × 44 × 44 × 32 × 20 × 20.

A 11×13×11 uniform mesh was designed as

x direction(mm): 28 × 28 × 28 × 28 × 28 × 28 × 28 × 28 × 28 × 28
y direction(mm): 52 × 52 × 52 × 52 × 52 × 52 × 52 × 52 × 52 × 52 × 52 × 52
z direction(mm): 32 × 32 × 32 × 32 × 32 × 32 × 32 × 32 × 32 × 32.

While it may not always be apparent how to best select a mesh pattern for a particular situation, the results shown in Table 5.4 demonstrate that a judiciously chosen non-uniform mesh can improve the accuracy of the solution and lead to some economic benefit by way of shorter CPU time. For every identical iteration step, the errors of potential and temperature are smaller using the non-uniform mesh than using the uniform mesh. To reach the same accuracy for vector potential or temperature, less iteration steps are needed, for example, to obtain a steady state (when T_error is less or equal to 1.0×10^{-5}), about 6000 main iteration steps for non-uniform mesh and 6300 main iteration steps for the uniform mesh, which is about 5% large, are needed.

Table 5.4 Effect of different types of mesh on accuracy and convergence

	*i*th step	P2_error	P3_error	T_error
Non-uniform mesh				
	3000	0.44E-04	0.40E-04	0.30E-03
	4000	0.58E-04	0.73E-04	0.10E-03
	5000	0.18E-04	0.23E-04	0.32E-04
	6000	0.56E-05	0.70E-05	0.96E-05
	6400	0.35E-05	0.43E-05	0.59E-05
uniform mesh				
	3000	0.73E-04	0.64E-04	0.40E-03
	4000	0.35E-04	0.29E-04	0.14E-03
	5000	0.32E-04	0.40E-04	0.47E-04
	6000	0.10E-04	0.12E-04	0.15E-04
	6400	0.64E-05	0.78E-05	0.91E-05

5.2 Comparison of Results with Previous Work

Beukema (1980, 1983) conducted experimental and numerical studies on natural convection in isotropic porous media with constant heat generation rates. The results obtained by Beukema agree very well with the numerical results found in this study under the same conditions. To examine the accuracy of our program, a test was performed by using exactly the same boundary conditions and the same physical properties to obtain the temperature field and temperature versus time curves which can be compared with those of Beukema's. The data used in Beukema's study and this test are shown in Table 5.5.

Table 5.5 Data used by Beukema (1980, 1983)

Constant rate of heat generation	60 W/m^3
Height of container	0.5 m
Width of container	0.76 m
Length of container	0.76 m
Initial temperature	28.75°C
Environment temperature	19.2°C
Isotropic permeability	1.45×10^{-6} m^2
Effective thermal conductivity	0.25 W/mK
Heat capacity, air $[\rho\ C_p]$ air	1230 J/mK
Heat capacity, media $[\rho\ C_p]_p$	2.3×10^6 J/mK
Dynamic viscosity, air, μ	1.77×10^{-5} kg/ms
Porosity	0.378
Convective heat transfer coefficient at top of the container	10 W/m^2K
Convective heat transfer coefficient at bottom of the container	7 W/m^2K
Convective heat transfer coefficient at side walls of the container	20 W/m^2K

A non-uniform grid mesh was designed to match those particular points that Beukema used to locate the temperature measurement. A 11x11x11 grid network was used with intervals of $0.11 \times 0.09 \times 0.1 \times 0.1 \times 0.11 \times 0.09 \times 0.1 \times 0.12 \times 0.09 \times 0.09$(m) in the x direction, and equal intervals in the *y* and *z* directions. The criterion set for vector potential convergence was 0.001.

Figure 5.2 shows the results of Beukema's work and Figure 5.3 shows the present results. Theses figures demonstrate that the temperature versus time characteristics calculated in the two studies agree quite well.

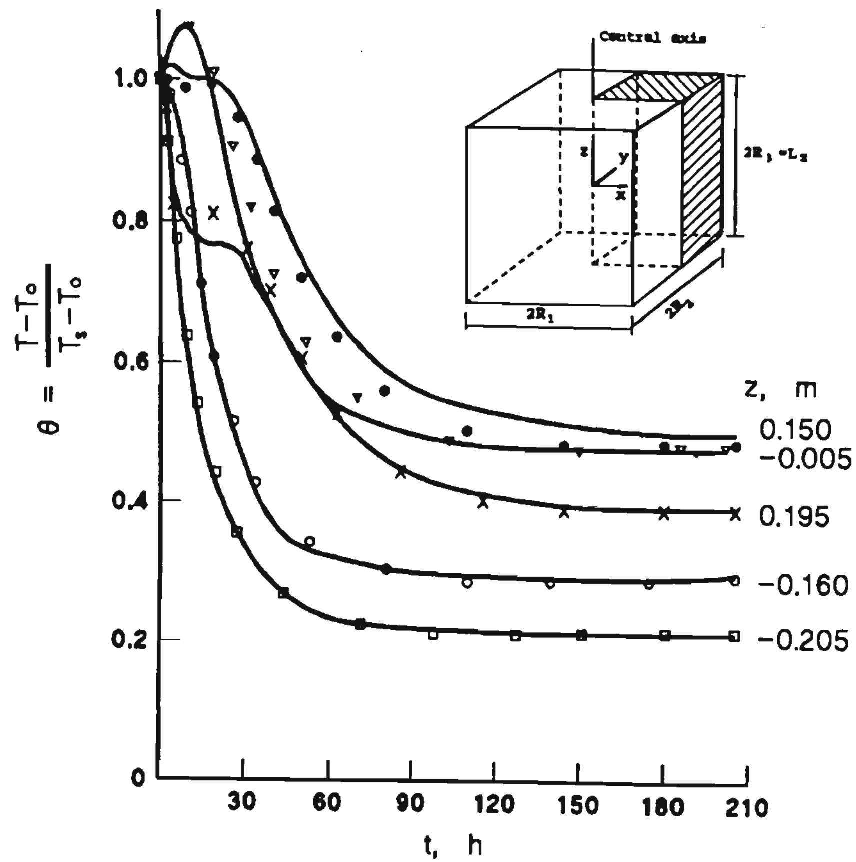


Figure 5.2 Beukema's results(1983): 'Measured and calculated (—) temperature changes with time at different locations along the central axis.

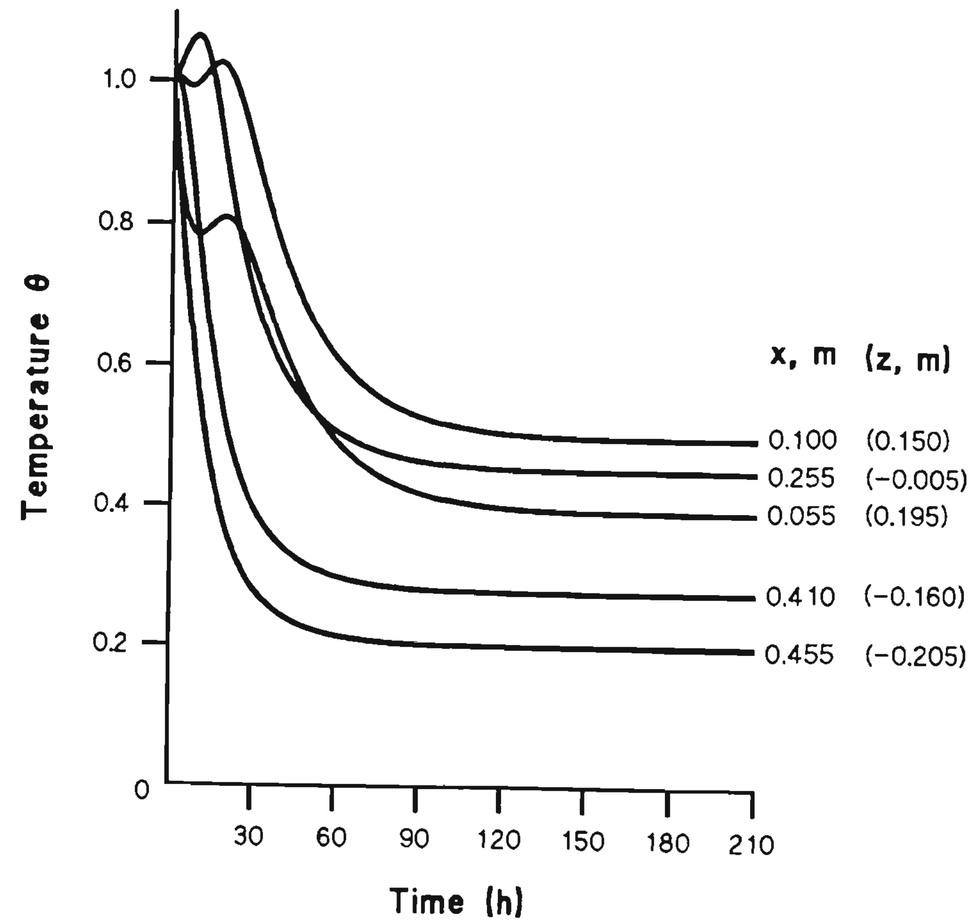


Figure 5.3 The temperature versus time curves along the central vertical axis calculated in this study. Locations correspond to those in Beukema's study.

CHAPTER 6

NUMERICAL RESULTS FOR COOLING STORED AGRICULTURAL PRODUCE

6.1 Introduction

The results of a series of numerical experiments designed to simulate the convective heat transfer processes occurring in the packed beds of selected fruits and vegetables as they are cooled are presented in this chapter. Besides modelling the characteristic variation of heat generated by respiration with temperature for each commodity, effects due to changes in the permeability of the packed bed are investigated by choosing fruits and vegetables exhibiting either isotropic or orthotropic permeability. For the comodities considered, natural convection with both adiabatic and isothermal boundary conditions imposed on the floor of the container are studied. A comprehensive discussion of the results has been reserved until the next chapter, however interpretations arising naturally from the results may be dealt with as they occur in this chapter.

Data from Table 2.2 which gives respiration rates for various commodities was used for constructing, by least squares, the respiration rate versus temperature function for apples, Brussels sprouts, carrots and asparagus in the program. The data on the physical properties of air used in the program are shown in Table 6.1.

Table 6.1 Physical data for air at atmospheric pressure, T=288K, Özisik (1985).

Specific heat, Cp_{air}, KJ/Kg·K	1.0056
Air viscosity, μ, Kg/m·s	1.8642×10^{-5}

6.2 Produce with Isotropic Permeability

Because of their roughly spherical shapes, apples and Brussels sprouts were chosen as typical of commodities, that when packed in boxes, may be treated as packed beds with isotropic permeability. A further consideration was the desirability of studying differences between a weakly respiring commodity, characterised by apples, and a strongly respiring one, such as Brussels sprouts.

The physical properties data used in the program to investigate natural convection in a container for storing apples are given in Table 6.2.

Table 6.2 Physical data for Apples, Fikiin (1983)

Water content, ϕ_{water}, %	83.5
Specific heat, Cp_s, KJ/Kg·K	3.724
Density, ρ_s, Kg/m^3	1066
Density of packed bed, ρ_p, Kg/m^3	657.5
Porosity ε [1)]	0.3832
Average diameter, m [2)]	0.067
Effective thermal conductivity in packed bed, KW/m·K	0.291×10^{-3}
Initial temperature, °C	30°
Cold wall temperature, °C	0°

Note: 1) $\varepsilon \approx 1 - \rho_p/\rho_s$;
2) Average diameter obtained from Gala Apples. It may vary from different grades or different cultivars.

The data on the physical properties of Brussels sprouts used in the program are shown in Table 6.3.

Table 6.3 Physical data for Brussels Sprouts

Water content, ϕ_{water}, %, (Hardenburg, 1986)	84.9
Specific heat, Cp_s, KJ/Kg·K, (Mohsenin,1980)	3.684
Density, ρ_s, Kg/m^3 [1)]	1060
Average diameter, m [2)]	0.029
Porosity, ε [3)]	0.3832
Effective thermal conductivity in packed bed, KW/m·K [3)]	0.291×10^{-3}
Initial temperature, °C	30°
Cold wall temperature, °C	0°

Note: 1) $\rho_s = 2670/(1.67+\phi_{water})$, Fikiin,1983;
2) May vary from different grades;
3) Assumed having the same value in Table 6.2.

6.2.1 Natural Convection with an Adiabatic Floor

The container was assumed to have an adiabatic floor, four cool isothermal side walls and a cool isothermal top. The box size was 320mm(w) × 520mm(l) × 280mm(h). The dimensions were taken from an actual package box for apples. A non-uniform grid mesh of 10 + 8 × 32.5 + 10 in the *x* direction, 10 + 10 x 50 + 10 in the *y* direction and 10 + 8 × 37.5 + 10 in the *z* direction was used. The convergence criterion was 0.001 and the main iteration time step was 0.02551.

The computed results of the evolving three-dimensional temperature and velocity fields with respiratory heat generation for apples with an adiabatic container floor are presented in Figures 6.1 to 6.12. Figure 6.1 shows the temperature response versus time at different locations along the central vertical axis of the box. At x=0.036 and x=0.152 the temperature initially decreases for some 1.5 to 3 hours, after which it plateaus and then begins to cool rapidly. While at x=0.268, x=0.384, an initial temperature rise can be observed. Some 20 hours after the cooling process first commenced a steady state temperature distribution was reached.

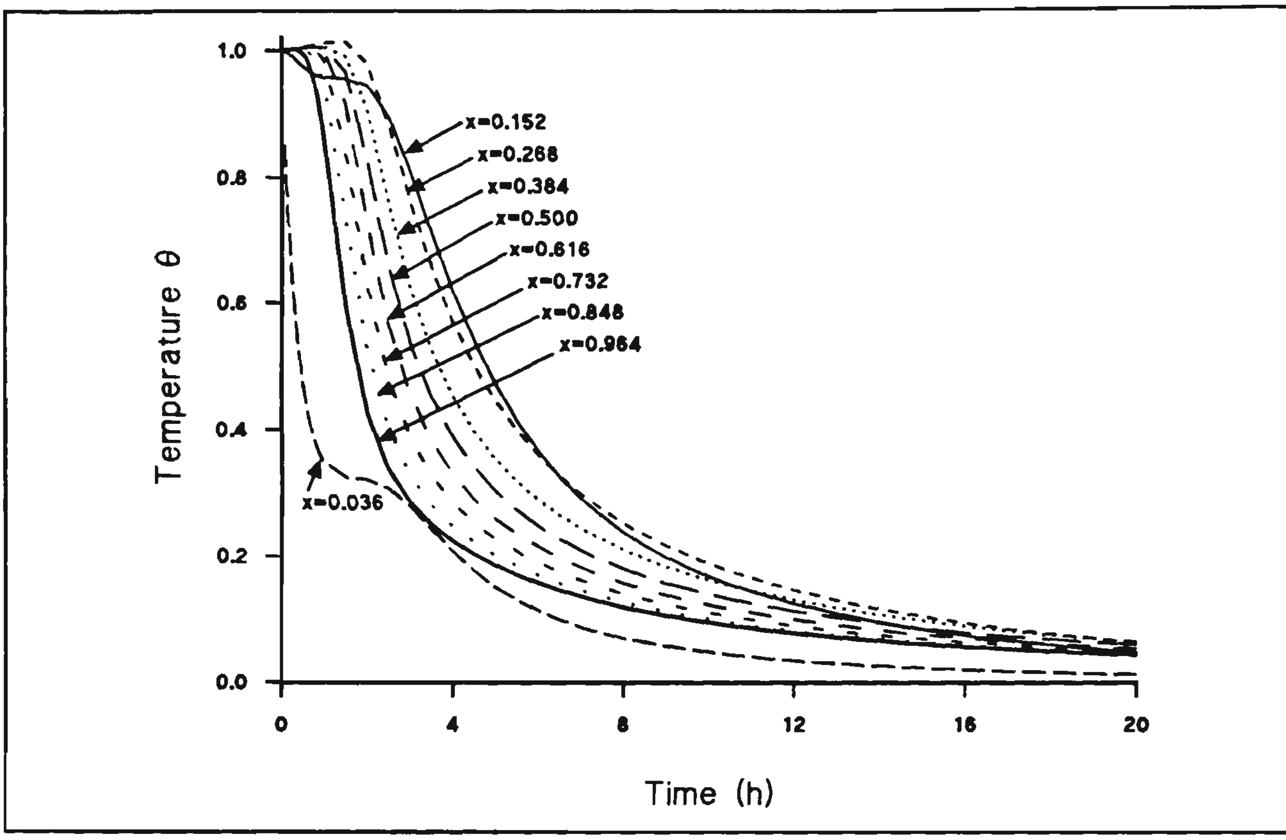


Figure 6.1 Temperature change with time along vertical central axis. Apples, adiabatic floor.

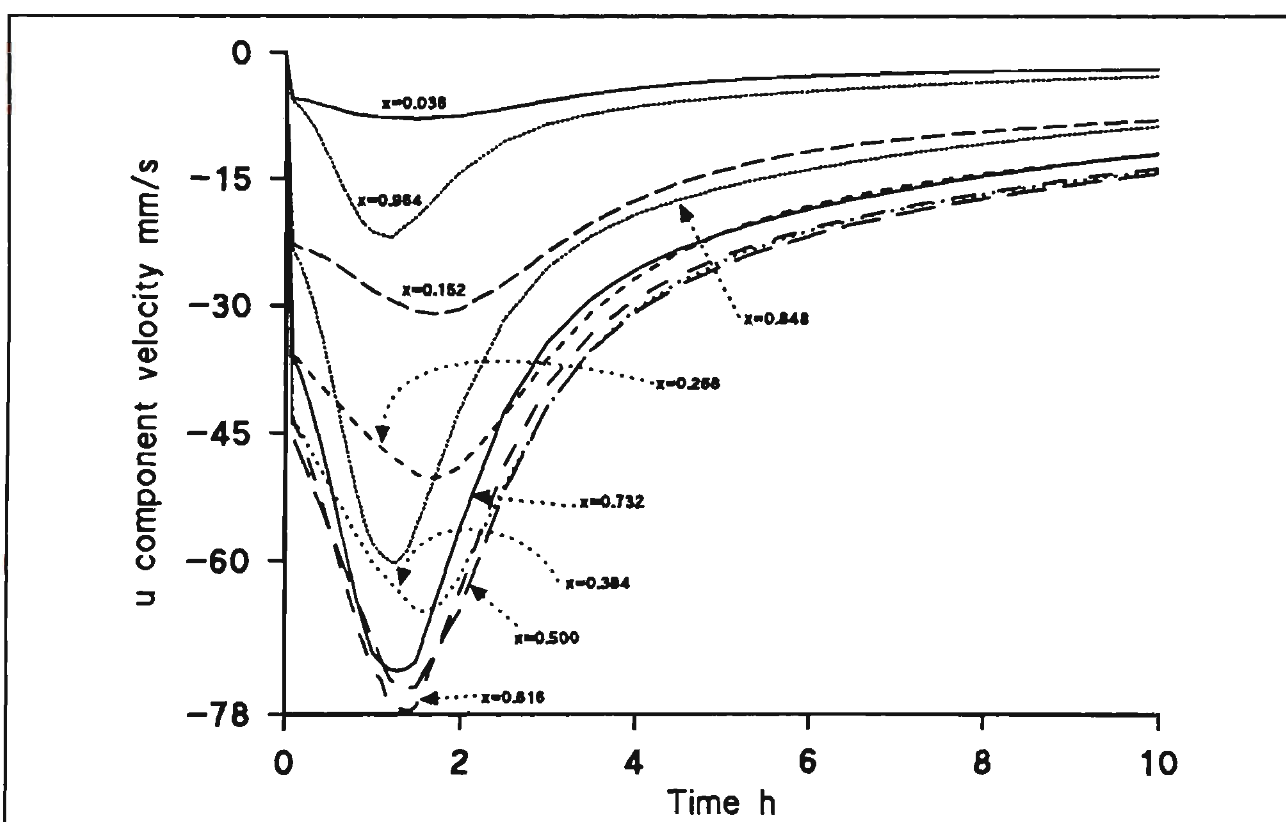


Figure 6.2 The x-component of velocity, u, versus time along the vertical central axis. Apples, adiabatic floor.

Figure 6.2 shows how the *x* component of velocity, *u*, changes along the vertical central axis with time. It apppears that strong air movement started about one hour after cooling commenced and reached its maximum after about 1.5 hours. The velocity at different positions reaches its peak value at different times. In the lower part of the container the velocity reached its maximum value before that in the upper part. The time delay between x=0.964 and x=0.152 was approximately 30 to 40 minutes.

Figures 6.1 and 6.2 indicate that high temperature gradients were associated with strong air movement inside the box. The *x* component velocity distribution near the top and bottom walls does not show sharp changes and the average value was relatively small compared to that in the central area.

Figures 6.3, 6.4 and 6.5 show the temperature distributions along three central axes of the box during the cooling process. Figures 6.6, 6.7 and 6.8 show the the *x* component velocity distributions along the three central axes. Because the system is symmetric in the $y=0.5y_0$ and $Z=0.5Z_0$ planes, the temperature and x component velocity are also symmetric along the Y and Z axes.

As shown in Figures 6.7 and 6.8, the downward air movements were limited to a narrow layer near the cold walls. Between the upward air flow and downward flow, there was a narrow zone where the velocity *u* was zero. It seems that the zero velocity *u* zone does not change position much. The absolute velocity values near the side walls were greater than those in the centre.

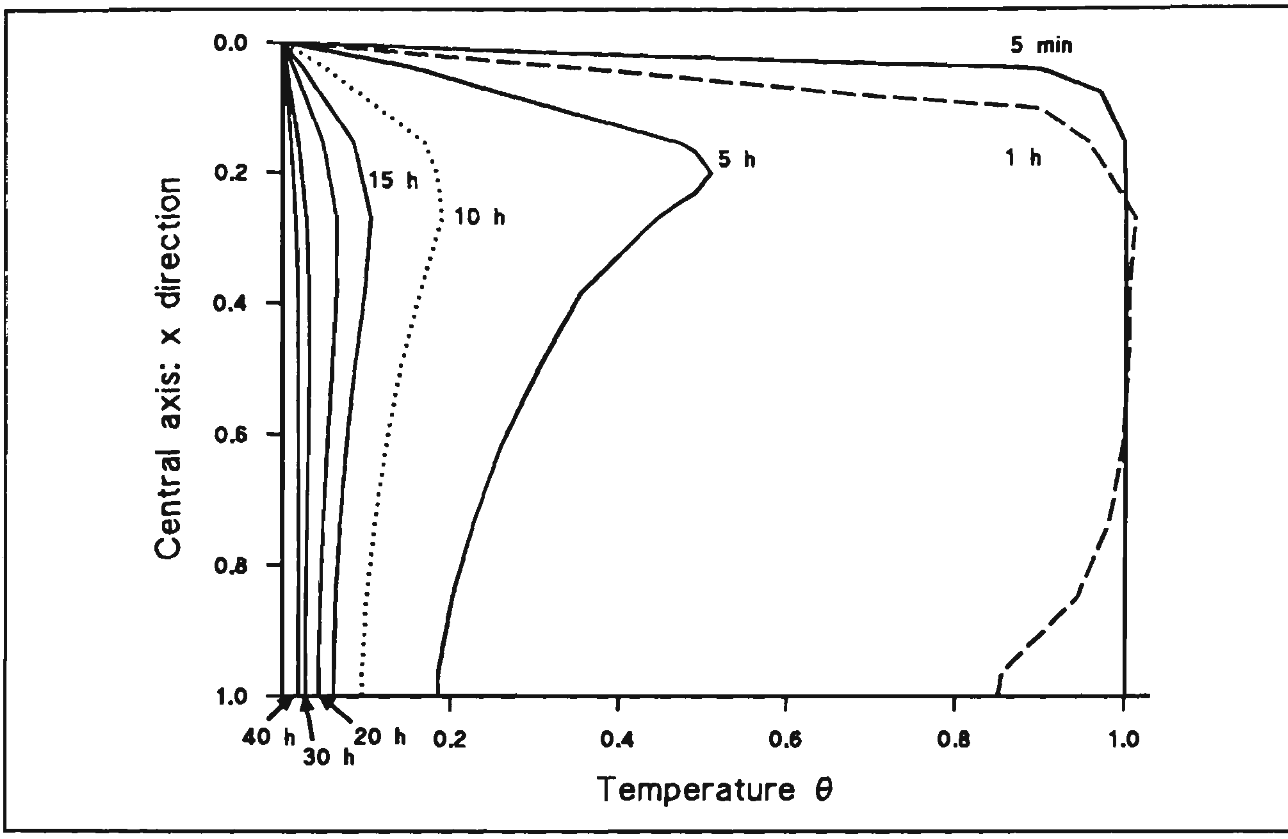


Figure 6.3 Temperature distribution at different cooling times along the central x axis. Apples, adiabatic floor.

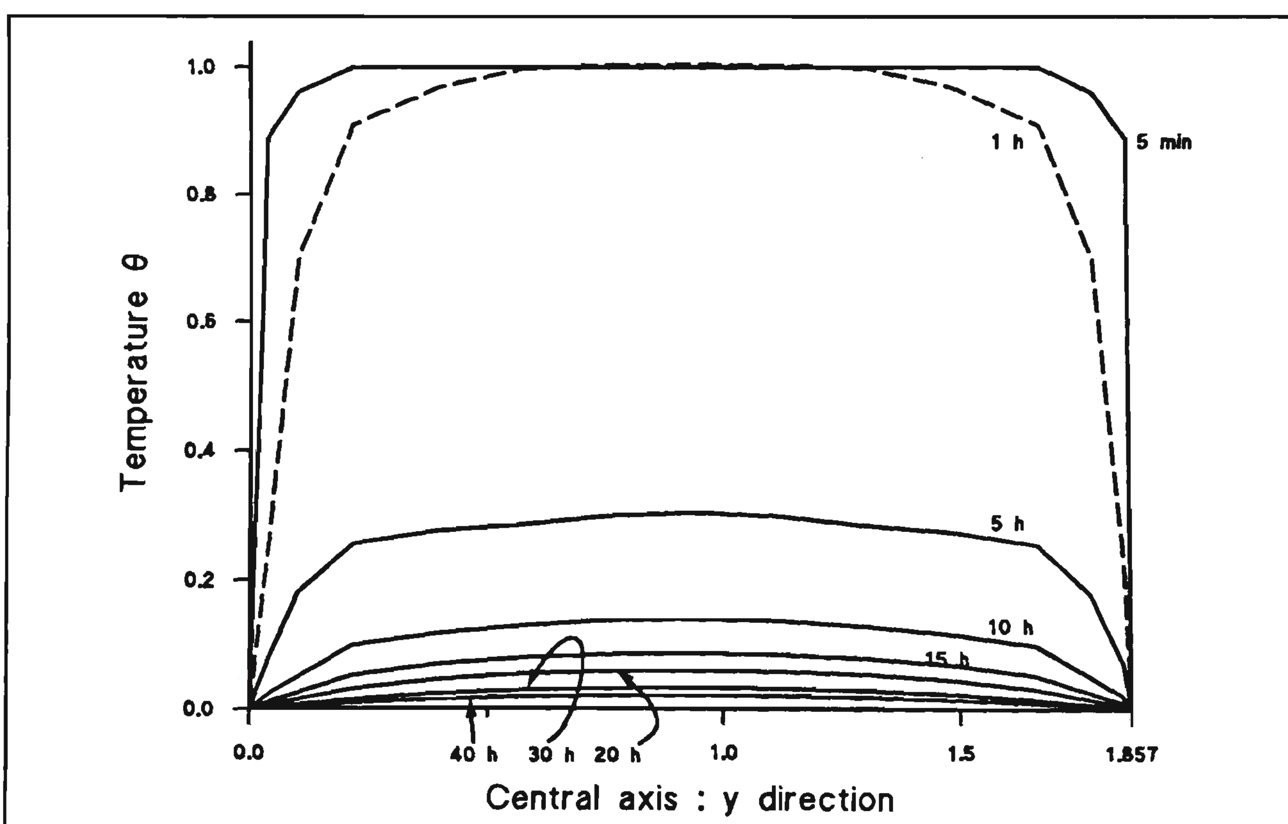


Figure 6.4 Temperature distribution at different cooling times along the central y axis. Apples, adiabatic floor.

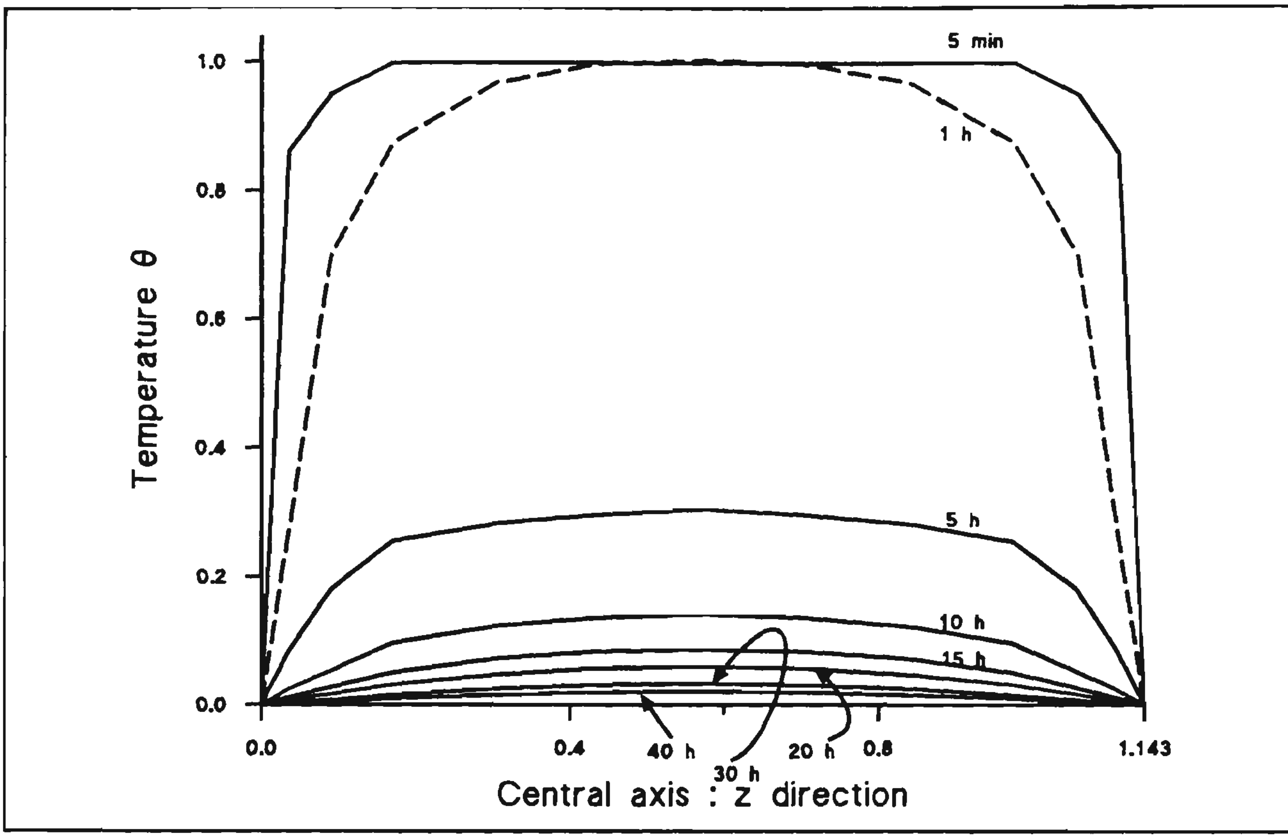


Figure 6.5 Temperature distribution at different cooling times along the central z axis. Apples, adiabatic floor.

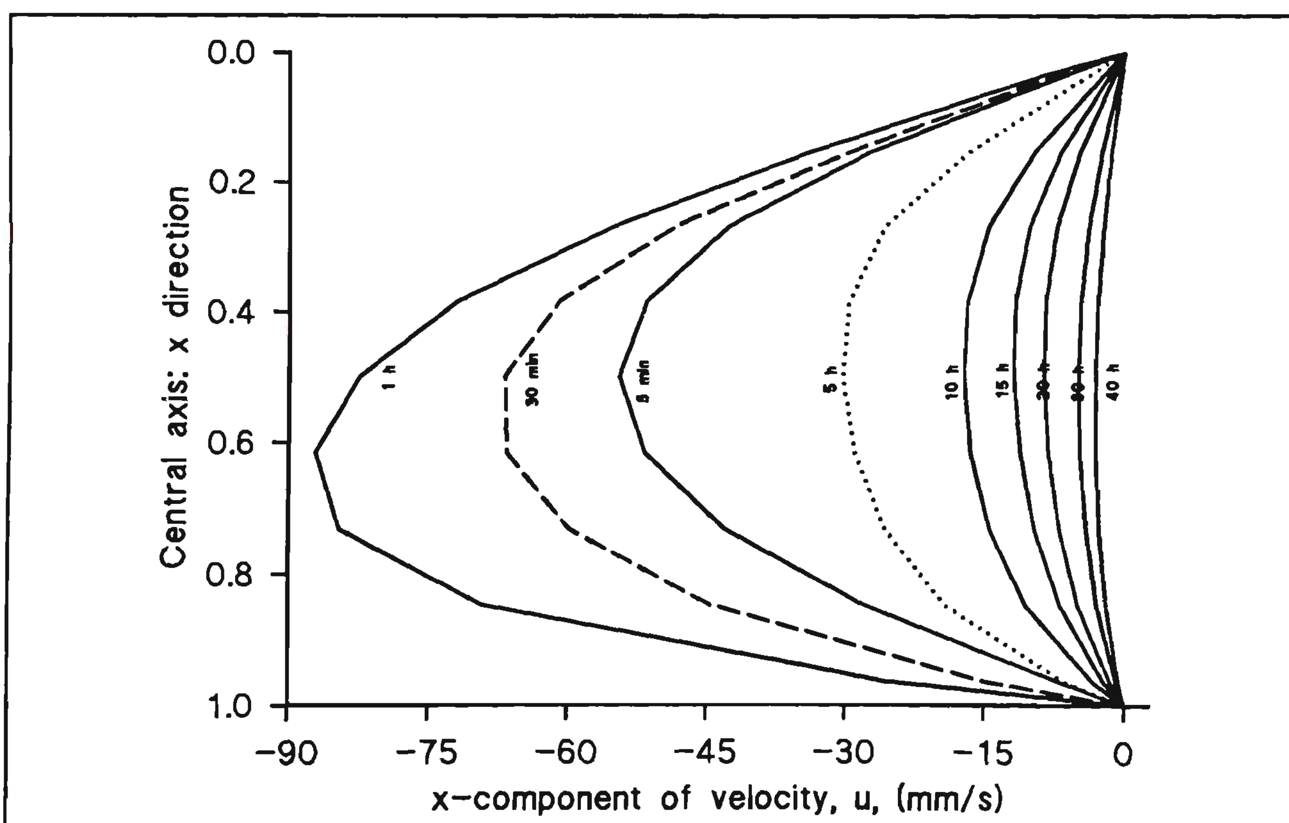


Figure 6.6 x component of velocity, u, distribution at different cooling times along the central x axis. Apples, adiabatic floor.

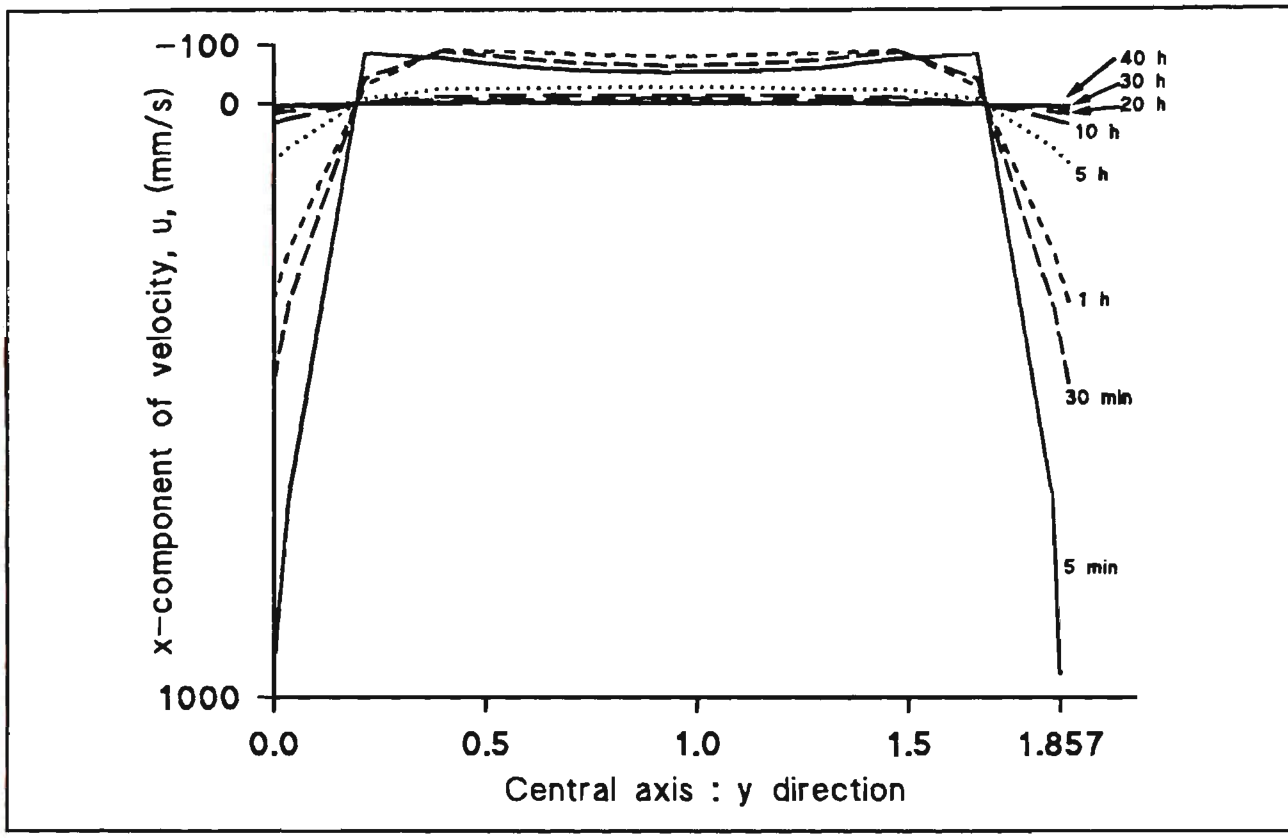


Figure 6.7 x component of velocity, u, distribution at different cooling times along the central y axis. Apples, adiabatic floor.

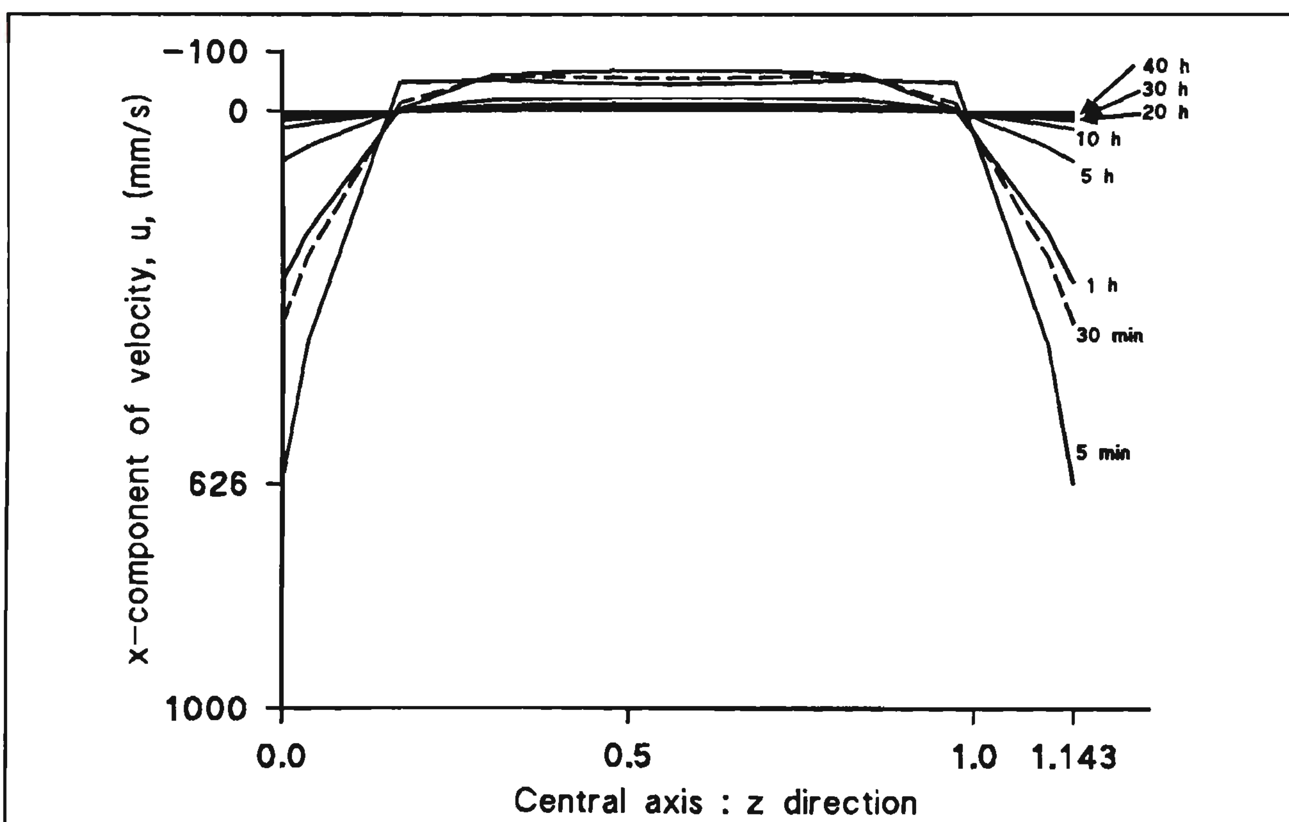


Figure 6.8 x component of velocity, u, distribution at different cooling times along the central z axis. Apples, adiabatic floor.

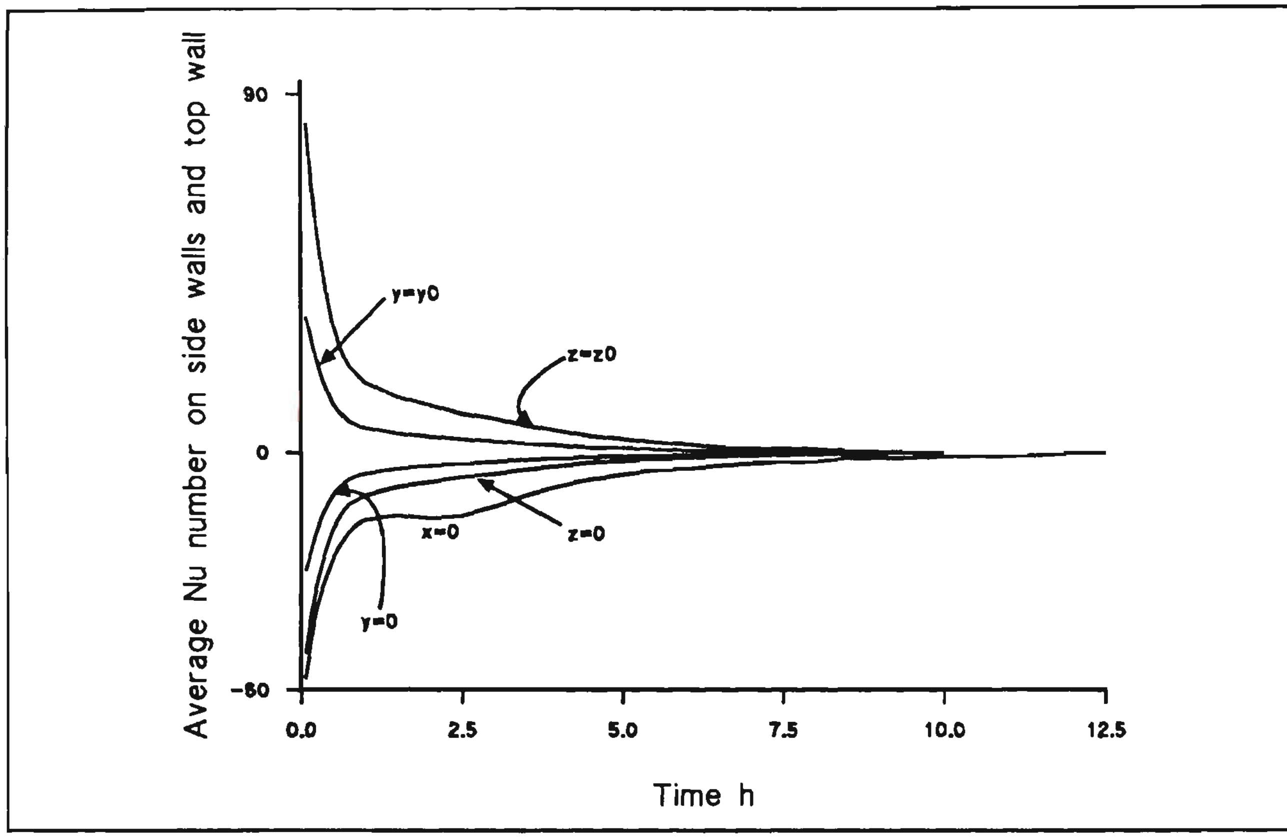


Figure 6.9 Nu number change with time on top and side walls. Apples, adiabatic floor.

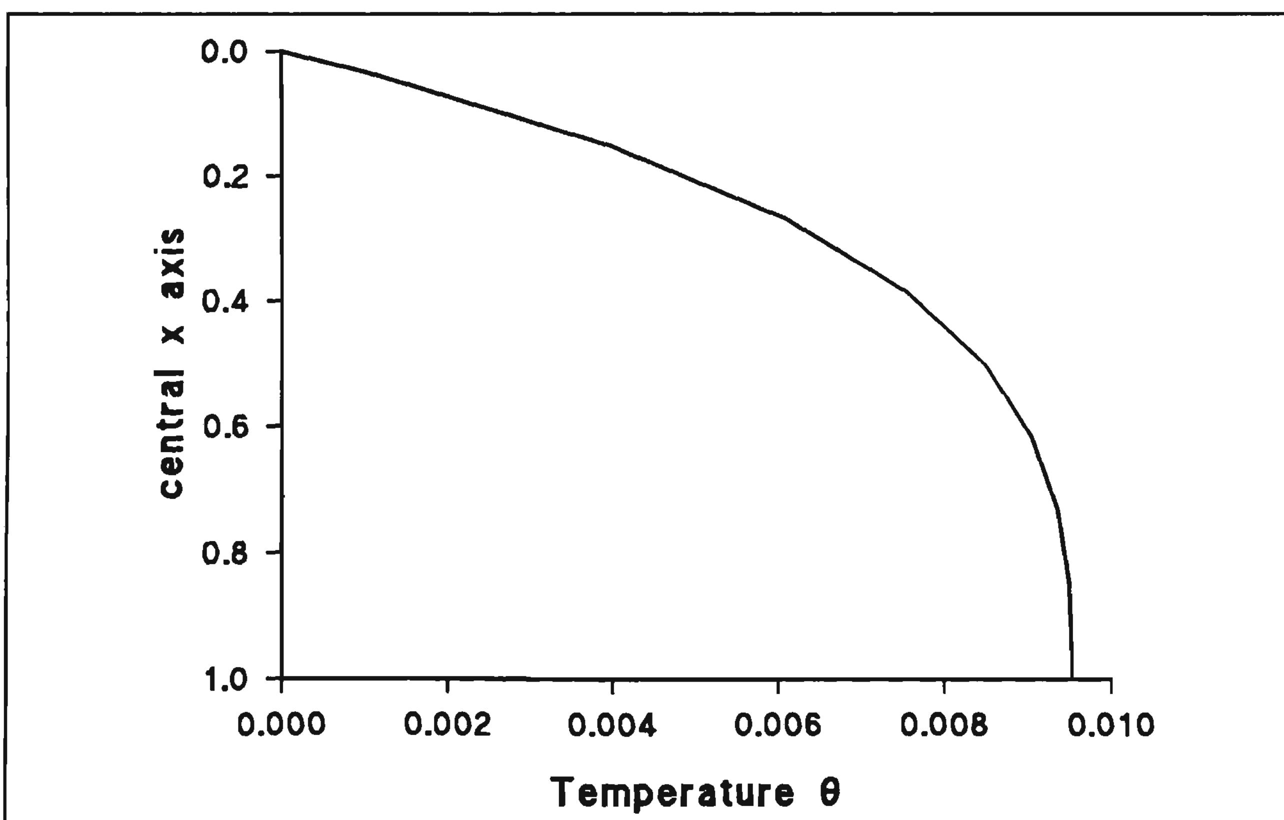


Figure 6.10 Steady state temperature distribution along the central x axis. Apples, adiabatic floor.

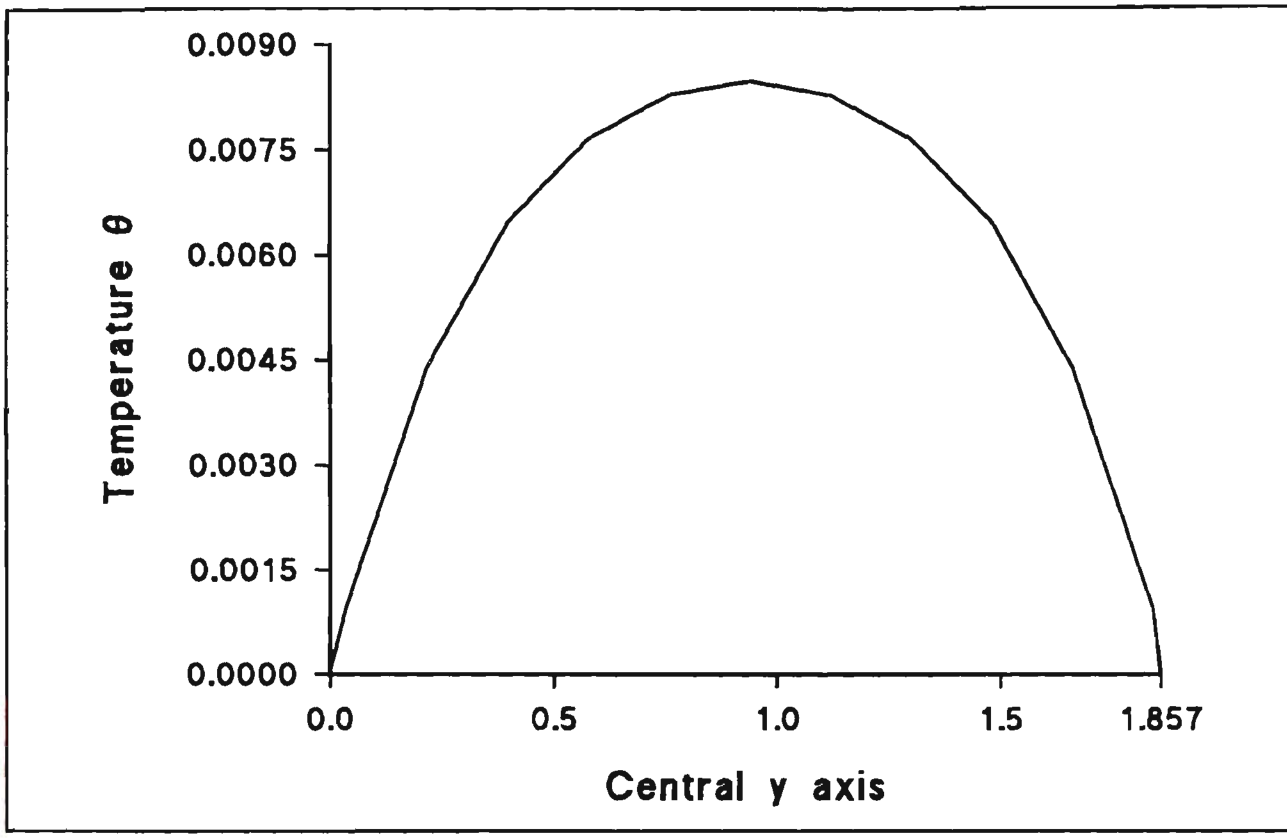


Figure 6.11 Steady state temperature distribution along the central y axis. Apples, adiabatic floor.

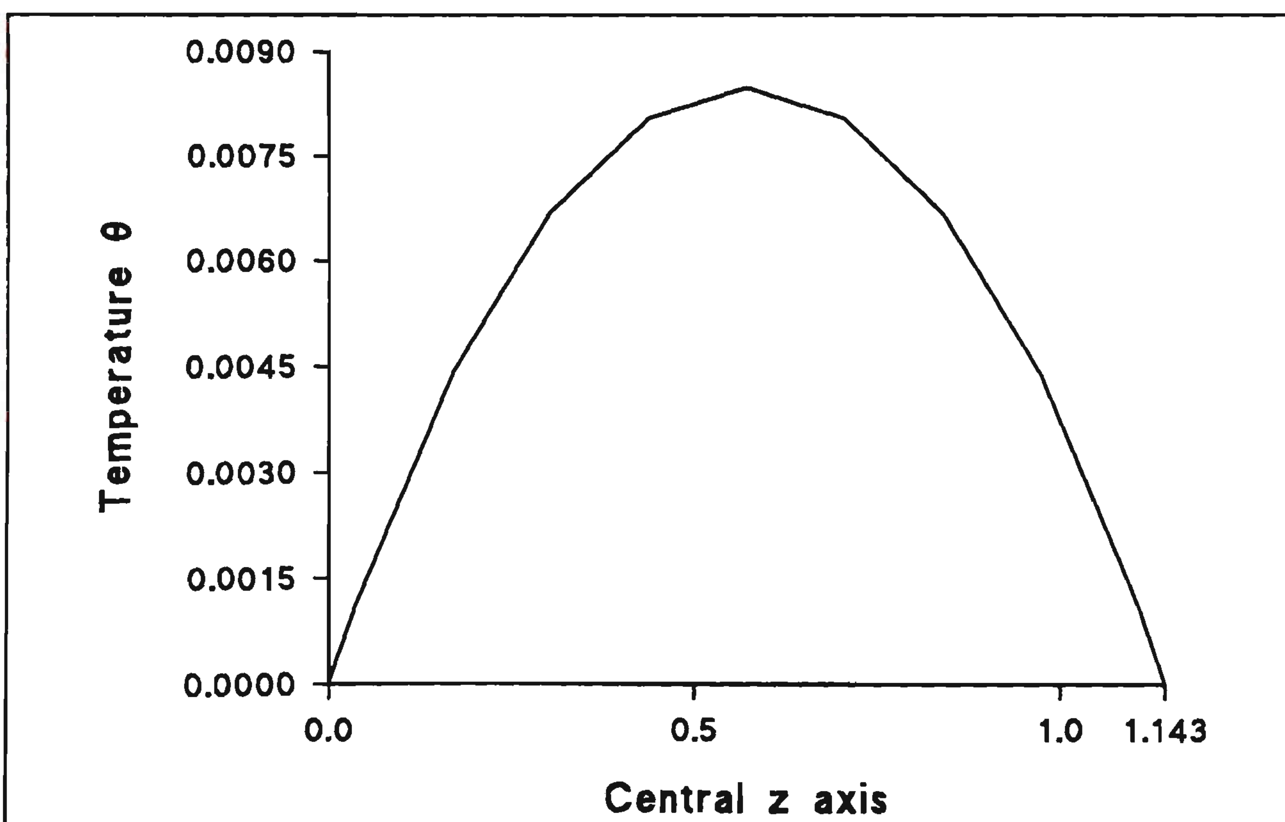


Figure 6.12 Steady state temperature distribution along the central z axis. Apples, adiabatic floor.

The average Nusselt number change with time on the side walls and the top wall is presented in Figure 6.9. In the first two hours of cooling, the Nu number had much larger values than in later stages of the process. This sugests that beyond three hours, conduction plays a more dominant role than convection in heat transfer.

Figure 6.10, 6.11, and 6.12 show the temperature distributions along the central x, y and z axes after steady state conditions were reach. The highest temperature zone was at the bottom of the box, with temperature reaching θ=0.009537 or T= 0.286^{0}C. The temperature in the central part of the box was $\theta \approx 0.008485$ or T=0.255^{0}C. At the steady state temperature, the velocity was very small and the influence of convection was weak since heat transfer was largely due to conduction. Appendix C contains the contour map of the local temperature distribution, local velocity distribution, vector potential and local Nusselt number distribution during the cooling process and at steady state.

Brussels sprouts were seleted to illustrate the ability of the program to deal with commodities characterised by high respiration rates. Brussels sprouts can be kept in good condition for a maximum period of 3 to 5 weeks at 0°C, according to Hardenbury (1986). The computed results of the developing temperature and velocity fields with respiratory heat generation for Brussels sprouts with adiabatic floor conditions are shown in Figures 6.13 to 6.24. The temperature and x-component of velocity, u, along the central vertical axis are shown in Figures 6.13 and 6.14 respectively. It is evident that a much greater time was needed to cool the sprouts than was the case for apples. The temperature distributions within the container along the central axes for the x,y and z coordinates, as it develops in time, are shown in Figure 6.15, 6.16 and 6.17.

The development of the velocity fields with time are illustrated in Figures 6.18, 6.19 and 6.20, where the x component of velocity distributions for the x,y and z coordinates along the central axes are shown. Figures 6.21, 6.22 and 6.23 depict the steady state temperature distributions along the central x,y and z cordinates respectively. The average Nusselt number change with time on the side walls and the top wall is presented in Figure 6.24.

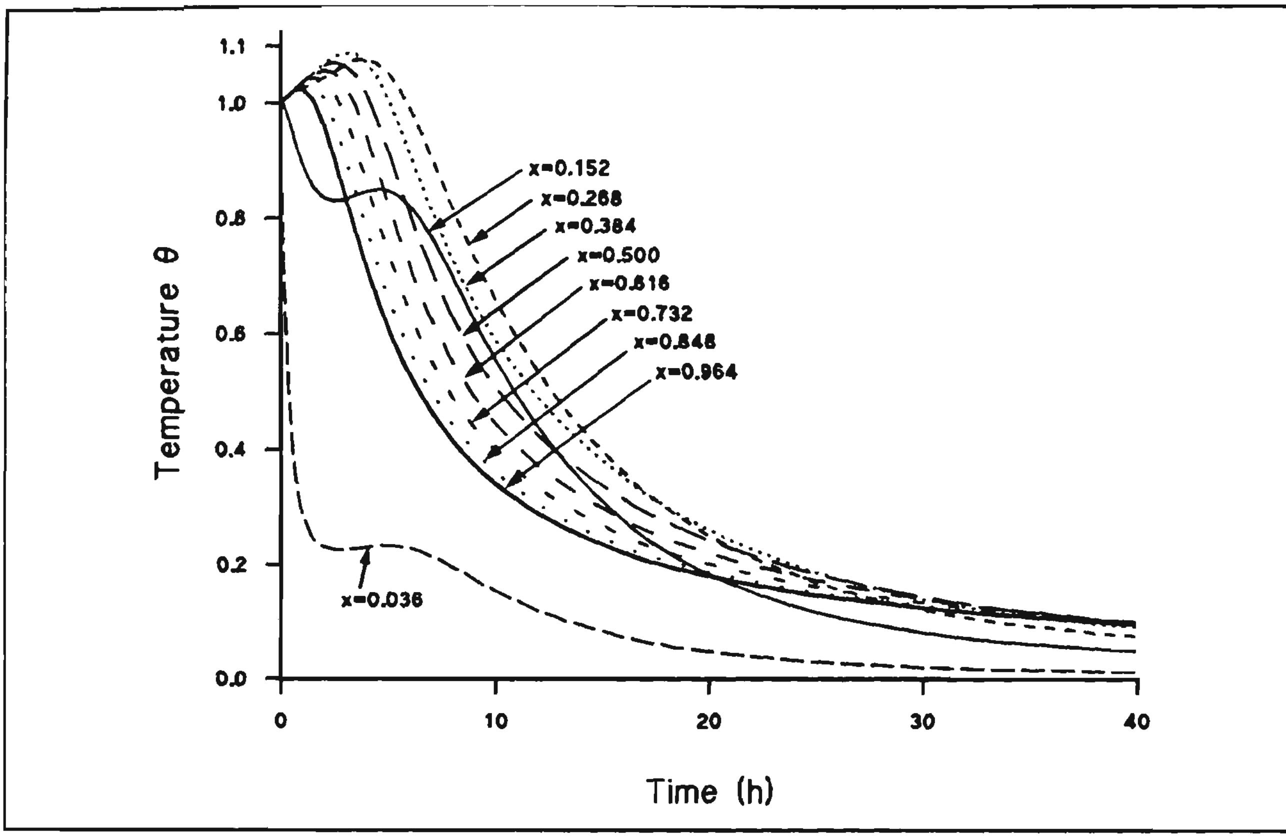


Figure 6.13 Temperature change with time along the vertical central x axis. Brussels sprouts, adiabatic floor.

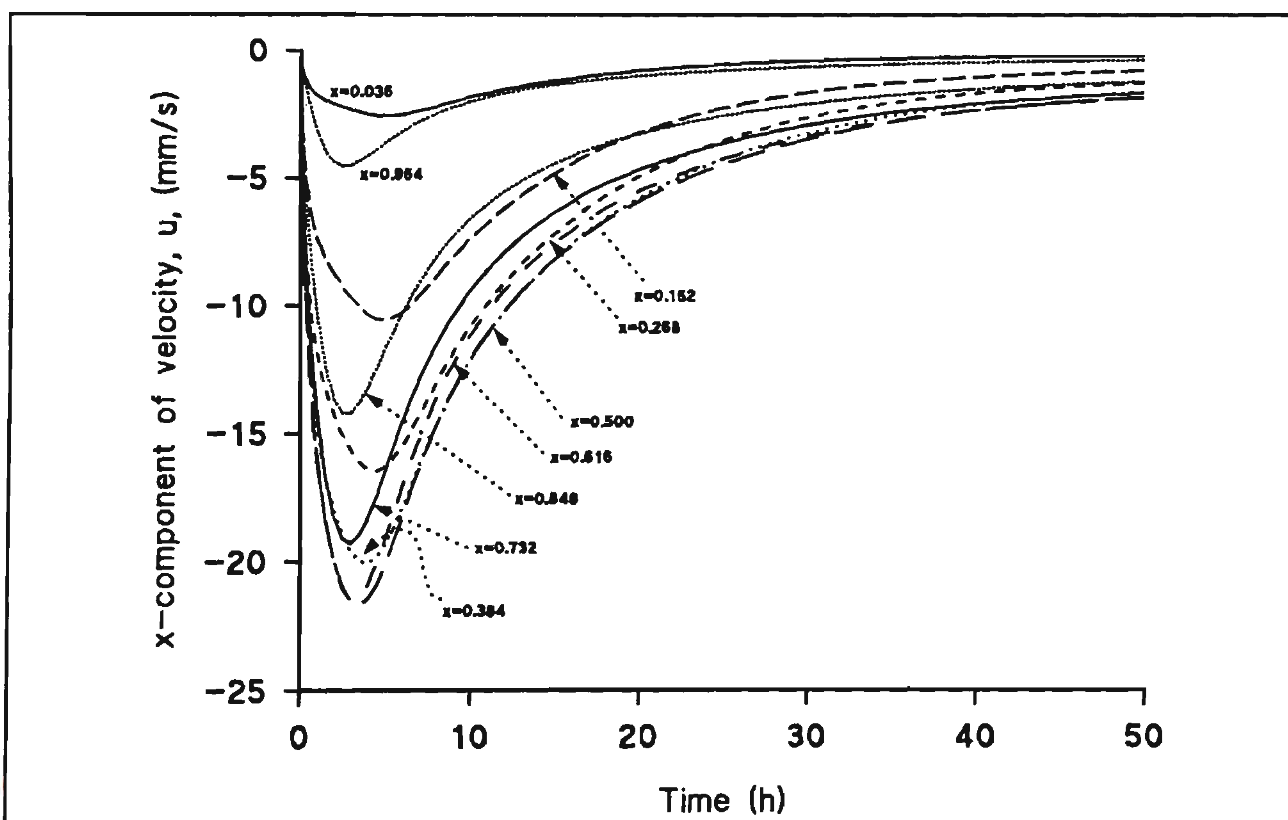


Figure 6.14 x-component of velocity, u, change with time along the vertical x central axis. Brussels sprouts, adiabatic floor.

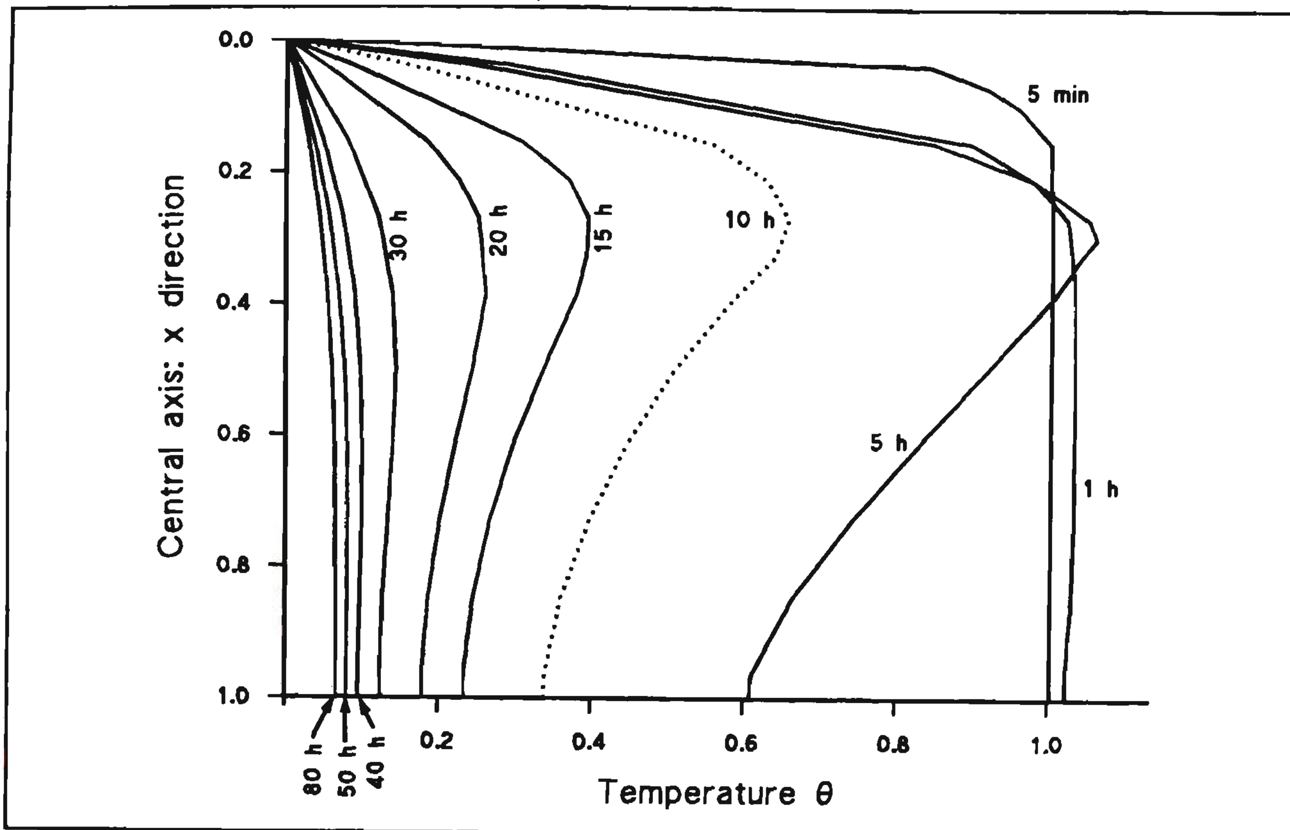


Figure 6.15 Temperature distribution at different cooling times along the central x axis. Brussels sprouts, adiabatic floor.

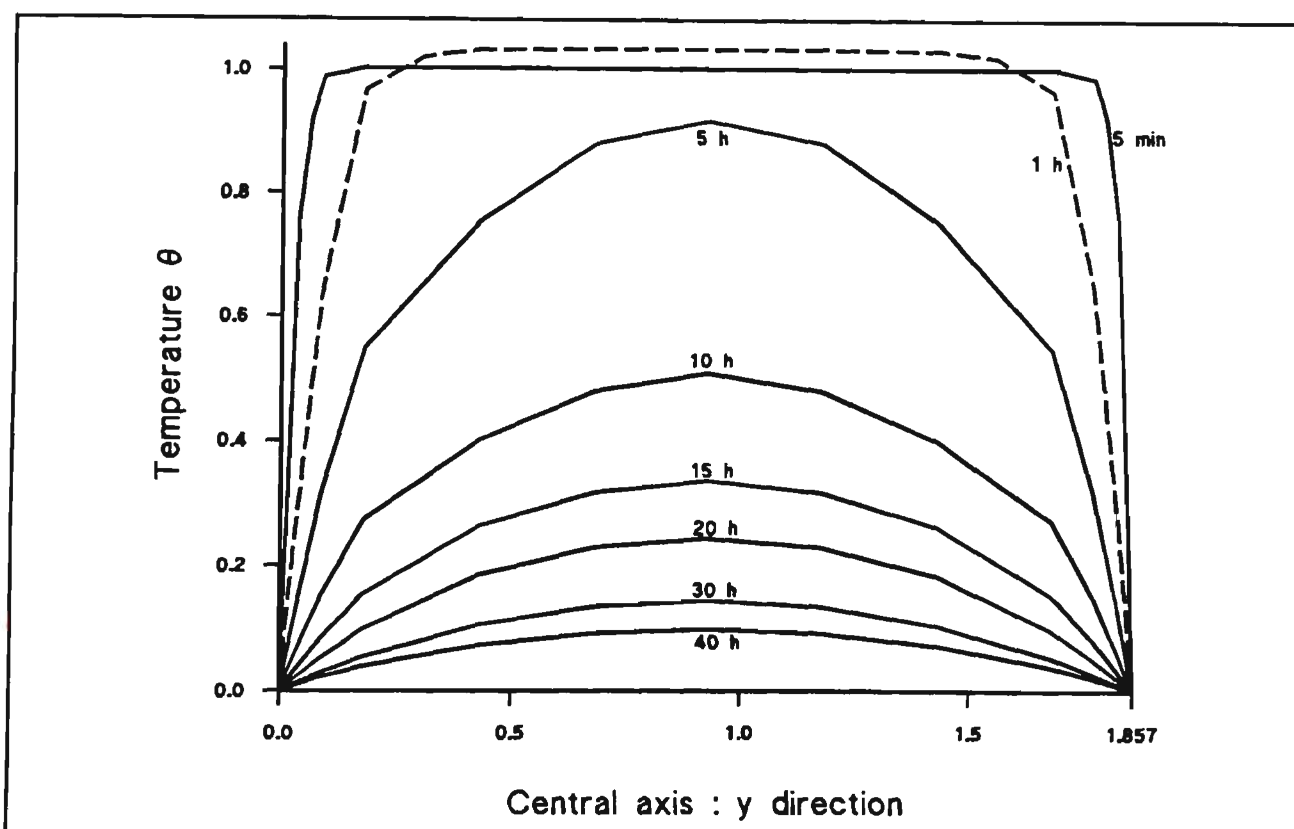


Figure 6.16 Temperature distribution at different cooling times along the central y axis. Brussels sprouts, adiabatic floor.

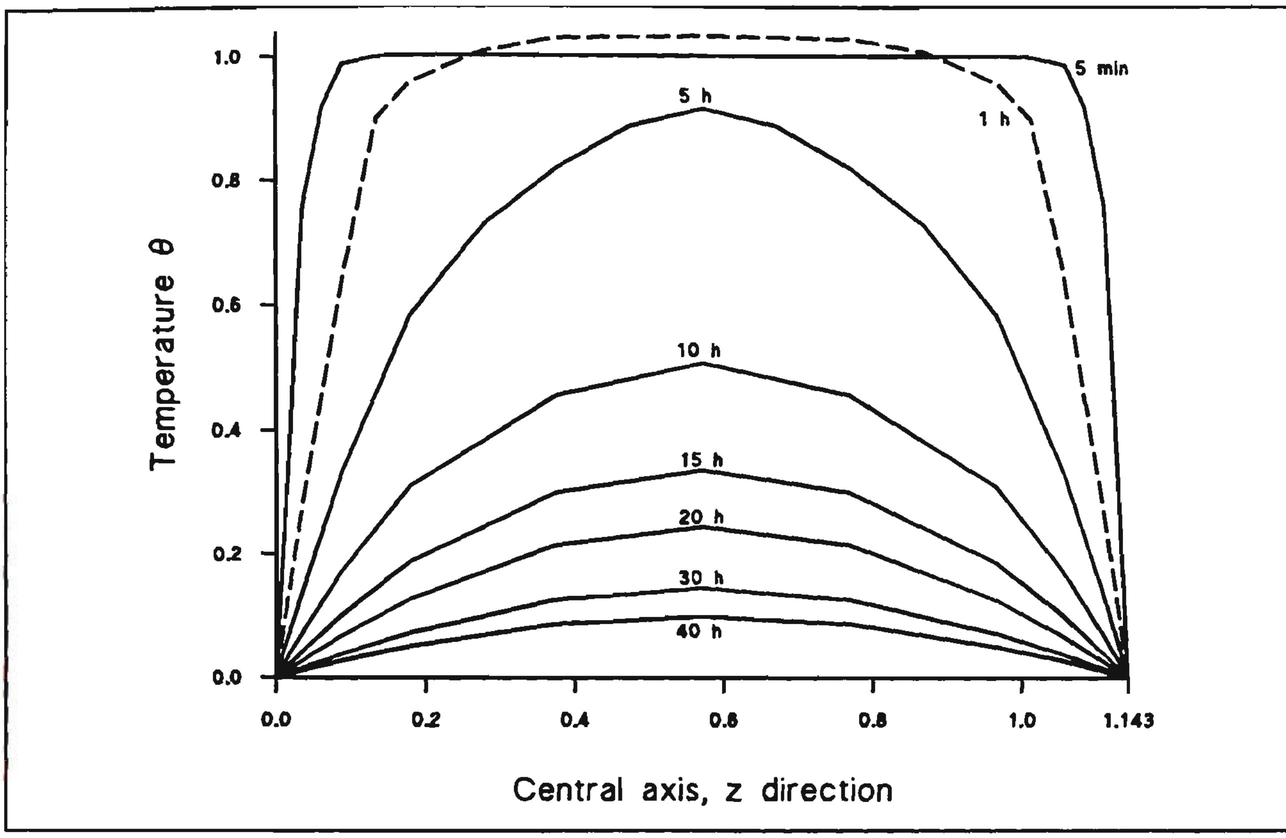


Figure 6.17 Temperature distribution at different cooling times along the central y axis. Brussels sprouts, adiabatic floor.

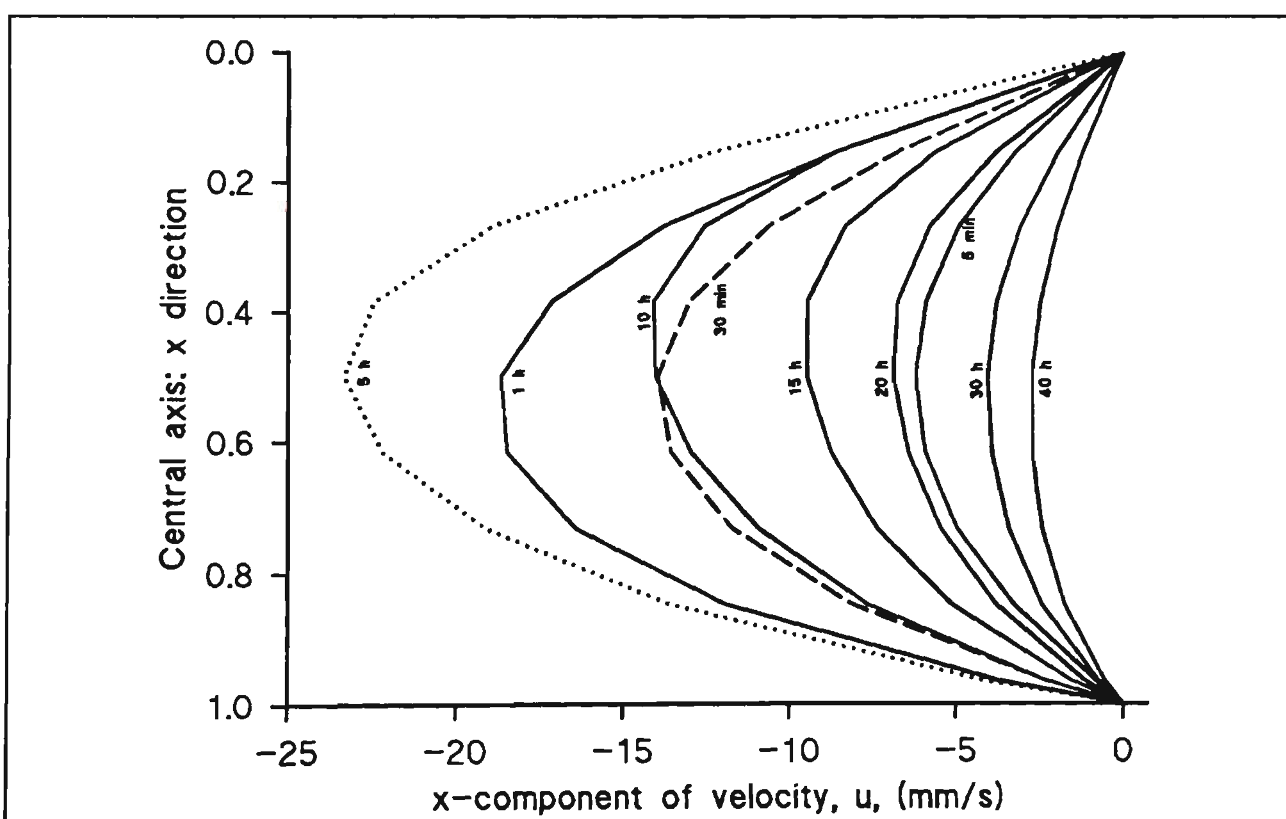


Figure 6.18 x-component of velocity, u, distribution at different cooling times along the central x axis. Brussels sprouts, adiabatic floor.

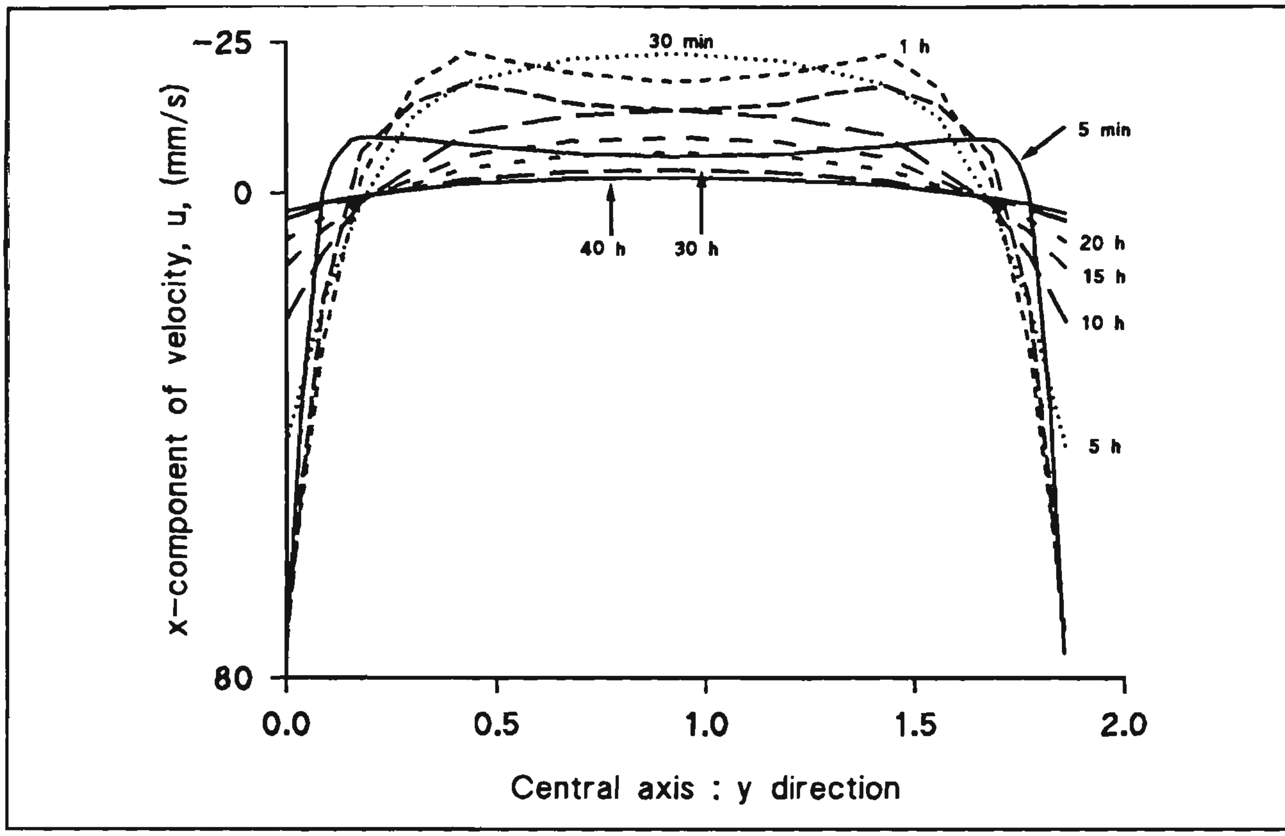


Figure 6.19 x-component of velocity, u, distribution at different cooling times along the central y axis. Brussels sprouts, adiabatic floor.

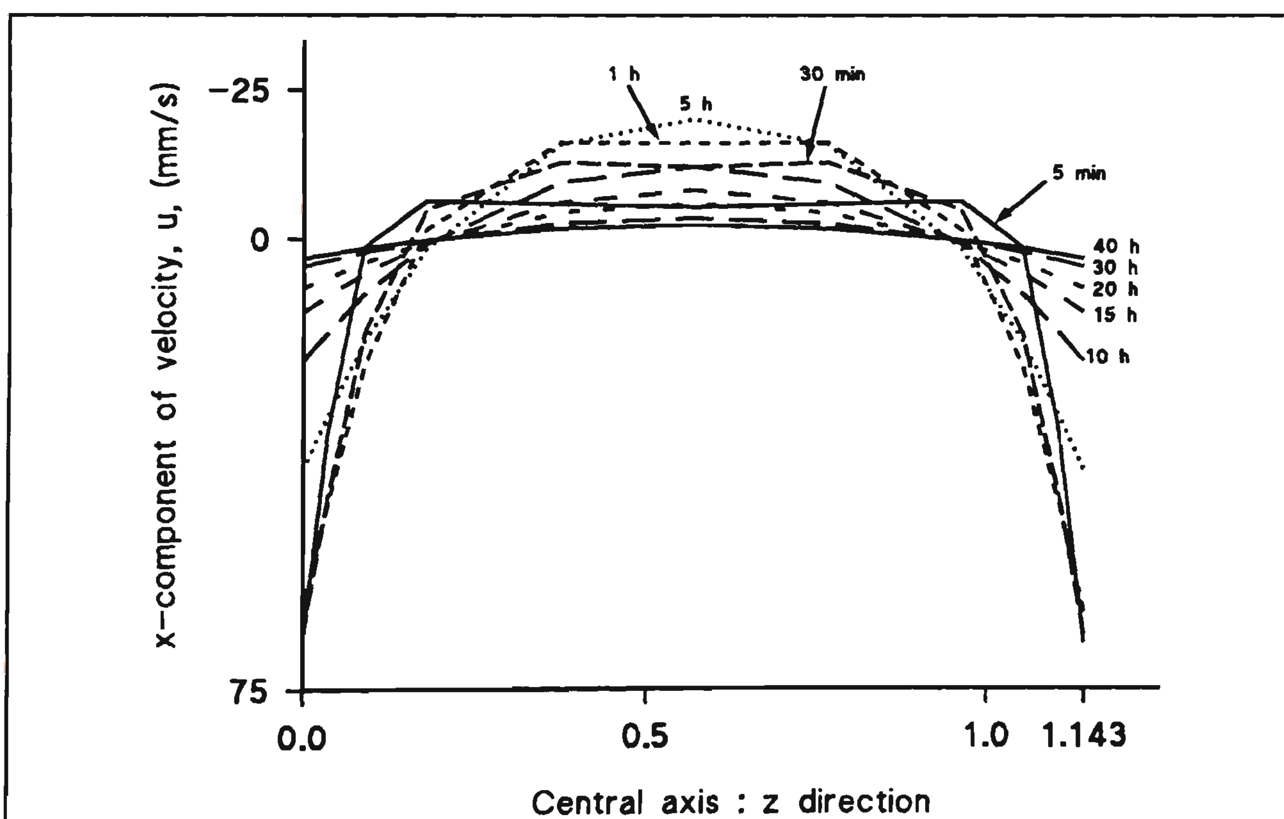


Figure 6.20 x-component of velocity, u, distribution at different cooling times along the central z axis. Brussels sprouts, adiabatic floor.

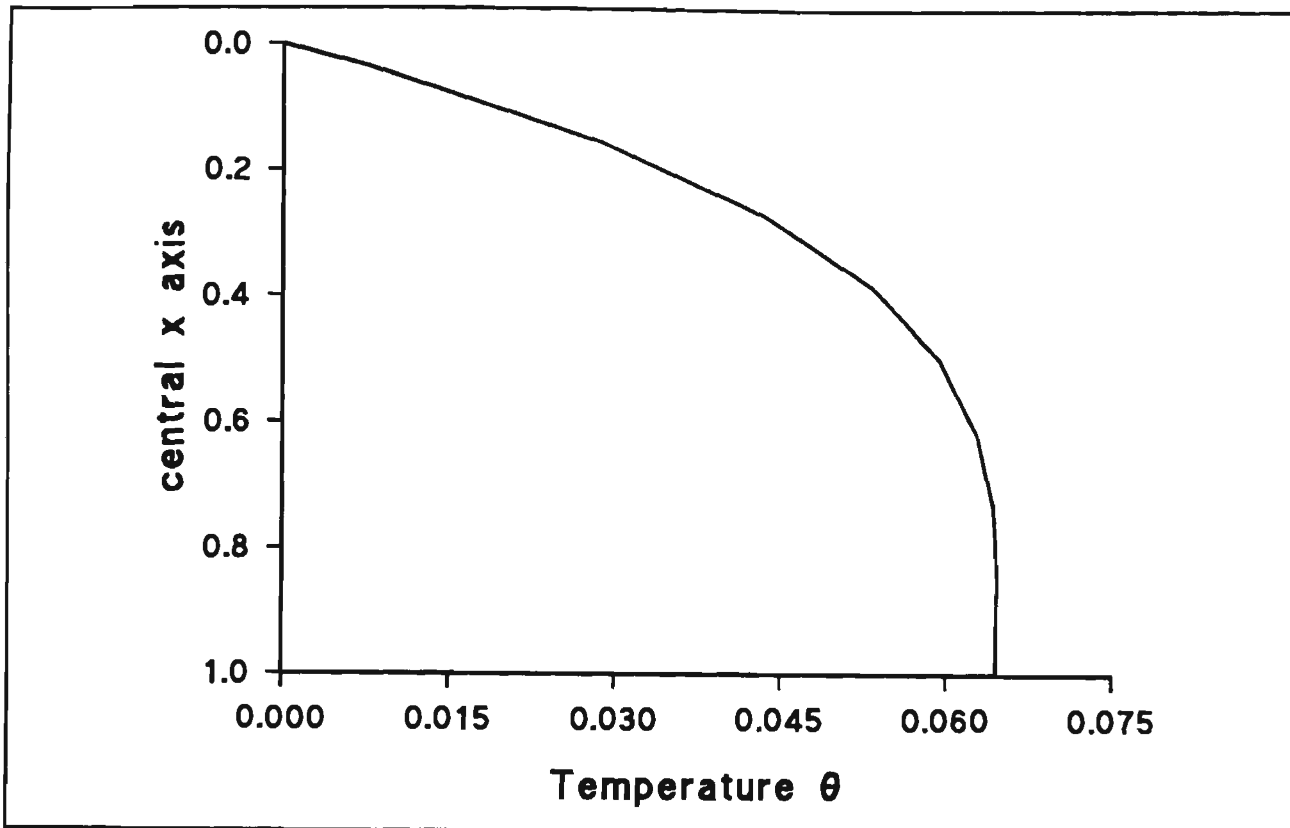


Figure 6.21 Steady state temperature distribution along the central vertical x axis. Brussels sprouts, adiabatic floor.

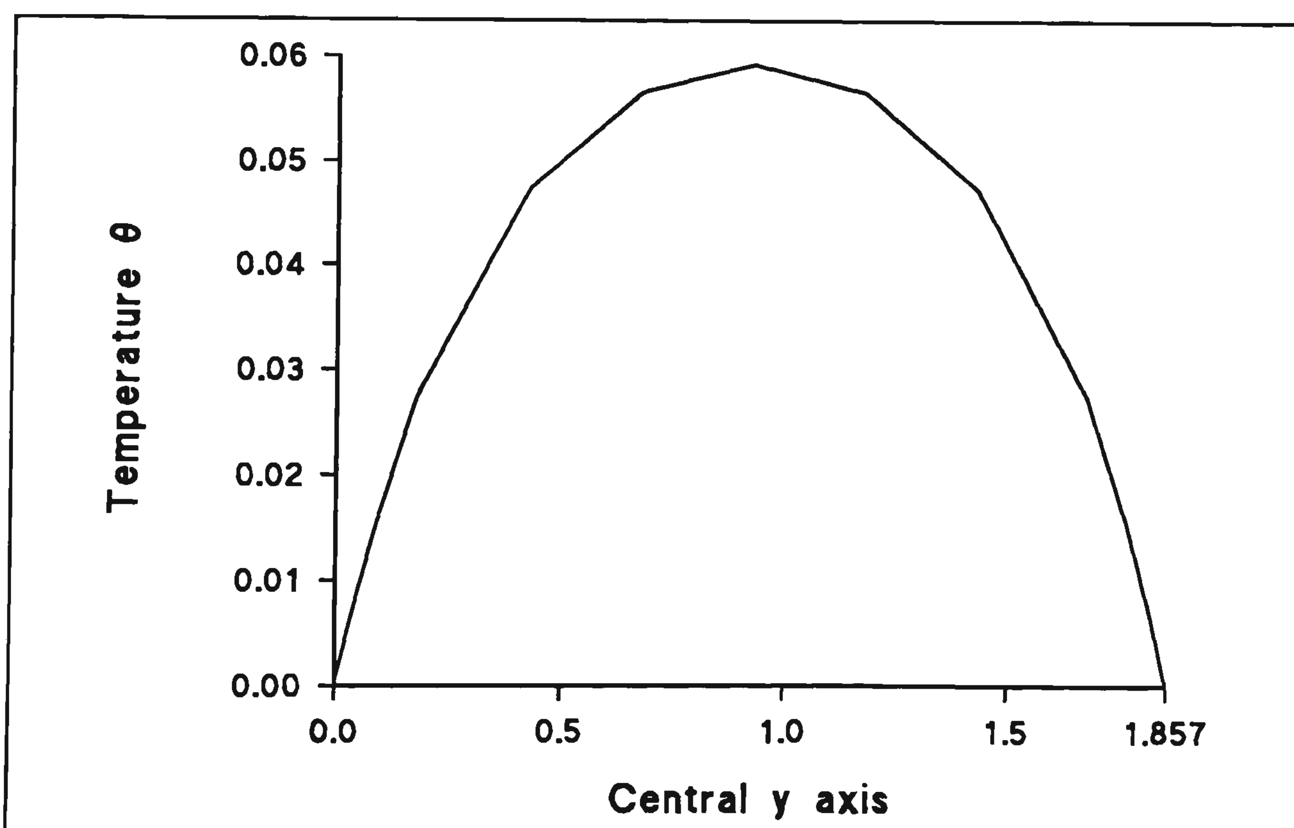


Figure 6.22 Steady state temperature distribution along the central horizontal y axis. Brussels sprouts, adiabatic floor.

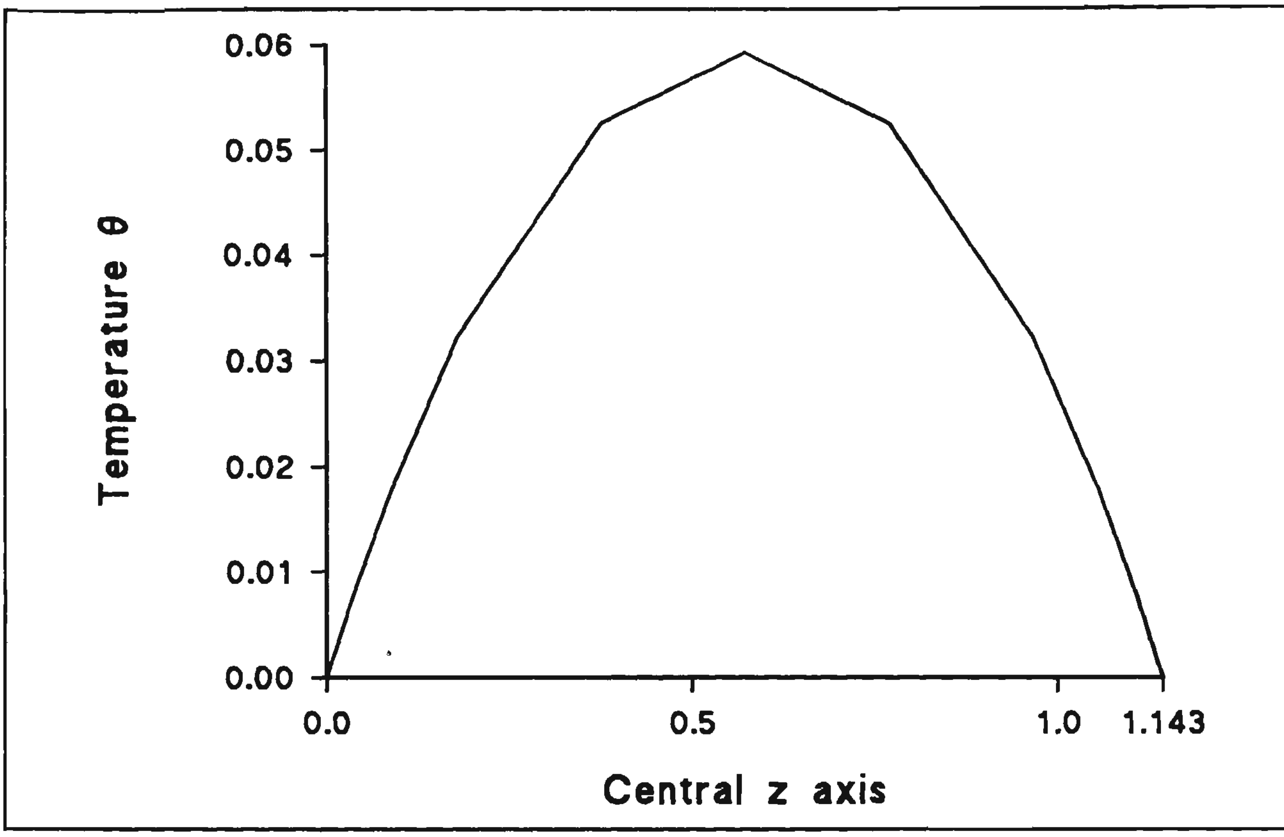


Figure 6.23 Steady state temperature distribution along the central z axis. Brussels sprouts, adiabatic floor.

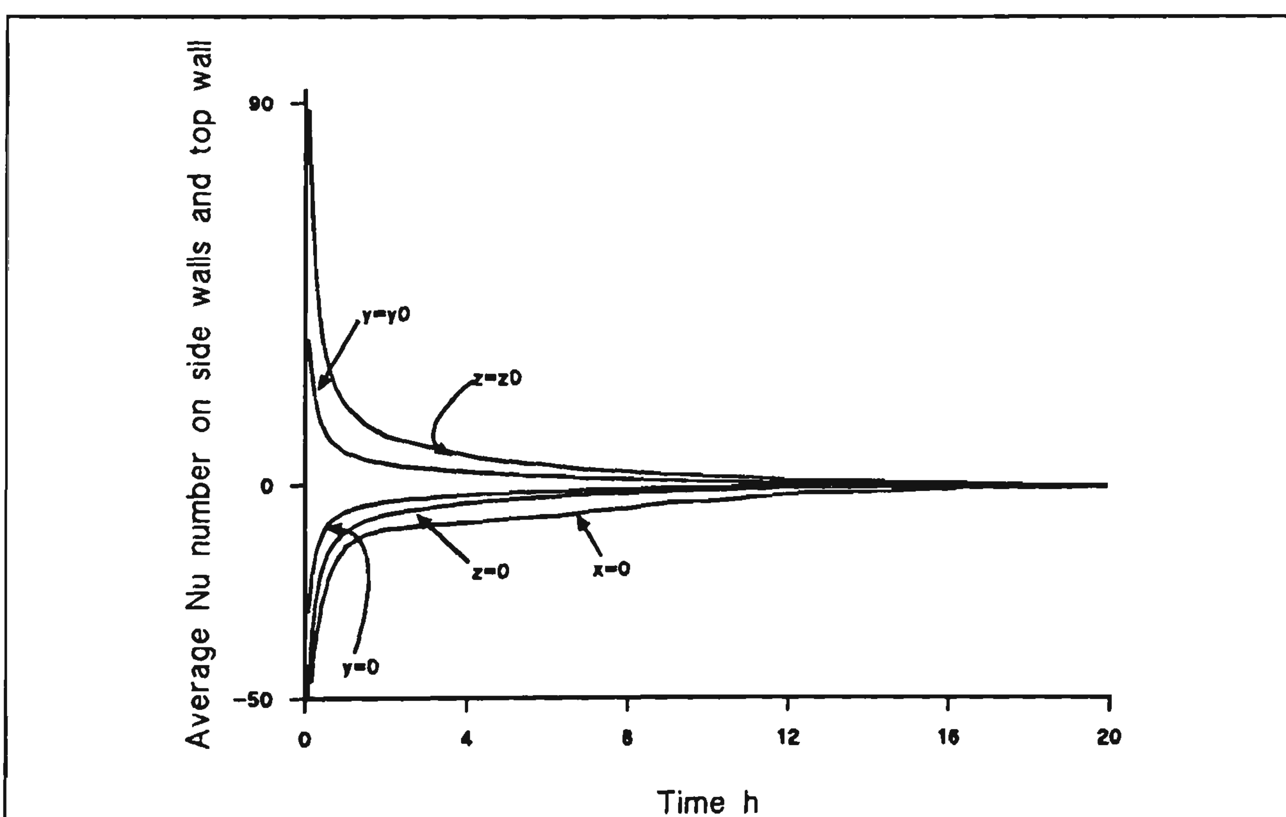


Figure 6.24 Nu number change with time on top and side walls. Brussels sprouts, adiabatic floor.

6.2.2 Natural Convection with an Isothermal Floor

The results computed for apples cooled under isothermal boundary conditions imposed on the floor of the storage container are presented below in Figures 6.25 to 6.36.

Progressive changes in the temperature destribution with time along the central x axis for apples is shown in Figure 6.25. After some 40 hours the temperature within the packed box reached steady state. Changes in the x-component of velocity, u, distribution with time along the central vertical axis are shown in Figure 6.26. Figures 6.27, 6.28 and 6.29 show the temperature distributions within the container along the central axes for the x,y and z coordinates, as it develops in time. Figures 6.30, 6.31 and 6.32 show the development of the x component of velocity with time along the central axes. The steady state temperature distributions along the central x,y and z directions are shown in Figures 6.33, 6.34 and 6.35 respectively. The average Nusselt number change with time on the side walls and the top wall is presented in Figure 6.36.

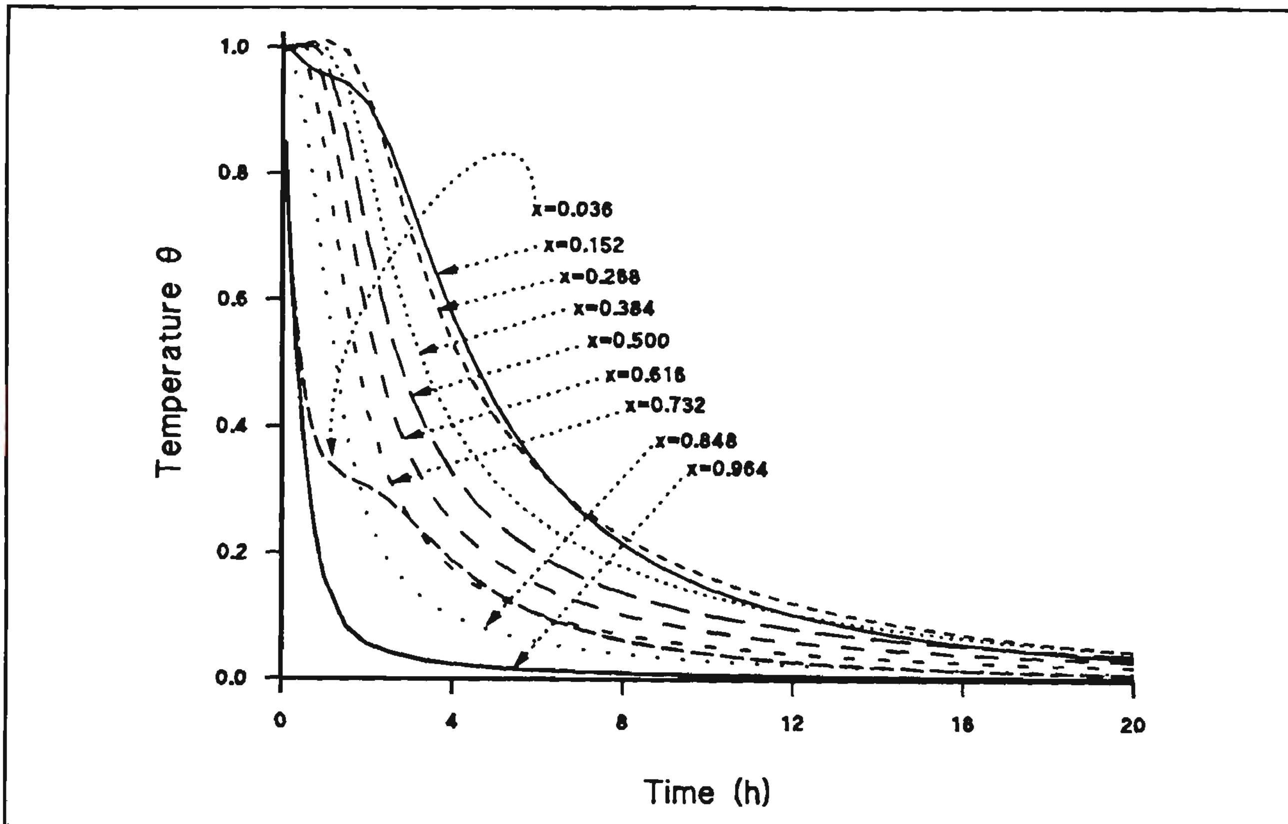


Figure 6.25 Temperature change with time along the vertical central axis. Apples, isothermal floor.

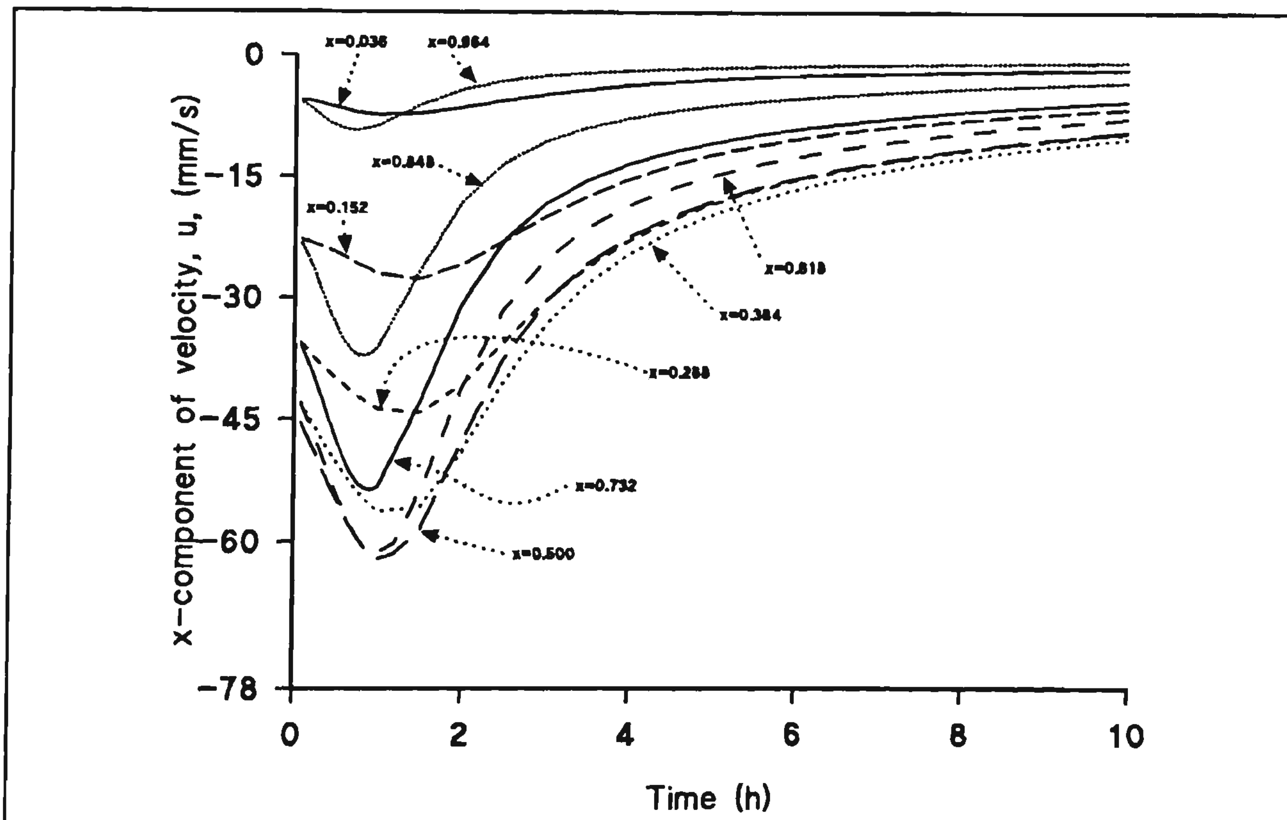


Figure 6.26 x-component of velocity, u, versus time along the vertical central axis. Apple, isothermal floor.

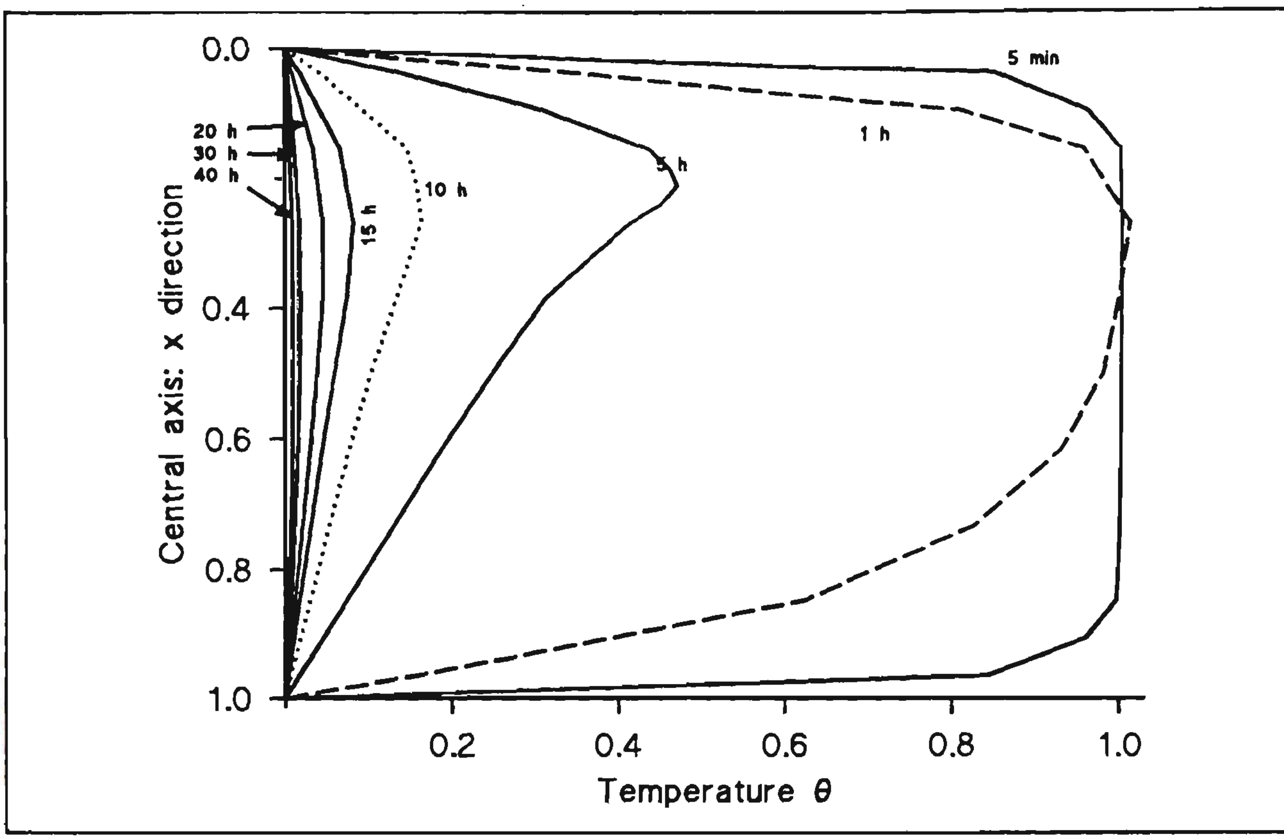


Figure 6.27 Temperature distribution at different cooling times along the central x axis. Apples, isothermal floor.

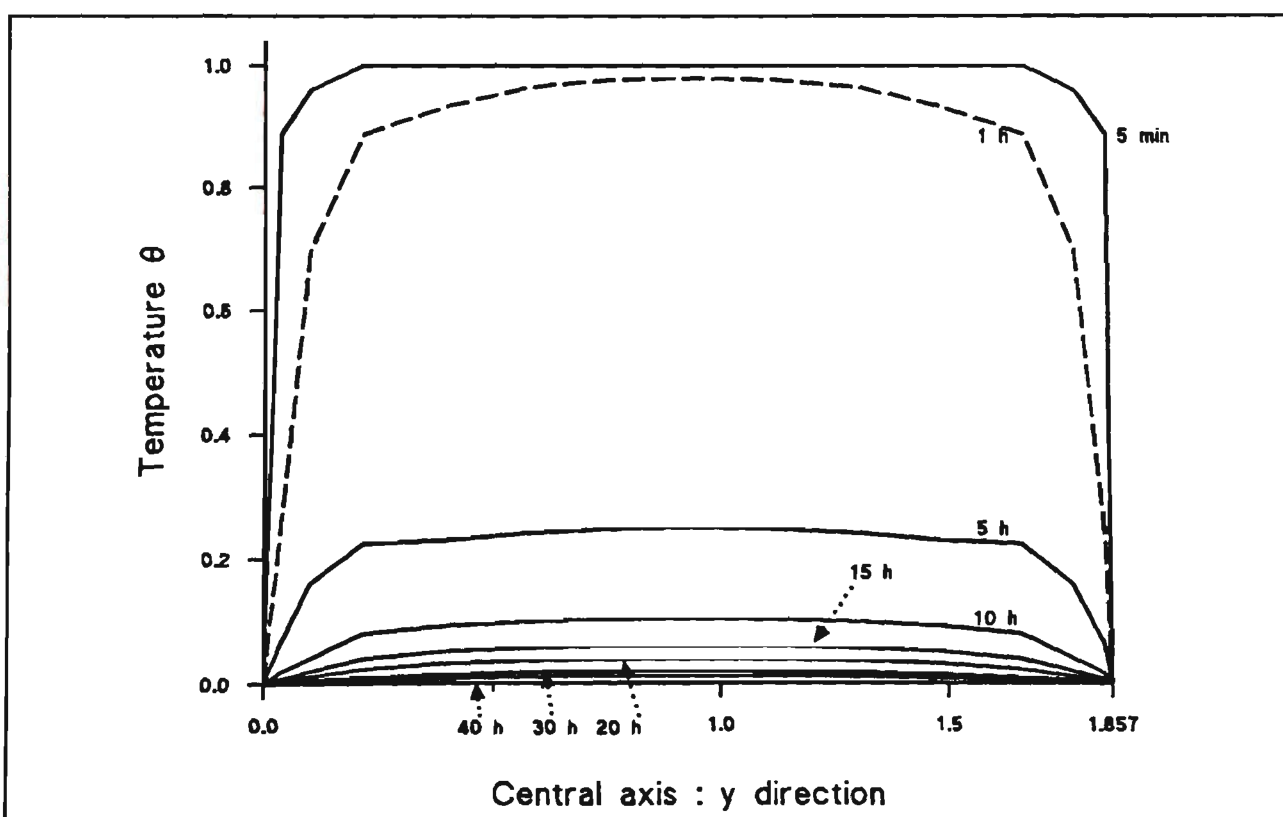


Figure 6.28 Temperature distribution at different cooling times along the central y axis. Apples, isothermal floor.

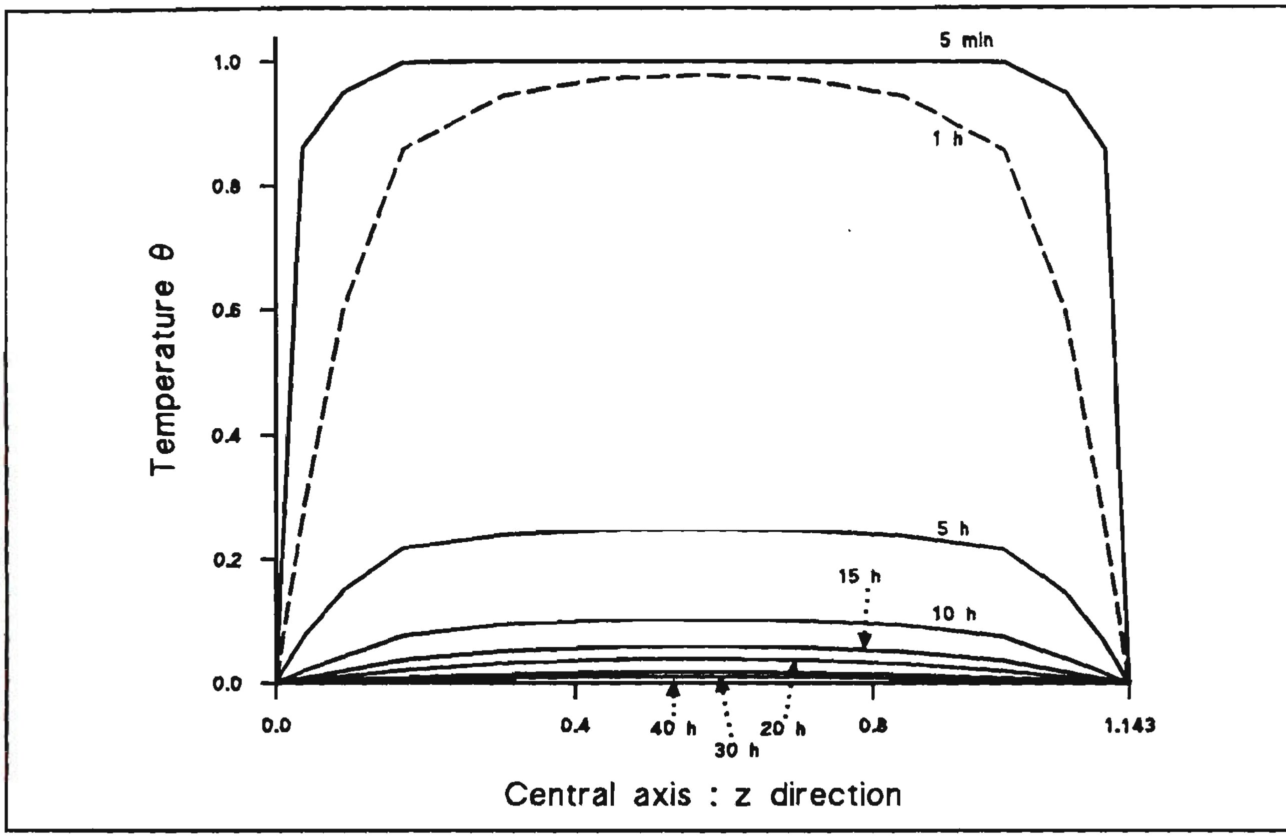


Figure 6.29 Temperature distribution at different cooling times along the central x axis. Apples, isothermal floor.

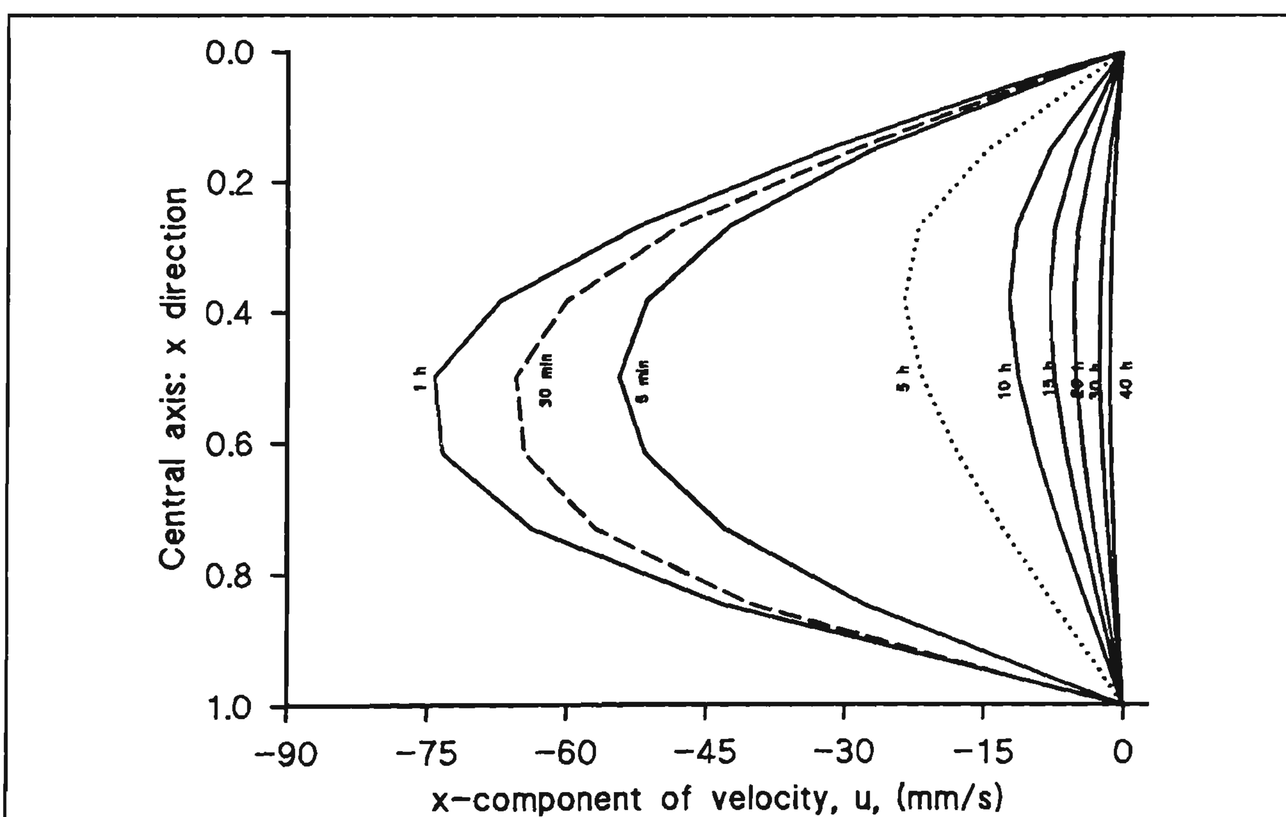


Figure 6.30 x-component of velocity, u, distribution at different cooling times along the central x axis. Apples, isothermal floor.

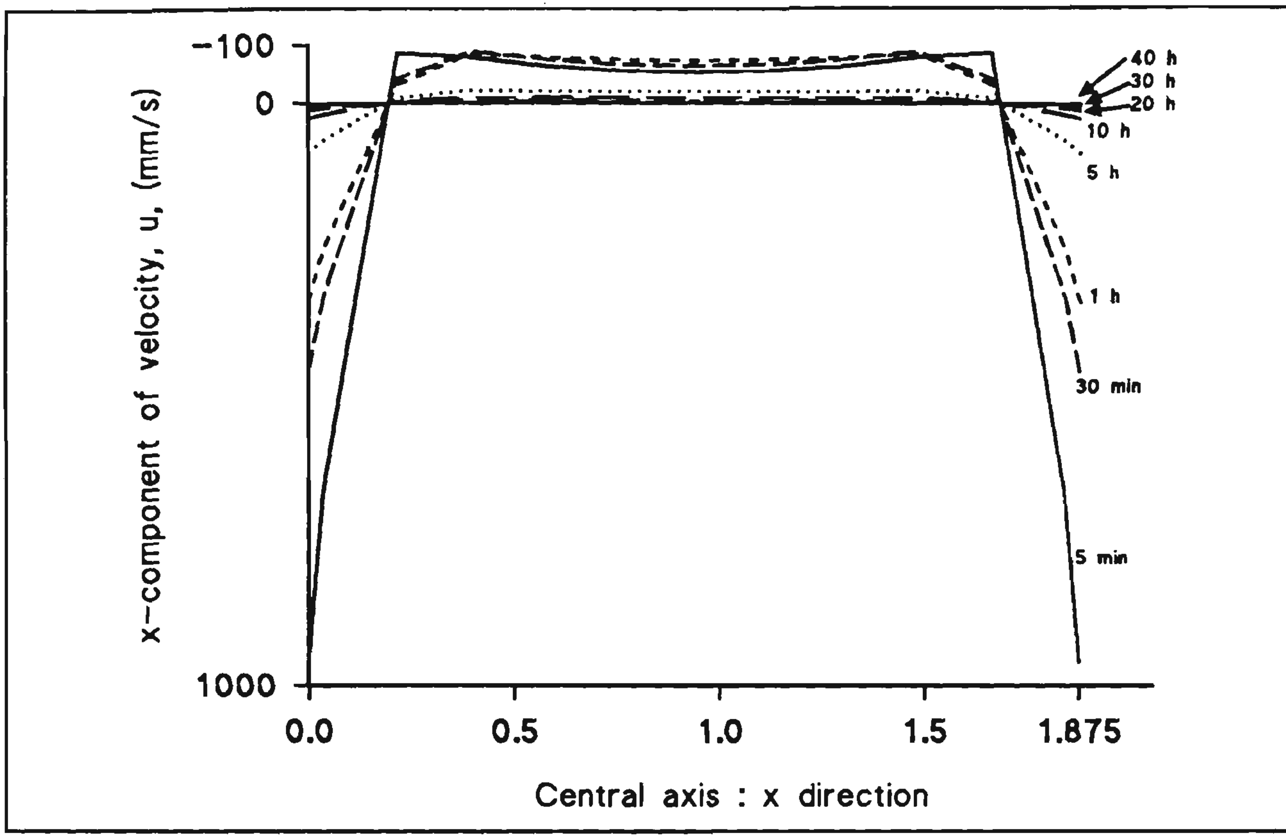


Figure 6.31 x-component of velocity, u, distribution at different cooling times along central y axis. Apples, isothermal floor.

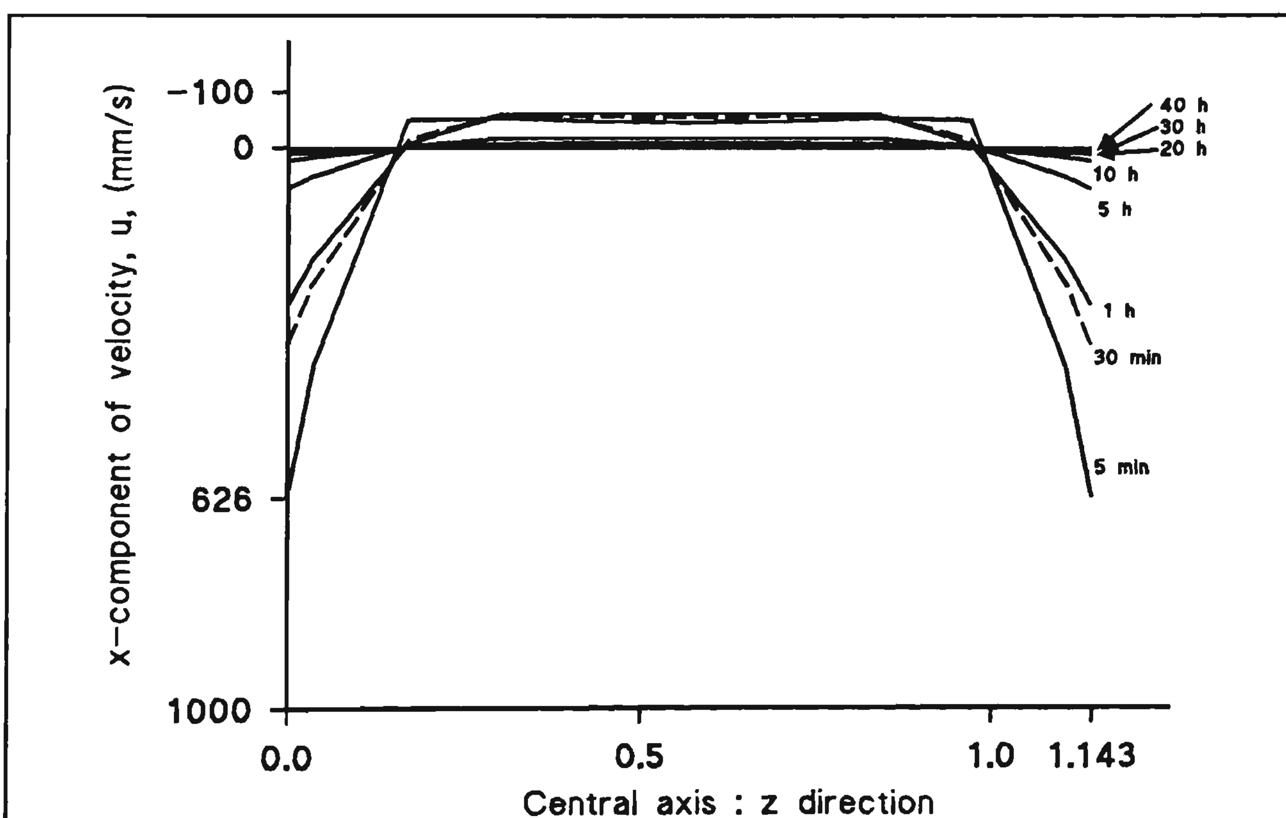


Figure 6.32 x-component of velocity, u, distribution at different cooling times along the central z axis. Apples, isothermal floor.

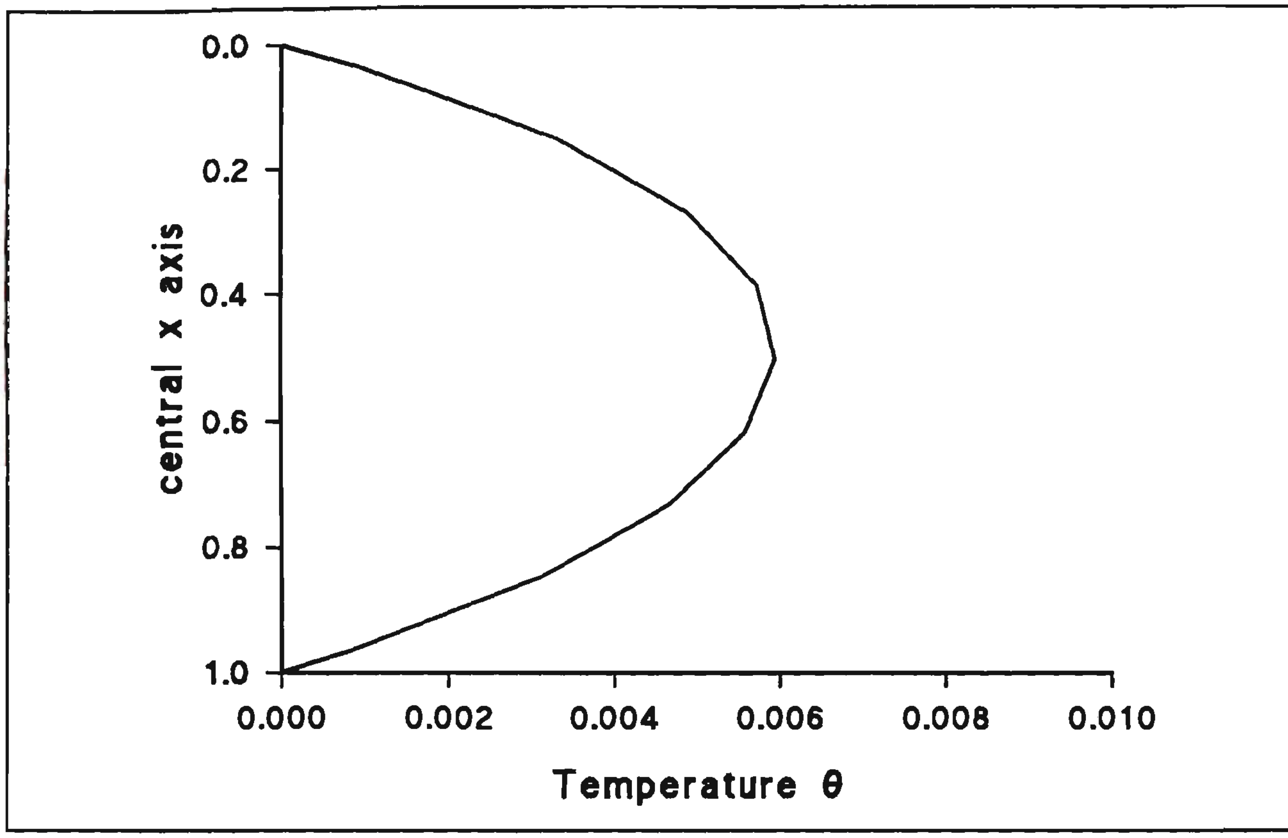


Figure 6.33 Steady state temperature distribution along the central x axis. Apples, isothermal floor.

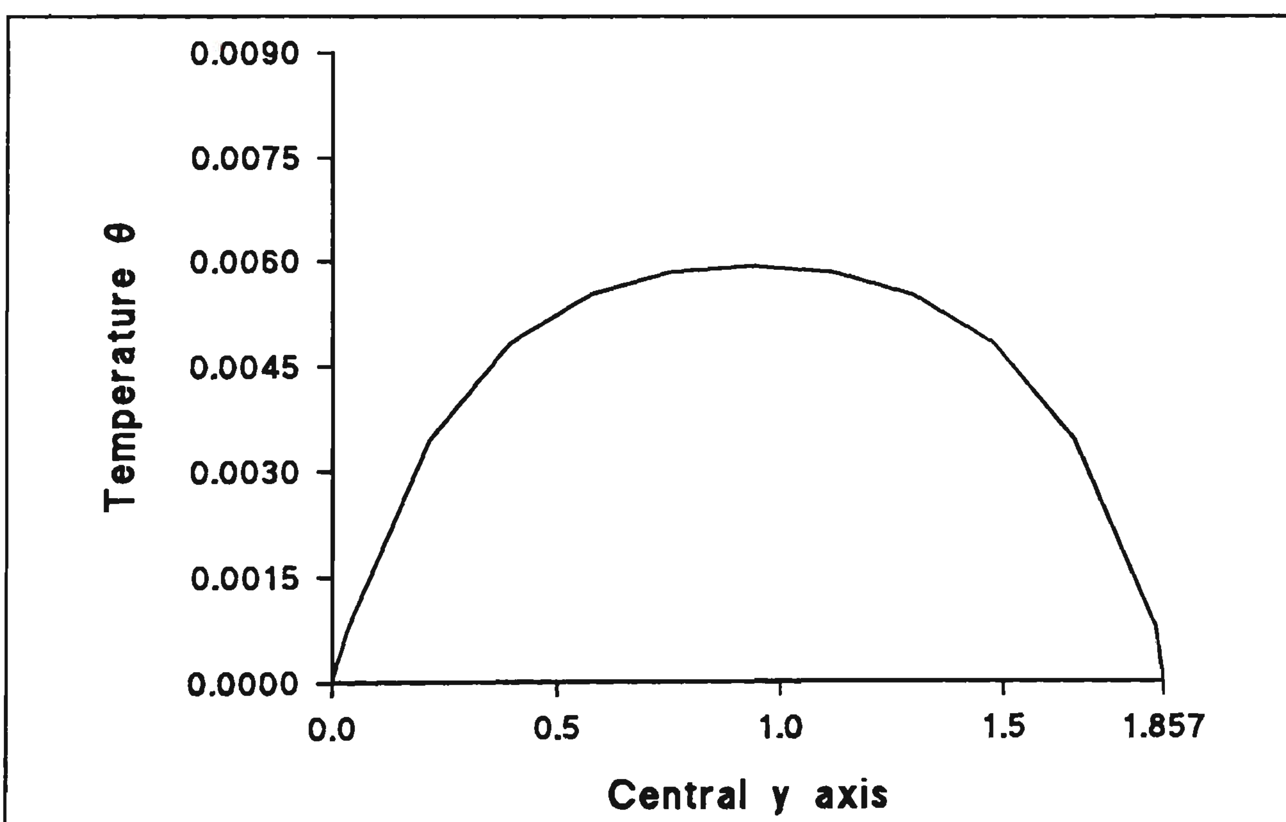


Figure 6.34 Steady state temperature distribution along the central y axis. Apples, isothermal floor.

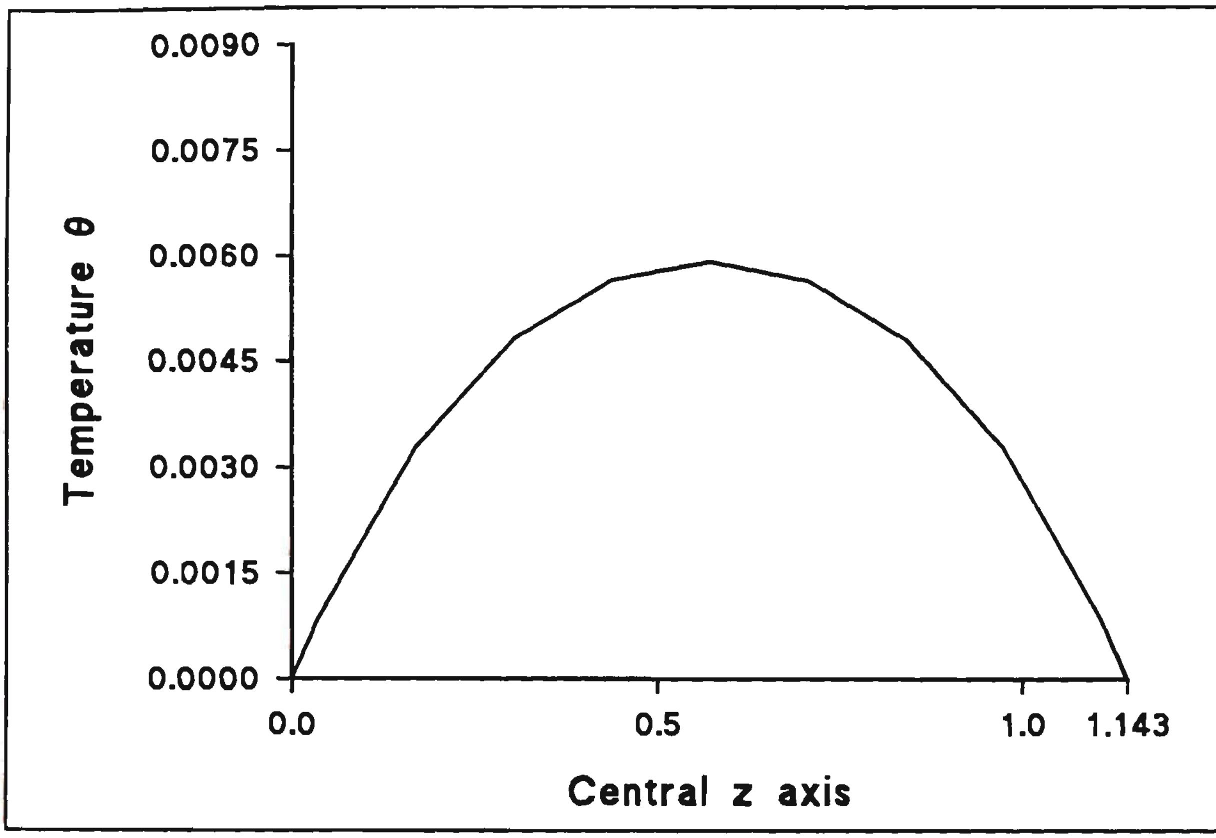


Figure 6.35 Steady state temperature distribution along the central z axis. Apples, isothermal floor.

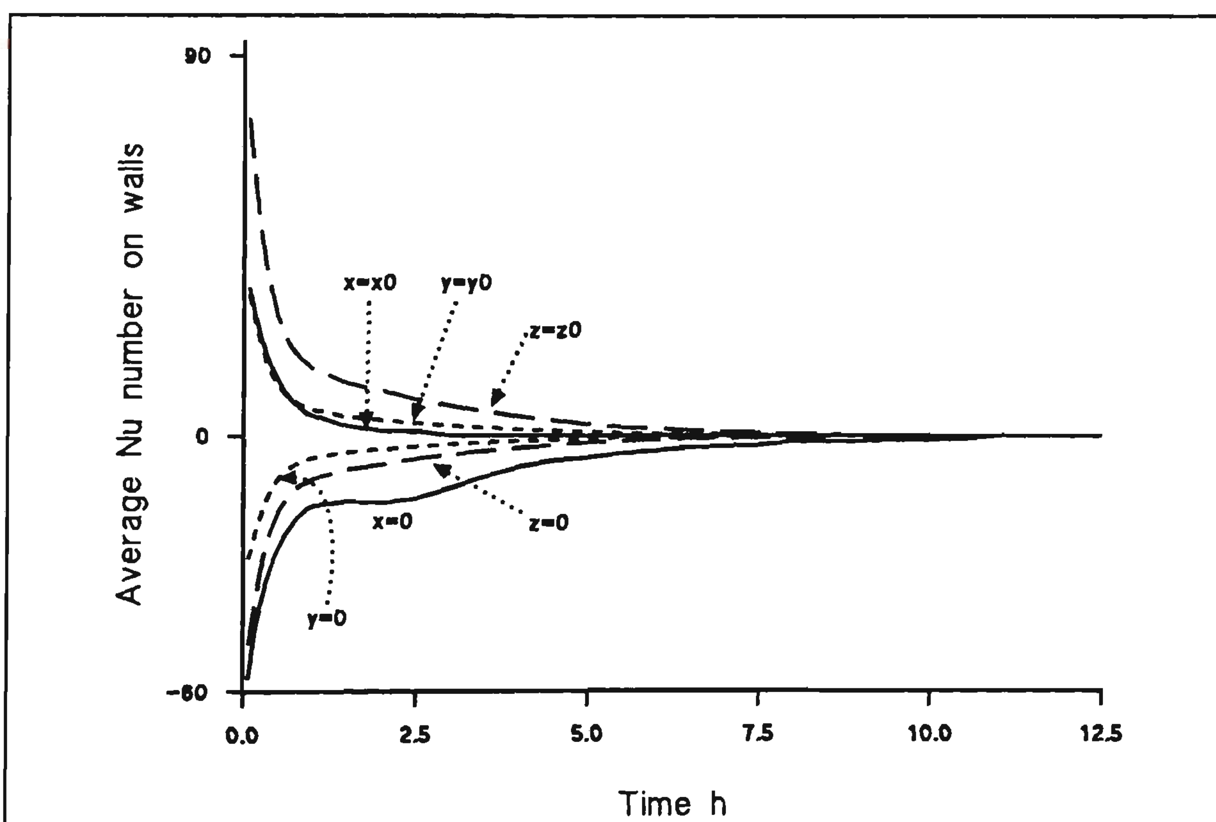


Figure 6.36 Nu number change with time on top and side walls. Apples, isothermal floor.

The results computed for Brussels sprouts cooled under isothermal boundary conditions imposed on the floor of the storage container are presented below in Figures 6.37 to 6.48.

Progressive changes in the temperature destribution with time along the central x axis for Brussels sprouts is shown in Figure 6.37. After some 40 hours the temperature within the packed box reached steady state. Changes in the x-component of velocity, u, distribution with time along the central vertical axis are shown in Figure 6.38. Figures 6.39, 6.40 and 6.41 show the temperature distributions within the container along the central axes for the x,y and z coordinates, as it develops in time. Figures 6.42, 6.43 and 6.44 show the development of the x component of velocity with time along the central axes. The steady state temperature distributions along the central x,y and z directions are shown in Figures 6.45, 6.46 and 6.47 respectively. The average Nusselt number change with time on the side walls and the top wall is presented in Figure 6.48.

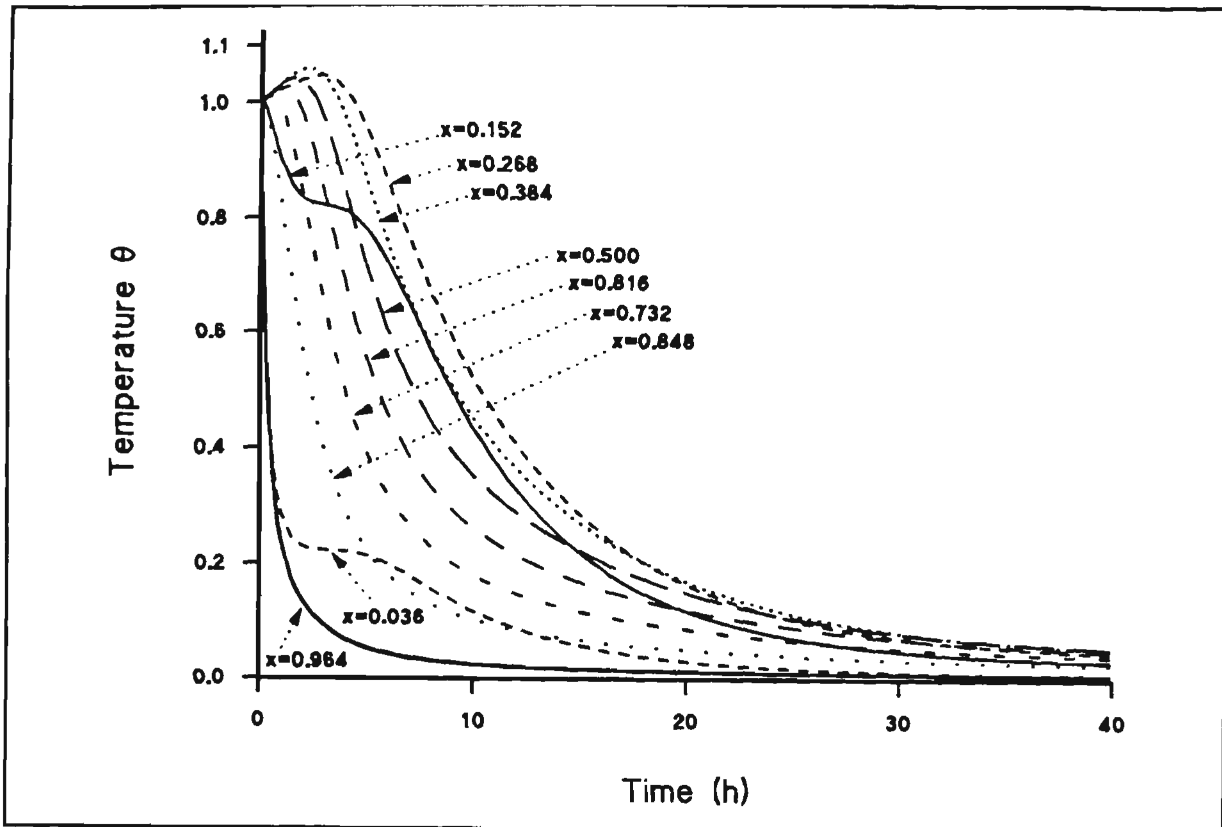


Figure 6.37 Temperature change with time along the vertical central x axis. Brussels sprouts, isothermal floor.

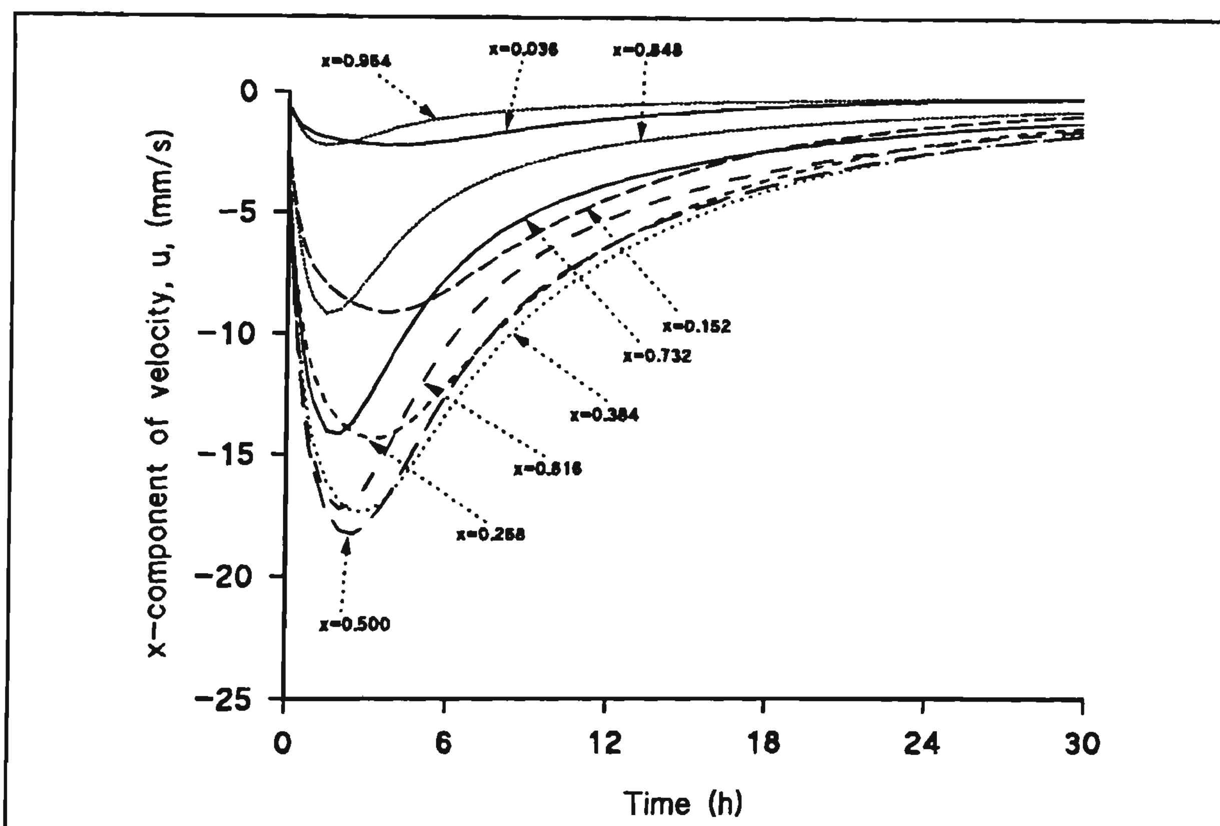


Figure 6.38 x-component of velocity, u, versus time along the vertical central axis. Brussels sprouts, isothermal floor.

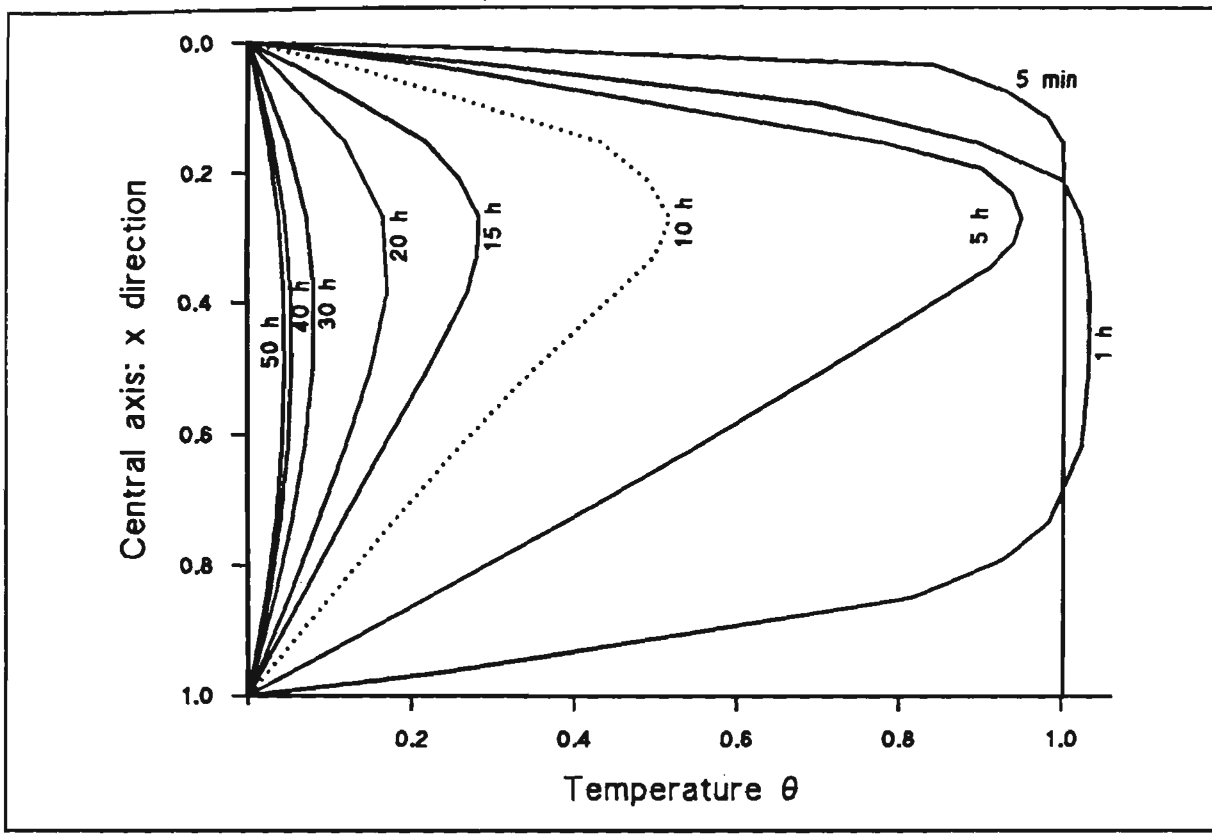


Figure 6.39 Temperature distribution at different cooling times along the central x axis. Brussels sprouts, isothermal floor.

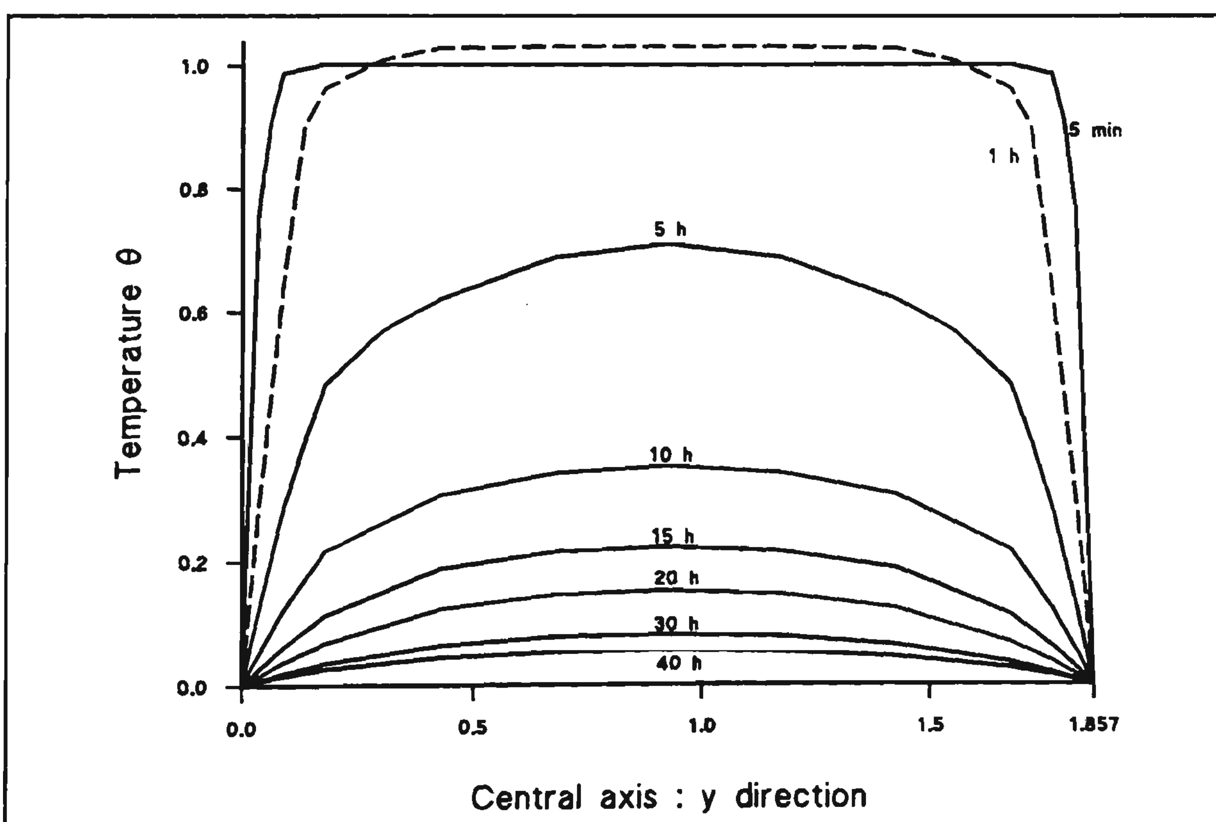


Figure 6.40 Temperature distribution at different cooling times along the central y axis. Brussels sprouts, isothermal floor.

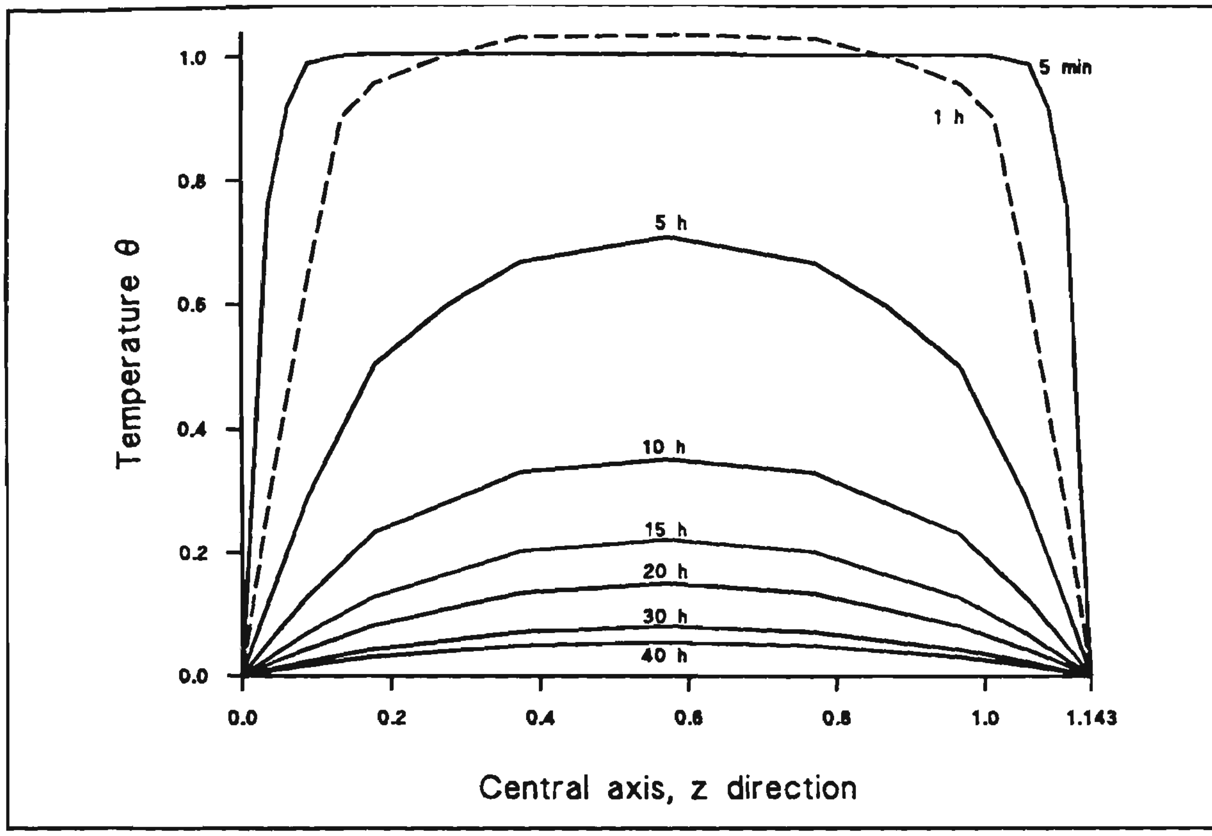


Figure 6.41 Temperature distribution at different cooling times along the central y axis, Brussels sprouts, isothermal floor.

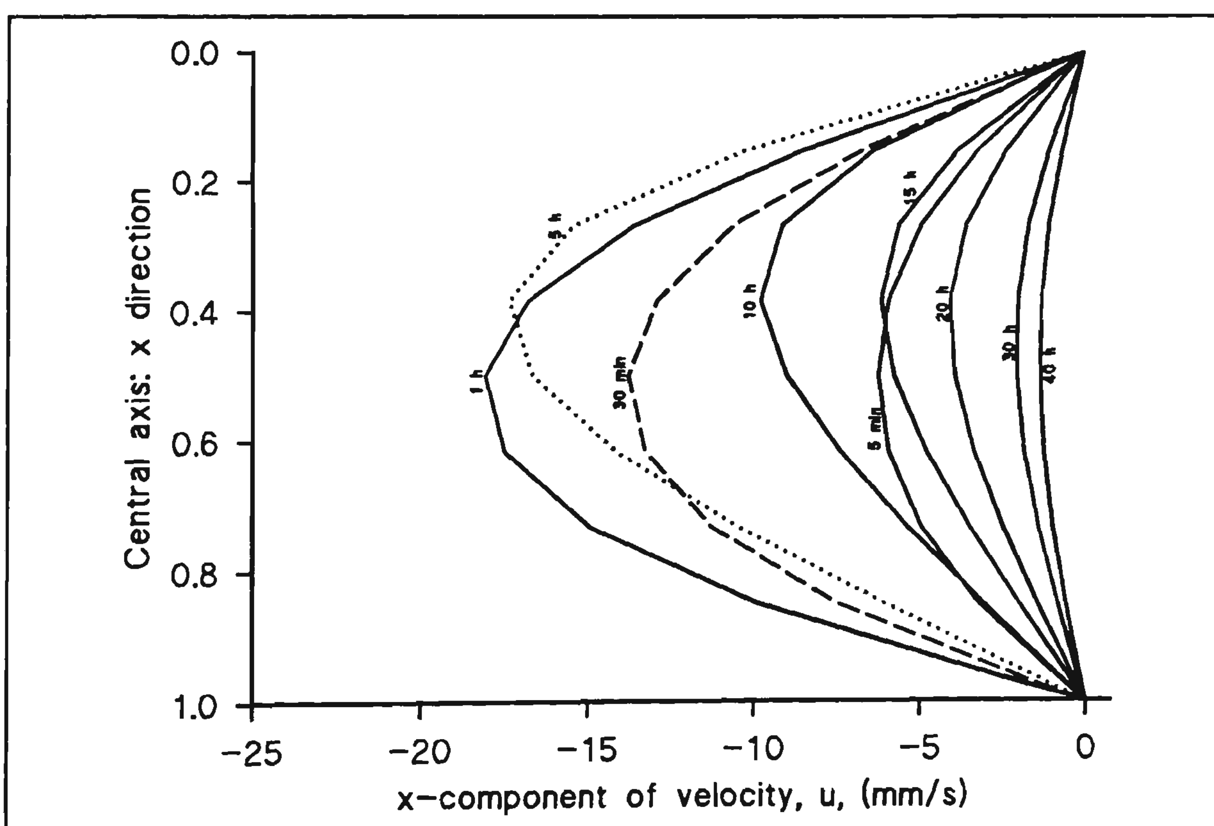


Figure 6.42 x-component of velocity, u, distribution at different cooling times along the central x axis. Brussels sprouts, isothermal floor.

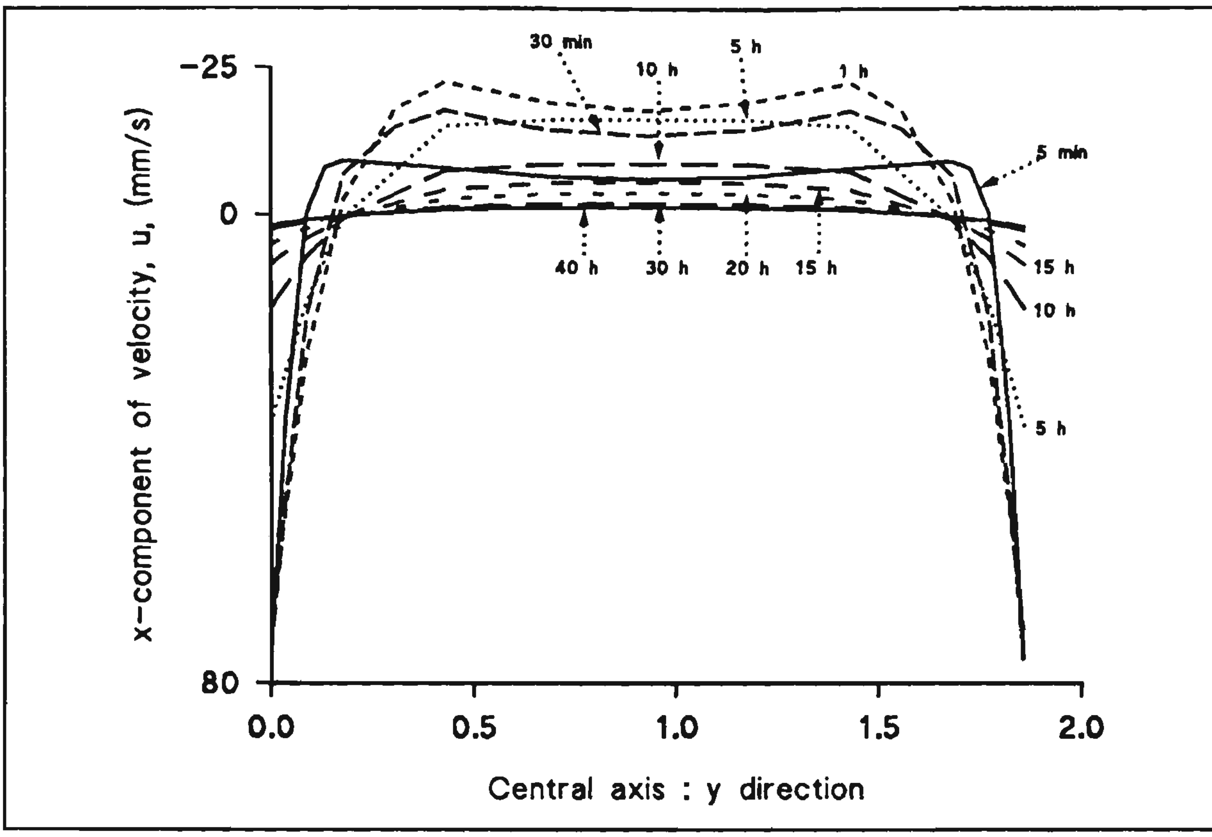


Figure 6.43 x-component of velocity, u, distribution at different cooling times along the central y axis. Brussels sprouts, isothermal floor.

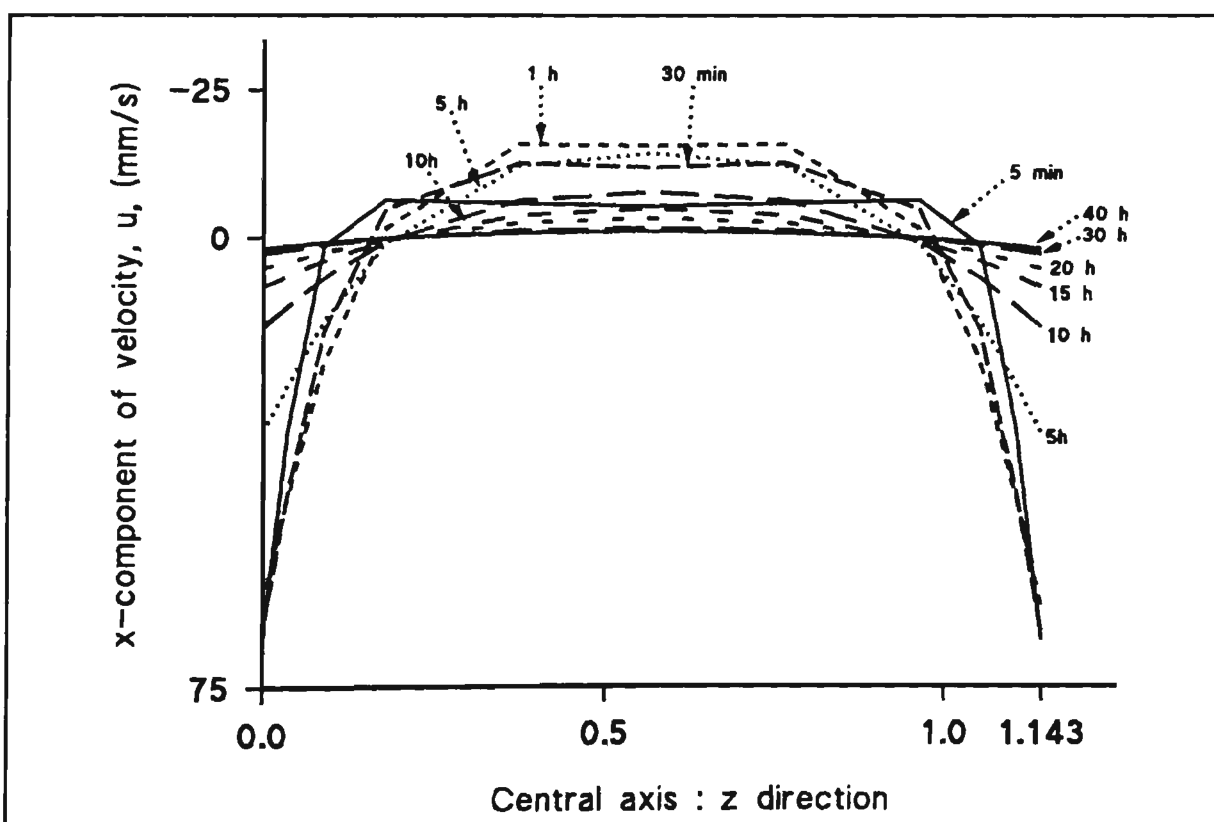


Figure 6.44 x-component of velocity, u, distribution at different cooling times along the central z axis. Brussels sprouts, isothermal floor.

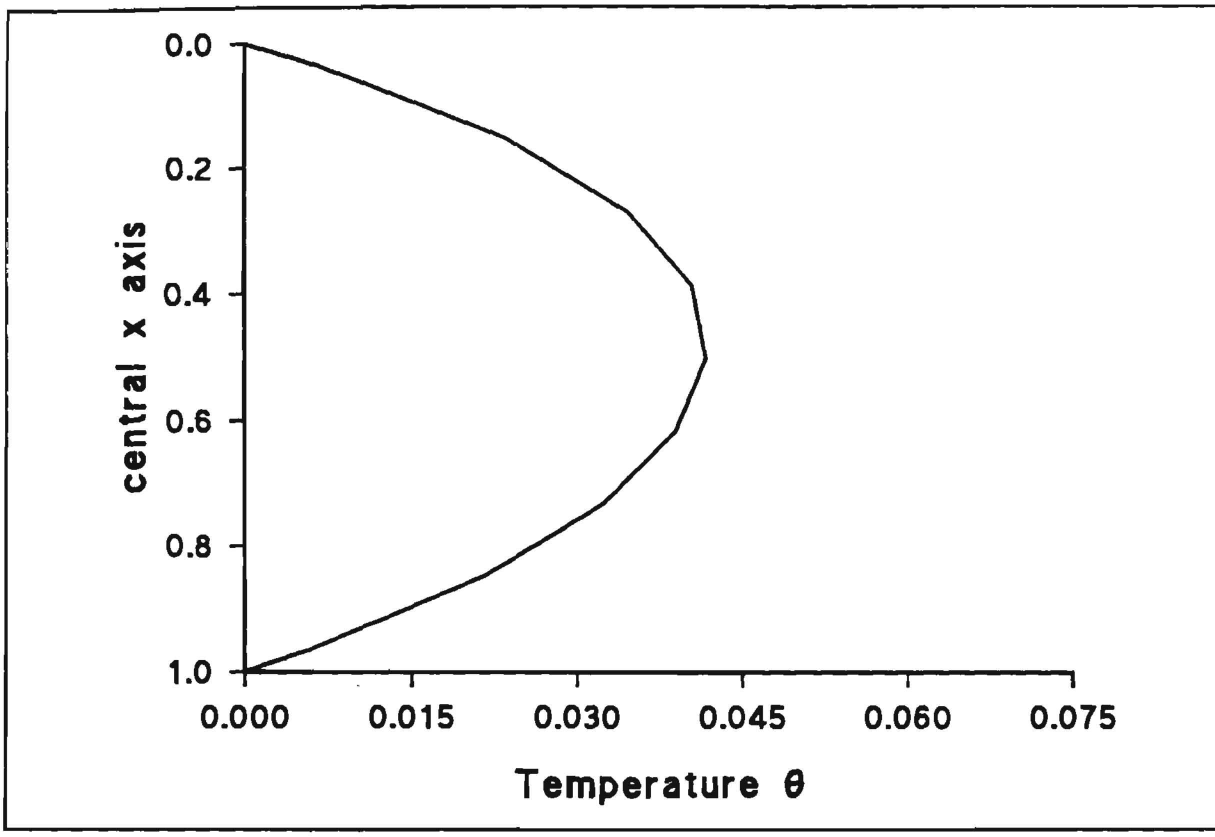


Figure 6.45 Steady state temperature distribution along the central vertical x axis. Brussels sprouts, isothermal floor.

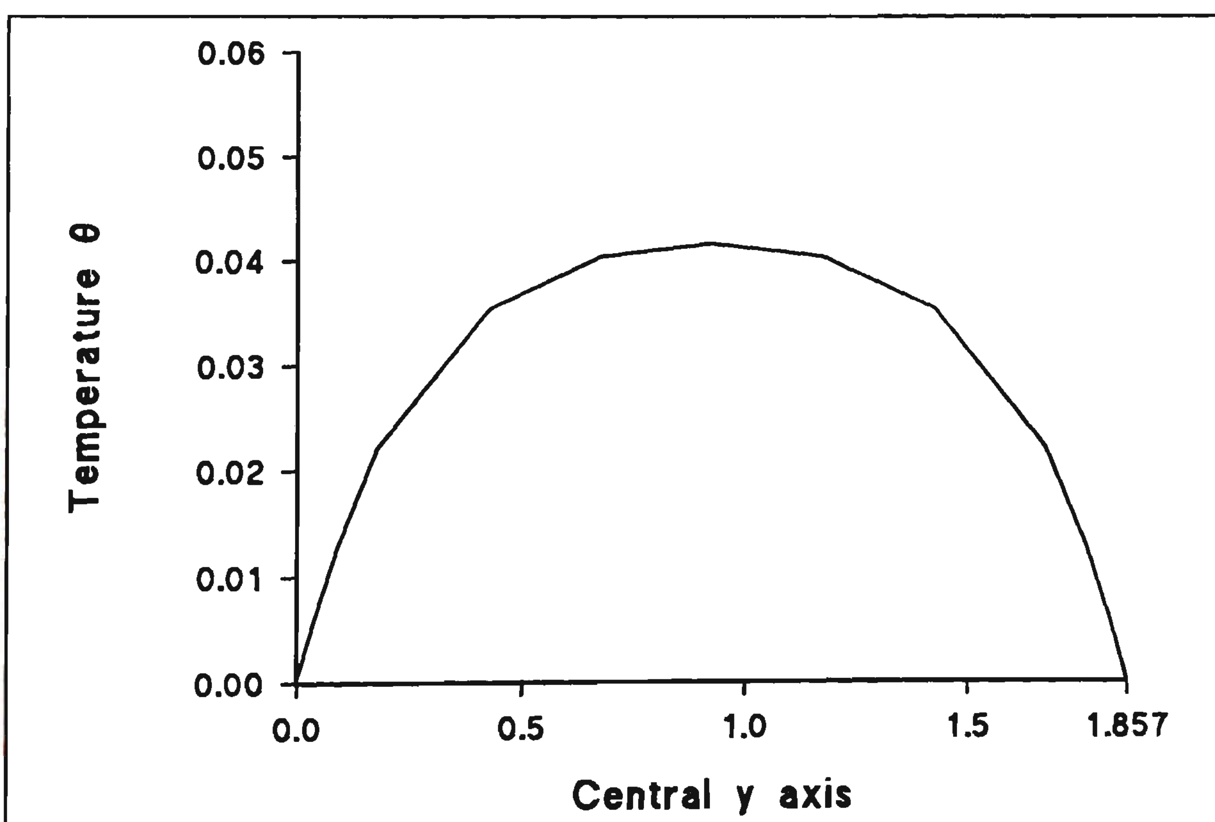


Figure 6.46 Steady state temperature distribution along the central horizontal y axis, Brussels sprout. isothermal floor.

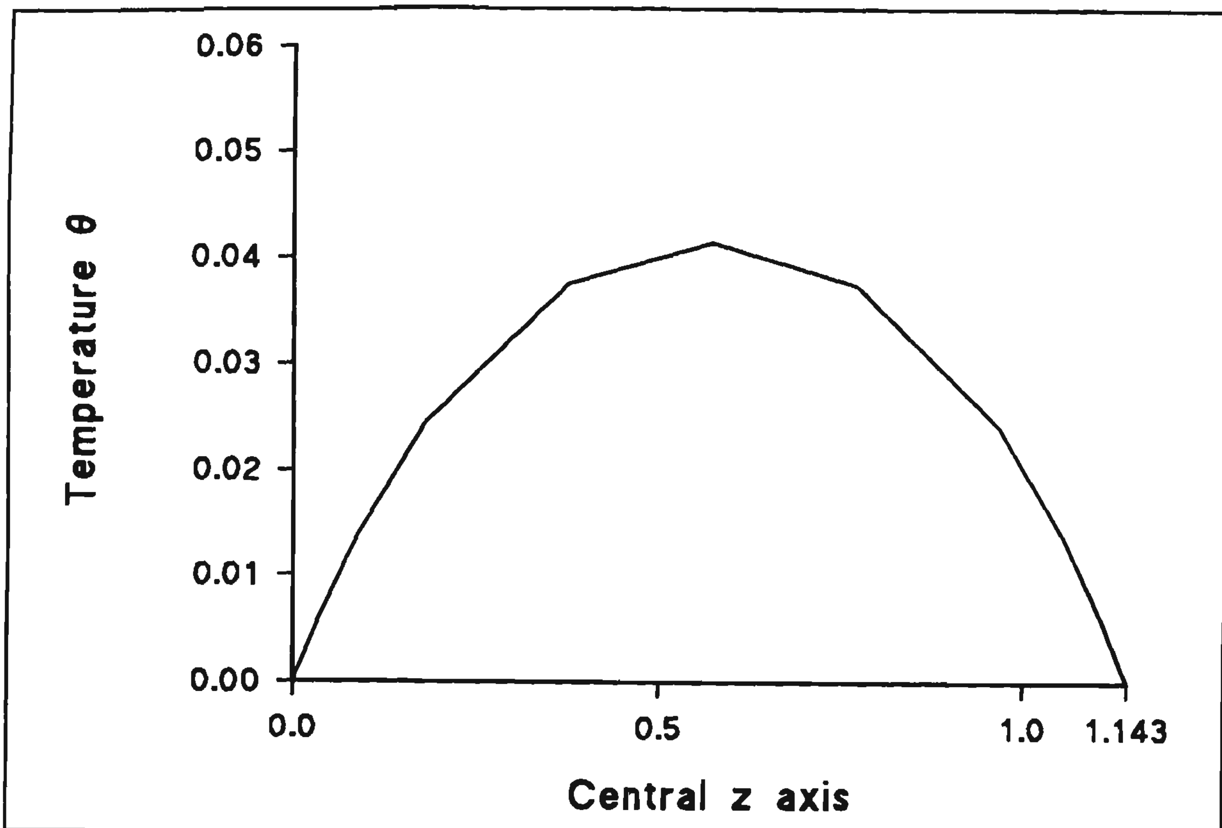


Figure 6.47 Steady state temperature distribution along the central z axis. Brussels sprouts, isothermal floor.

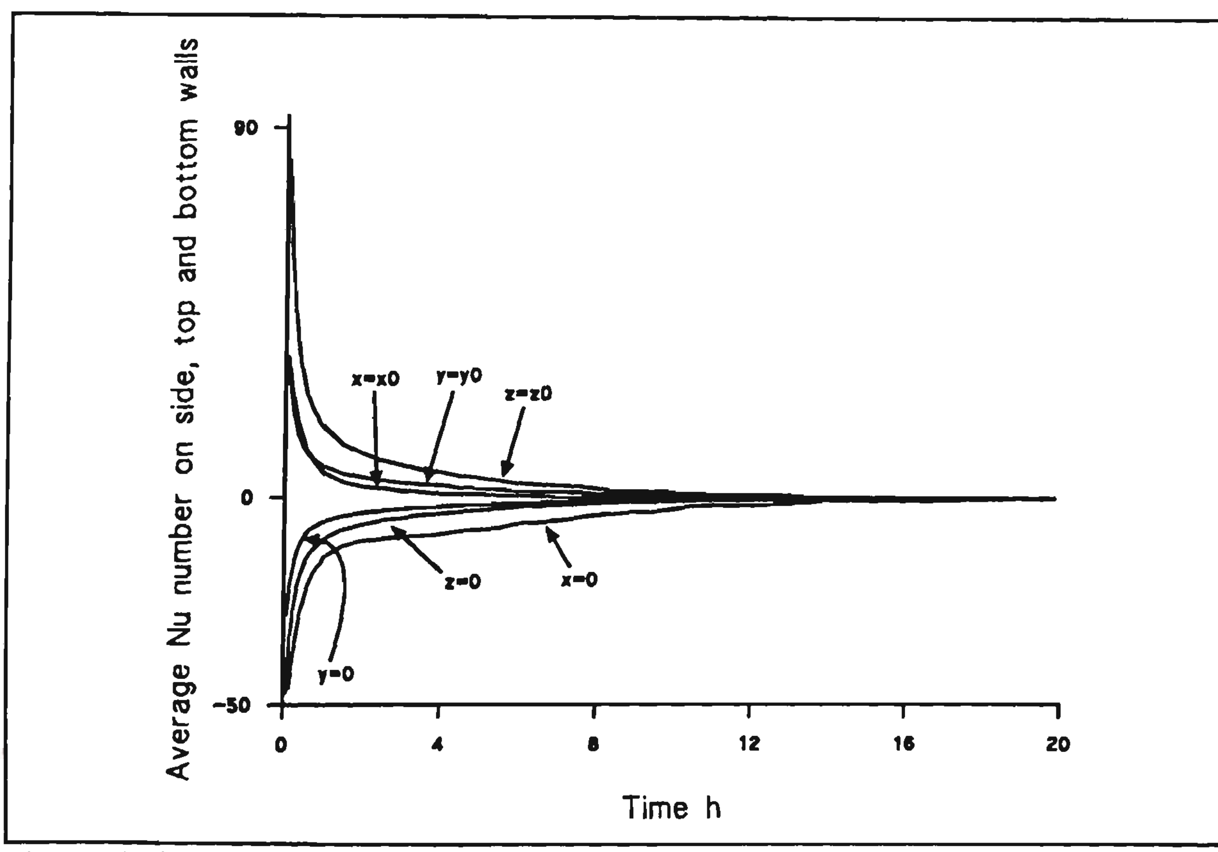


Figure 6.48 Nu number change versus time on top and side walls. Brussels sprouts, isothermal floor.

6.3 Produce with Orthotropic Permeability

Carrots and asparagus packed in boxes 0.28m × 0.52m × 0.32m were chosen as representative of the orthotropic permeability characteristics of packed produce undergoing convective heat transfer. Data describing the physical properties of carrot, which has a low respiration rate, and asparagus, which has a high respiration rate are shown in Tables 6.4 and 6.5 repectively.

Table 6.4 Physical properties of Carrots

Water content, ϕ_{water}, %, (Hardenburg, 1986)	88.2
Specific heat, Cp_s, KJ/Kg·K, (Mohsenin,1980)	3.768
Density, ρ_s, Kg/m^3 1)	1046
Density of packed bed, ρ_p, Kg/m^3, (Mohsenin,1986)	641
Average effective diameter, m 2)	0.041
Porosity, ε 3)	0.3872
Effective thermal conductivity in packed bed, KW/m·K 4)	0.291×10^{-3}
Initial temperature, °C	30°
Cold wall temperature, °C	0°

Note: 1) $\rho_s = 2670/(1.67+\phi_{water})$, Fikiin,1983;
2) May vary from different grades. See Equation 3.23;
3) $\varepsilon \approx 1-\rho_p/\rho_s$;
4) Assumed having the same value in Table 6.2.

Table 6.5 Physical properties of Asparagus

Water content, ϕ_{water}, %, (Hardenburg, 1986)	93.0
Specific heat, Cp_s, KJ/Kg·K, (Mohsenin,1980)	3.936
Density, ρ_s, Kg/m^3 1)	1027
Density of packed bed, ρ_p, Kg/m^3, (Mohsenin,1986)	577
Average effective diameter, m 2)	0.02
Porosity, ε 3)	0.4382
Effective thermal conductivity in packed bed, KW/m·K 4)	0.291×10^{-3}
Initial temperature, °C	30°
Cold wall temperature, °C	0°

Note: 1) $\rho_s = 2670/(1.67+\phi_{water})$, Fikiin,1983;
2) May vary from different grades. See Equation 3.23;
3) $\varepsilon \approx 1-\rho_p/\rho_s$;
4) Assumed having the same value in Table 6.2.

As previously, Table 2.2 was used to determine the respiration rate versus temperature functions for carrots and asparagus. The numerical experiments described in this section were performed on boxes experencing adiabatic floor conditions and isothermal cooling on the sides and top.

6.3.1 Horizontally Laid Pack

When produce such as asparagus, cucumber, celery, beans or carrots are laid horizontally in boxes, one of the horizontally directed permeabilities, say K_y, has a value different to the other two, K_x, K_z. In order to examine the effect of horizontal packing on heat transfer, four different relationships between the components of permeability were used in computing the results for carrots. The relationships used were: $Da_y = Da_x = Da_z$; $Da_y = 2Da_x = 2Da_z$; $Da_y = 3Da_x = 3Da_z$; $Da_y = 4Da_x = 4Da_z$.

The temperature versus time curves computed for the four different permeabilities for horizontally packed carrots are shown in Figure 6.49 and the temperature distribution along the central axes in the x, y, and z directions are shown in Figures 6.50, 6.51 and 6.52 respectively. Velocity profiles have been computed after cooling for one hour and after five hours, shown in Figures 6.53, 6.54 and 6.55.

Temperature distributions along the x, y and z diresctions were computed at steady state conditions and the results are shown in Figures 6.56, 6.57 and 6.58 for carrots.

A paralllel set of results using the same boundary conditions and permeability relationships were computed for horizontally laid asparagus. These results are presented in Figures 6.59 through to 6.68.

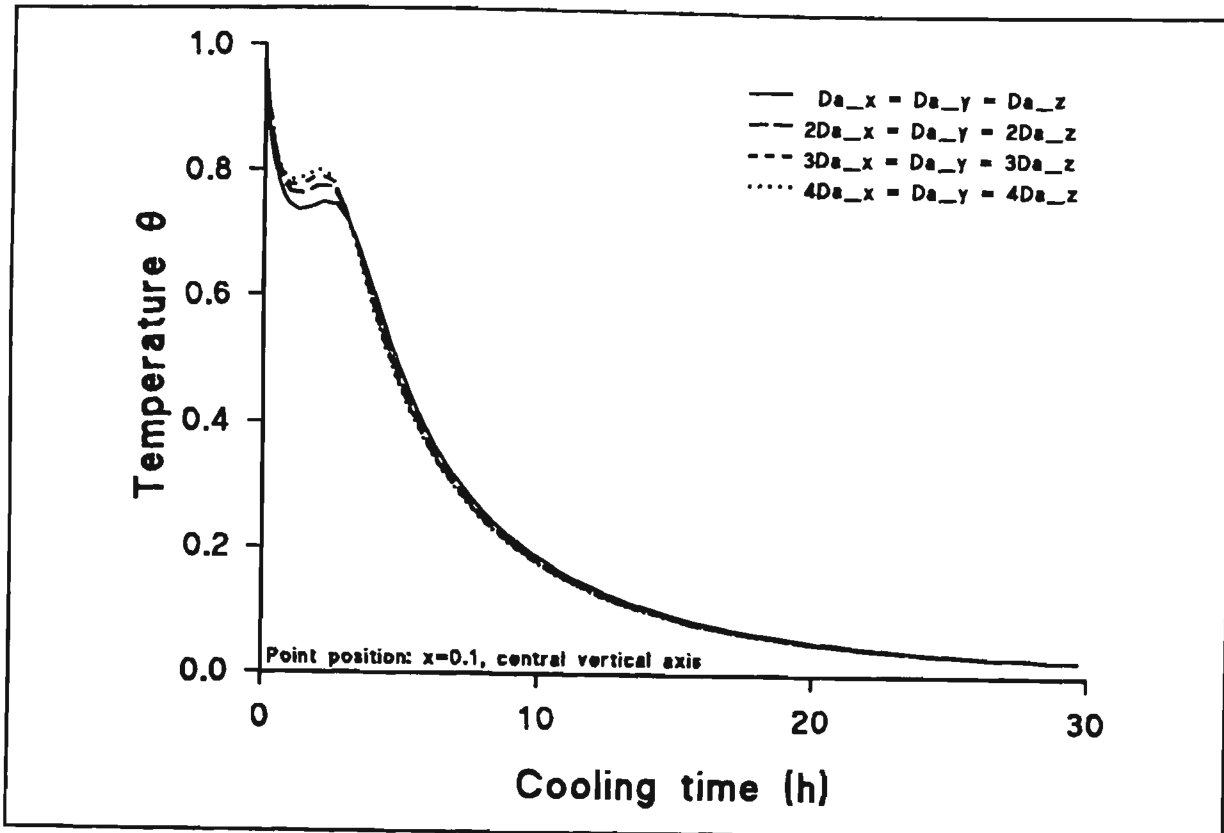


Figure 6.49 Temperature versus time curves for horizontally laid carrots, adiabatic floor.

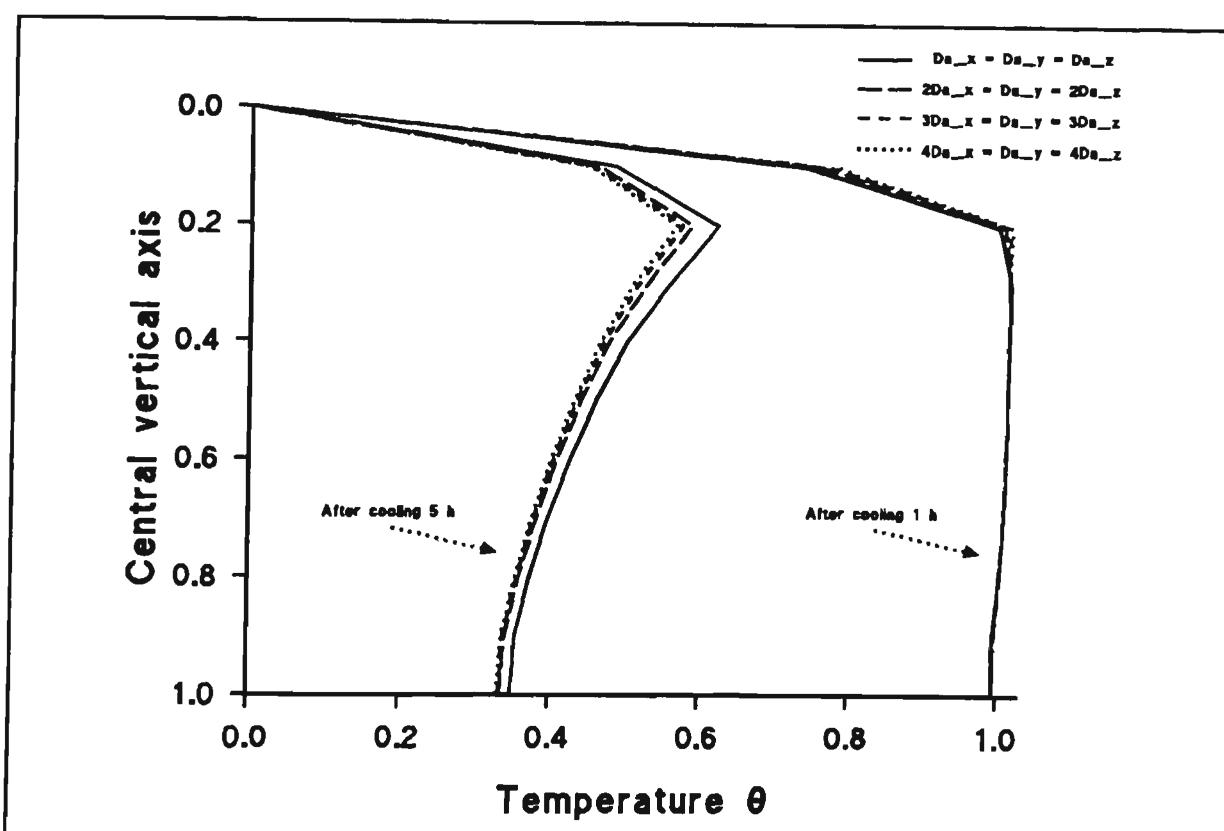


Figure 6.50 Temperature distributions along the central vertical axis for horizontally laid carrots, adiabatic floor.

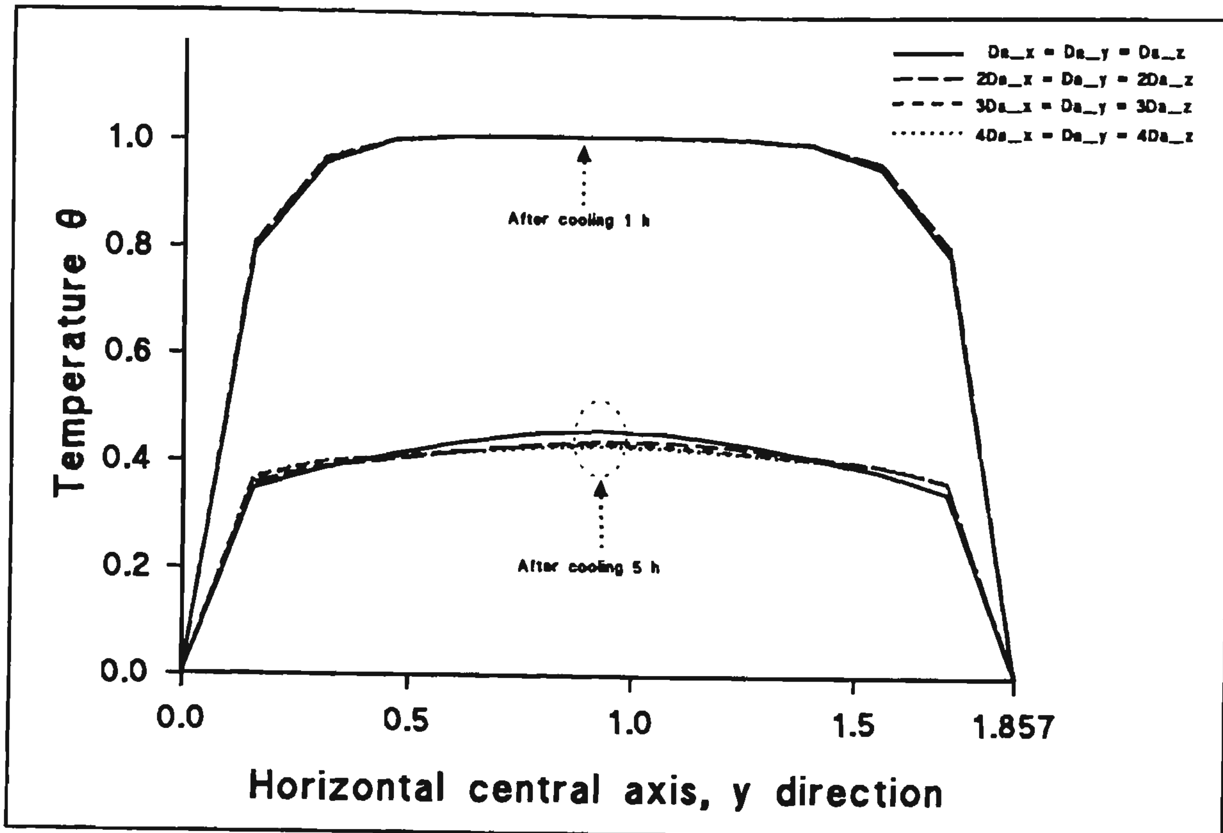


Figure 6.51 Temperature distributions along the central horizontal y axis for horizontally laid carrots, adiabatic floor.

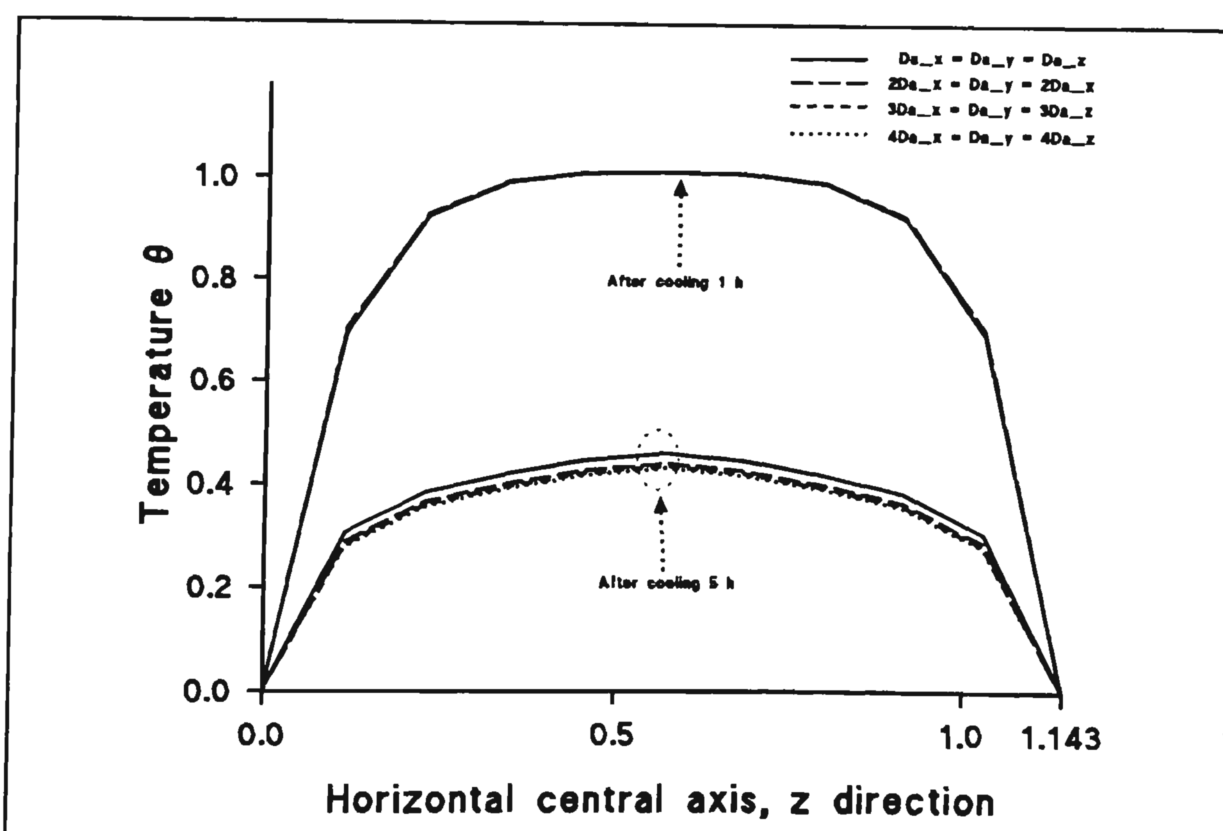


Figure 6.52 Temperature distributions along the central horizontal z axis for horizontally laid carrots, adiabatic floor.

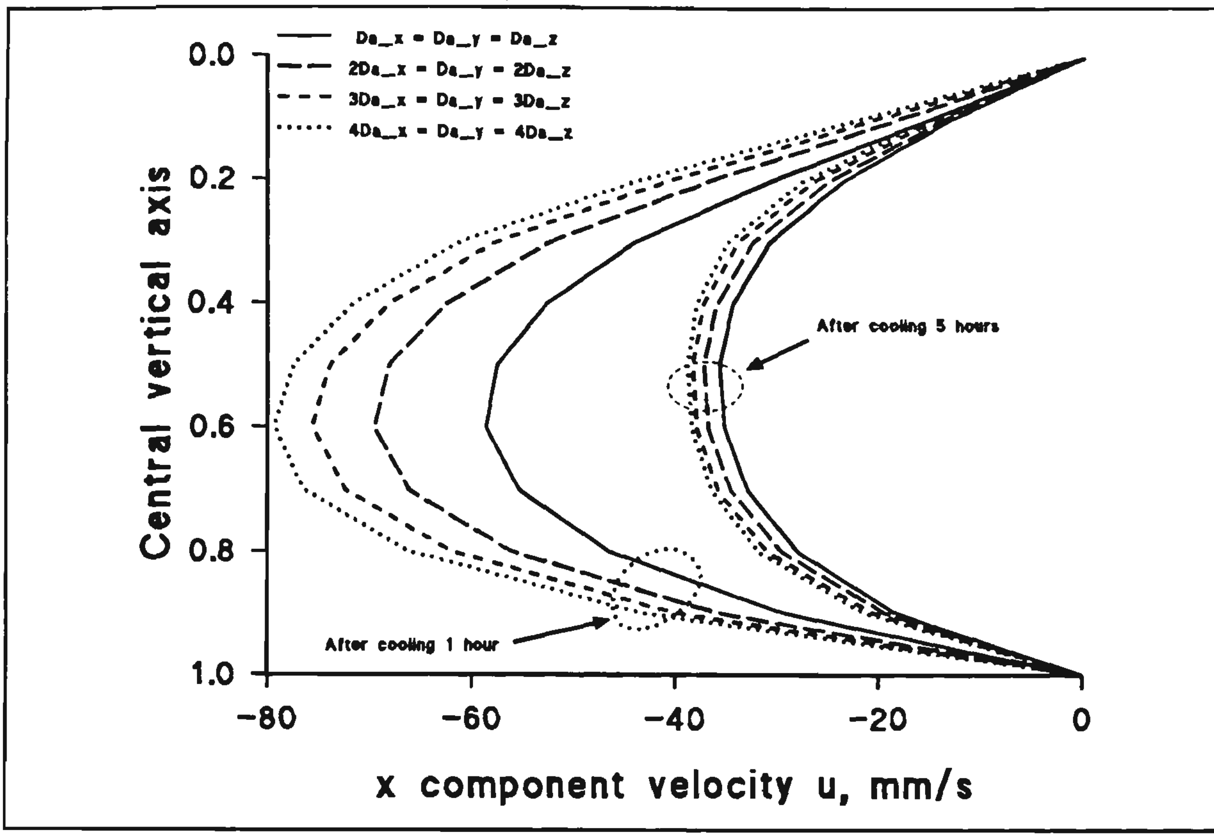


Figure 6.53 x-component of velocity, u, distributions along the central vertical x axis for horizontally laid carrots, adiabatic floor.

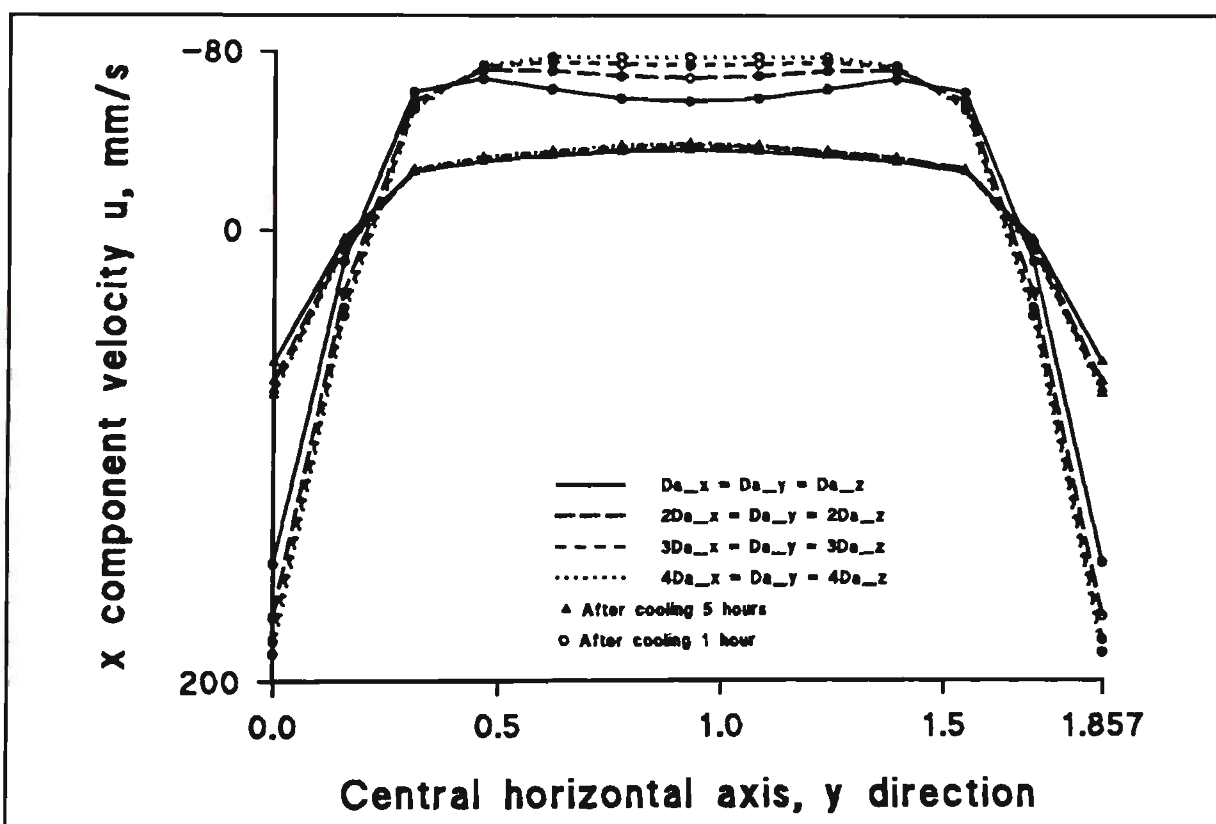


Figure 6.54 x-component of velocity, u, distributions along the central horizontal y axis for horizontally laid carrots, adiabatic floor.

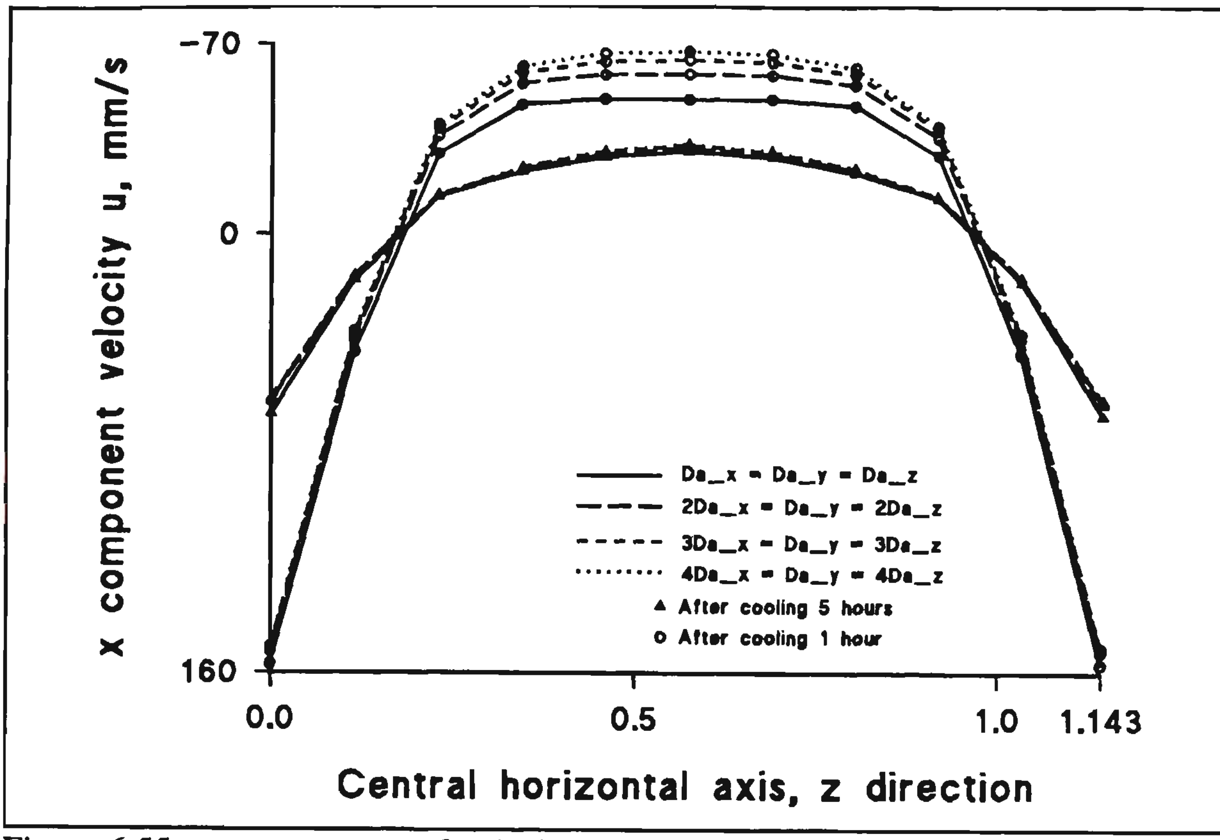


Figure 6.55 x-component of velocity, u, distributions along the central horizontal z axis for horizontally laid carrots, adiabatic floor.

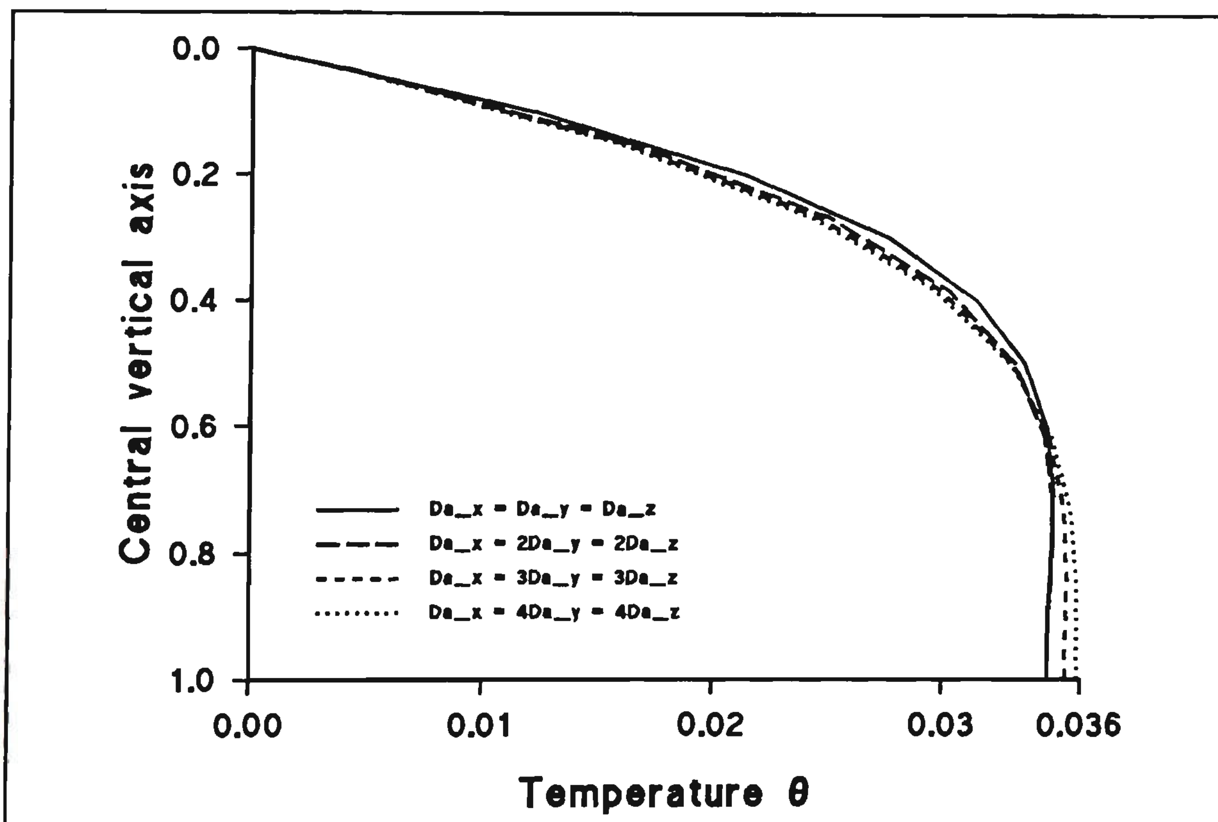


Figure 6.56 Temperature distributions along the central vertical x axis at steady state for horizontally laid carrots, adiabatic floor.

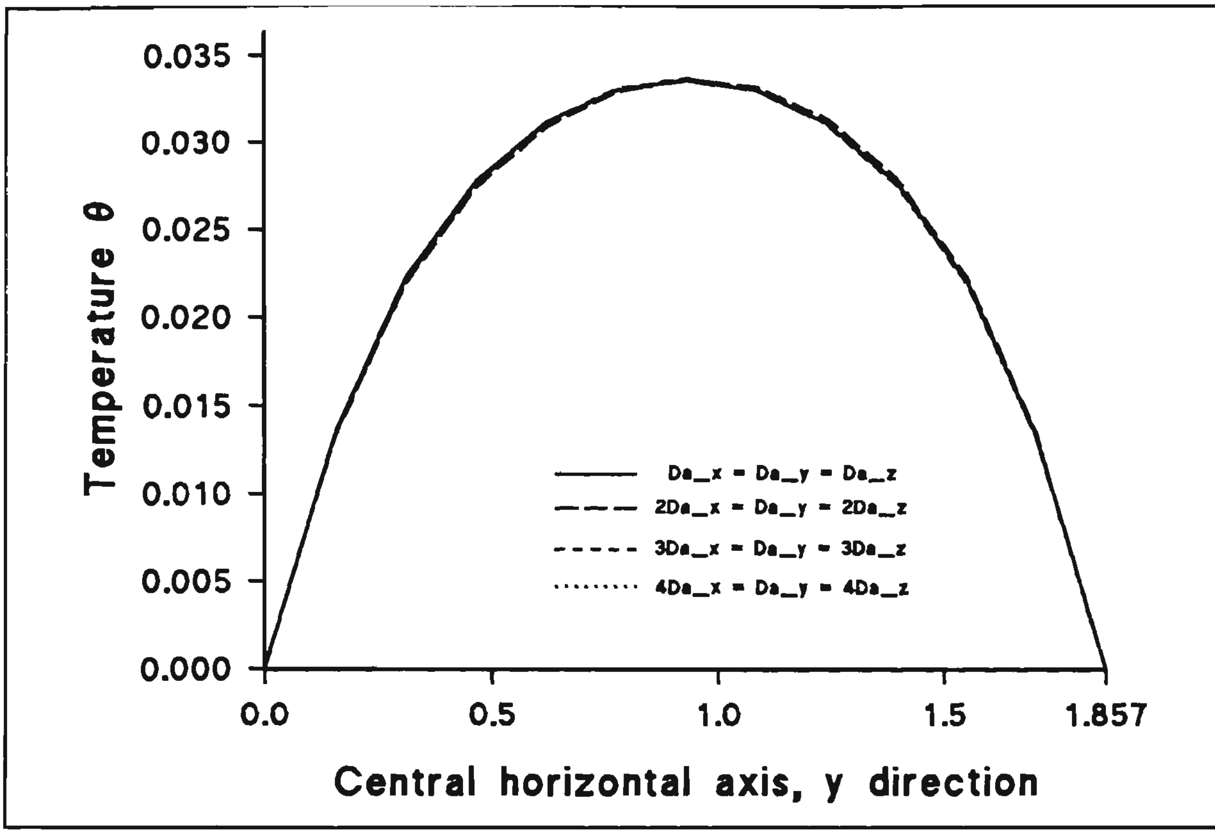


Figure 6.57 Temperature distributions along the central horizontal y axis at steady state for horizontally laid carrots, adiabatic floor.

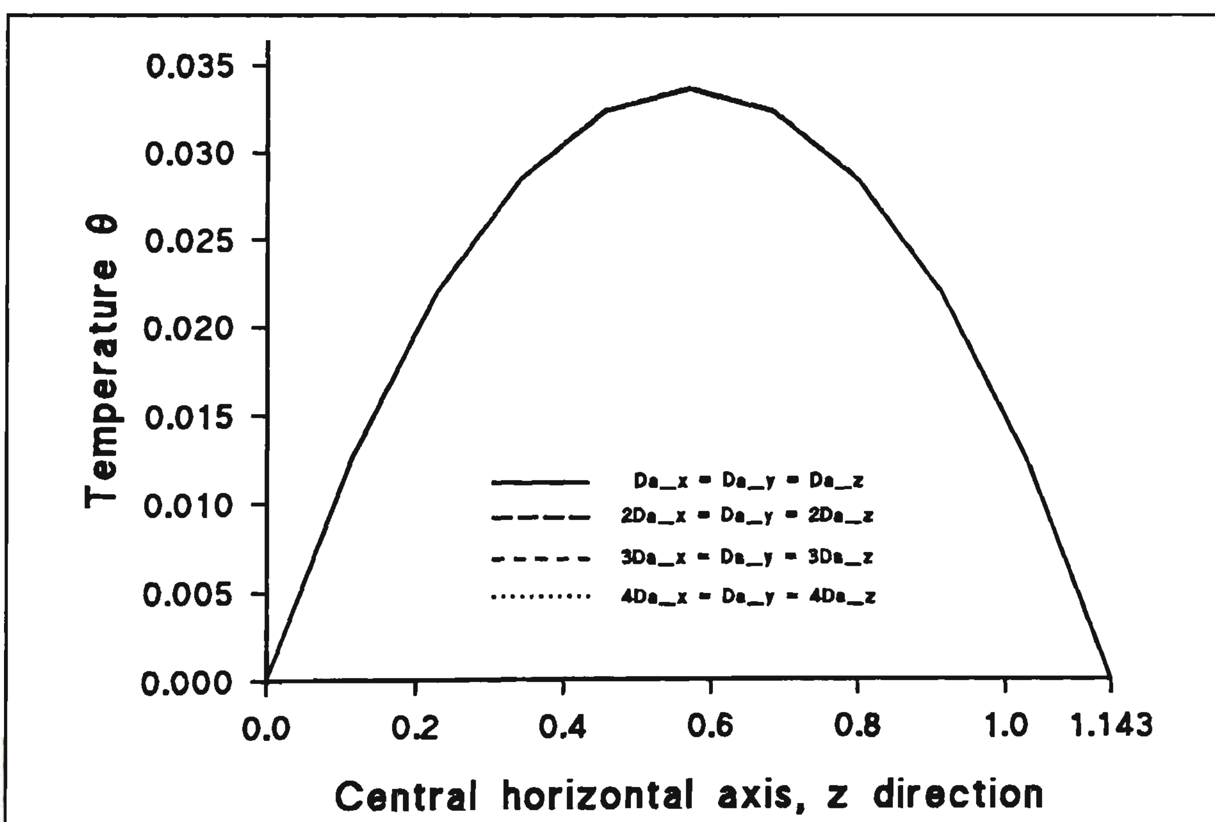


Figure 6.58 Temperature distributions along the cental horizontal z axis at steady state for horizontally laid carrots, adiabatic floor.

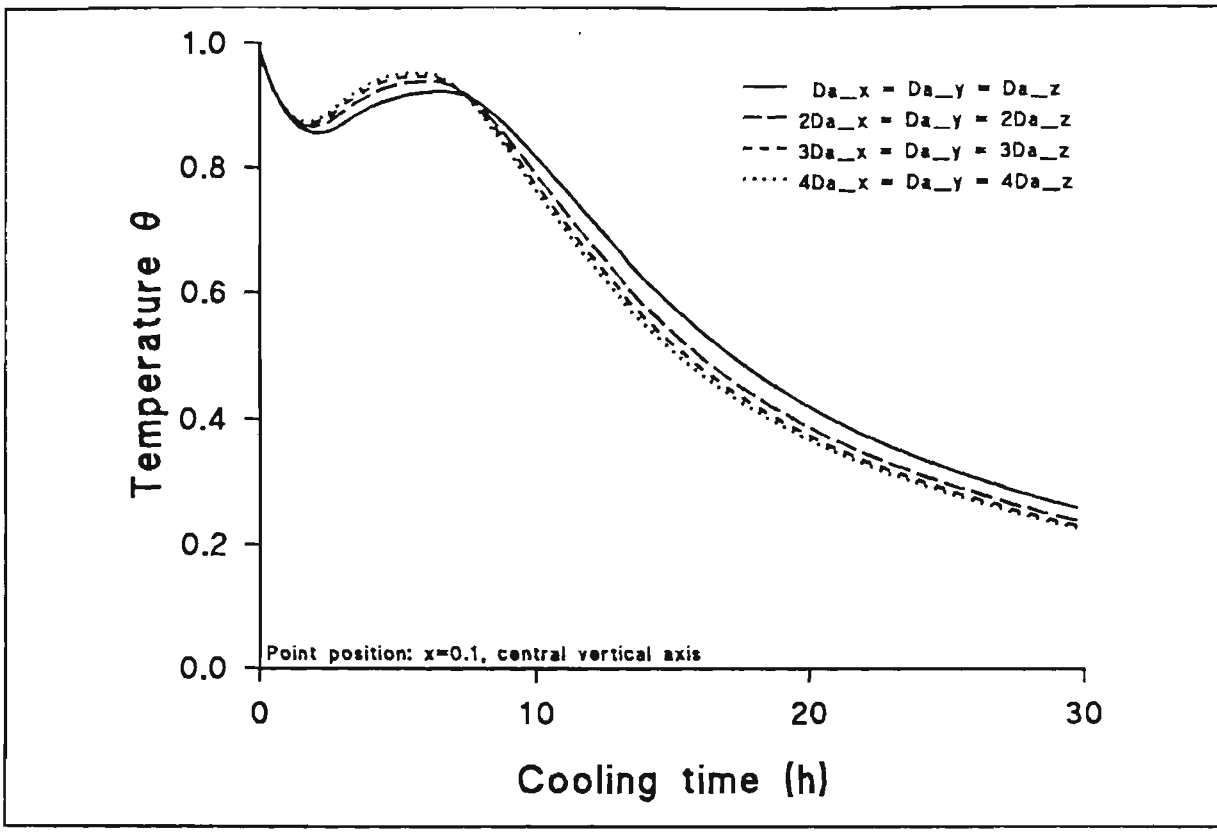


Figure 6.59 Temperature versus time curves for horizontally laid asparagus, adiabatic floor.

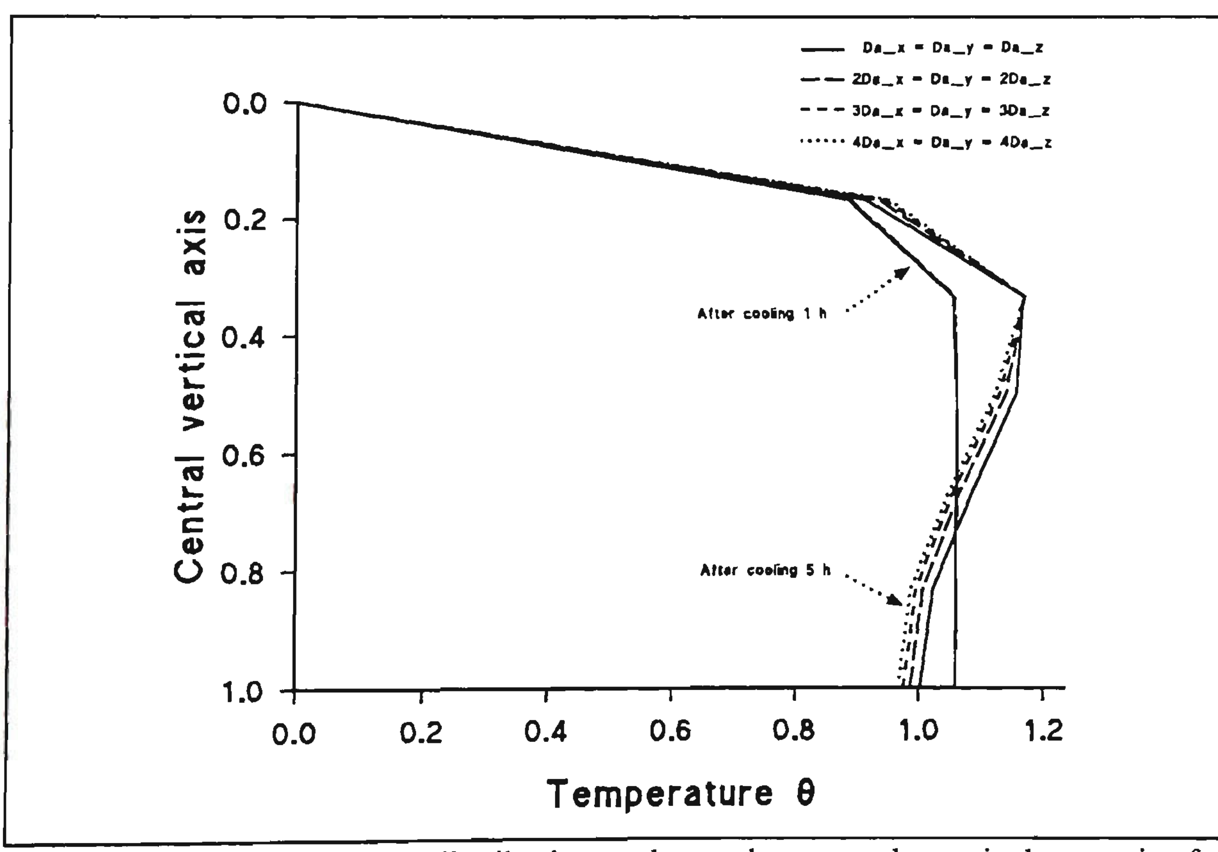


Figure 6.60 Temperature distributions along the central vertical x axis for horizontally laid asparagus, adiabatic floor

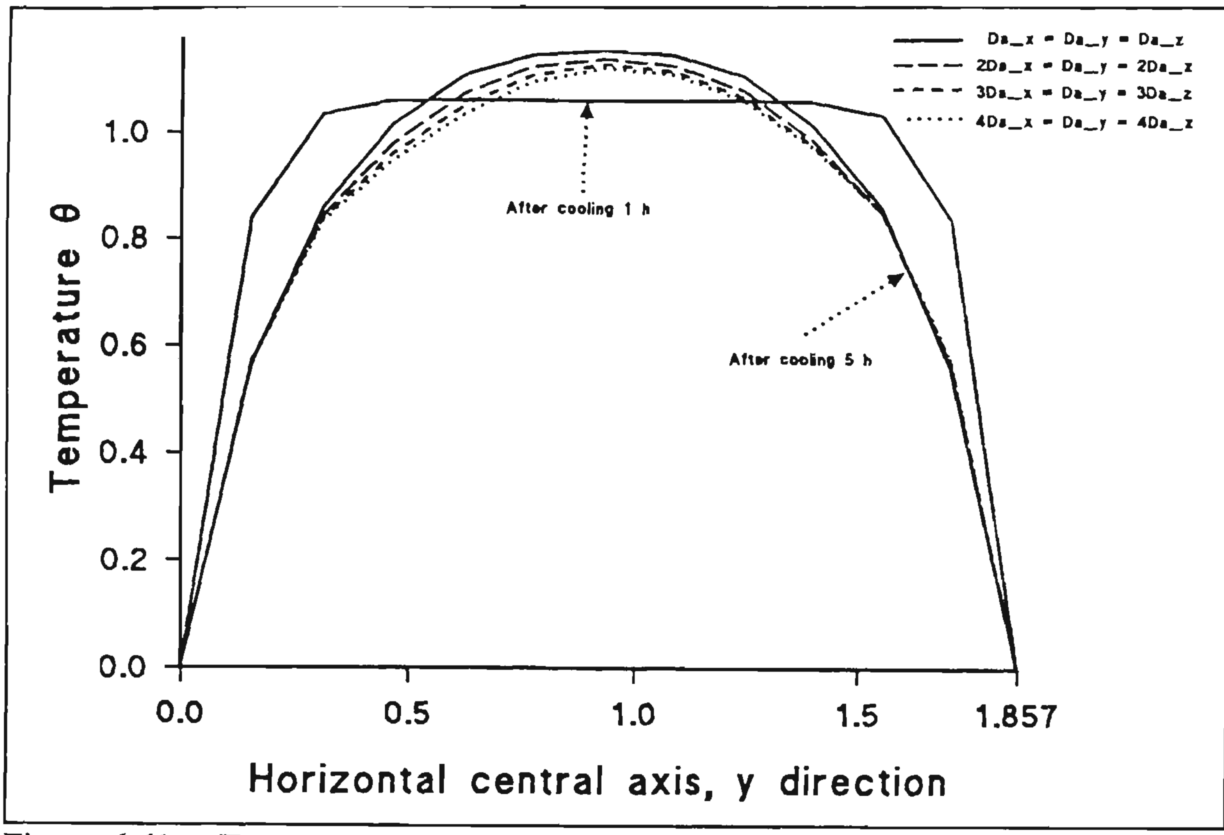


Figure 6.61 Temperature distributions along the central horizontal y axis for horizontally laid asparagus, adiabatic floor.

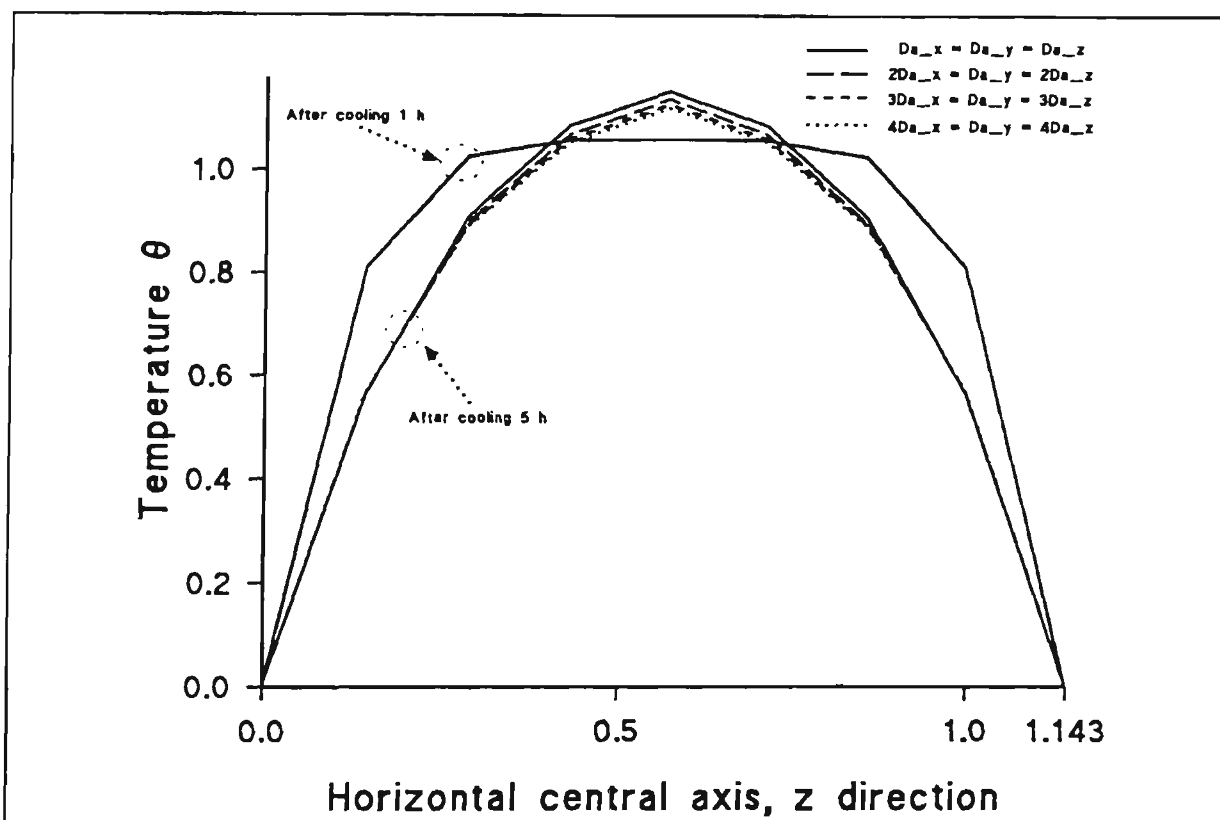


Figure 6.62 Temperature distributions along the central horizontal z axis for horizontally laid asparagus, adiabatic floor.

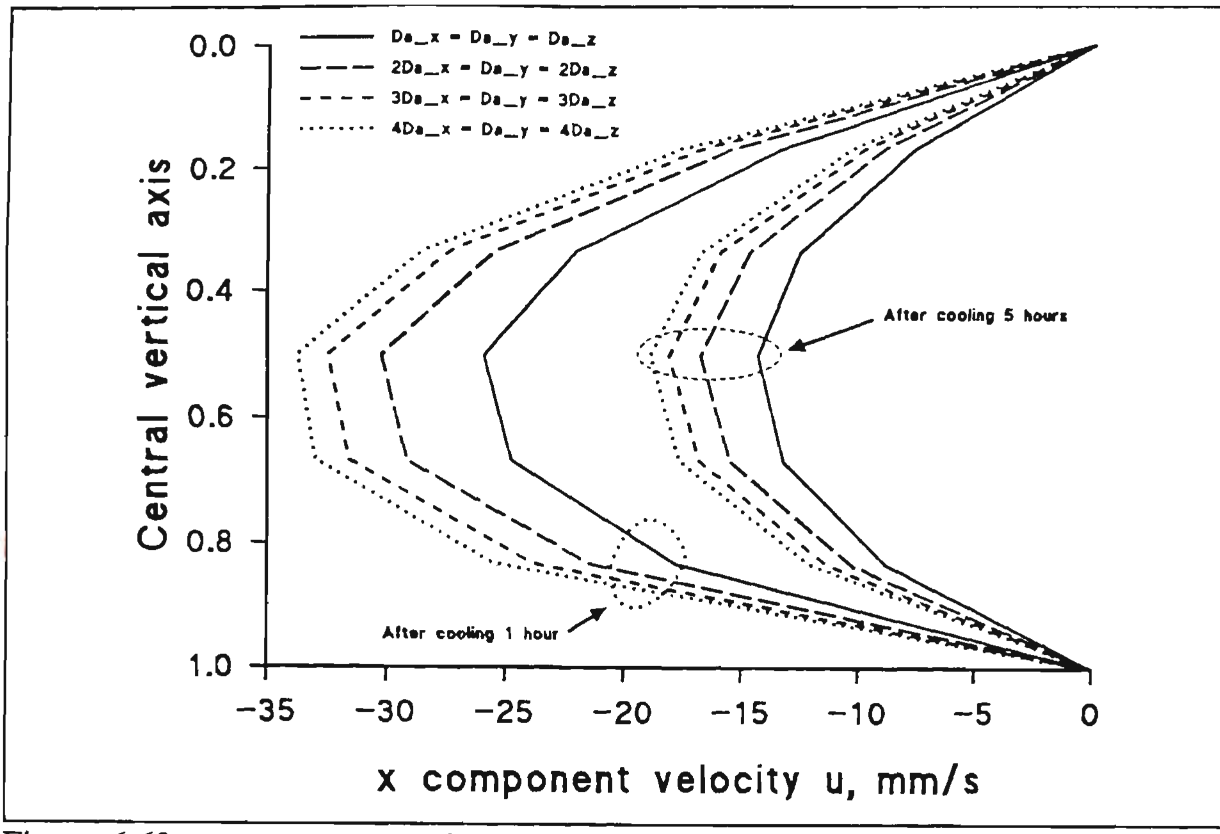


Figure 6.63 x-component of velocity, u, distributions along the central vertical x axis for horizontally laid asparagus, adiabatic floor.

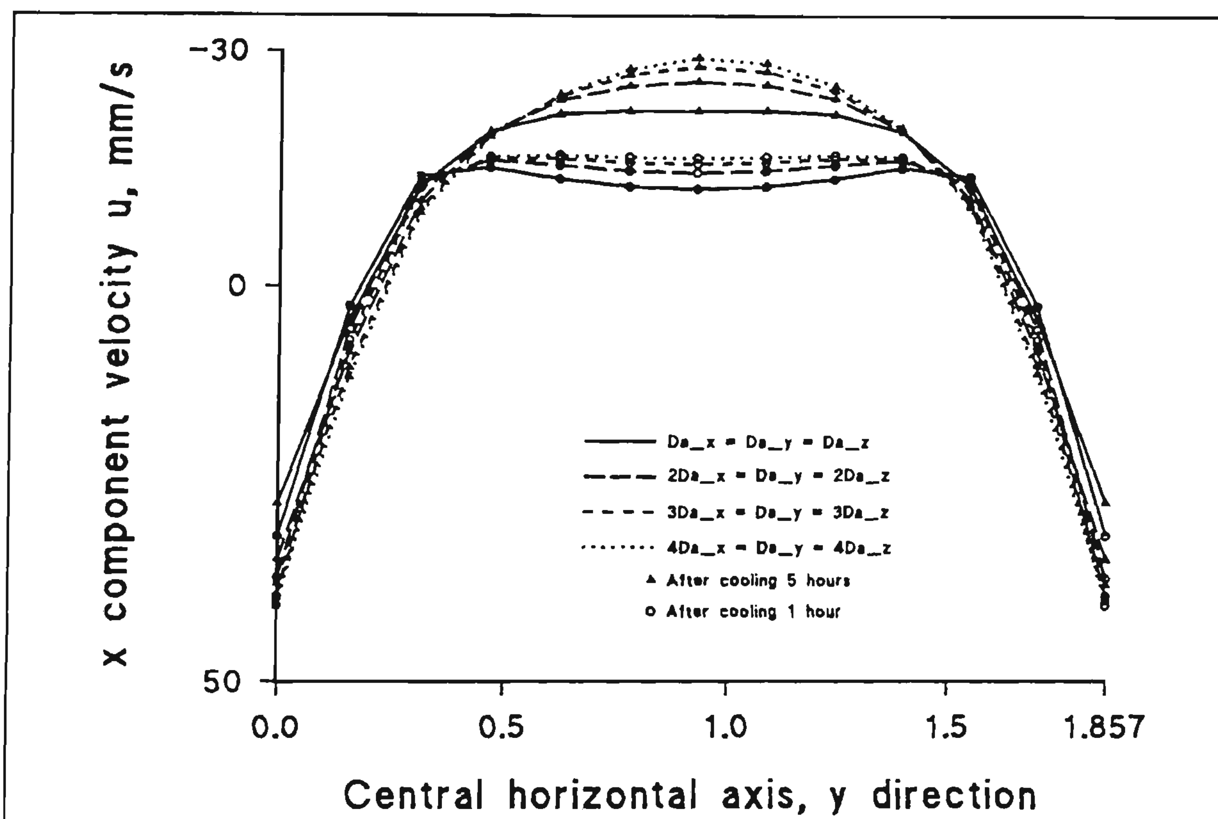


Figure 6.64 x-component of velocity, u, distributions along the central horizontal y axis for horizontally laid asparagus, adiabatic floor.

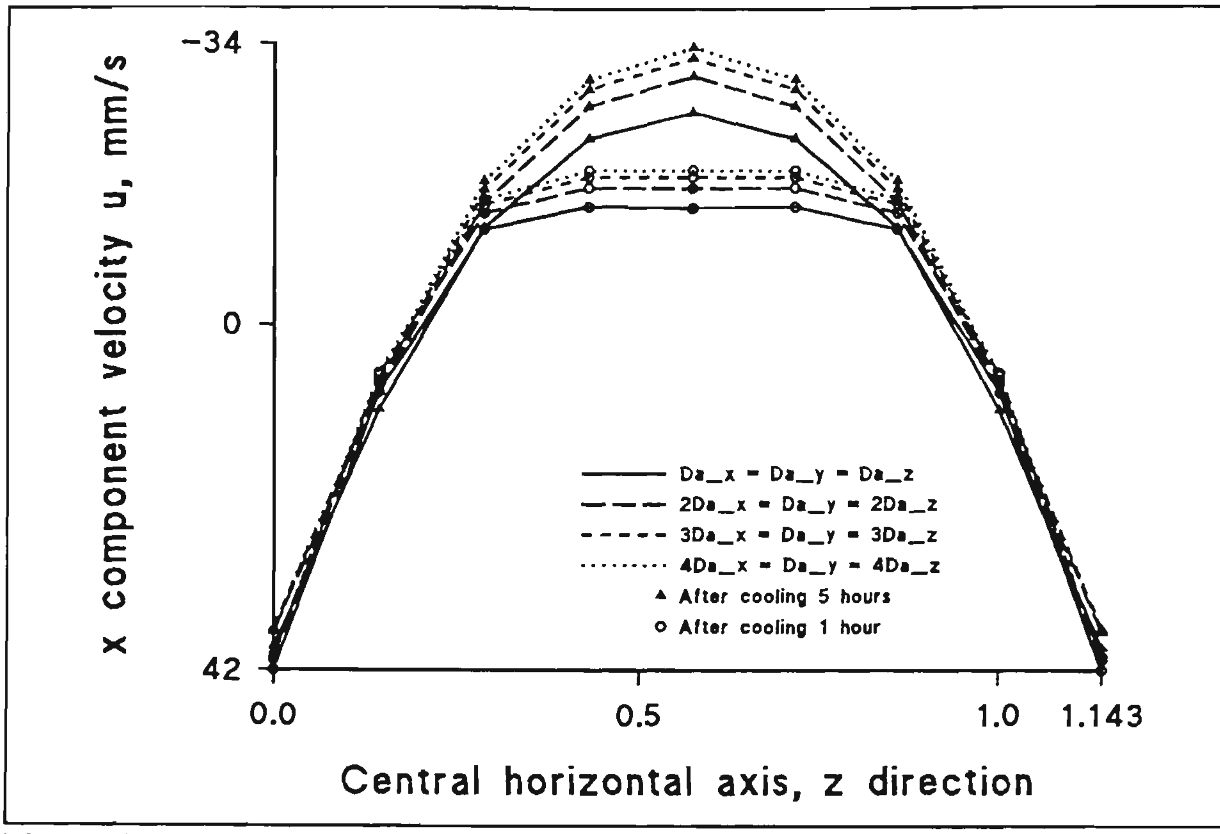


Figure 6.65 x-component of velocity, u, distributions along the central horizontal z axis for horizontally laid asparagus, adiabatic floor.

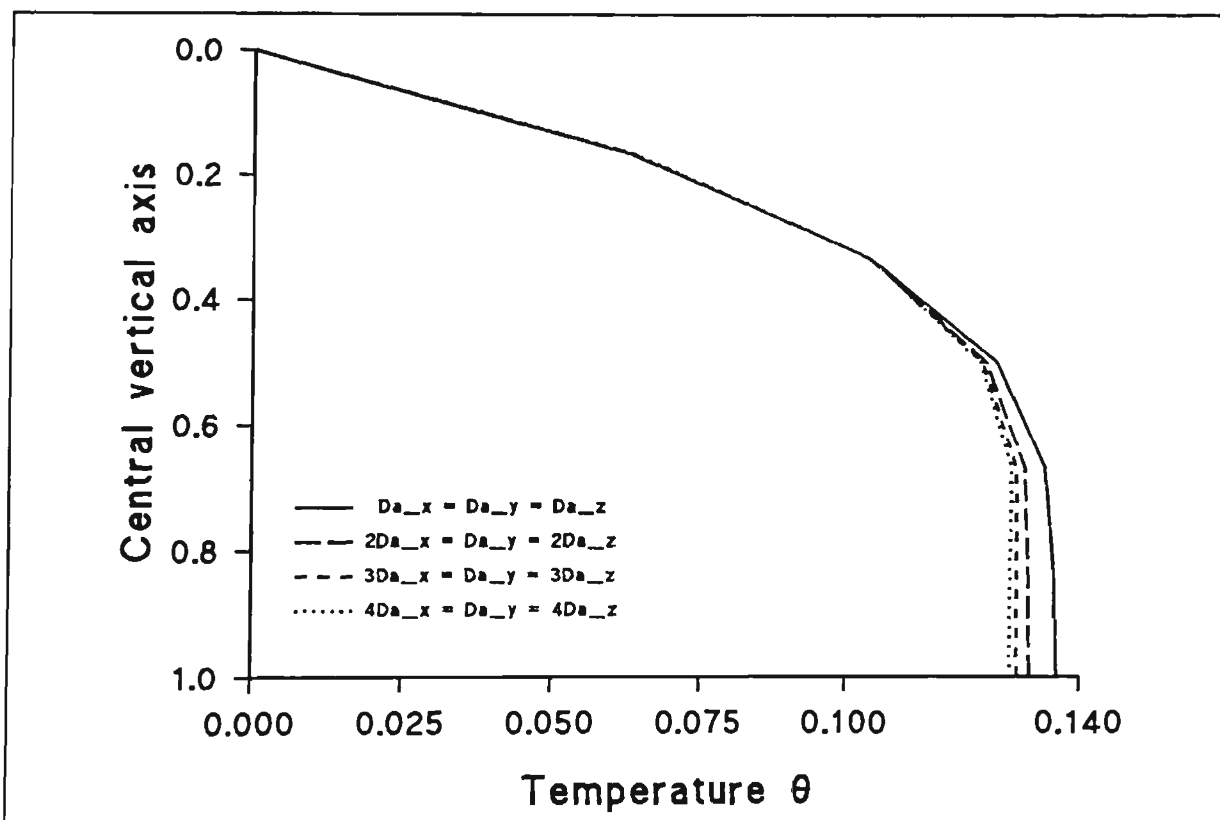


Figure 6.66 Temperature distributions along the central vertical x axis at steady state for horizontally laid asparagus, adiabatic floor.

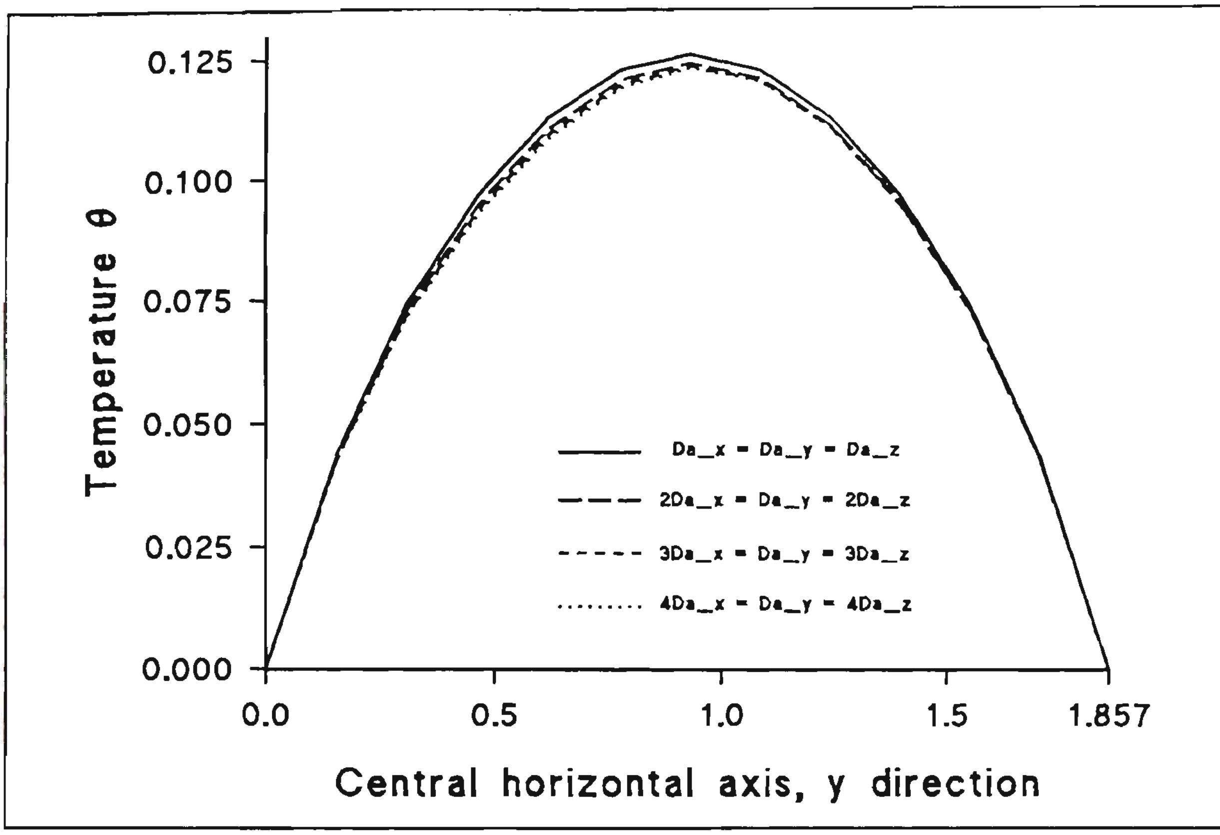


Figure 6.67 Temperature distributions along the central horizontal y axis at steady state for horizontally laid asparagus, adiabatic floor.

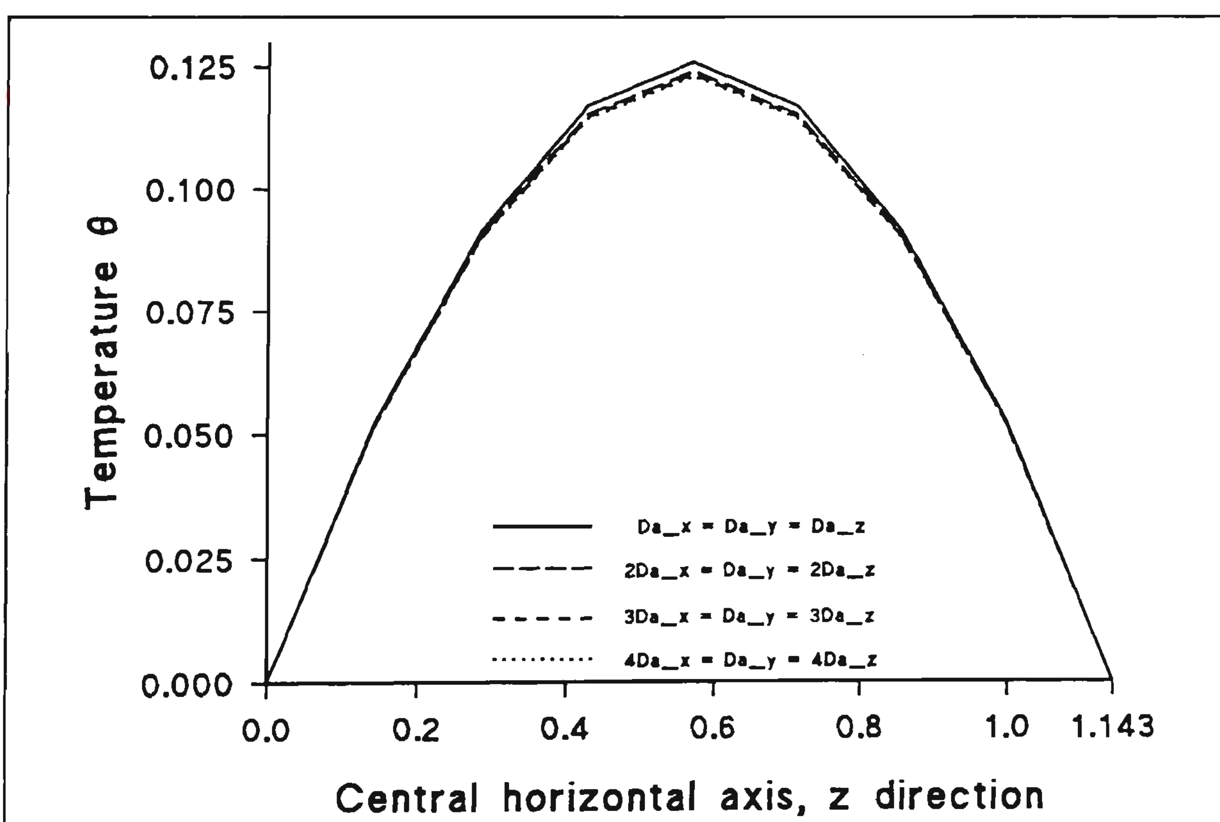


Figure 6.68 Temperature distributions along the central horizontal z axis at steady state for horizontally laid asparagus, adiabatic floor.

6.3.2 Vertically Laid Pack

In packed bed convective heat transfer processes significant differences are expected to occur when vertically packed beds and horizontally packed beds are used. The results presented in this section are for vertically laid carrots and asparagus, as in the horizontally laid pack, four different relationships between the components of permeability were used, for this case they were: $Da_x = Da_y = Da_z$; $Da_x = 2Da_y = 2Da_z$; $Da_x = 3Da_y = 3Da_z$; $Da_x = 4Da_y = 4Da_z$. The boundary conditions for this study are identical to those used for the horizontally laid pack.

The results of this numerical experiment are presented in identical sequence to the horizontally laid carrots and asparagus and Figure 6.69 throught to 6.88 encompass the entire set of results.

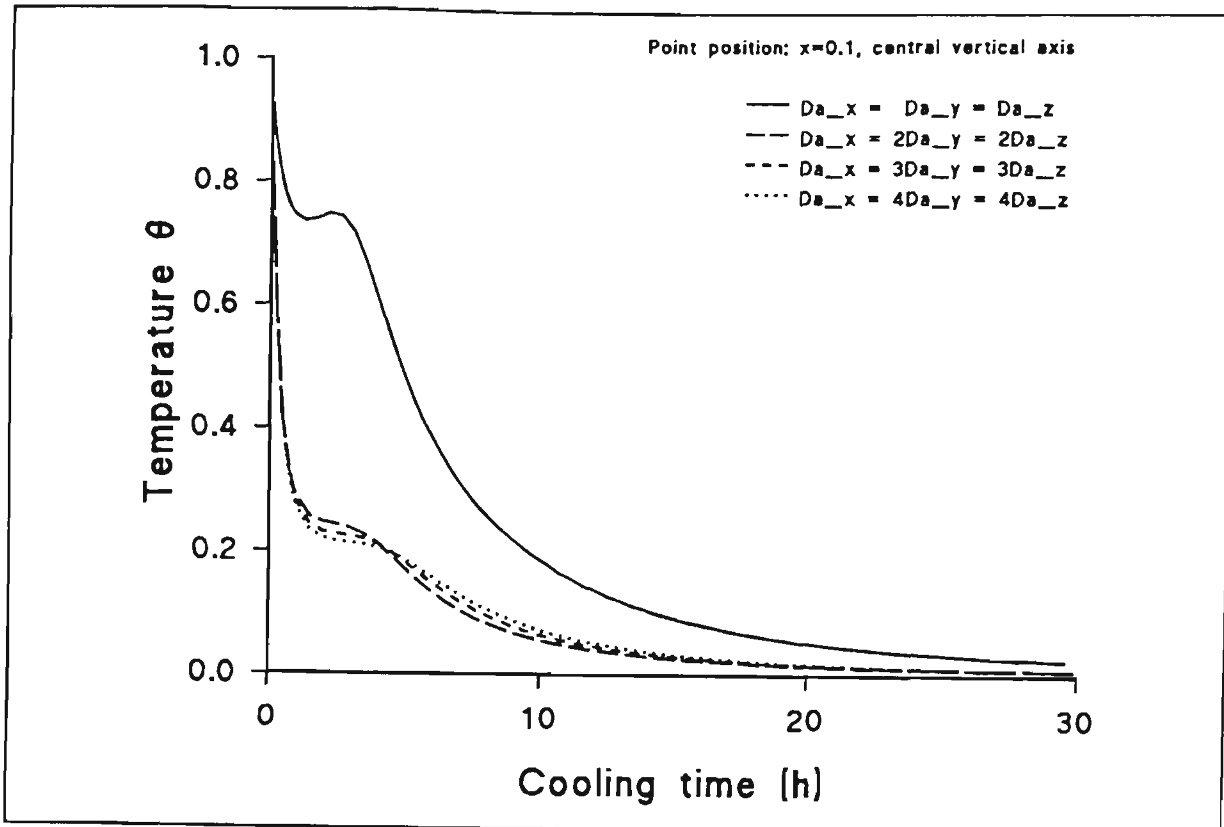


Figure 6.69 Temperature versus time curves for vertically laid carrots, adiabatic floor.

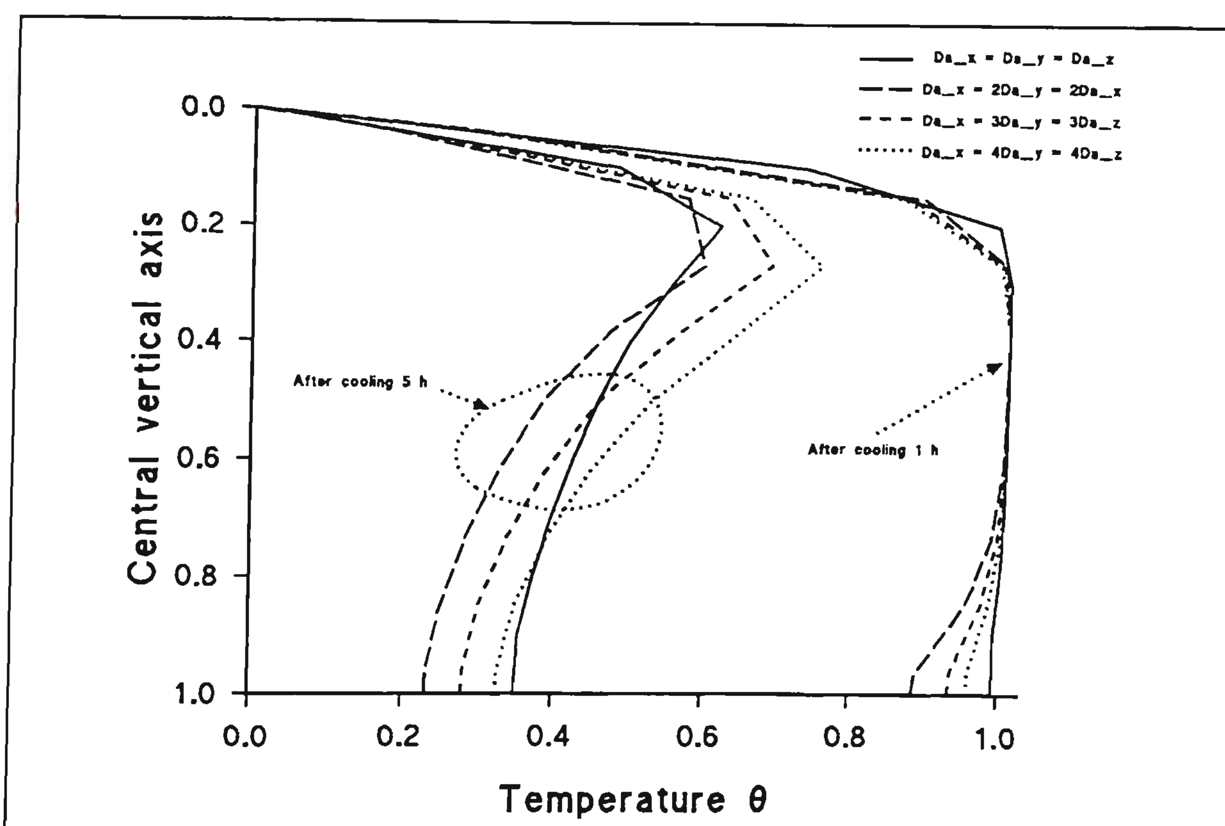


Figure 6.70 Temperature distributions along the central vertical x axis for vertically laid carrots, adiabatic floor.

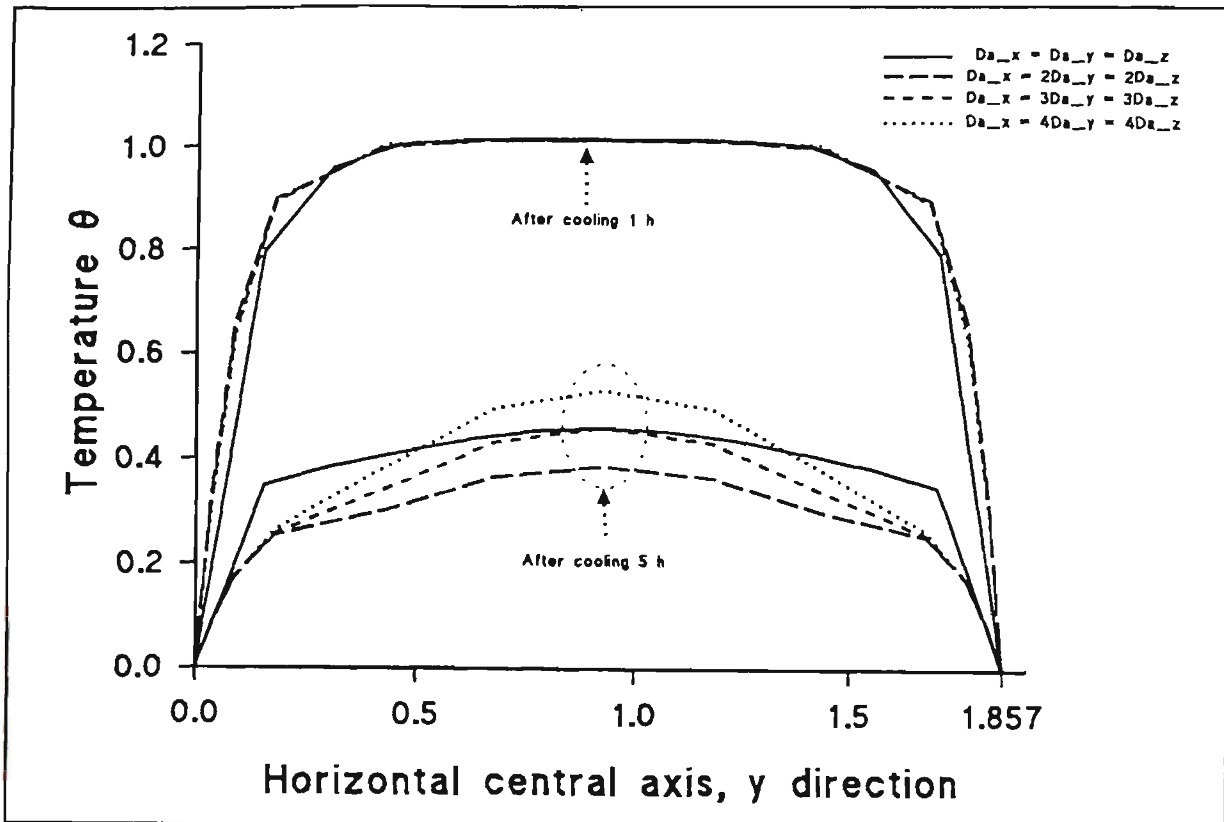


Figure 6.71 Temperature distributions along the central horizontal y axis for vertically laid carrots, adiabatic floor.

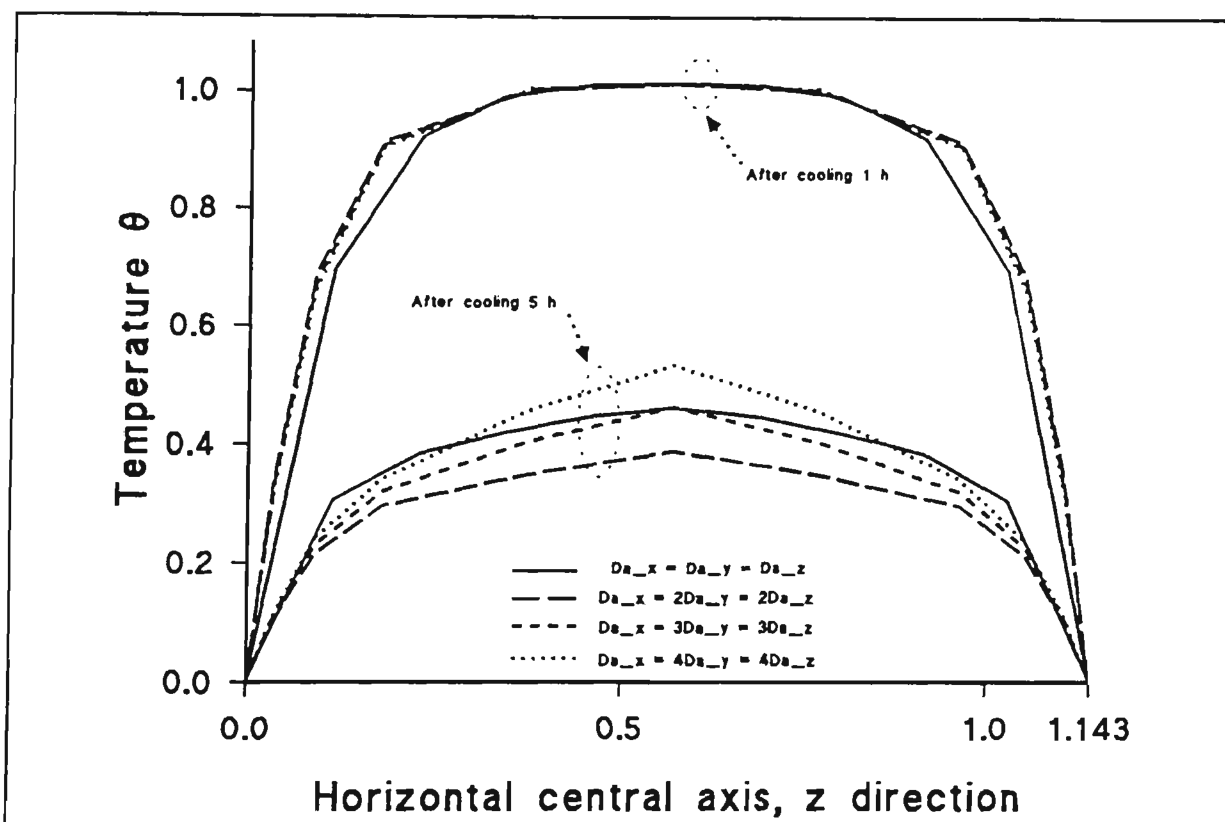


Figure 6.72 Temperature distributions along the central horizontal z axis for vertically laid carrots, adiabatic floor.

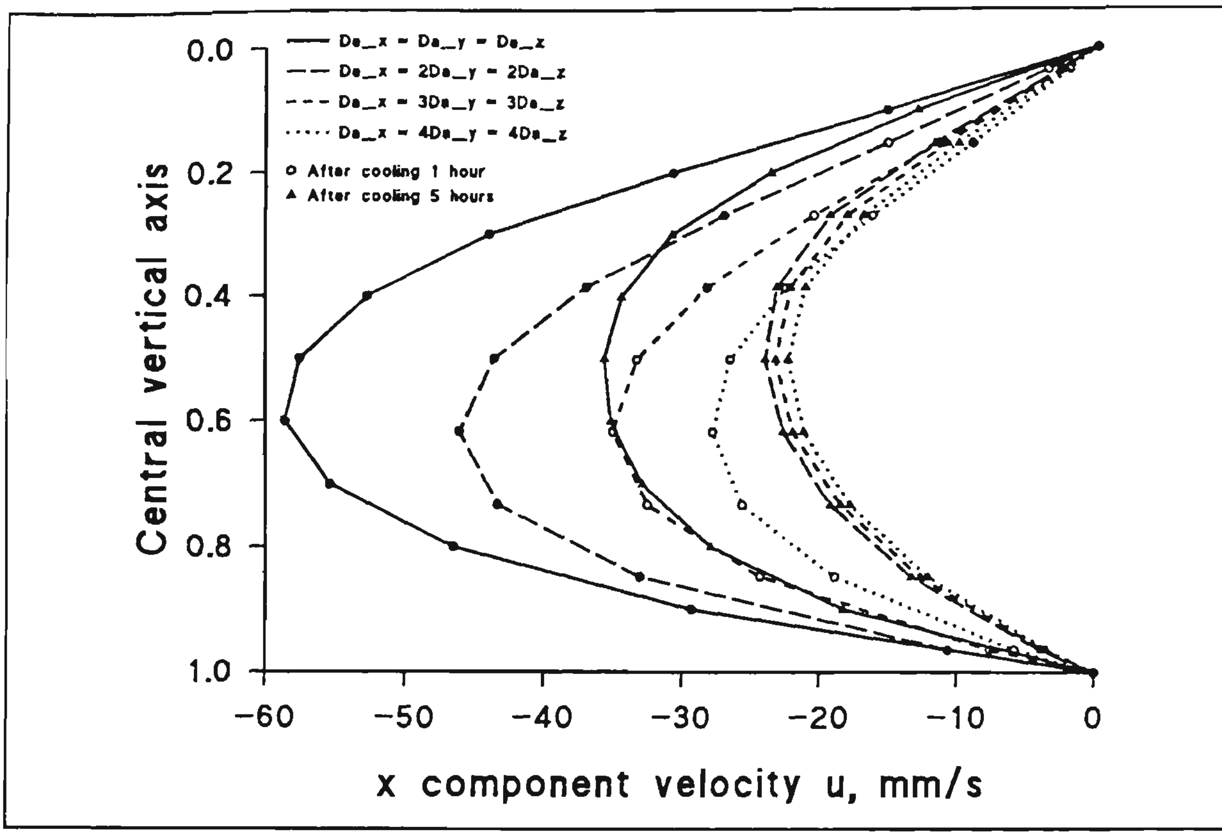


Figure 6.73 x-component of velocity, u, distributions along central vertical x axis for vertically laid carrots, adiabatic floor.

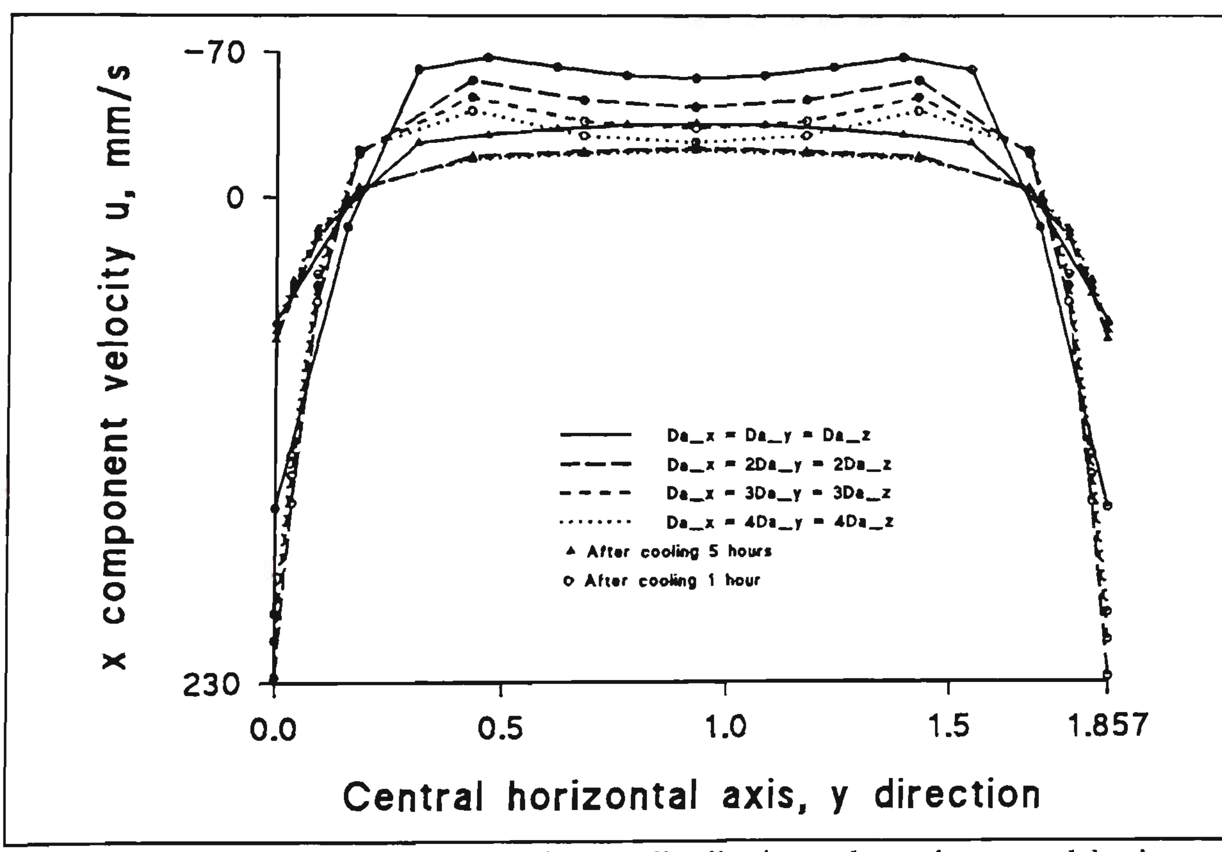


Figure 6.74 x-component of velocity, u, distributions along the central horizontal y axis for vertically laid carrots, adiabatic floor.

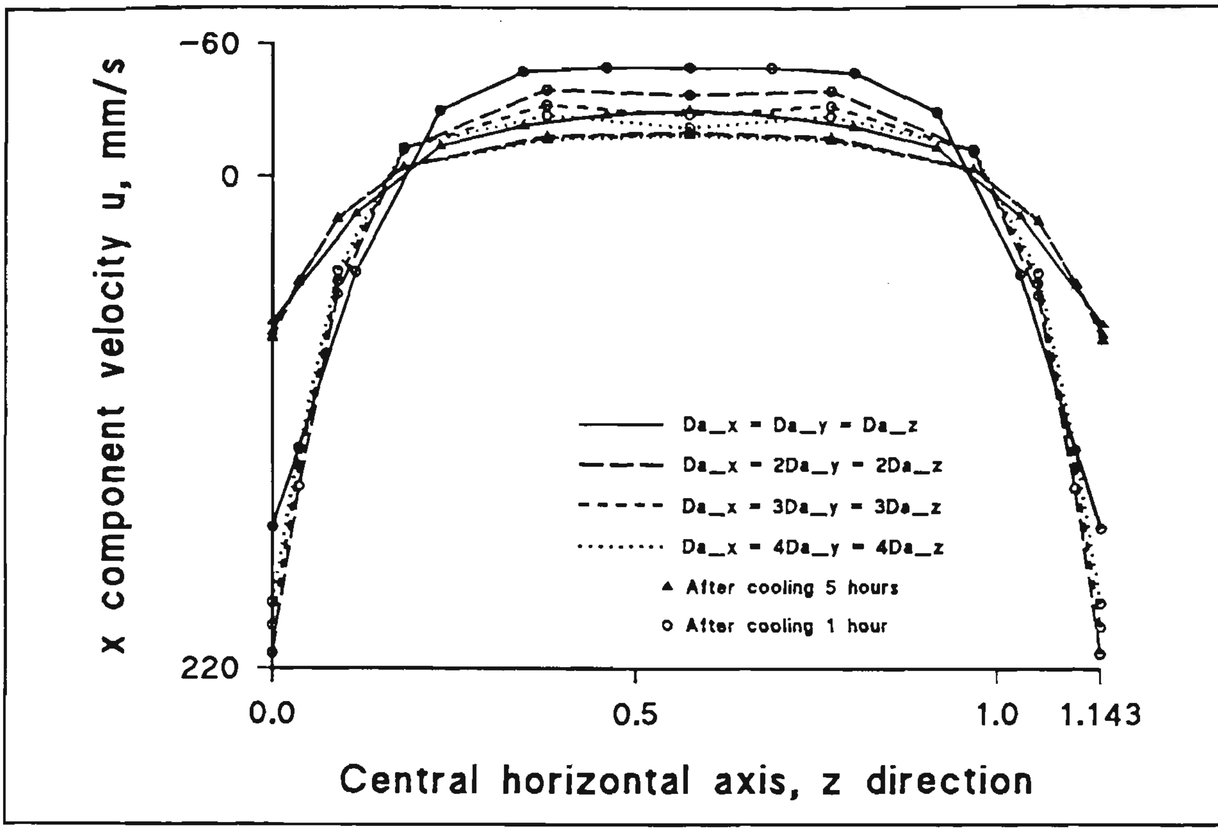


Figure 6.75 x-component of velocity, u, distributions along the central horizontal z axis for vertically laid carrots, adiabatic floor.

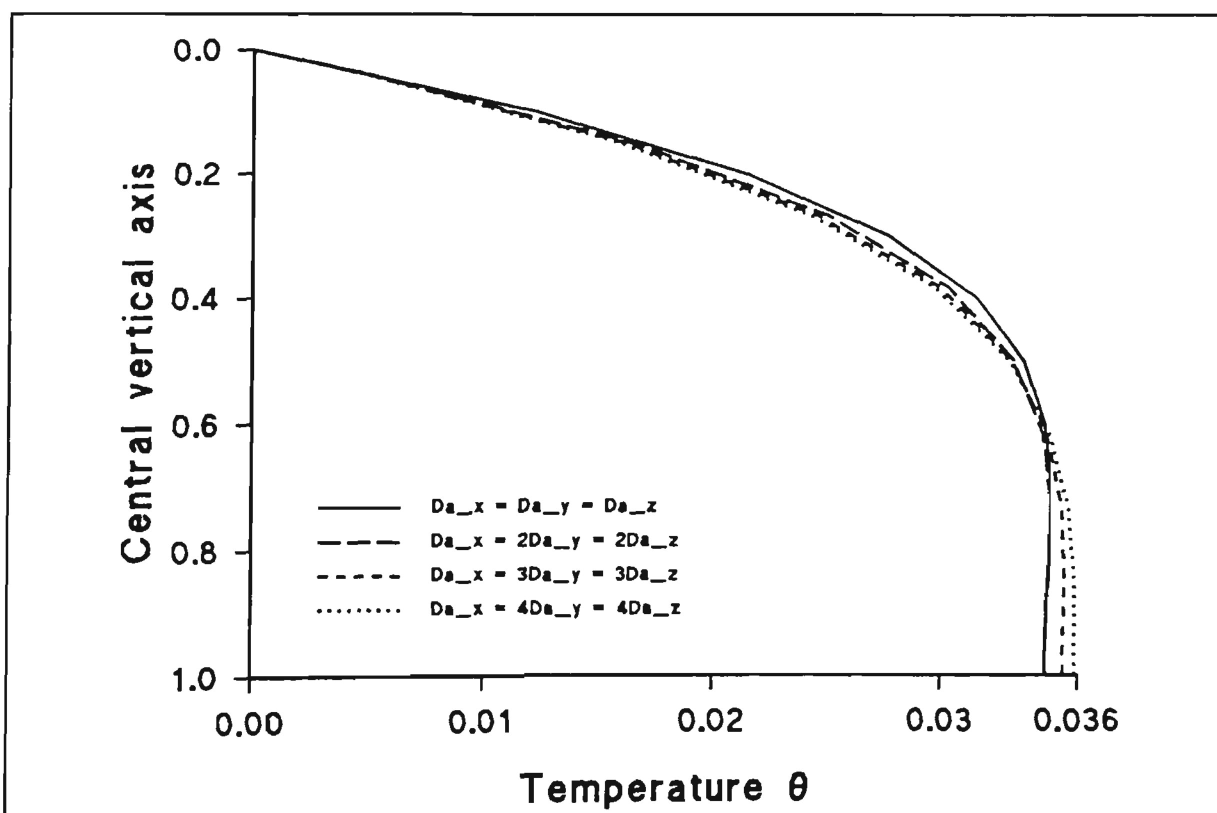


Figure 6.76 Temperature distributions along the central vertical x axis at steady state for vertically laid carrots, adiabatic floor.

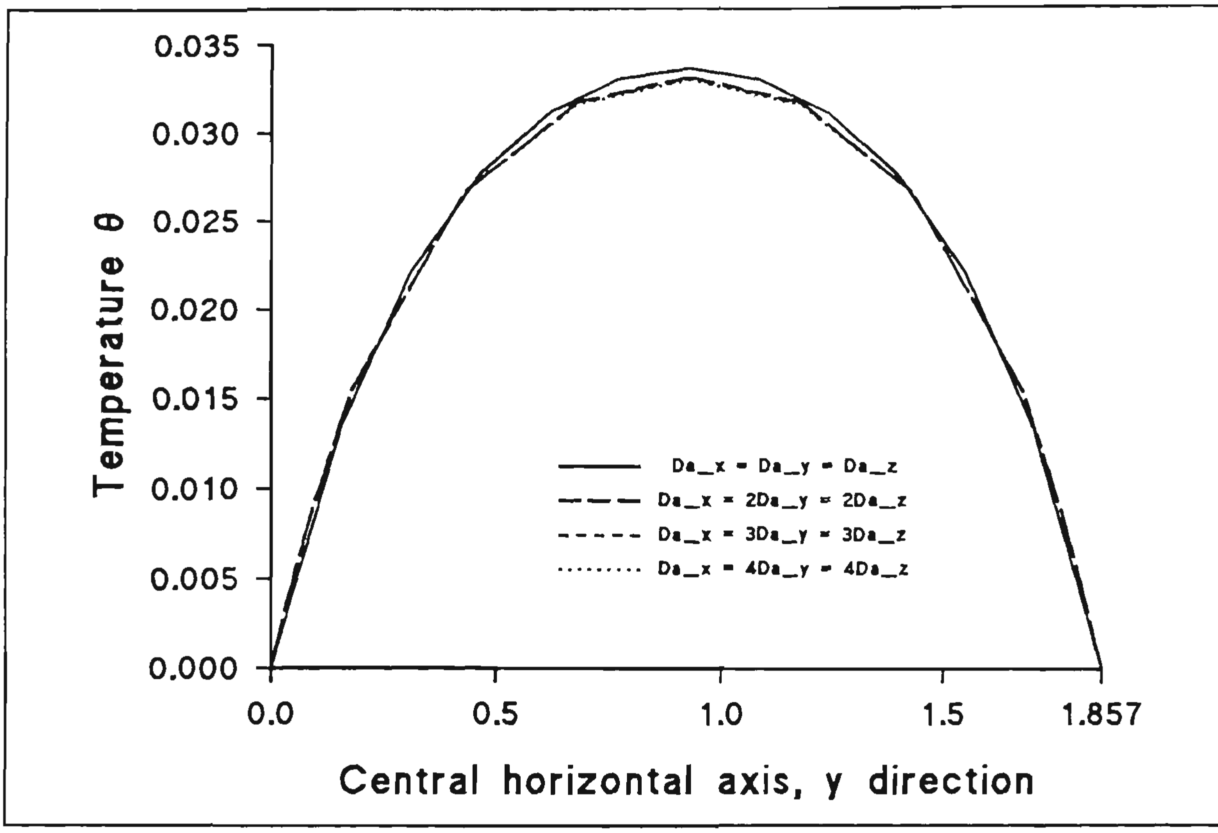


Figure 6.77 Temperature distributions along the central horizontal y axis at steady state for vertically laid carrots, adiabatic floor.

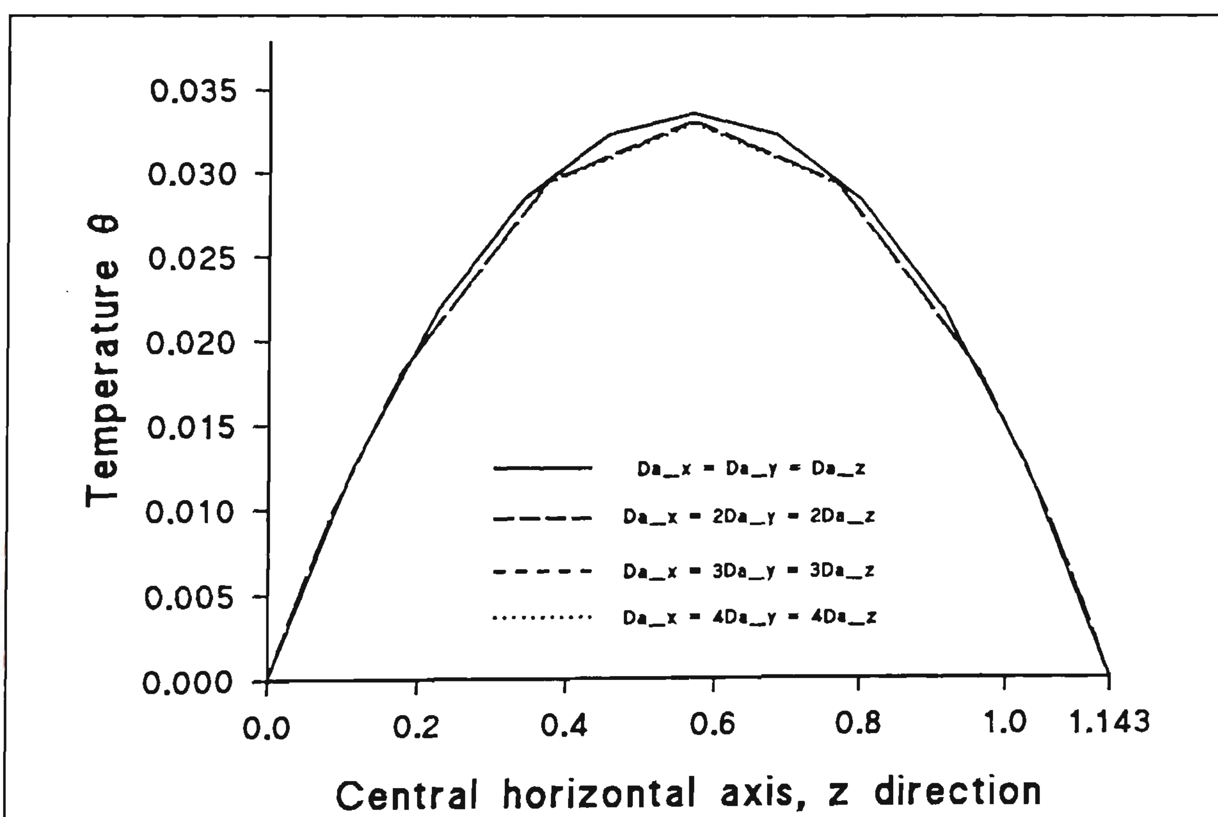


Figure 6.78 Temperature distributions along central horizontal z axis at steady state for vertically laid carrots, adiabatic floor.

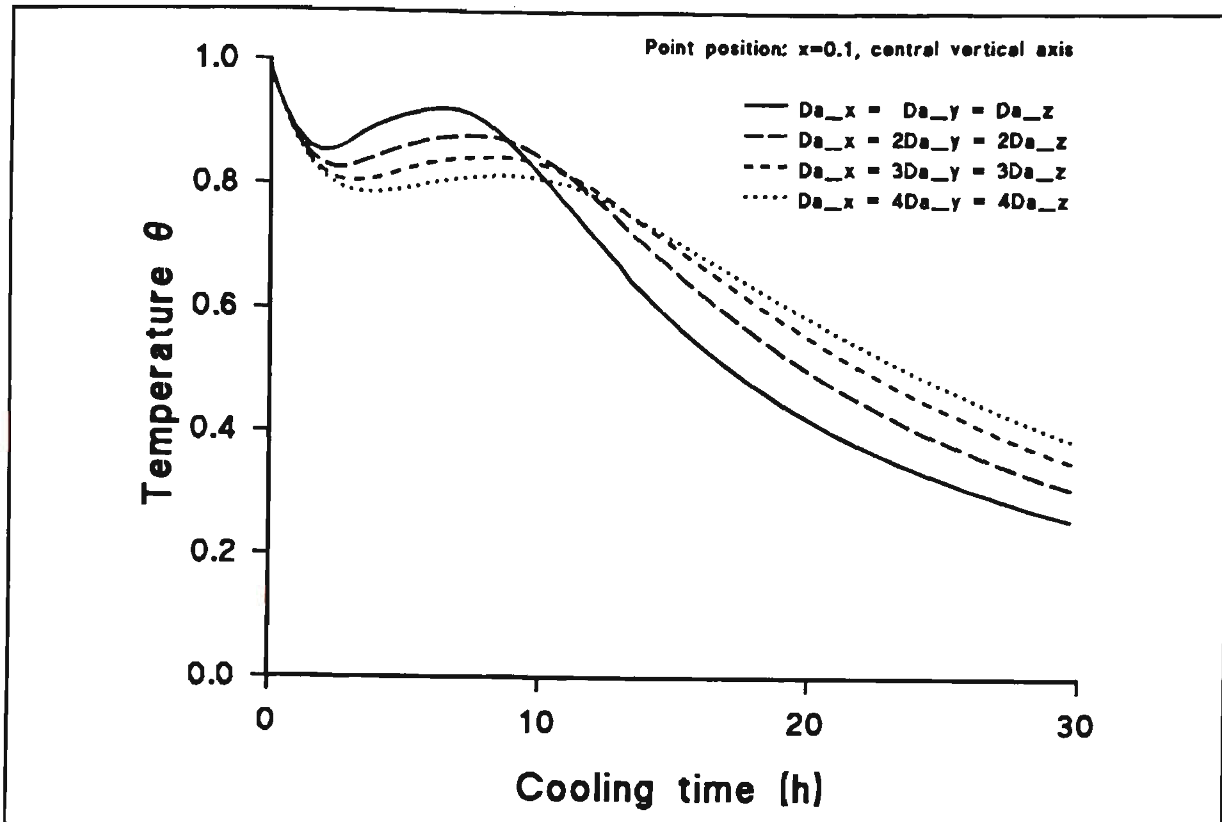


Figure 6.79 Temperature versus time curves for vertically laid asparagus, adiabatic floor.

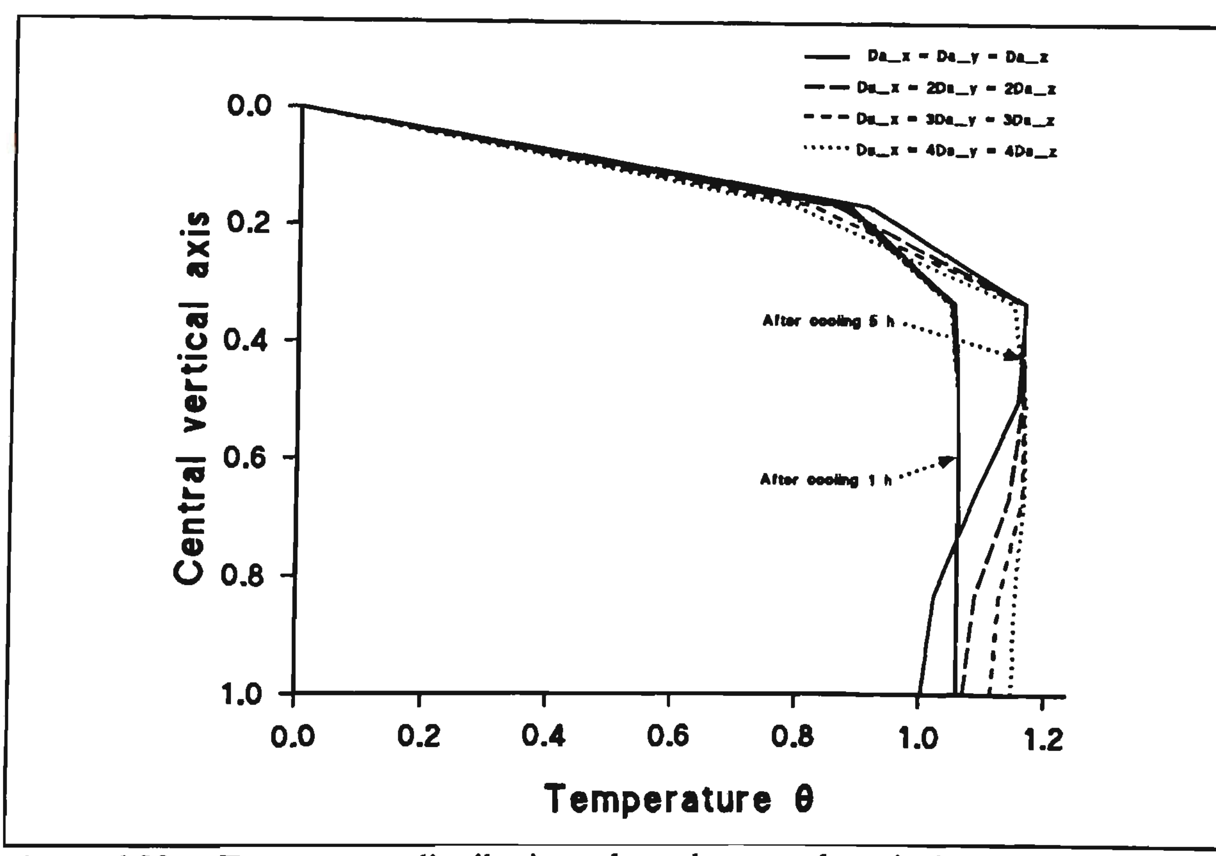


Figure 6.80 Temperature distributions along the central vertical x axis for vertically laid asparagus, adiabatic floor.

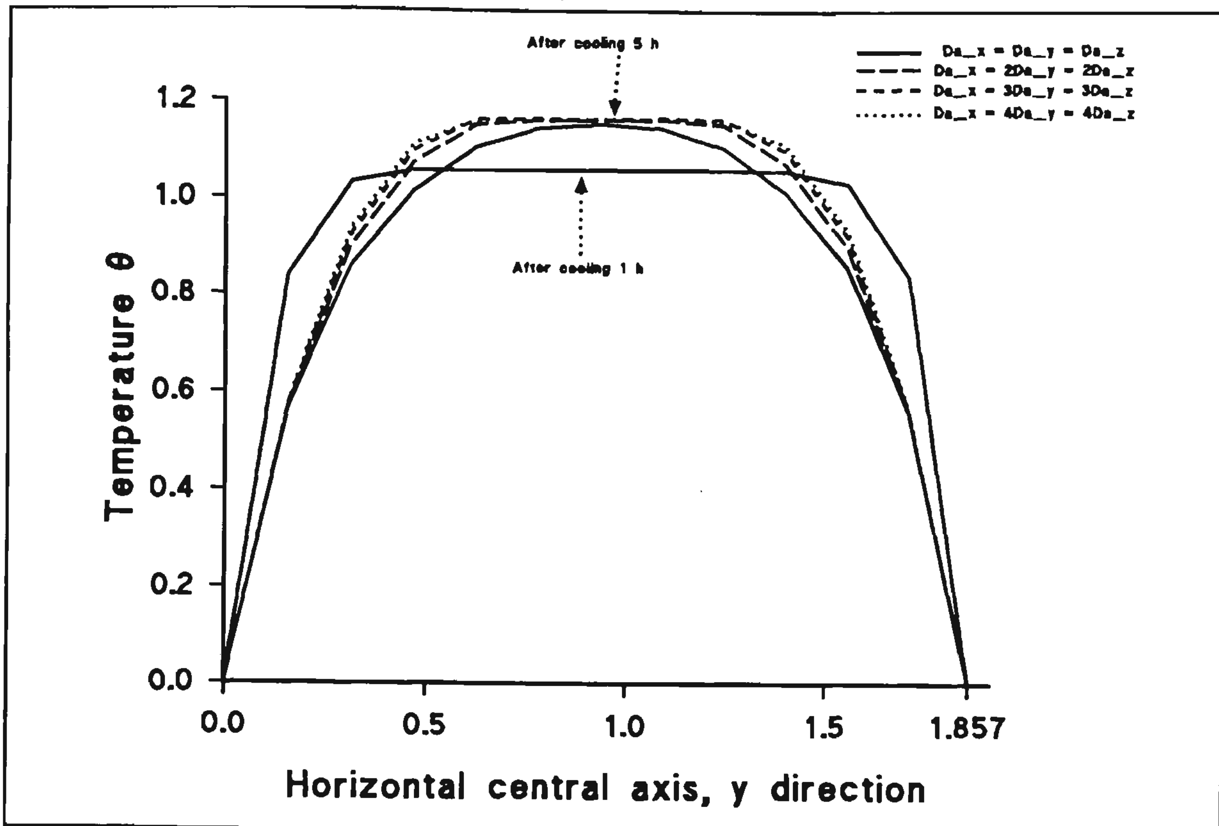


Figure 6.81 Temperature distributions along the central horizontal y axis for vertically laid asparagus, adiabatic floor.

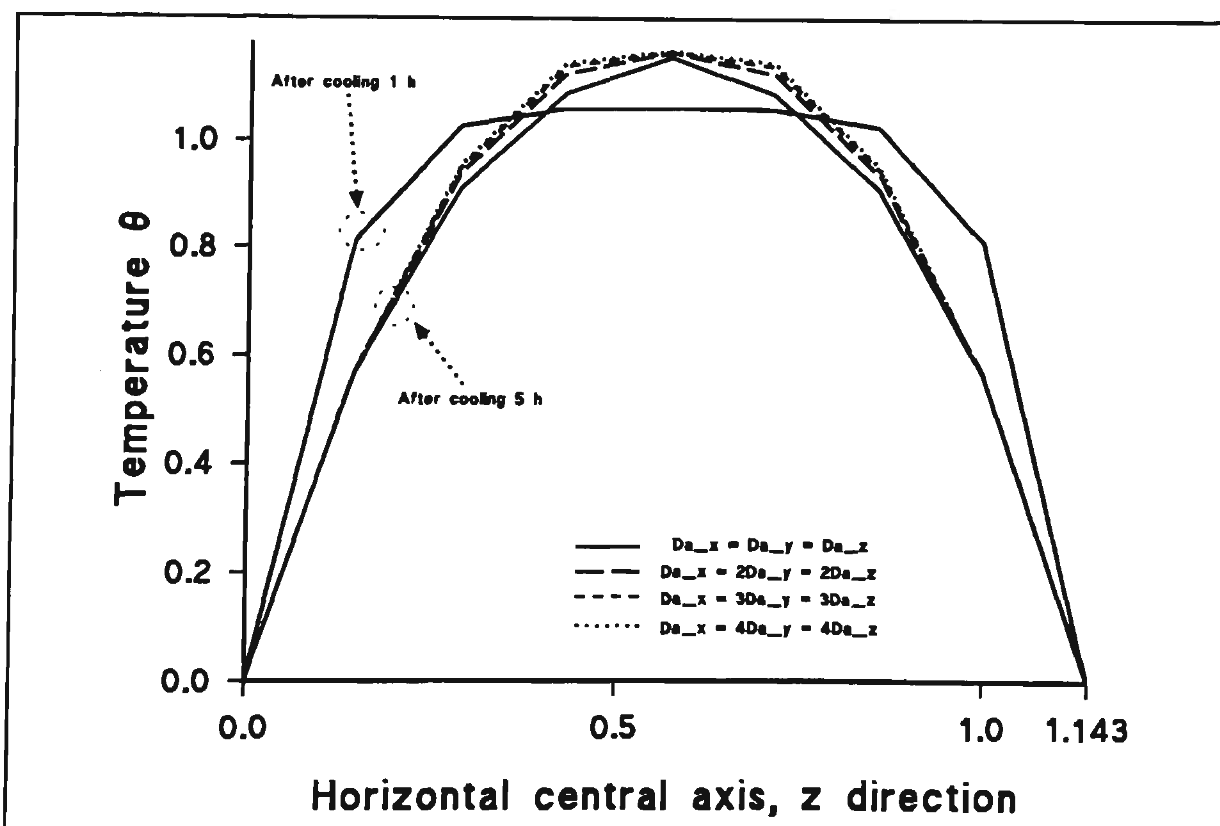


Figure 6.82 Temperature distributions along the central horizontal z axis for vertically laid asparagus, adiabatic floor.

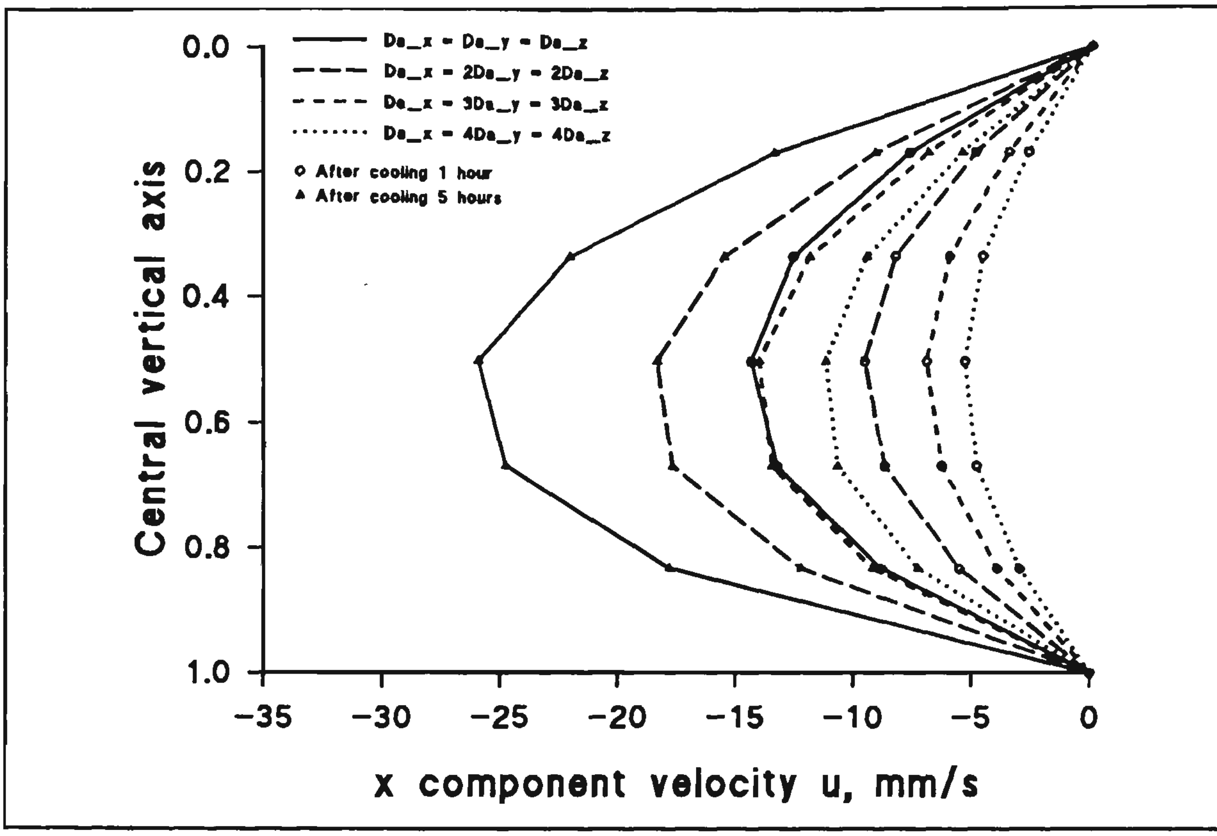


Figure 6.83 x-component of velocity, u, distributions along the central vertical x axis for vertically laid asparagus, adiabatic floor.

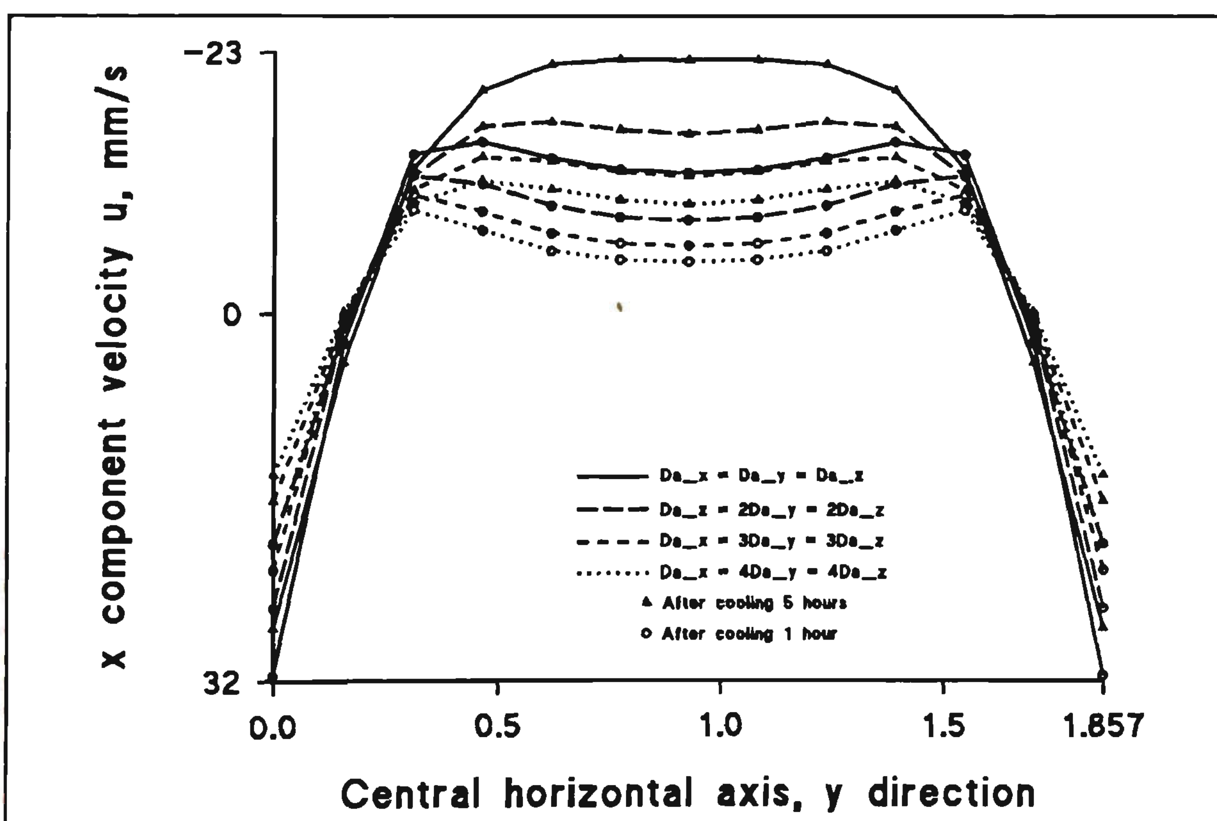


Figure 6.84 x-component of velocity, u, distributions along the central horizontal y axis for vertically laid asparagus, adiabatic floor.

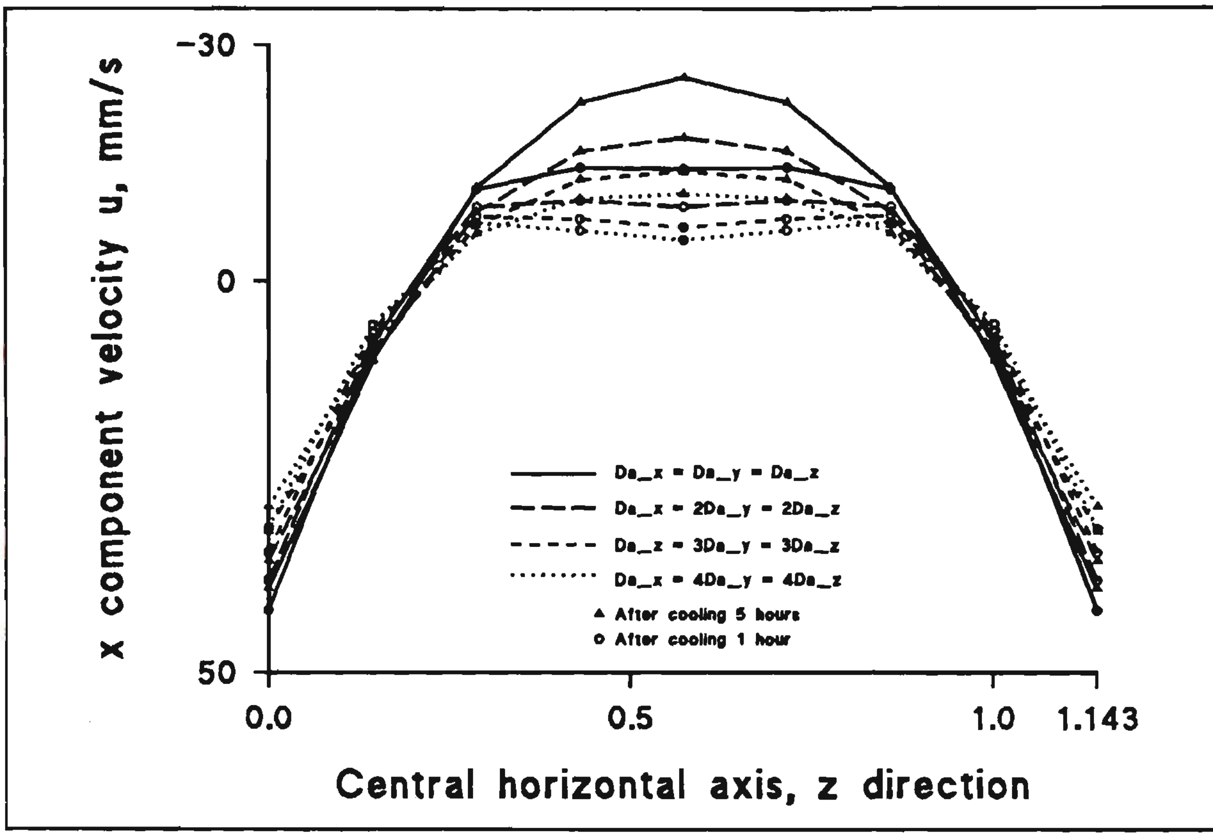


Figure 6.85 x-component of velocity, u, distributions along the cental horizontal z axis for vertically laid asparagus, adiabatic floor.

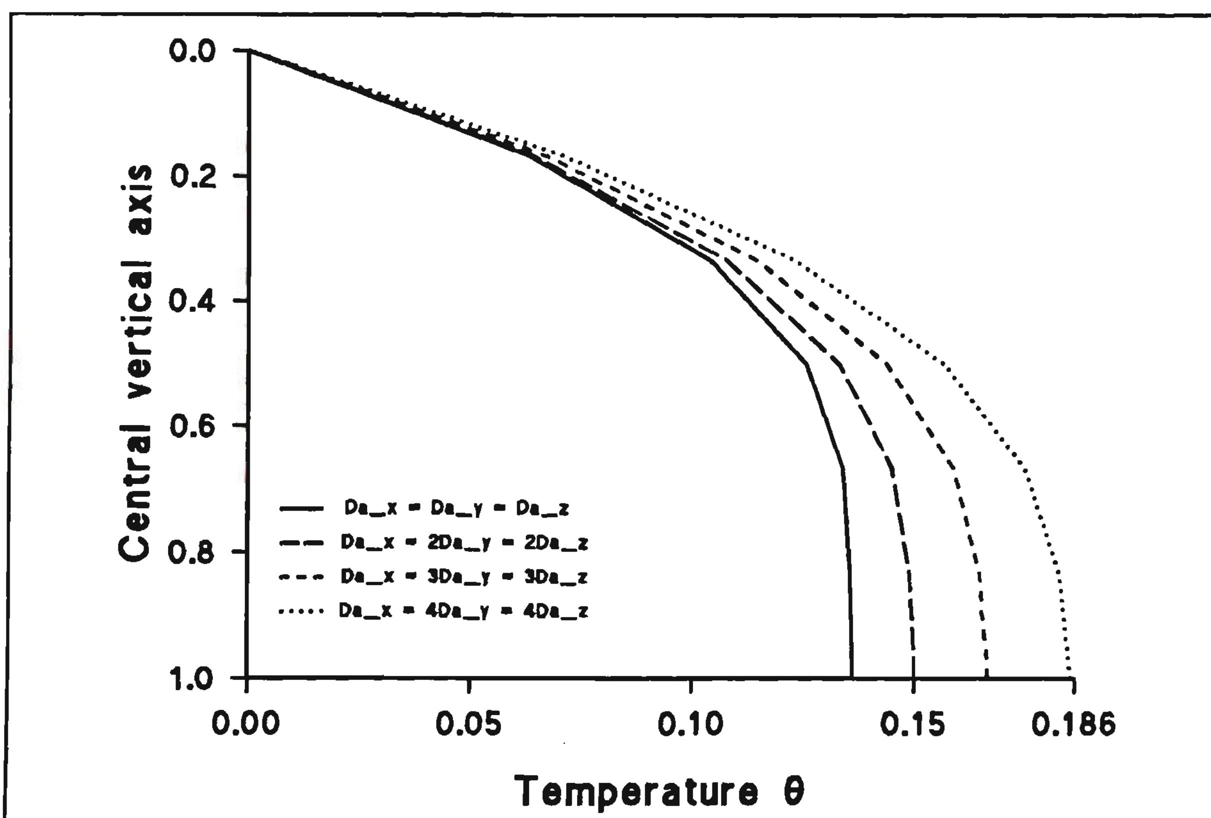


Figure 6.86 Temperature distributions along the central vertical x axis at steady state for vertically laid asparagus, adiabatic floor.

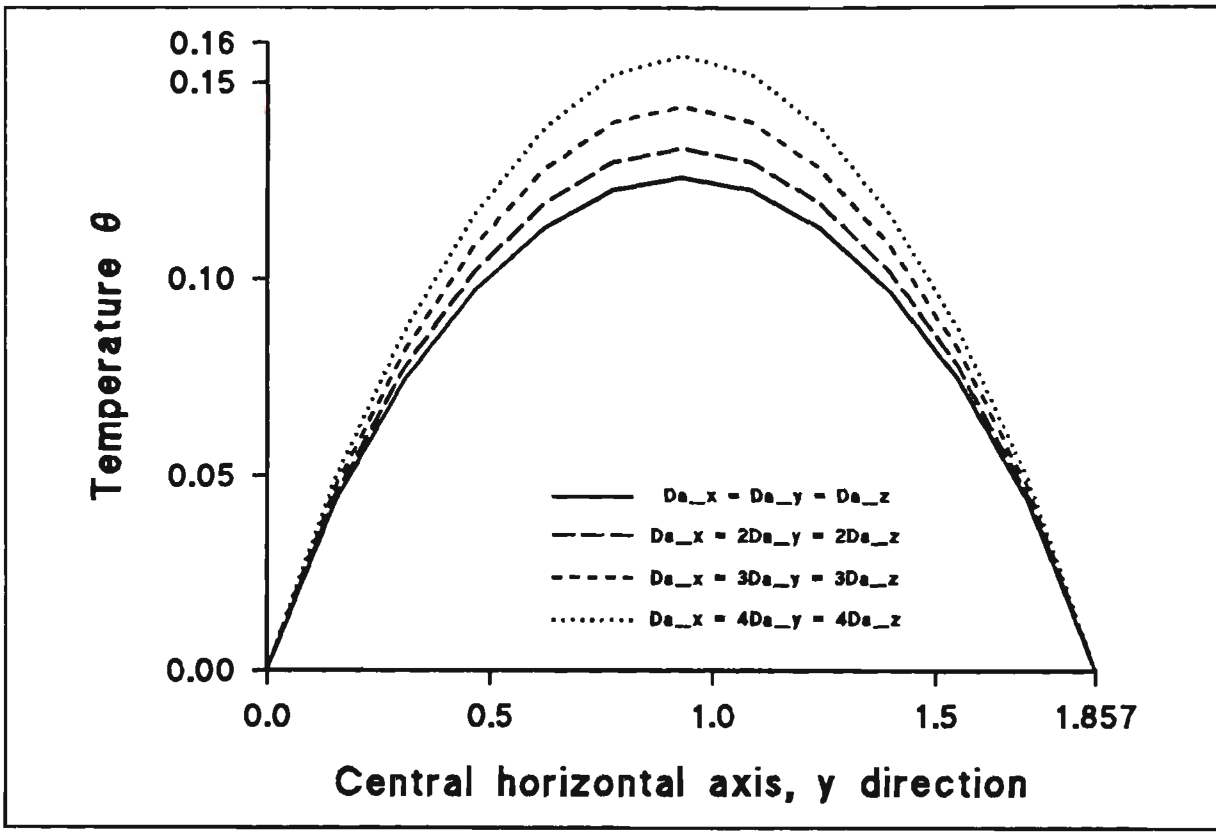


Figure 6.87 Temperature distributions along the cental horizontal y axis at steady state for vertically laid asparagus, adiabatic floor.

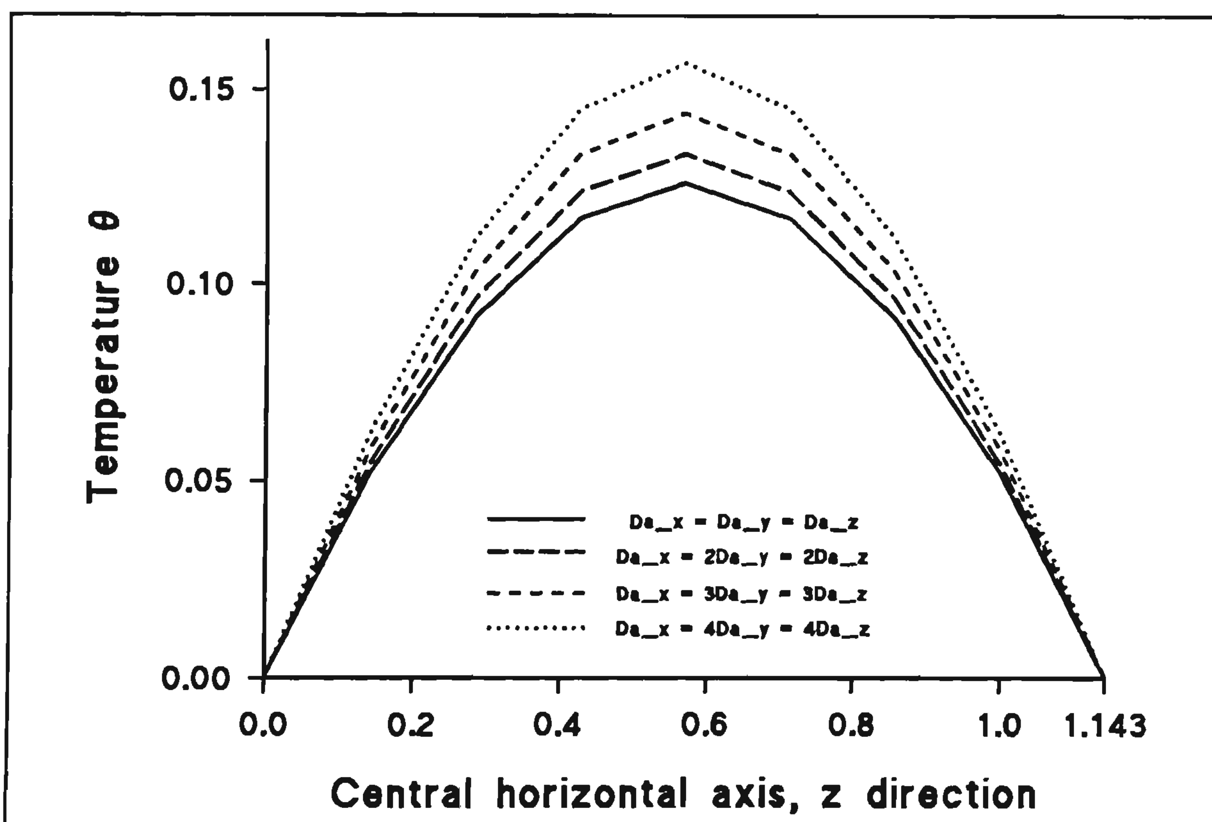


Figure 6.88 Temperature distributions along the central horizontal z axis at steady state for vertically laid asparagus, adiabatic floor.

CHAPTER 7

DISCUSSION

7.1 Numerical Model

In bringing the numerical model to an effective state, firstly the formulation of the underlying principles governing observed phenomena in cooling or heating of stored agricultural produce was accomplished. As distinct from most other established practice, the propositions advanced in this study were concerned with treating transient three-dimensional natural convection in stored respiring produce, as that in a confined porous media with internal heat generation and isotropic or orthotropic permeability. Secondly, the solution of the set of non-linear partial differential equations developed to describe the heat transfer process in packed agricultural produce was obtained numerically, using the method of false transient and the Samarskii-Andreyev implicit alterative direction iteration scheme, as outlined previously. Finally, the model was evaluated and so far as possible compared with other work before applying it to a number of practical applications. It was found the convergence, accuracy and computing time required to obtain solutions depended on a number of factors. These included: mesh type, grid size, time step, false transient parameters, convergence criterion, internal heat source function and container size.

Results obtained using the model were compared with the numerical and experimental results presented by Beukema (1980, 1983), and shown in Chapter 5; the comparison was made using the same internal heat generation, physical property values and boundary conditions.

The results were in good agreement, but Beukema's model does not include a temperature dependent respiratory heat generation function, so realistic comparisons with our results for isotropic fruits and vegetables were not possible. In fact, we were unable to find other work with which to compare our numerical results. This study has provided a valuable perception of the problems associated with the storage and distribution of agricultural produce, but until more detailed information on permeability and respiration rates are available, and the computed numerical results are borne out reasonably well experimentally, we should at the present stage of its development, treat the results obtained with a degree of reserve. In the following sections the results of the study are discussed and ideas for further research are suggested.

7.2 Temperature Distribution

The cooling processes for apples and Brussels sprouts in a standard container with adiabatic or isothermal floor are shown in Figure 6.1, 6.13, 6.25 and 6.37. As expected the temperature changing profiles were different. With isothermal floor the temperatures dropped faster at corresponding positions than those with adiabatic floor. It indicated that the isothermal floor improved the cooling process. The difference of these boundary conditions affected most on the temperature distributions in the lower part of the container, as shown in Table 7.1.

The difference at the bottom, x=0.964, reached up to 80%. At the centre, x=0.5, there was only about 0.02-2.5% difference in temperature between these two boundary conditions. Air was cooled by the side walls and moved downward to the bottom. With the adiabatic floor,

Table 7.1 Comparison of temperature values along the central vertical axis for apples and Brussels sprouts.

Position x	θ_{app_adb}	θ_{app_iso}	θ_{Brsl_adb}	θ_{Brsl_iso}
0.0360	0.3490	0.3473	0.2871	0.2869
0.1520	0.9570	0.9549	0.8966	0.8963
0.2680	1.0123	1.0104	1.0234	1.0232
0.3840	1.0055	1.0008	1.0340	1.0338
0.5000	1.0048	0.9795	1.0342	1.0328
0.6160	1.0017	0.9308	1.0339	1.0251
0.7320	0.9828	0.8279	1.0331	0.9862
0.8480	0.9457	0.6260	1.0307	0.8193
0.9640	0.8574	0.1605	1.0238	0.2411

Note: 1. θ_{app_adb} = temperature of apples with adiabatic floor;
2. θ_{app_iso} = temperature of apples with isothermal floor;
3. θ_{Brsl_adb} = temperature of Brussels sprouts with adiabatic floor;
4. θ_{Brsl_iso} = temperature of Brussels sprouts with isothermal floor;
5. The values here were obtained at 1 hour cooling time.

the cold air was first warmed up by the produce before it reached the central region of the lower part, thus a small temperature difference between air and produce resulted. With the isothermal floor, when air moved into the central area, it was still cooled by the cold floor and this enabled the air temperature to be lower than of the produce, even in the central area near the bottom. But when the air left the cold bottom to move up, it was heated to near the produce's temperature. This is why there was a big difference in the bottom area and a small one in the central and upper area when adiabatic and isothermal floor were introduced. There was a overall improvement when using the isothermal floor but the temperature behaviour in the central and upper area were quite similar.

When the steady state was reached, the temperature profiles were also different. With the adiabatic floor, the highest temperature was located at the bottom centre. With the isothermal floor the highest temperature was lower than that of the adiabatic floor and was located at the centre of the container.

These results suggest that in cooling practice, attention should be paid to the central and upper area where produce are not easily cooled for both adiabatic and isothermal floor conditions. While in storage practice, if an adiabatic floor is used, the produce at the bottom might undergo damage most easily.

An interesting lower temperature decrease after cooling for a period time can be observed at upper positions, as shown in Figures 5.2, 5.3, 6.1, 6.13, 6.25 and 6.37. This can be explained as follows. At the beginning of cooling, the temperature differences in the container were small and thus natural convection was weak. Heat transfer could be mainly attributed to conduction. A temperature rise, due to the internal heat generation took place in the cental area along with a rapid temperature drop near the cold walls. This process could last hours and led to a significant temperature difference as the natural convection developed completely. The air flowed downwards near the cold walls and upwards in the central area in the container. The rising air was warmed resulting in heat transport from the centre or bottom to the upper part of the container.

The respiration heating also had a significant effect on temperature behaviour. Brussels sprouts have a much higher rate than apples do. It also took much longer time to cool down to the same temperature level. As a comparison with Figure 6.25 and 6.37 we can find that at x=0.268, it took about a hour to cool apple to 6°C and about 18 hours for Brussels sprouts. The temperatures for Brussels sprouts at steady state were much higher than those for apples. This is only because of the high heating rate.

With the symmetric system, the temperature distributions were also symmetric as shown in

Appendix C. The contour map of the temperature distribution shows a clear view of the temperature field and its changing progress inside the container.

7.3 Velocity Distribution

Velocity distribution is relative to temperature changing. As shown in Figures 6.1, 6.2, 6.13, 6.14, 6.25, 6.26, 6.37 and 6.38, both rapid temperature changes and high velocity happened within the same time period. When the temperature difference was large, the velocity was also large. This is because with large temperature differences the buoyant force was also large, thus air movement was strong. The strong air movement inside the container started at a certain time after cooling commenced, about 1.0 hour for apples having an adiabatic floor, and reached its maximum value later, about 1.5 hours for apples. In the lower part of the container, velocity first reached its peak value, then those in the central and upper part. It indicates that the commodities in the bottom would be cooled first. This was also reflected in the temperature distribution figures. After a period of cooling, the temperature difference inside the box was small again and thus the velocity became small and uniform.

As shown in figures of x-component of velocity along the central horizontal axes, the downward air flows were limited to a narrow layer near the cold walls. Between the central upward air flow and the downward air flow, the zero velocity u zones were very small and did not change their positions much.

The velocity value near the side walls were greater than those in the centre. This is because that the net mass flow through the plane must be zero. Thus the downward flow rate should be equal to the upward flow rate. With a lager area surface, the velocity in central area should be smaller. It is found that the velocity profile is quite even within the central area. This means that the temperature in it must be even within the same horizontal plane. Figure 6.4 and 6.5 show the same results. Contour maps of velocity distribution inside the container are shown in Appendix C.

The differences in velocity profiles between adiabatic and isothermal floor were small. This is because in both conditions, the air flow patterns were similar. Air was cooled by side-walls and moved downward near the walls, then it was warmed and moved upward in the central area. When warm air moved away from the bottom, the space was filled by cold air when a isothermal floor was presented, or warm air when adiabatic floor was used.

Appendix C provides a view of the velocity distribution in central planes. At the top of the container, air moved diverging from the centre while at bottom air moved converging to the centre.

7.4 Nusselt Number

As described in last section, strong air movement happened near the side-walls. Strong heat transfer by convection should also happen within this zone. The convection phenomena in this zone could be a good indication to identify the starting, developing and ending of natural convection inside the container. As defined in Chapter 3, Nusselt numbers are used to describe the natural convection occurring on the surface. When $Nu \leq 1$, it implies that there is no convection.

Figures 6.9, 6.24, 6.36 and 6.48 display the average Nusselt numbers' change with time on side-walls and top and bottom. At the beginning of cooling process, Nu had large values. Thus heat transfer by convection played a main role. When cooling reached a certain step, Nu numbers dropped to a low level. This indicated that due to the fact that the temperature differences were small, natural convection was weak and conductive heat transfer was important at this stage.

7.5 Effects of Permeability

7.5.1 Isotropic Permeability

Permeability has an significant effect on natural convection in porous media. As shown in Figure 7.1, the temperature profiles were different when isotropic permeability changed. The larger the permeability, the faster the temperature changed and the less time was needed to cool down the produce. This was because that when permeability increased, the flow resistance inside the packed bed decreased. Air moved more freely thus the convection was enhanced. On the other hand, high permeability also means the packed bed may have a high porosity enabling more space for air and less space for the produce. The heat capacity and the power of the internal heat sources may also be reduced. Thus the packed bed can be cooled quickly.

7.5.2 Orthotropic Permeability

Figure 6.49 to 6.58 show the result of horizontally laid carrots in the package and the results for horizontally laid asparagus are presented in Figure 6.59 to 6.68.

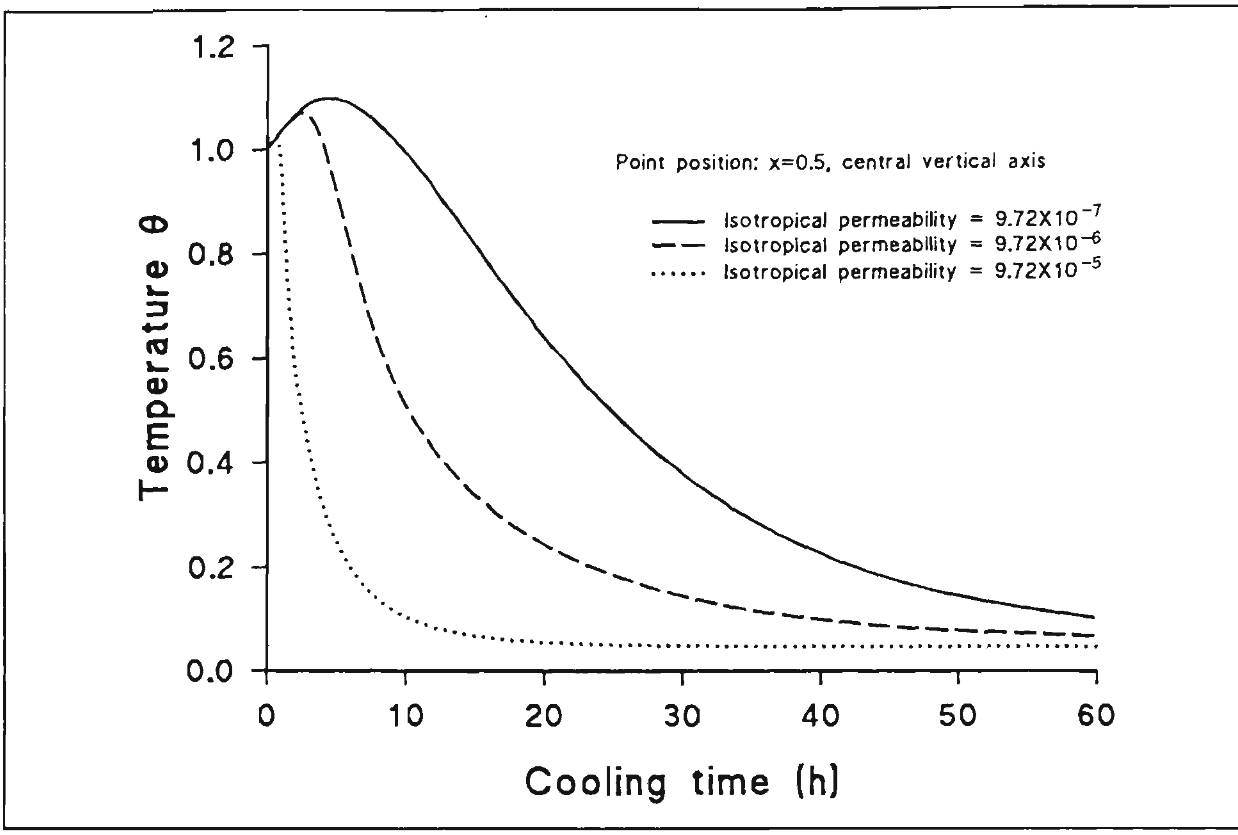


Figure 7.1 Temperature change versus time at central point of the container under different isotropic permeability.

For horizontally laid produce, the results indicate that the cooling process was not significantly affected by the different permeability relationships tried. The temperature distributions were also similar to each other, however slight differences in the velocity fields were observed. Velocity increased when the departure from isotropy increases. When produce were packed horizontally, for example, the axes of produce were parallel to y direction, air channels were formed in y direction. Thus resistance of air movement in y direction was reduced. When natural convection occurred inside the package, as described before, at top of the container, air flowed out from central area to side-walls area and at bottom flowed back to the central from outside region. Horizontally laid pattern improved the air flow in y direction in top and bottom area, but did not improve much in z direction. With no vertical channel formed the vertical air flow did not increase its velocity either.

When produce were packed vertically, vertical channels were formed. Thus resistance in x direction decreased. The downward air flow near side-walls and upward air flow in the central area were enhanced. In natural convection, vertical air movement plays a very important role in heat transfer. Due to the vertical air movement enhancement, the cooling processes were also improved, shown in Figure 6.69 to 6.78 for vertically laid carrots. Because the cooling process was more efficient, the temperature differences in the produce were smaller in the non-isotropic than those in the isotropic arrangements. The velocity was also smaller due to a smaller temperature difference.

Figure 6.79 to 6.88 show the results for vertically oriented asparagus. The vertical package did not show much difference from the isotropic results, shown in Figure 7.2. In the late cooling stage and the steady state, the isotropic package showed a better result. This is because when choosing different relations of permeability in x, y and z direction, permeability in x direction, κ_x, was set to a constant value, forcing κ_y and κ_z smaller when the differences between κ_y, κ_z and κ_x increasing. A low value of κ_y and κ_z implied that the resistance for air movement in y and z directions increased. Figure 7.3 shows the results when κ_y and κ_z were set to a fixed value but κ_x increased to perform the horizontally laid effect. The cooling process was improved with κ_x increasing. When κ_x was larger than a certain value, the cooling process did not improve. This may be explained as when κ_x was large enough, resistances in y and z directions became critical. Thus, increasing κ_x would not improve the convection unless κ_y and κ_z were improved.

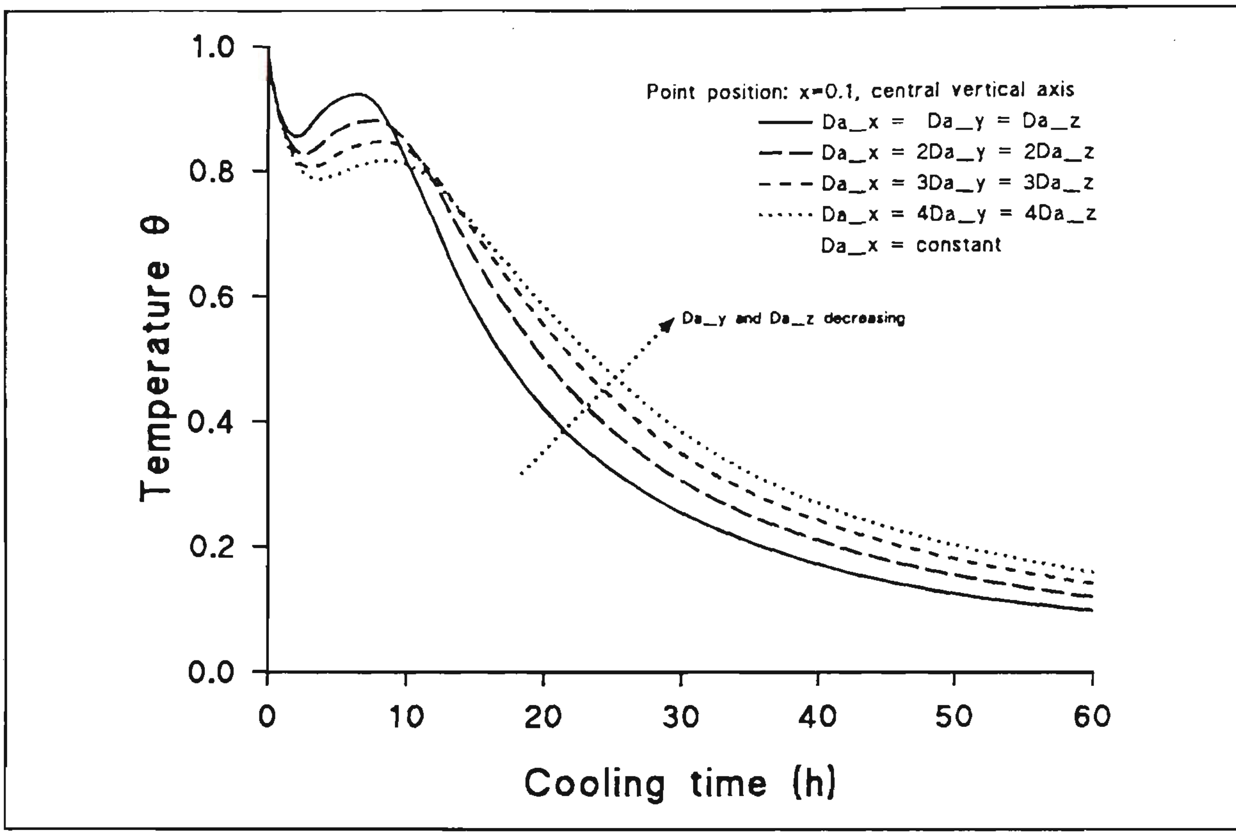


Figure 7.2 Temperature versus time curves for vertically laid asparagus, adiabatic floor. Resistance in y and z direction increasing.

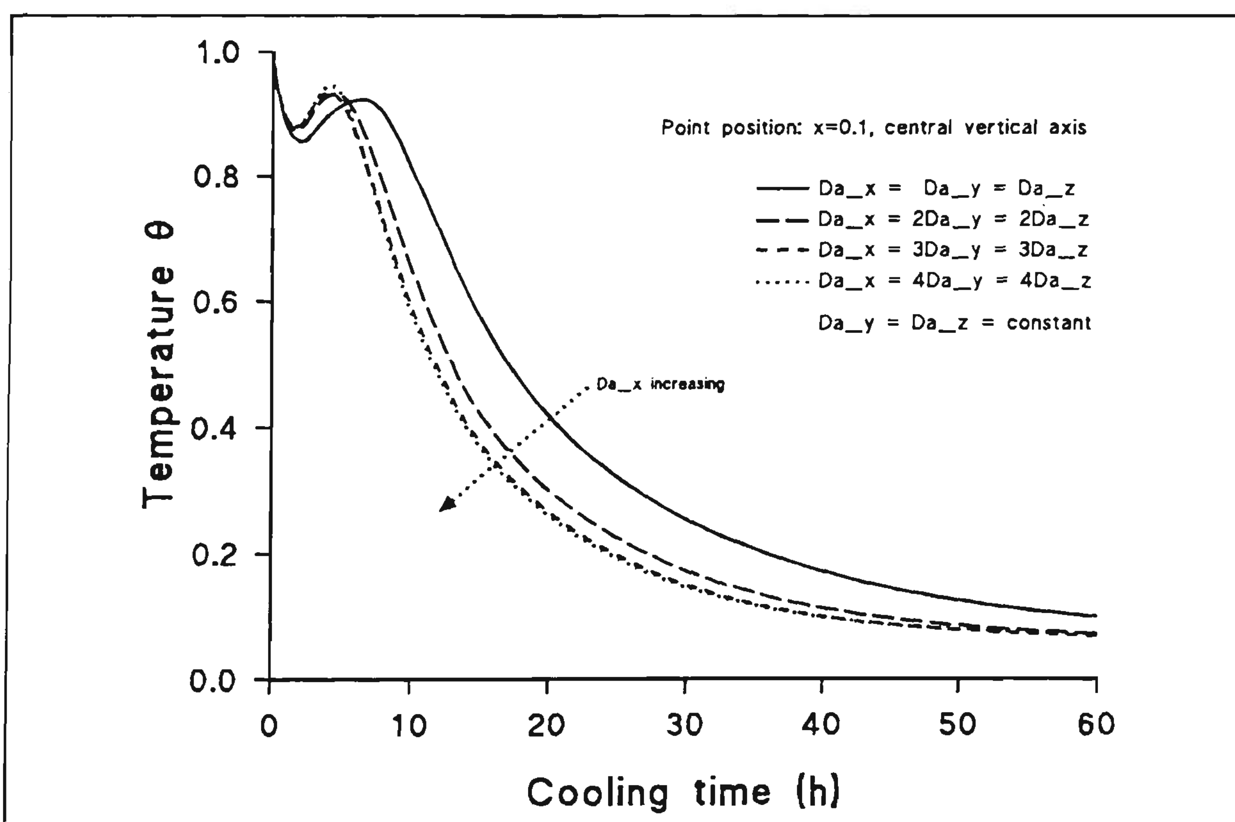


Figure 7.3 Temperature versus time curves for vertically laid asparagus, adiabatic floor. Resistance in x direction decreasing.

7.6 Respiration Effects on Container Size

A comparison of cooling processes for apples in a large container with and without respiration is shown in Figure 7.4. Because the respiration rate for apples was small, even at high temperatures, the temperature differences shown in figure 7.4 were also small. At steady state, the temperature was uniform. Without respiration heat, the temperature of the produce finally equilibrated to the environmental temperature $\theta = 0$. With respiring heating, the final temperature was still higher than the environmental temperature.

As discussed later, a large container is not suitable for cooling or storing produce having a high respiration rate. But even for a small container, the inclusion of respiration heat produces a significant temperature difference, as show in Figure 7.5.

Currently many different sized containers are used to store fruits or vegetables. When produce is harvested, the field heat should be removed as quickly as possible and usually this is done within a few hours. Delays between harvesting and cooling produce are certain to increase deterioration when ambient temperatures are high. Figure 7.6 and 7.7 show the numerical results of the cooling process for apples and Brussels sprouts in different size boxes. It is evident that the bigger the size of the box, the higher is the temperature reached and the longer is the time needed to cool the produce to steady state. It may be clearly seen that a small size container is better suited for rapid cooling than large boxes provided the cold walls can be maintained.

Referring to Figure 7.7, when a large size container was used to cool and store high

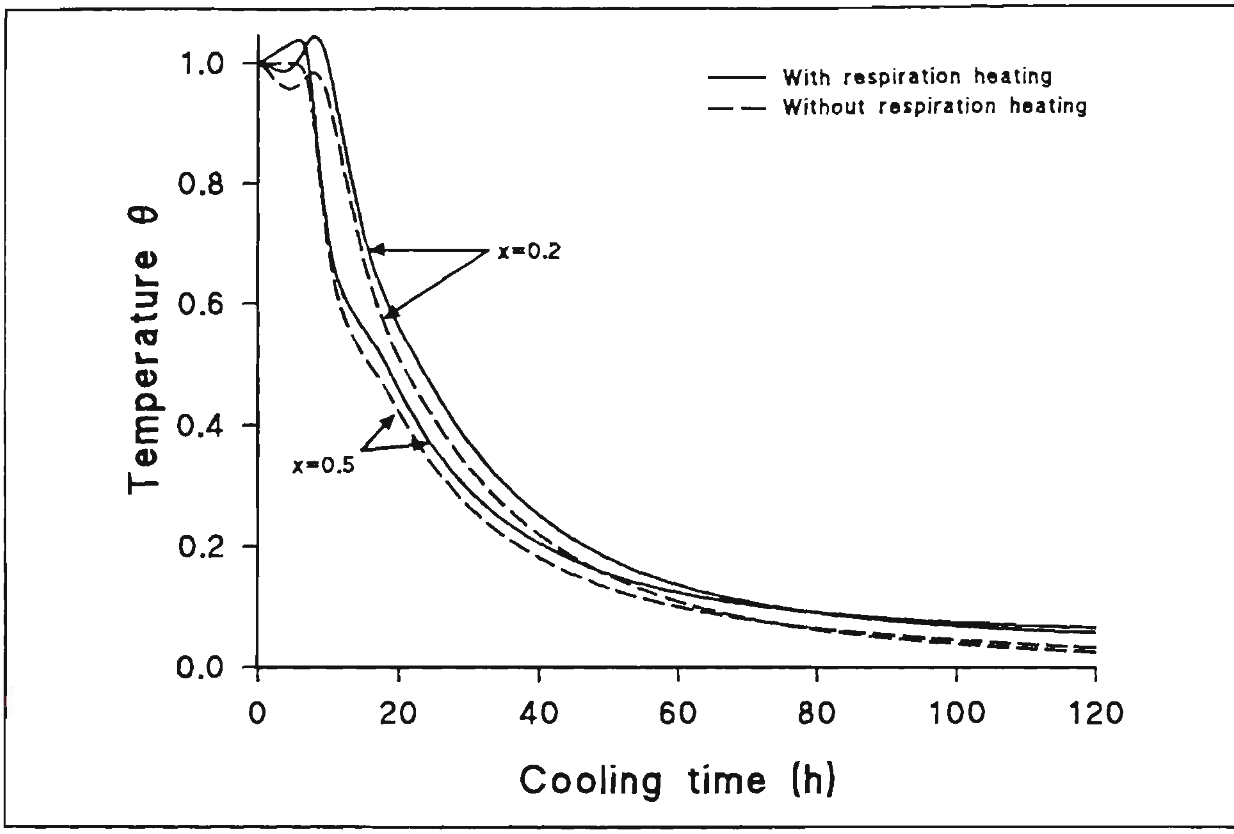


Figure 7.4 Temperature changing with time at central vertical axis, container size: 0.5 x 1.2 x 1.0 m, Apples, adiabatic floor

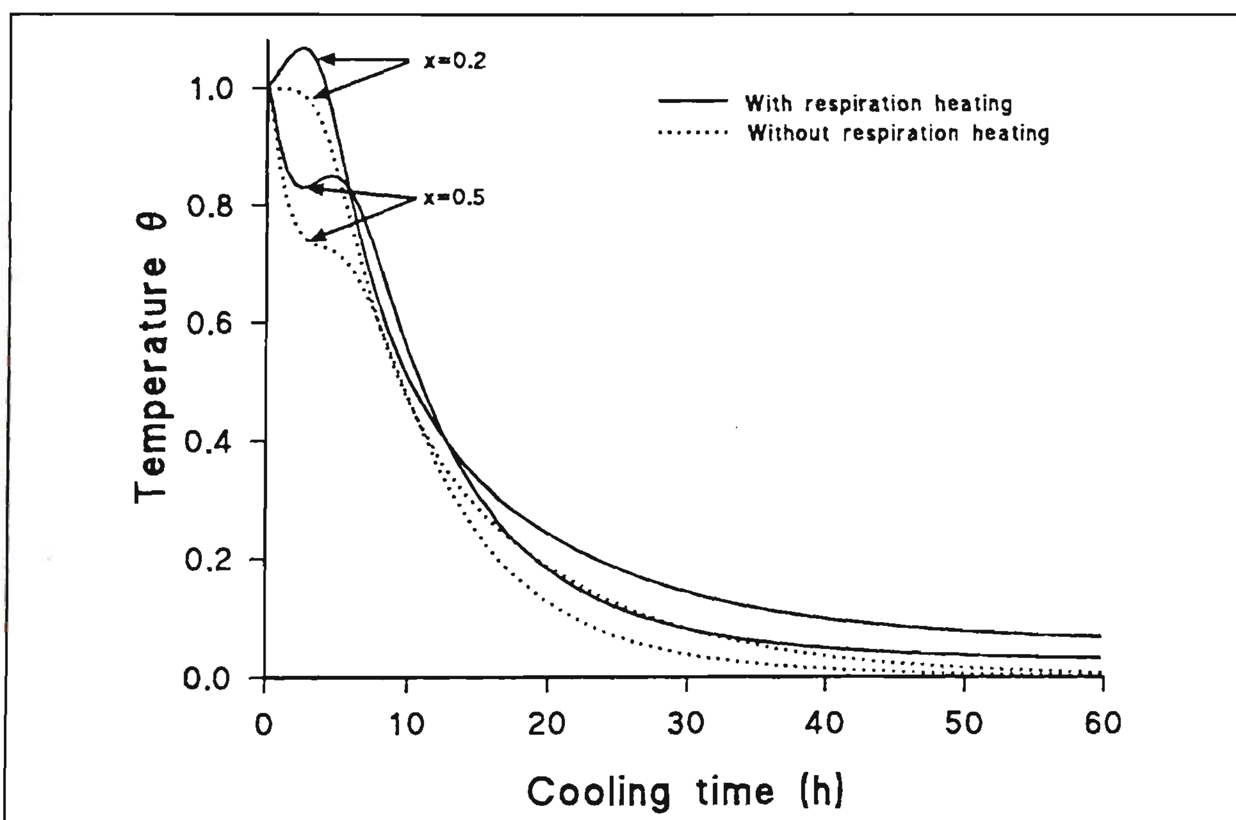


Figure 7.5 Temperature changing along central vertical axis, Brussels sprouts, container size: 0.28 x 0.52 x 0.32 m

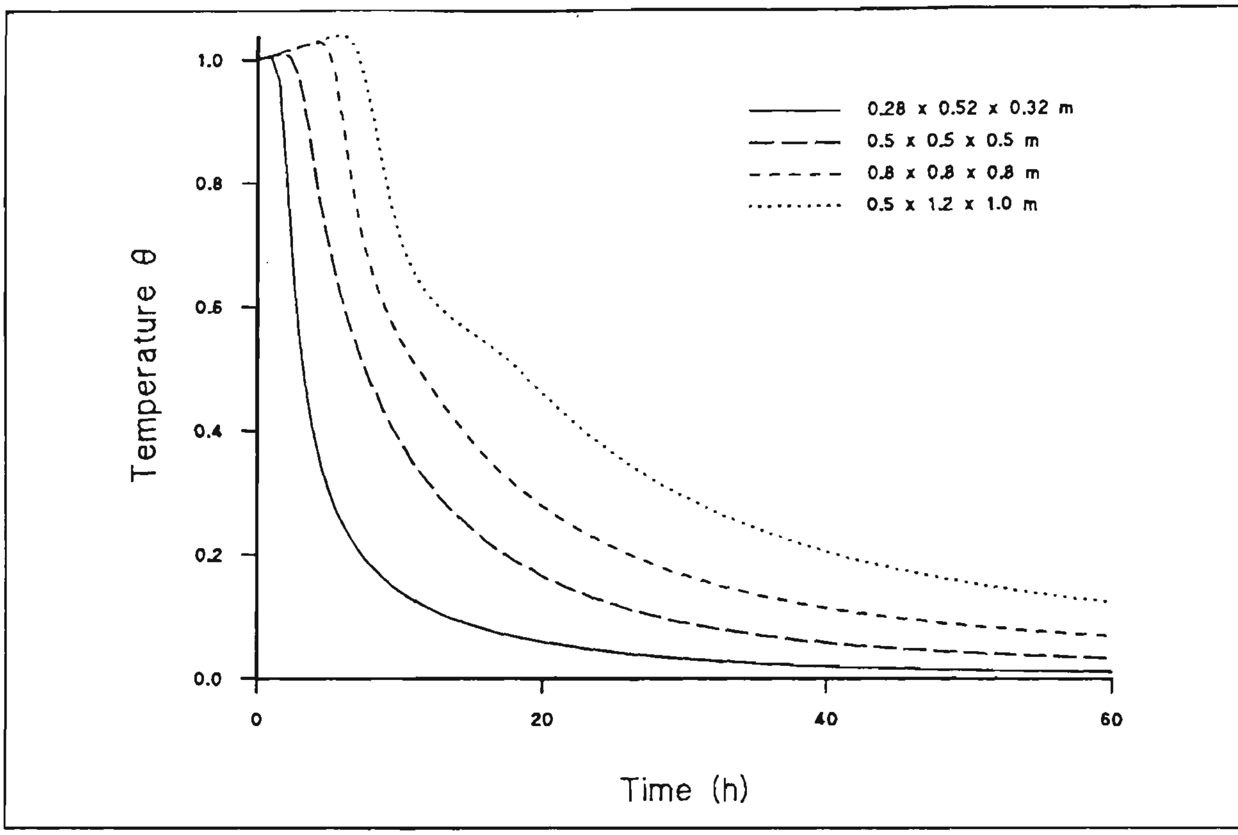


Figure 7.6 Effect of different size boxes on the cooling process for apple

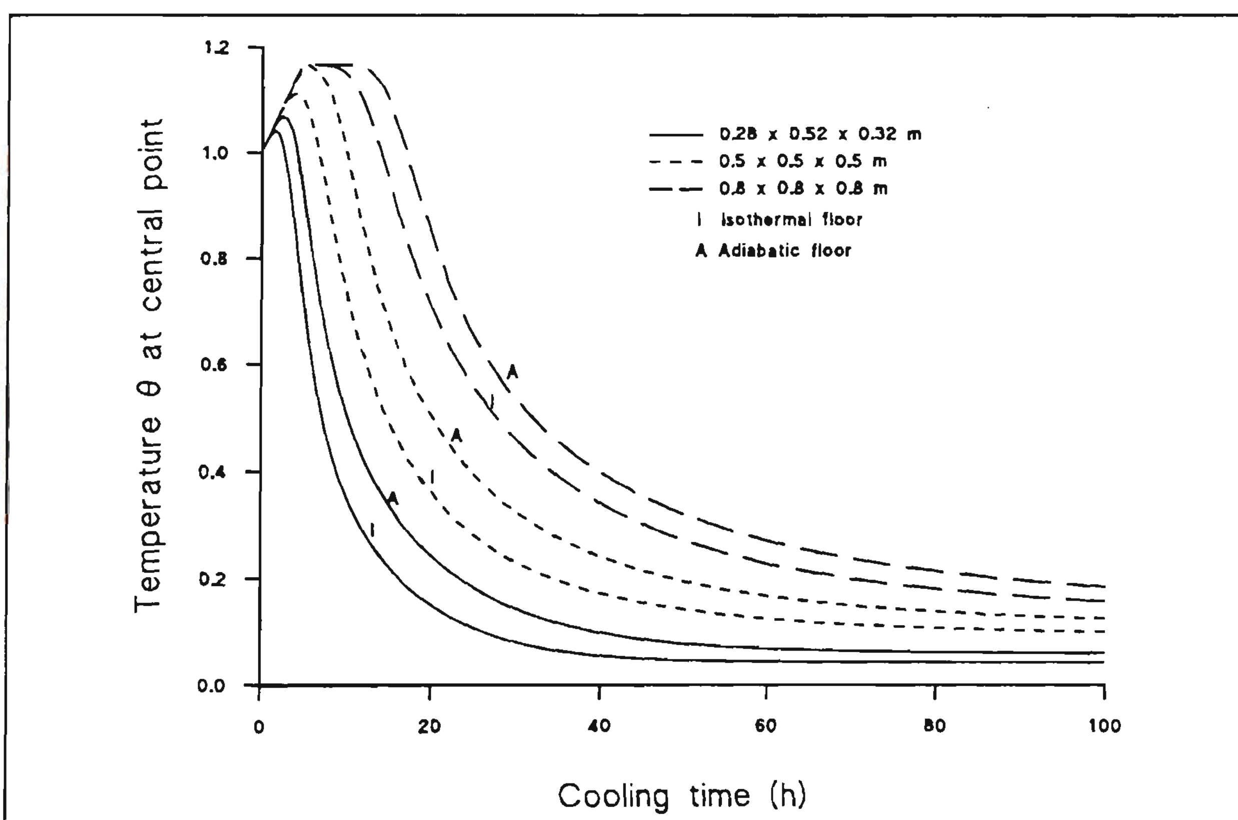


Figure 7.7 Effect of different size boxes on the cooling process for Brussels sprouts

```
x,y central plate.                         x,z central plate.

. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .


Note:        '.' represents live produce and '*' represents thermal death
             produce.
```

Figure 7.8 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.28x0.52x0.32m, adiabatic floor, initial temperature 30°C.

```
x,y central plate.                         x,z central plate.

. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .
. . . . . . . . . . .                      . . . . . . . . .


Note:        '.' represents live produce and '*' represents thermal death
             produce.
```

Figure 7.9 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.28x0.52x0.32m, isothermal floor, initial temperature 30°C.

```
x,y central plate.                    x,z central plate.

. . . . . . . . .                     . . . . . . . . .
. . * * * * * . .                     . . * * * * * . .
. . . * * * . . .                     . . . * * * . . .
. . . * * * . . .                     . . . * * * . . .
. . . . * . . . .                     . . . . * . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .

Note:      '.' represents live produce and '*' represents thermal death
           produce.
```

Figure 7.10 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.5x0.5x0.5m, adiabatic floor, initial temperature 30°C.

```
x,y central plate.                    x,z central plate.

. . . . . . . . .                     . . . . . . . . .
. . * * * * * . .                     . . * * * * * . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .
. . . . . . . . .                     . . . . . . . . .

Note:      '.' represents live produce and '*' represents thermal death
           produce.
```

Figure 7.11 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.5x0.5x0.5m, isothermal floor, initial temperature 30°C.

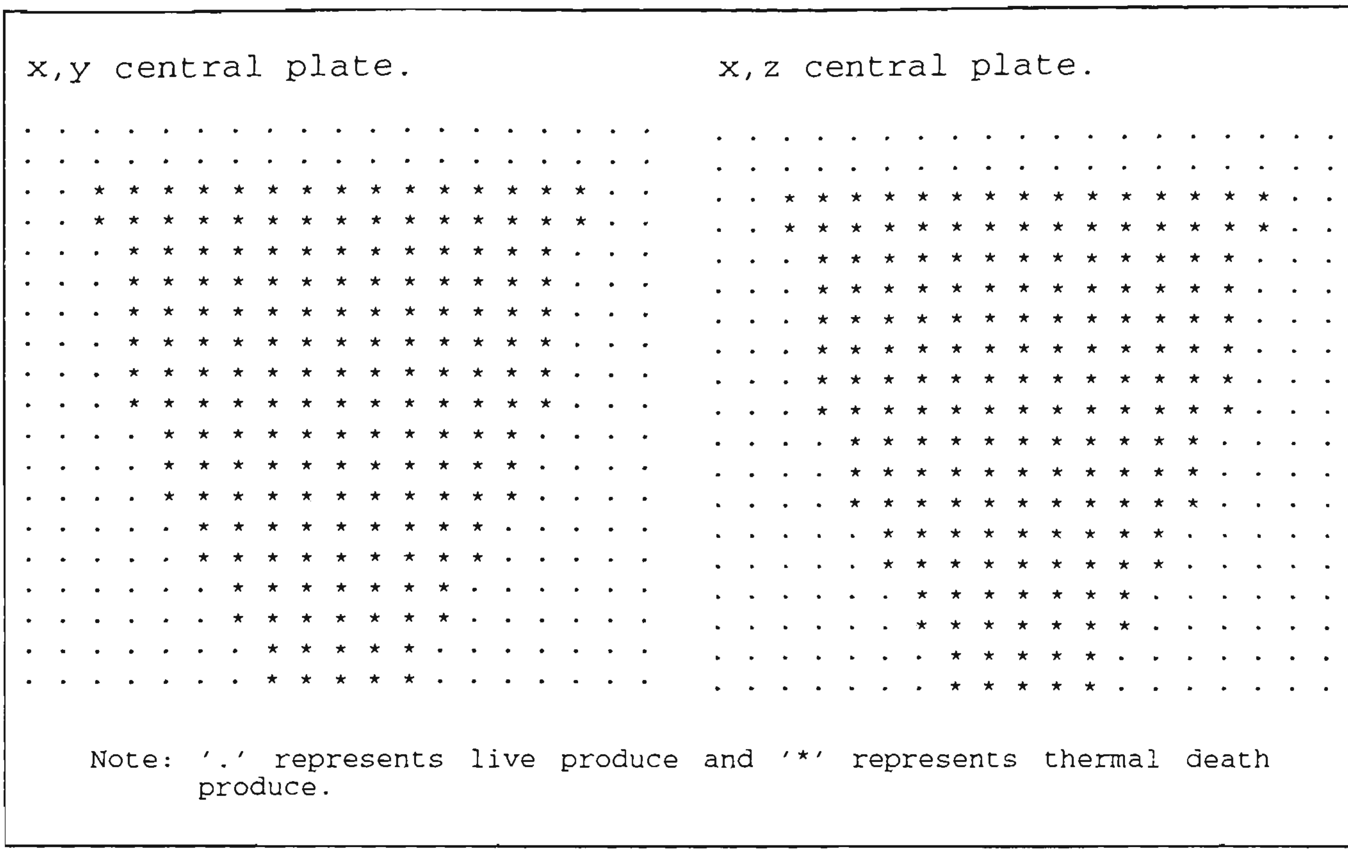


Figure 7.12 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.8x0.8x0.8m, adiabatic floor, initial temperature 30°C.

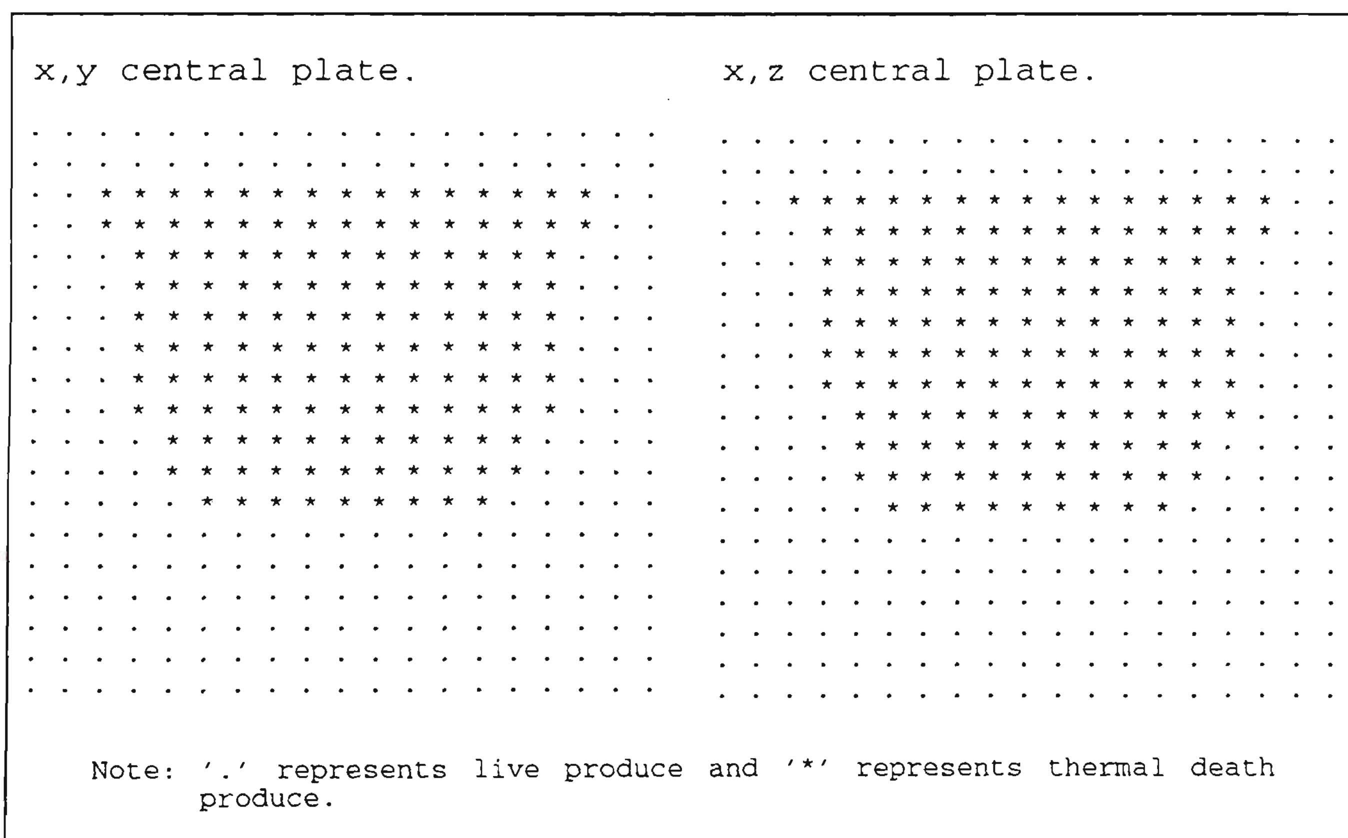


Figure 7.13 Positions of live and thermal dead Brussels sprouts on two central plates when reaching steady state. Box size: 0.8x0.8x0.8m, isothermal floor, initial temperature 30°C.

respiration rate produce, such as Brussels sprouts, a critically high temperature can be reached so that the produce suffers "thermal death". No life activities take place and no heat can be generated after reaching this point. Figure 7.8 to 7.13 display the distribution positions of alive and thermally dead Brussels sprouts after cooling process. When a small size container was used, no thermal death occurred. When using a large container, produce in the central area became thermally dead. Due to the cold top, produce in the surface layer still remained alive. Thus it might not be easy to identify the quality of the produce by just seeing the surface.

Even for the low respiration rate produce such as apple, a small container is still not good enough for efficient cooling to remove the field heat. As shown in Figure 7.6, for a standard 0.28 x 0.52 x 0.32m box, it took about 8 hours to cool apples from $\theta = 1$ (t=30°C) to $\theta = 0.2$ (t=6°C). With such a long time at high temperature, apples probably suffer from deterioration.

A fast pre-cooling procedure usually is used to remove the field heat efficiently. Different heating rates also had strong effects on temperature at steady state even when the produce was prechilled to 0°C, as shown in Figure 7.14. For example, Brussels sprouts reached a steady state temperature as high as $\theta = 0.06$, ie t=0.18°C and the temperature distribution was not uniform, as shown in Figures 6,25, 6.26 and 6.27. Figure 7.15 reveals the temperature versus time characteristic for stored Brussels sprouts after pre-cooling to 0°C and Figure 7.16 shows the steady state temperature distribution along the central vertical axis, when different sized containers were chosen. The final temperature was high and respiration remains at a relatively strong level, thus demonstrating that Brussels sprouts usually can only be stored for a short time.

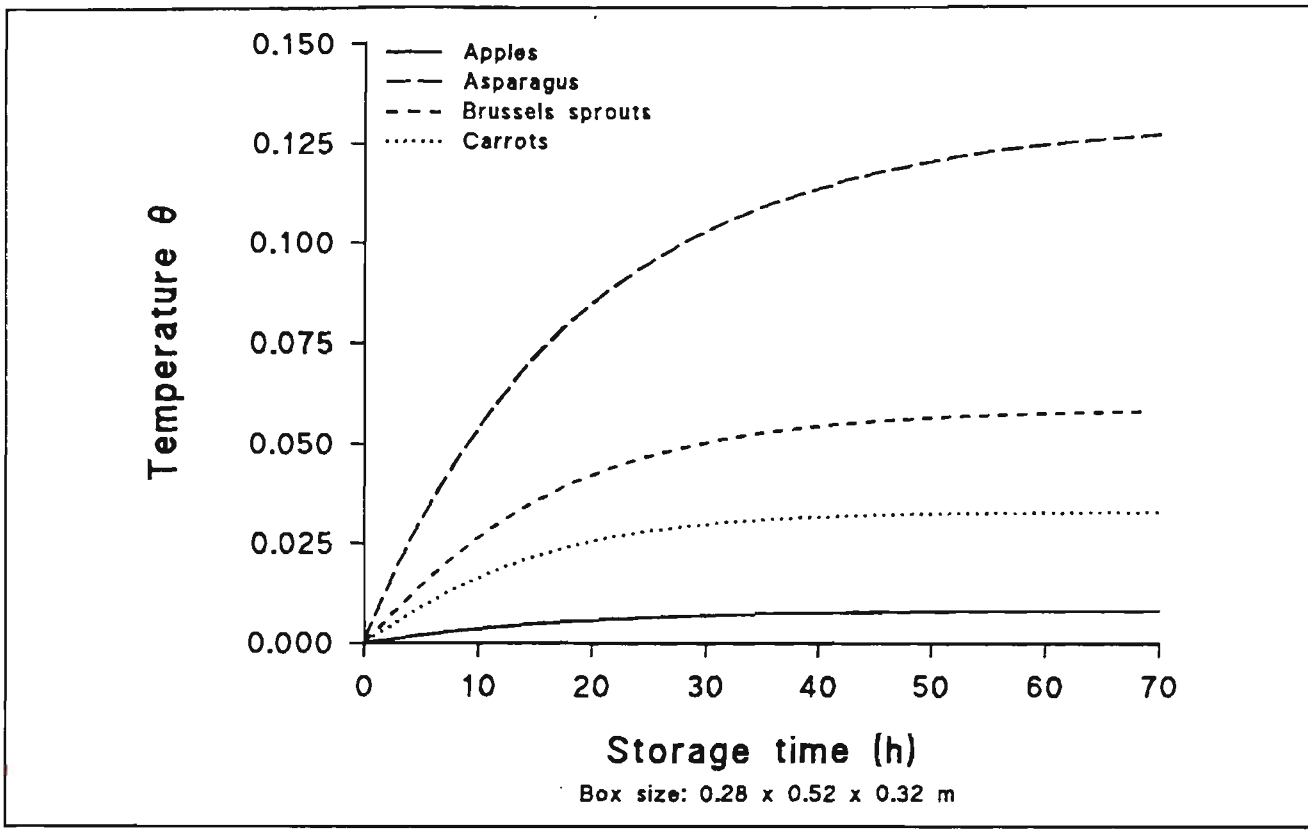


Figure 7.14 Effect of different respiration heating rate on temperature changing at central point of the container, 0.28 x 0.52 x 0.32 m, adiabatic floor, prechilled temperature 0 °C.

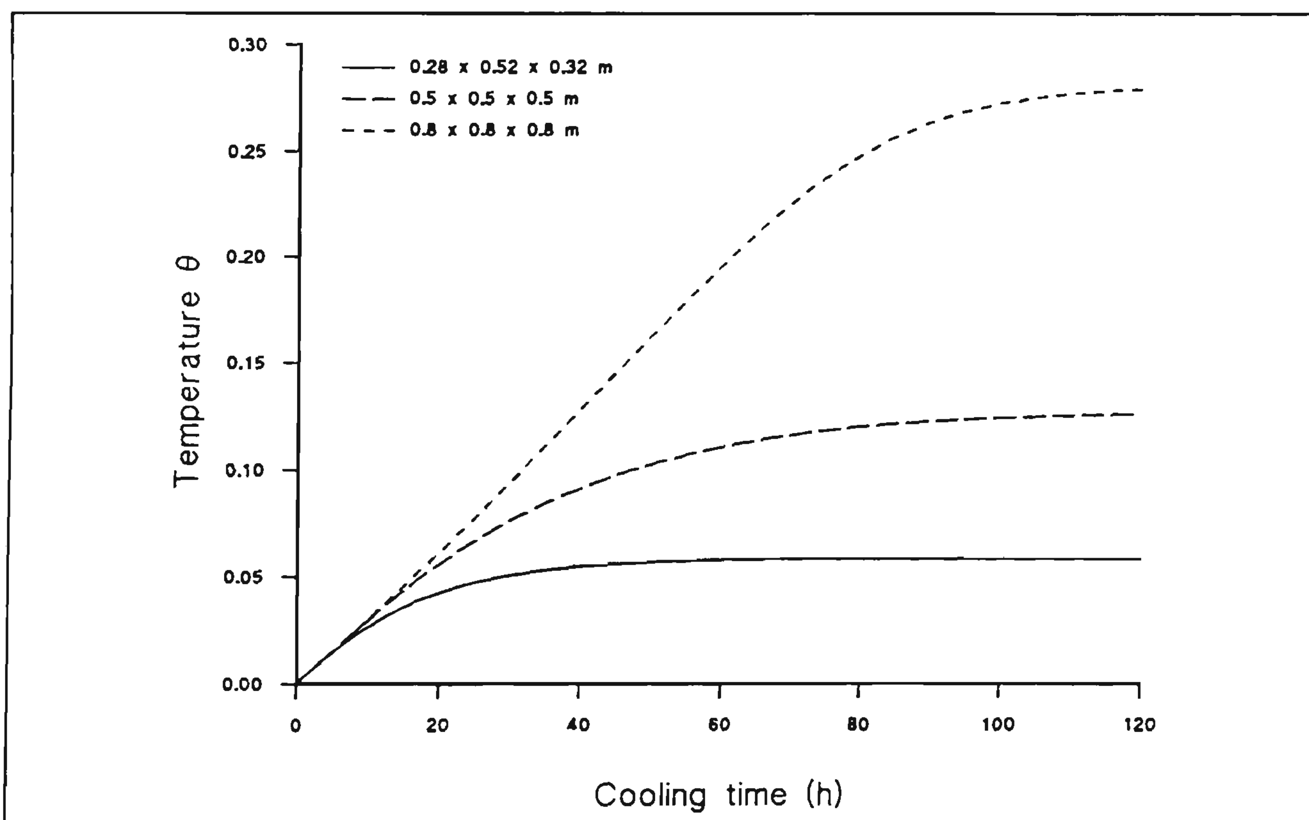


Figure 7.15 Effect of different size boxes on storage Brussels Sprouts after prechilling to 0 °C.

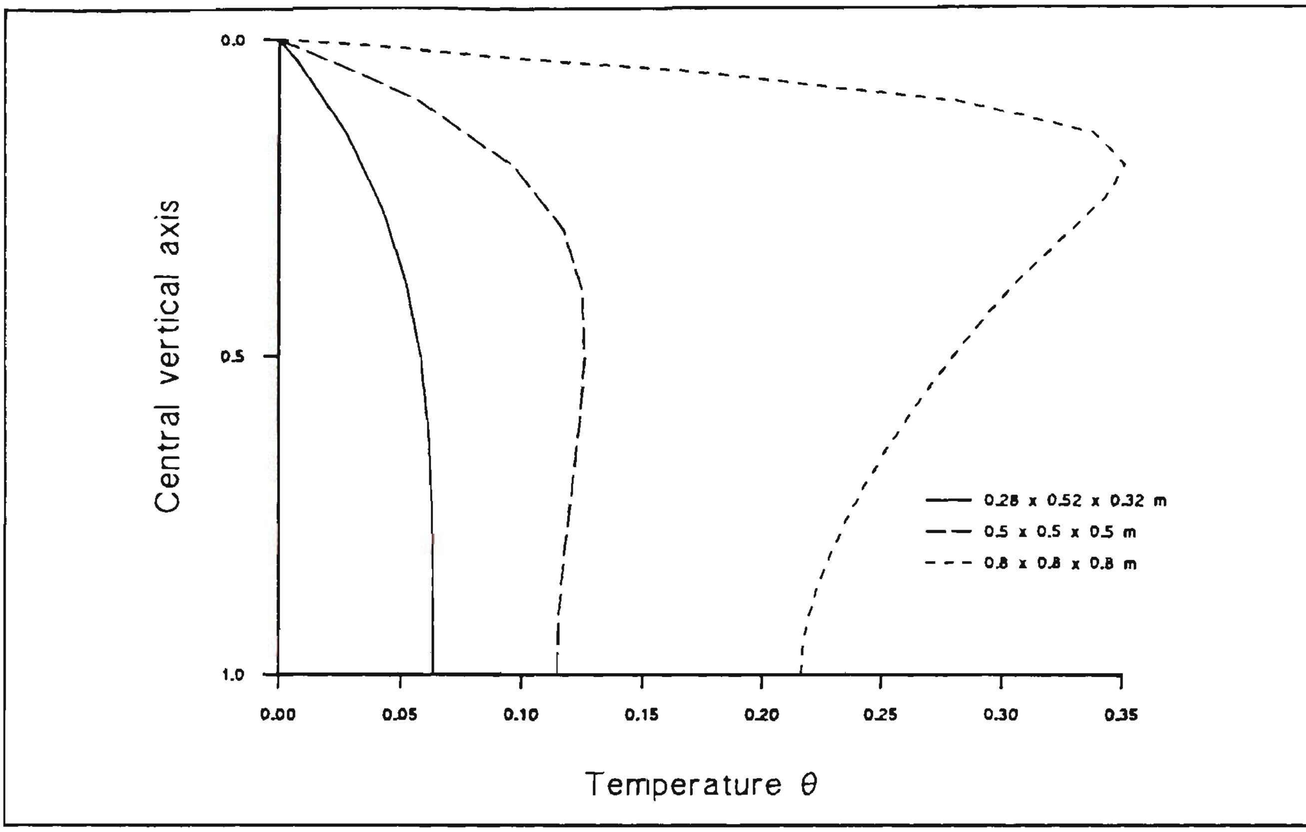


Figure 7.16 Effect of box size on temperature distribution along central vertical axis, Brussels Sprouts, prechilled to 0 °C.

7.7 Suggestions for Further Research

7.7.1 Permeability

Experimentally determined permeability measurements for various fruits and vegetables in of paramount importance and if this information becomes available it would enhance the usefulness of the model.

7.7.2 Respiration Function

Effects should be made to develop methods, based on measurements, of rapidly determining the characteristic temperature dependent respiration function for various commodities. Critical temperature of thermal death, both freezing point and hot point, for various fruits and vegetables are important to determine the respiration functions in the model.

7.7.3 Transpiration Rate

Foremost of all the improvements which could be made to the model would be to account for transpiration or moisture loss in the produce. The factors influencing transpiration rates are temperature, humidity and barometric pressure in storage or transportation vehicle. A suggestion for doing this would be to incorporate the cooling effect of evaporation by adding an additional negative term to the respiratory heat generation - its form would be "rate of moisture loss × latent heat of vaporization". Also a moisture transfer equation could be considered into the equations group to obtain a mass-heat transfer model.

CHAPTER 8

CONCLUSIONS

1. The theory of natural convective heat transfer with internal heat generation as applied to the storage of agricultural or horticultural produce, was extended by successfully incorporation a characteristic temperature dependent respiratory heat generation term.

2. A three-dimensional transient model of the natural convective heat transfer processes occurring in stored agricultural produce in closed containers with isotropic or orthotropic permeability and temperature dependent respiratory heating was developed to an effective state.

3. In stored produce, heat generated by respiration significantly influences the temperature and velocity distributions. Respiration also affects the final uniform steady state temperature - low respiration rates give lower steady state temperatures and smaller temperature differences within the container, whereas higher respiration rates lead to greater steady state temperatures and more non-uniform temperature distributions.

4. Container size is an important factor in cooling and storage of agricultural produce. Commodities with lower respiration rates may be packed in larger containers, but for higher respiration rate commodities, restrictions on the size of container used may be necessary to avoid high and non-uniform temperatures.

5. The nature of the permeability also affects the natural convective heat transfer process. The results of this study demonstrated that a package pattern which produces lower resistance to flow along the direction of gravity gives more effective heat transfer.

REFERENCES

1. Aziz, K. and Hellums, J. D., 1967. *Numerical solution of the three-dimensional equations of motion of laminar natural convection.* Phys. Fluid. Vol. 10, pp314-324.

2. Beck, J. L., 1972. *Convection in a box of porous material saturated with fluid.* Phys. Fluids, Vol. 15, pp1377-1383.

3. Beckermann, C., Ramadhyani, S. and Viskanta, R., 1987. *Natural convection flow and heat transfer between a fluid layer and a porous layer inside a rectangular enclosure.* Transactions of the ASME, Journal of Heat Transfer, Vol. 109, pp363-370.

4. Beukema, K. J., 1980. *Heat and mass transfer during cooling and storage of agricultural products as influenced by natural convection.* Agric. Res. Rep. 897, ISBN 9022007286, Centre for Agricultural Publishing and Documentation, Wageningen.

5. Beukema, K. J., Bruin, S., and Schenk, J., 1983. *Three-dimensional natural convection in a confined porous medium with internal heat generation.* Int. J. Heat Mass Trans. Vol. 26, pp451-459.

6. Buretta, R. J. and Berman, A. S., 1976. *Convective heat transfer in a liquid saturated porous layer.* J. Appl. Mech. Vol. 43, pp249-253.

7. Carnahan, B., Luther, H.A. and Wilkes, J. O. 1969. *Applied Numerical Methods* John Wiley & Sons, Inc.

8. Chan, B. K. C., Ivey, C. M. and Barry, J. M., 1970. *Natural Convection in enclosed porous media with rectangular boundaries.* ASME J. of Heat Transfer, Vol. 2, pp21-27.

9. Chorin, A. J., 1968. *Numerical solution of the Navier-Stokes equations.* Mathematics of Computation, Vol. 22, pp745-762.

10. Douglas, Jr. J. and Rachford, Jr. H. H., 1956. *On the numerical solution of heat conduction problems in two and three space variables.* Trans. Am. Math. Soc. Vol. 82, pp421-439.

11. Dullien, F. A. L., 1979. *Porous media fluid transport and pore structure.* Academic Press.

12. Ergun, S., 1952. *Fluid flow through packed columns.* Chemical engineering progress. Vol. 48. No. 2, pp89-94.

13. Fikiin, A. G., 1983. *Etude des facteurs d'intensification de l'échange de chaleur dans le refroidissement des fruits et légumes.* International Journal of Refrigeration, Vol. 6, No. 3, pp176-181.

14. Gasser, R. D., and Kazimi, M. S., 1976. *Onset of convection in a porous medium with internal heat generation.* Transactions of ASME, Journal of Heat Transfer, Vol. 98, pp49-54.

15. Goh, L.P., Leonardi, E. and de Vahl Davis, G., *FRECON3D - Users Manual. A program for the numerical solution of mixed convection in a three-dimensional rectangular cavity.* The University of New South Wales, School of Mech. and Ind. Engineering, report 1988/fmt/7, (1988).

16. Hardee, H. C. and Nilson, R. H.,1977. *Natural convection in porous media with hear generation.* Nucl. Sci. Engng., Vol. 63, pp119-132.

17. Hardenburg, R. E., Watada, A. E., and Wang, C. Y., 1986. *The commercial storage of fruits, vegetables, and florist and nursery stocks.* U.S. Department of Agriculture, Agriculture Handbook No.66

18. Hirasakii, G. J. and Hellums, J. D., 1968. *A general formulation of the boundary conditions on the vector potential in three-dimensional hydrodynamics.* Q. Appl. Math. Vol. 26, pp311-342.

19. Hirasaki, G. J. and Hellums, J. D., 1970. *Boundary conditions on the vector and scalar potentials in viscous three-dimensional hydro-dynamics.* Quart. Appl. Math. Vol. 28, pp293-297.

20. Johnson, A. T., Kirk, G. D., Moon, S. H., and Shih, T. M., 1988. *Numerical and experimental analysis of mixed forced and natural convection about a sphere.* Transactions of the ASAE, American Society of Agricultural Engineers. Vol. 31:1, pp293-299.

21. Kulacki, F. A. and Freeman, R. G., 1979. *A note on thermal convection in a saturated. heat generating porous layer.* ASME J. Heat Transfer, Vol 101, pp169-171.

22. Mallinson, G. D. and De vahl Davis, G., 1973. *The method of false transient for the solution of coupled elliptic equations.* Journal of Computational Physics. Vol. 12, pp435-461.

23. Mohsenin, N. N., 1980. *Thermal properties of foods and agricultural materials.* Gordon and Breach Science Publishers.

24. Mohsenin, N. N., 1986. *Physical properties of plant and animal materials.* Gordon and Breach Science Publishers.

25. Moon, S. H., Johnson, S. T. and Shih, T. M., 1987. *Numerical analysis of mixed convection from horizontal cylinders.* Journal of Agricultural Engineering Research. Vol. 38:4, pp289-300.

26. Neale, G. and Nader, W., 1974. *Practical significance of Brinkman's extension of Darcy law's: coupled parallel flows within a channel and a bounding porous medium.* Canadian Journal of Chemical Engineering, Vol. 52, pp475-478.

27. Nield, D. A., 1991. *The limitations of the Brinkman-Forchheimer equation in modelling flow in a saturated porous medium and at an interface.* Int. J. Heat and Fluid Flow, Vol. 12, No. 3, pp269-272.

28. Nield, D. A. and Bejan, A., 1992. *Convection in porous media.* Springer-Verlag New York Inc.

29. Özisik, M. N., 1985. *Heat Transfer.* McGraw-Hill Book Company.

30. Plumb, O. A. and Huenefeld, J. C., 1981. *Non-Darcy natural convection from heated surfaces in saturated porous media.* Int. J. Heat Mass. Trans. Vol. 24(5), pp765-768.

31. Rhee, S. J., Dhir, V. K. and Catton, I., 1978. *Natural convection heat transfer in beds of inductively heated particles.* Transactions of the ASME, Journal of Heat Transfer. Vol. 100, pp78-85.

32. Richardson, S. M. and Cornish, A. R. H., 1977. *Solution of three-dimensional flow problems.* J. Fluid Mech. Vol. 82, pp309-319.

33. Robinson, J., 1988. *Convective air flow in fruit store.* Engineering Advances for Agriculture and Food. pp313-314.

34. Rudraiah, N. and Prabhamani, P. R., 1974. *Thermal diffusion and convective stability of two component fluid in a porous medium.* Proc. 5th Int. Heat Transf. Conf., Tokyo (5), pp79-82.

35. Ryall, A. L. and Lipton, W. J., 1979. *Handling, transportation and storage of fruits and vegetables, Vol. 1.* AVI Publishing Company.

36. Salunkhe, D.K. 1974. *Storage, processing, and nutritional quality of fruits and vegetables.* CRC Press Inc.

37. Samarskii, A. A., and Andreyev, V. B., 1963. *On a high-accuracy difference scheme for an elliptic equation with several space variables.* U.S.S.R. Journal of Computational Mathematics and Mathematical Physics, Vol. 3, pp1373-1382.

38. Singh, A. K., Leonardi, E. and Thorpe, G. R., 1993. *Three-dimensional natural convection in a confined.* Transactions of the ASME, Journal of Heat Transfer. Vol. 115, No 3. pp.631-638.

39. Tang, L. Y. and Johnson, A. T., 1992. *Mixed convection about fruits.* Journal of Agricultural Engineering Research. Vol. 51:1, pp15-27.

40. Tong, T. W. and Subramanian, E., 1985. *A boundary-layer analysis for natural convection in vertical porous enclosures -- use of the Brinkman-extended Darcy model.* J. Heat Mass Transfer. Vol. 28, pp563-572.

41. Tveitereid, M. 1977. *Thermal convection in a horizontal porous layer with internal heat sources.* Int. J. Heat Mass Transfer, Vol. 20, pp1045-1050.

42. Walker, K. and Homsy, G. M., 1977. *A note on convective instabilities in Boussinesq fluids and porous media.* J. Heat Transfer 99: pp338-339.

43. Wills, R. B. H., Hall, E. G., Lee, T. H., McGlasson, W. B., and Graham, D., 1977. *An introduction to postharvest fruit and vegetables*, Australian Development Assistance Bureau.

44. Whitaker, S., 1966. *The equation of motion in porous media.* Chem. Engng Sci, Vol. 21, pp291-300.

45. Whitaker, S., 1969. *Advances in theory of fluid motion in porous media.* Ind. Engng chem. Vol, 61, pp15-28.

46. Williams, G. P., 1969. *Numerical integration of the three-dimensional Navier-Stokes equations for incompressible flow.* Fluid Mech. Vol. 37, pp727-750.

47. Woods, J. L. 1990. *Moisture loss from fruits and vegetables.* Postharvest News and Information. 1:3, pp195-199.

APPENDIX A

SOLUTION OF LAPLACE EQUATIONS

In Chapter 3 we have two Laplace equations. One is the scale potential ϕ:

$$\nabla^2\phi = 0 \tag{3.48}$$

Another is the equation for the x component of vector potential when $k_y = k_z$ and the x axis is along the gravity direction:

$$\nabla^2\psi_x = 0 \tag{3.87}$$

With their boundary conditions it is found that ϕ is an arbitary constant and $\psi_x = 0$. In this appendix the detailed solution of ϕ is demonstrated.

The solution procedure consists of four steps. Applying the theory of Separation of Variables, we assume that the solution of ϕ exist as products of a function X(x) of x alone, a function Y(y) of y alone and a function Z(z) of z alone, thus we may write

$$\phi(x,y,z) = X(x)Y(y)Z(z) \tag{A1}$$

Step 1

Firstly at eight corners of the enclosure, for example, at (0,0,0), it is simply found that

$$\left.\frac{\partial\phi}{\partial x}\right|_{(0,0,0)} = \left.\frac{dX}{dx}\right|_{x=0} Y(0)Z(0) = 0 \tag{A2}$$

$$\left.\frac{\partial\phi}{\partial y}\right|_{(0,0,0)} = \left.\frac{dY}{dy}\right|_{y=0} X(0)Z(0) = 0 \tag{A3}$$

$$\left.\frac{\partial\phi}{\partial z}\right|_{(0,0,0)} = \left.\frac{dZ}{dz}\right|_{z=0} X(0)Y(0) = 0 \tag{A4}$$

supose that

$$X(0) \neq 0 \ , \qquad Y(0) \neq 0 \ , \qquad Z(0) \neq 0 \tag{A5 - A7}$$

From equations A2 to A4 we have

$$X(0)=c_1, \ Y(0)=c_2, \ z(0)=c_3 \tag{A8 - A10}$$

or,

$$\phi=c_1c_2c_3=C \tag{A11}$$

where ϕ is an arbitrary constant at the corners.

Step 2

We next solve for ϕ on the edges of the enclosure, for example, at x=0 and y=0. At the edge, equation 3.47 becomes one dimensional and can be readily solved. We have

$$\frac{\partial^2\phi}{\partial z^2}=0 \tag{A12}$$

$$\frac{\partial\phi}{\partial z}=c_4 \tag{A13}$$

$$\phi=c_4z+c_5 \tag{A14}$$

Applying the above results at z=0 and $z=z_0$ in step 1 we find

$$\phi|_{z=0}=c_5=C \tag{A15}$$

$$\phi|_{z=z_0}=c_4z_0+C=C \tag{A16}$$

$$c_4=0, \ ie, \ \phi=c_4z+c_5=C \tag{A17}$$

Step 3

Now the results on the edges can be used as the boudary conditions for solving ϕ on the surfaces, on which equation 3.48 becomes two dimensional, for example, on the surface z=0, we have

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=0 \tag{A18}$$

hence

$$X''Y+Y''X=0$$

or

$$\frac{X''}{X}=-\frac{Y''}{Y} \tag{A19}$$

Now the left hand side member of the above equation is independent of y. Hence the right-hand side of it must also be independent of y since it is identically equal to the expression on the left. Similarly each member of the equation must be independent of x. Therefore, being independent of both x and y, each side of the equation must be a constant, say, C^*, we can write

$$\frac{X''}{X}=-\frac{Y''}{Y}=C^* \tag{A20}$$

if $C^*<0$, say, $C^*=-\lambda^2$ $(\lambda>0)$, we have,

$$X''=-\lambda^2 X \quad \Rightarrow \quad X = a_1\cos\lambda x + a_2\sin\lambda x + a_3 \tag{A21}$$

$$Y''=\lambda^2 Y \quad \Rightarrow \quad Y = a_4\cosh\lambda x + a_5\sinh\lambda x + a_6 \tag{A22}$$

Using the result of step 2, viz,

$$\phi|_{x=0} = X*Y = (a_1+a_3)*Y = C \tag{A23}$$

$$\phi|_{y=0} = X*Y = X*(a_4+a_6) = C \tag{A24}$$

From these two expressions we can say that both ***X, Y*** are constant and their product is equal to *C*, thus, we have

$$\phi = X*Y = C \tag{A25}$$

if $C^*=0$, then

$$X''=0 \quad \Rightarrow \quad X=a_1x+a_2 \tag{A26}$$

$$Y''=0 \quad \Rightarrow \quad Y=a_3y+a_4 \tag{A27}$$

Applying the boundary conditions

$$\phi|_{x=0} = C \tag{A28}$$

$$\phi|_{y=0} = C \tag{A29}$$

leads to

$$a_1 = a_3 = 0 \qquad a_2*a_4 = C$$

$$\phi = C \tag{A30}$$

if $C^*>0$, then

$$X'' = \lambda^2 X \quad \Rightarrow \quad X = a_1 \cosh\lambda x + a_2 \sinh\lambda x + a_3 \tag{A31}$$

$$Y'' = -\lambda^2 Y \quad \Rightarrow \quad Y = a_4 \cos\lambda x + a_5 \sin\lambda x + a_6 \tag{A32}$$

Using the result of step 2, viz,

$$\phi|_{x=0} = X * Y = (a_1 + a_3) * Y = C \tag{A33}$$

$$\phi|_{y=0} = X * Y = X * (a_4 + a_6) = C \tag{A34}$$

From these two expressions we can say that both ***X***, ***Y*** are constant and their product is equal to ***C***, thus, we have

$$\phi = X * Y = C \tag{A35}$$

Step 4

Now considering ϕ on any z plane, we can use the result of step 3 and since the equation and boundary conditions are exactually the same as those in step 3. We obtain $\phi = C$.

APPENDIX B

THOMAS METHOD FOR SOLVING A TRI-DIAGONAL MATRIX EQUATION

In solving temperature and vector potential fields, an equation set having a tri-diagonal matrix is encountered. The equation set takes the form

$$
\begin{aligned}
b_1x_1 + c_1x_2 \qquad\qquad\qquad\qquad &= d_1 \\
a_2x_1 + b_2x_2 + c_2x_3 \qquad\qquad &= d_2 \\
a_3x_2 + b_3x_3 + c_3x_4 \qquad &= d_3 \\
\cdots\cdots \\
a_jx_{j-1} + b_jx_j + c_jx_{j+1} &= d_j \\
\cdots\cdots \\
a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n &= d_{n-1} \\
a_nx_{n-1} + b_nx_n &= d_n
\end{aligned}
\tag{B1}
$$

or in a matrix form

$$
\begin{bmatrix}
b_1 & c_1 & & & & & \\
a_2 & b_2 & c_2 & & & & \\
 & a_3 & b_3 & c_3 & & & \\
 & \cdots\cdots & & & & & \\
 & & a_j & b_j & c_j & & \\
 & \cdots\cdots & & & & & \\
 & & & & a_{n-1} & b_{n-1} & c_{n-1} \\
 & & & & & a-n & b_n
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_j \\ \cdots \\ x_{n-1} \\ x_n \end{bmatrix}
\begin{bmatrix} d_1 \\ d-2 \\ d-3 \\ \cdots \\ d_j \\ \cdots \\ d_{n-1} \\ d_n \end{bmatrix}
\tag{B2}
$$

The idea behind the Gaussian elimination scheme is to manipulate the equations into the form

$$
\begin{bmatrix}
\beta_1 & c_1 & & & & & \\
 & \beta_2 & c_2 & & & & \\
 & & \beta_3 & c_3 & & & \\
 & & & \cdots\cdots & & & \\
 & & & \beta_j & c_j & & \\
 & & & & \cdots\cdots & & \\
 & & & & \beta_{n-1} & c_{n-1} & \\
 & & & & & & \beta_n
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_j \\ \cdots \\ x_{n-1} \\ x_n \end{bmatrix}
\begin{bmatrix} \delta_1 \\ \delta-2 \\ \delta-3 \\ \cdots \\ \delta_j \\ \cdots \\ \delta_{n-1} \\ \delta_n \end{bmatrix}
\tag{B3}
$$

It is then easy to find x_n, since

$$\beta_n x_n = \delta_n$$

or

$$x_n = \delta_n / \beta_n \tag{B4}$$

Knowing x_n enables us to find x_{n-1} from

$$\beta_{n-1} x_{n-1} + c_{n-1} x_n = \delta_{n-1}$$

$$or \quad x_{n-1} = \frac{\delta_{n-1} - c_{n-1} x_n}{\beta_{n-1}} \tag{B5}$$

and x_{n-2} from

$$x_{n-2} = \frac{\delta_{n-2} - c_{n-2} x_{n-1}}{\beta_{n-2}} \tag{B5a}$$

and so on.

The question now arises as to how to manipulate the system of equations into the desired form.

Firstly set

$$\beta_1 = b_1 \quad and \quad \delta_1 = d_1 \ . \tag{B6}$$

So the first two equations become

$$\beta_1 x_1 + c_1 x_2 = \delta_1 \tag{B7}$$

$$a_2 x_1 + b_2 x_2 + c_2 x_3 = d_2 \tag{B8}$$

Multiply equation (B7) by a_2/β_1

$$a_2 x_1 + \frac{a_2}{\beta_1} c_2 x_2 = \delta_1 \frac{a_2}{\beta_1} \tag{B7a}$$

and subtract this result from equation (B8) to get

$$(b_2 - \frac{a_2}{\beta_1} c_2) x_2 + c_2 x_3 = d_2 - \delta_1 \frac{a_2}{\beta_1} \tag{B8a}$$

Now set $b_2 - \frac{a_2}{\beta_1} c_2 = \beta_2$ and $d_2 - \delta_1 \frac{a_2}{\beta_1} = \delta_2$

So equation (B8a) may be written as

$$\beta_2 x_2 + c_2 x_3 = \delta_2 \tag{B8b}$$

Which is exactly the form we require.

Now consider the equations

$$\beta_2 x_2 + c_2 x_3 = \delta_2 \qquad \text{(B8b)}$$

$$a_3 x_2 + b_3 x_3 + c_3 X_4 = d_3 \qquad \text{(B9)}$$

Multiply equation (B8b) by $\dfrac{a_3}{\beta_2}$ to get

$$a_3 x_2 + \frac{a_3}{\beta_2} c_2 x_3 = \frac{\delta_2}{\beta_2} a_3 \qquad \text{(B8c)}$$

Subtracting equation (B8c) from equation (B9) yields

$$(b_3 - \frac{a_3}{\beta_2} c_2) x_3 + c_4 x_4 = d_3 - \frac{\delta_2}{\beta_2} a_3 \qquad \text{(B9a)}$$

so if $\beta_1 = b_3 - \dfrac{a_3}{\beta_2} c_2$ and $\delta_3 = d_3 - \dfrac{\delta_2}{\beta_3} a_3$
equation (B9) becomes

$$\beta_3 x_3 + c_4 x_4 = \delta_3 \qquad \text{(B9b)}$$

and so on.

In general we have

$$\beta_j = b_j - \frac{a_j}{\beta_{j-1}} c_{j-1} \qquad \text{(B10)}$$

and

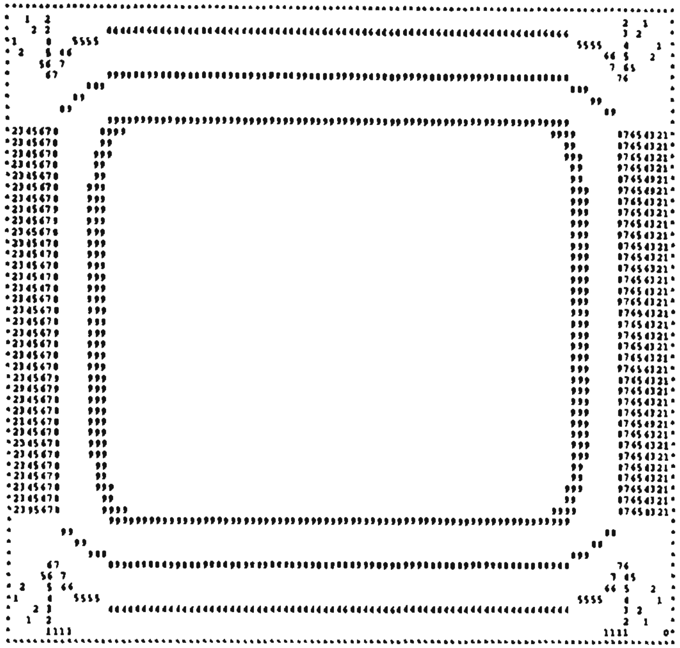
$$\delta_j = d_j - \frac{\delta_{j-1}}{\beta_{j-1}} a_j \qquad \text{(B11)}$$

remembering that we have set $\beta_1 = b_1$ *and* $\delta_1 = d_1$, therefore all the β and δ can be found and thus all the x_i can be found.

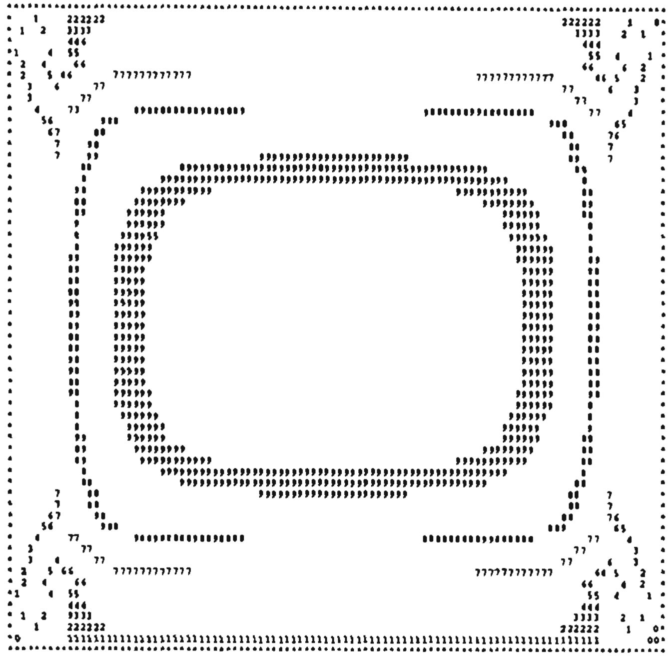
APPENDIX C

CONTOUR MAPS OF TEMPERATURE DISTRIBUTION, VELOCITY DISTRIBUTION AND VECTOR POTENTIAL FIELD

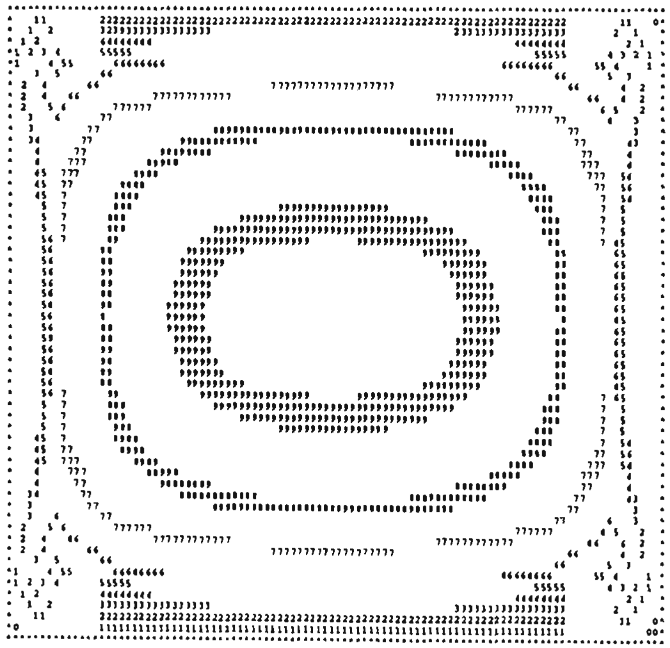
realtime= 0.4829609
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
0.79 3.96 7.13 10.30 13.47 16.64 19.81 22.99 26.15 29.31



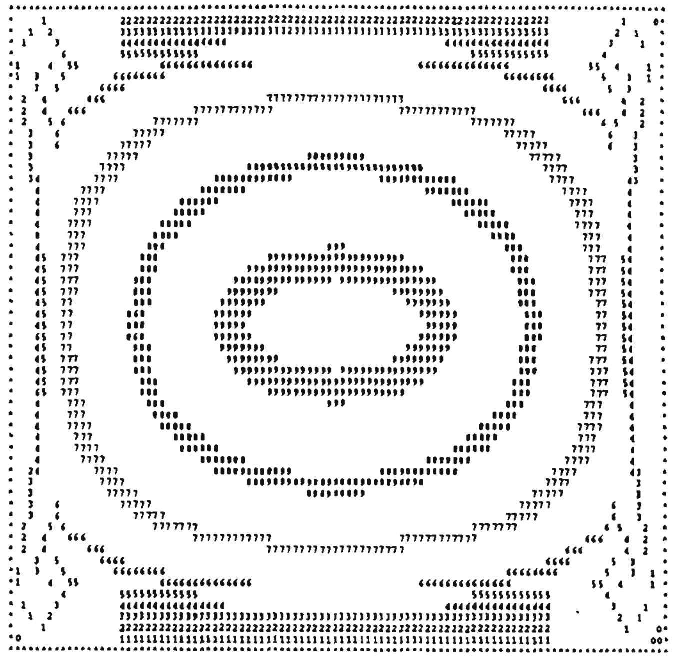
realtime= 0.9843096
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
0.79 3.97 7.15 10.33 13.51 16.68 19.86 23.04 26.22 29.40



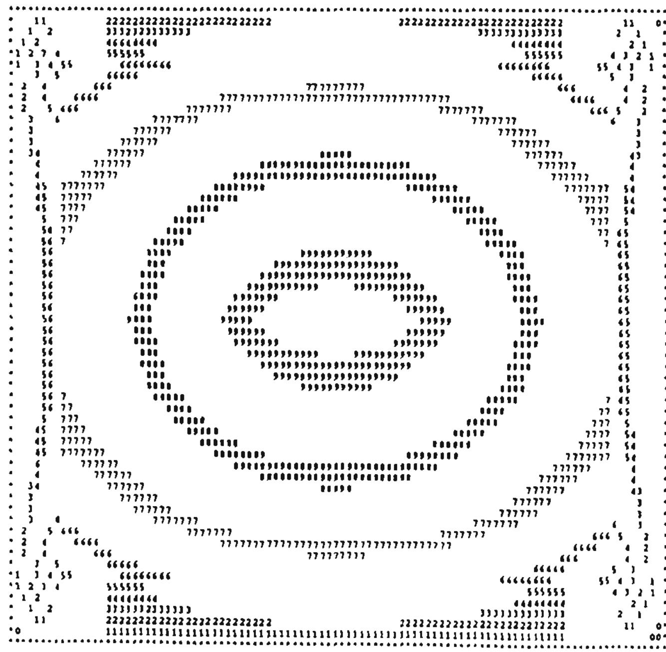
realtime= 1.485750
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
0.79 3.94 7.09 10.24 13.39 16.54 19.69 22.84 25.98 29.13



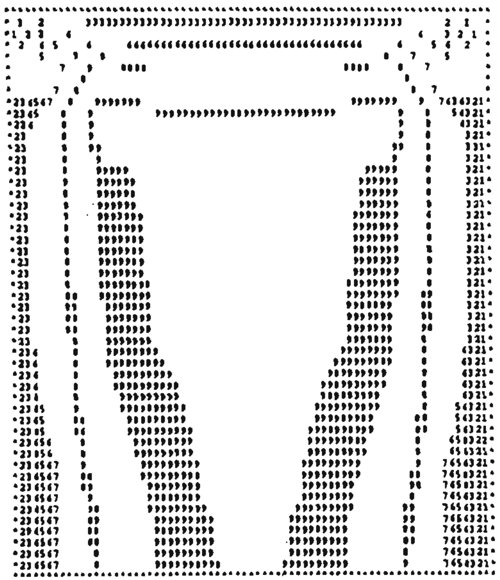
realtime= 1.987191
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
0.74 3.70 6.66 9.62 12.58 15.54 18.50 21.46 24.42 27.38



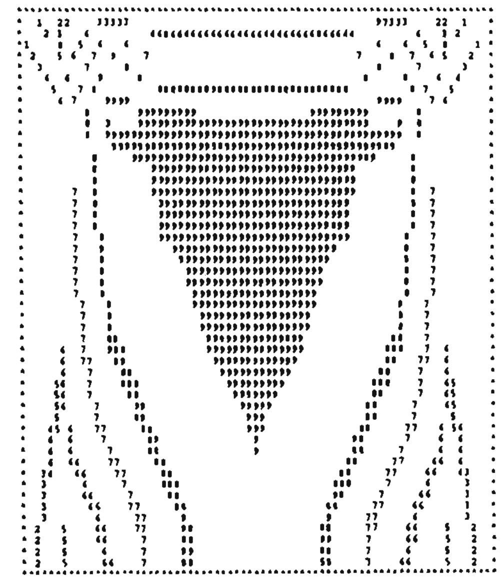
realtime= 2.489632
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
0.65 3.25 5.85 8.45 11.04 13.64 16.24 18.84 21.44 24.04



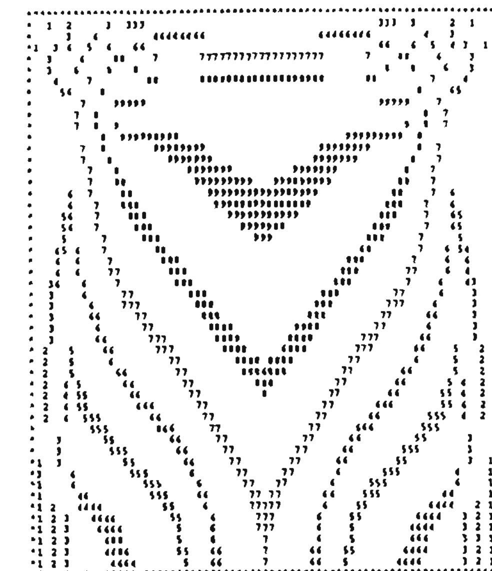
realtime= 0.4528669
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE ;
0.81 4.06 7.30 10.55 13.79 17.04 20.28 23.53 26.77 30.02



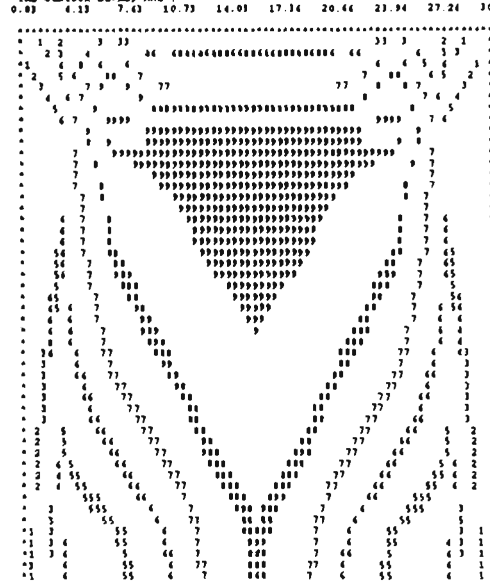
realtime= 0.9943098
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE ;
0.83 4.16 7.48 10.81 14.14 17.46 20.79 24.11 27.44 30.77



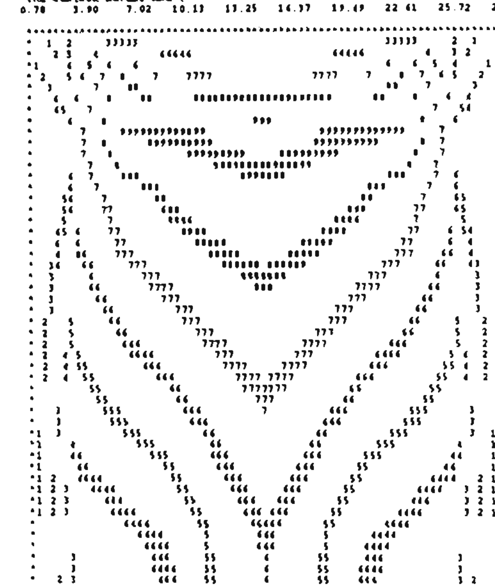
realtime= 1.987191
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE ;
0.80 4.00 7.21 10.41 13.61 16.81 20.02 23.22 26.42 29.63



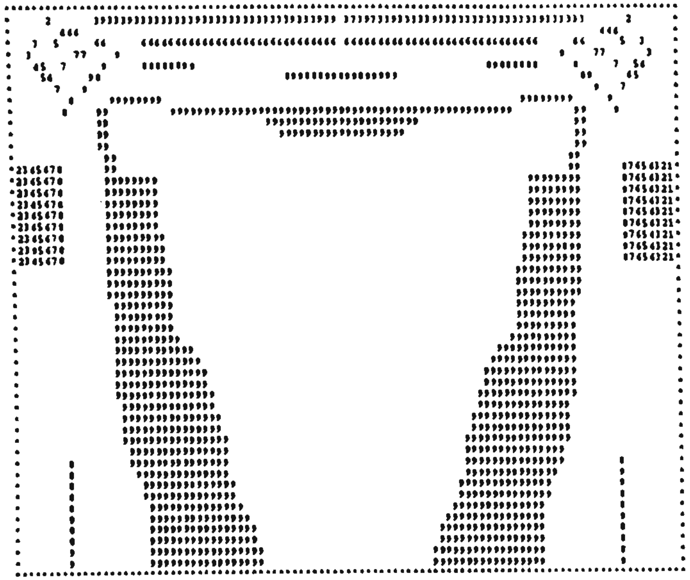
realtime= 1.485750
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE ;
0.83 4.13 7.43 10.73 14.03 17.34 20.64 23.94 27.24 30.54



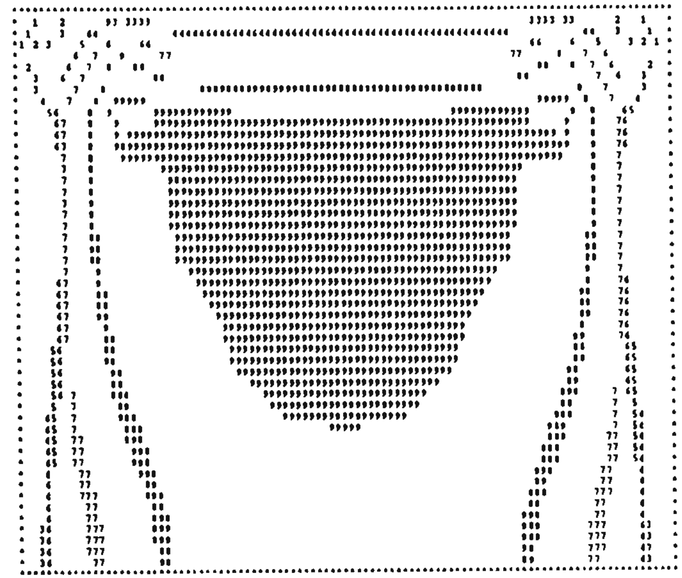
realtime= 2.488432
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE ;
0.78 3.90 7.02 10.13 13.25 16.37 19.49 22.61 25.72 28.84



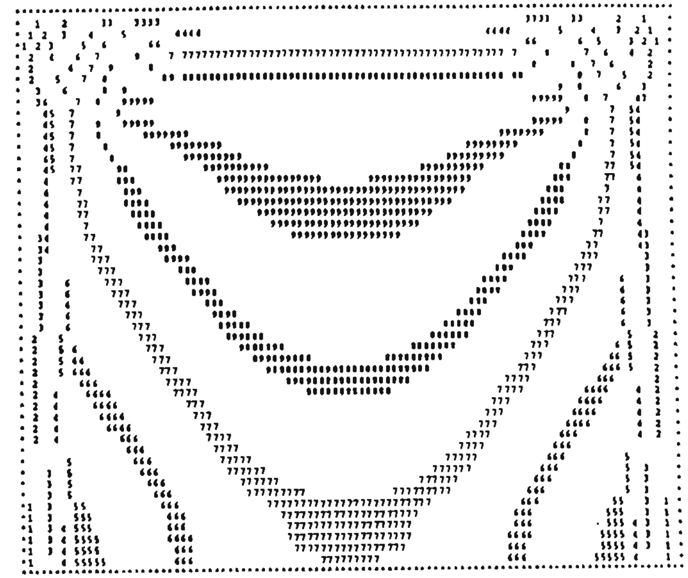
realtime= 0.4828689
***** CONTOUR MAP OF TEMPERATURE ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.91 4.06 7.32 10.57 13.82 17.07 20.32 23.57 26.92 30.07



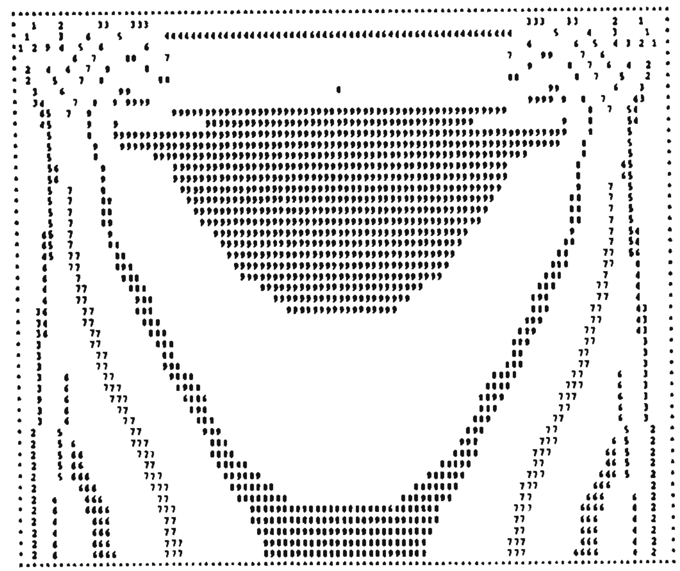
realtime= 0.9941896
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.83 4.17 7.50 10.93 14.17 17.50 20.83 24.17 27.50 30.93



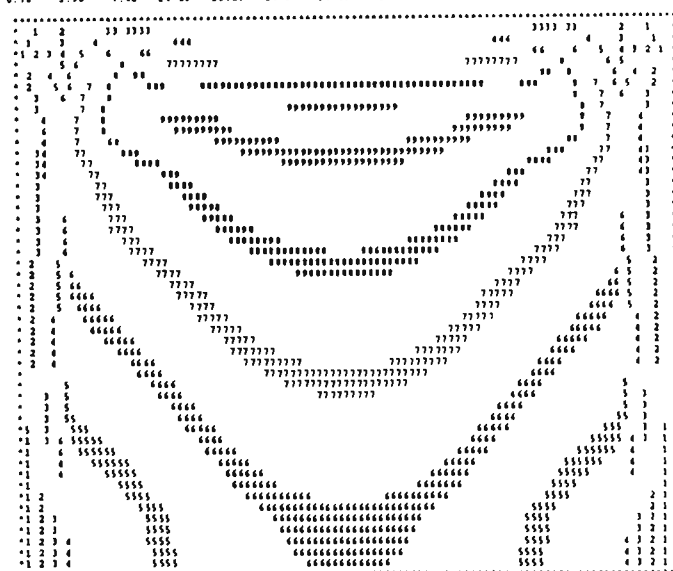
realtime= 1.987191
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.80 4.01 7.21 10.41 13.62 16.82 20.03 23.23 26.43 29.64



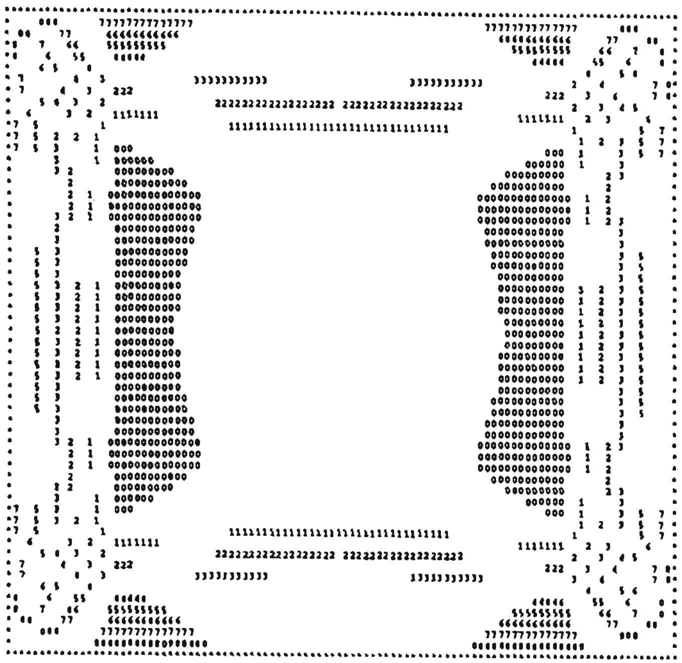
realtime= 1.485750
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.92 4.12 7.41 10.71 14.00 17.30 20.59 23.89 27.59 30.48



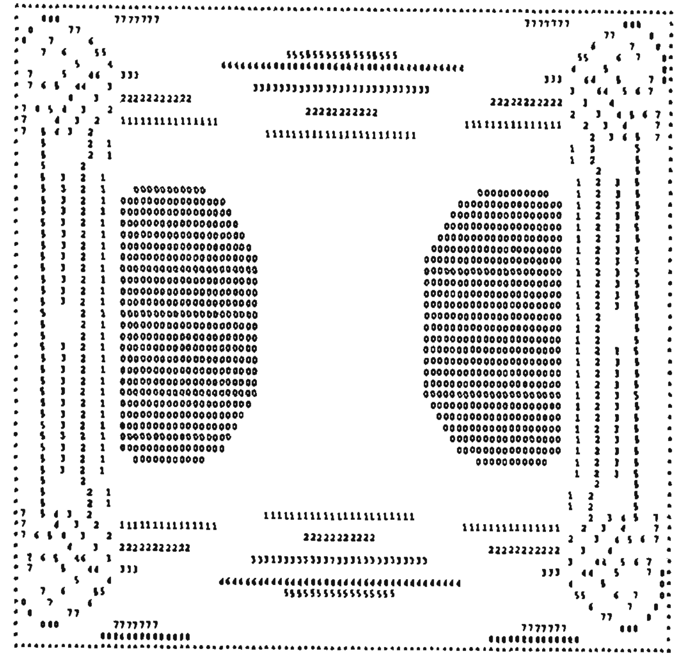
realtime= 2.480612
1 ***** CONTOUR MAP OF TEMPERATURE ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.78 3.90 7.02 10.13 13.25 16.37 19.69 22.61 25.72 28.84



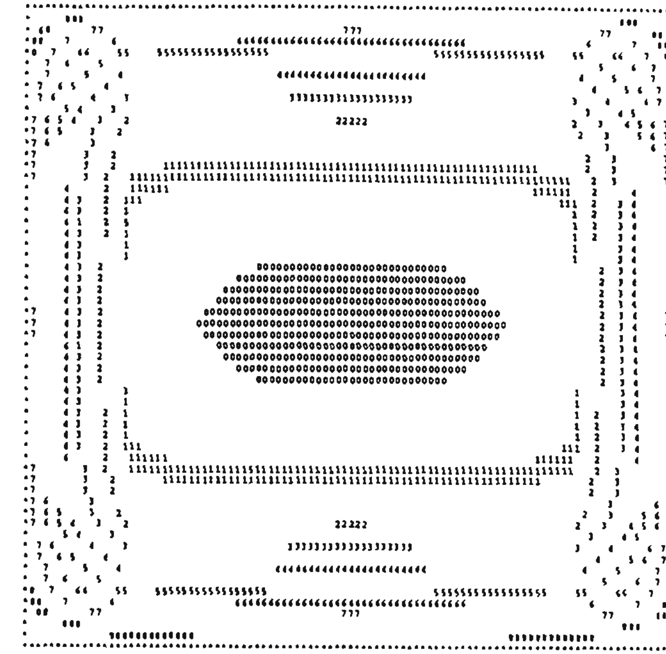
realtime= 0.4825600
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-62.27 -31.00 -3.89 25.30 54.49 83.69 112.88 142.07 171.26 200.45



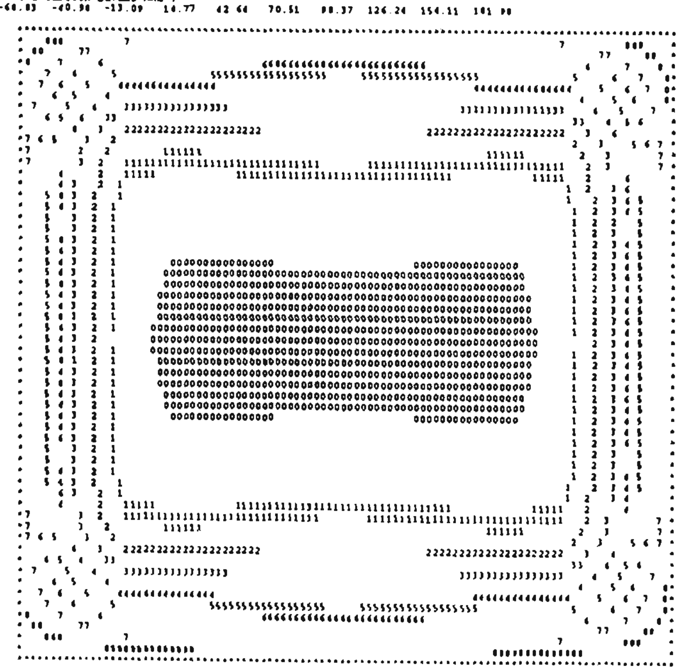
realtime= 0.9843090
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-65.90 -34.82 -7.67 21.49 50.65 79.81 108.97 138.13 167.29 196.45



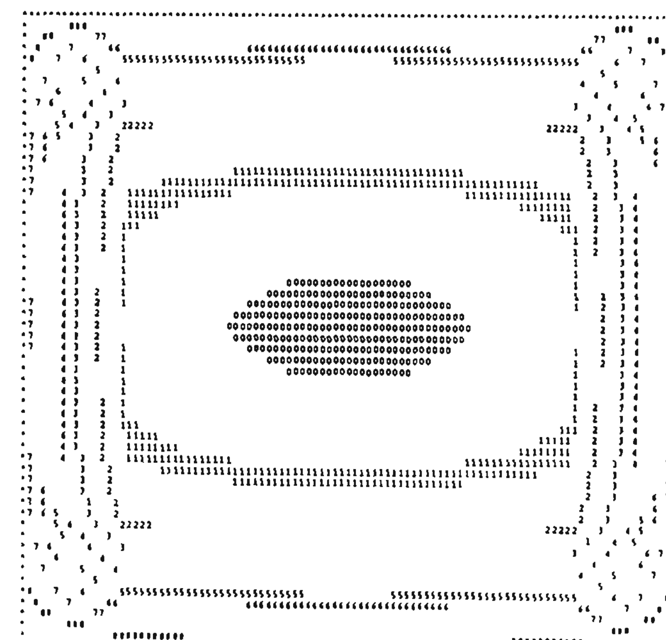
realtime= 1.987101
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-67.82 -42.28 -16.94 8.50 33.95 59.39 84.83 110.27 135.72 161.16



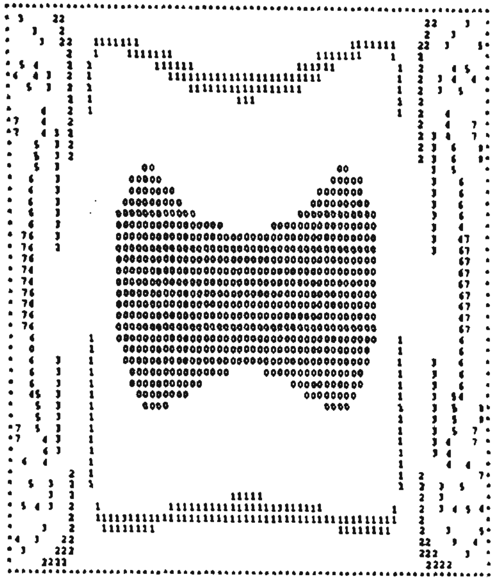
realtime= 3.445750
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-68.83 -40.96 -13.09 14.77 42.64 70.51 98.37 126.24 154.11 181.98



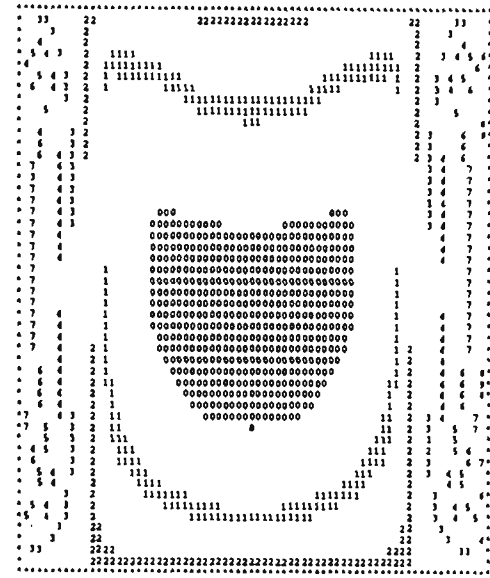
realtime= 2.488432
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-60.36 -38.07 -15.78 6.52 28.81 51.10 73.39 95.68 117.97 140.26



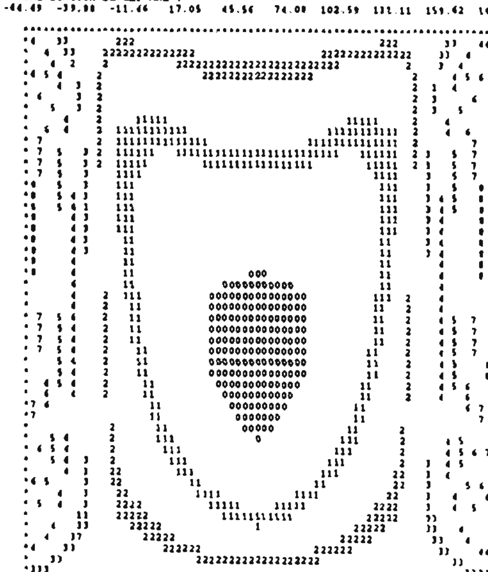
realtime= 0.4020649
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-46.54 -19.16 8.22 36.40 62.87 90.35 117.73 145.10 172.48 199.86



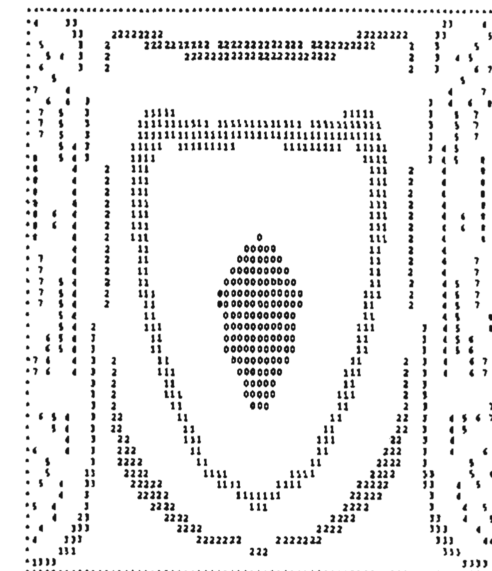
realtime= 0.8843096
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-56.45 -28.11 0.23 28.57 56.90 85.24 113.58 141.92 170.25 198.59



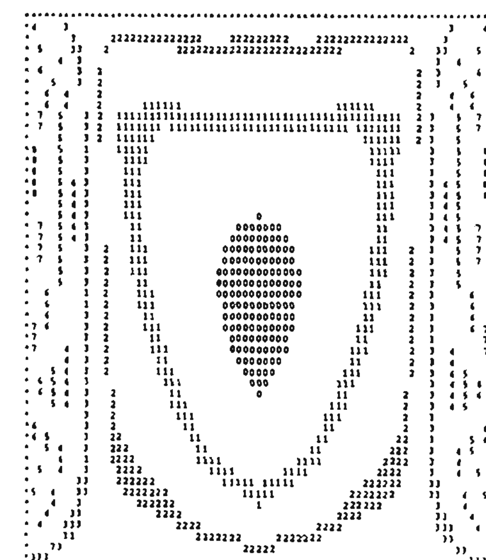
realtime= 1.485750
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-68.49 -39.98 -11.46 17.05 45.56 74.08 102.59 131.11 159.62 188.14



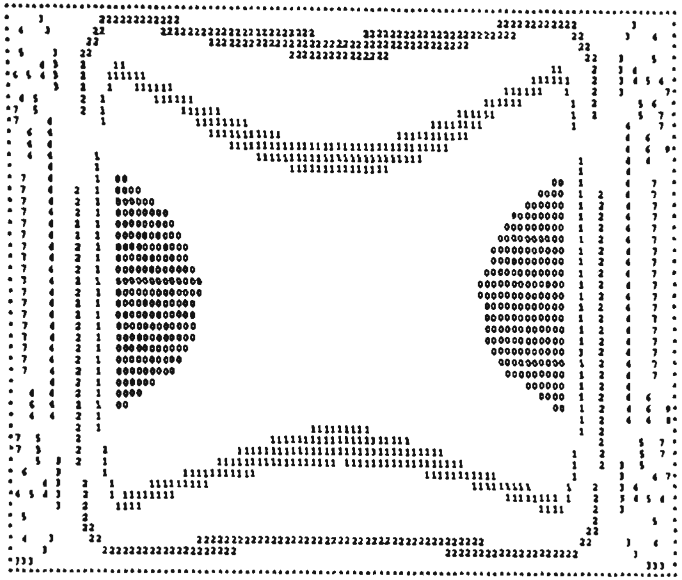
realtime= 1.987191
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-68.51 -41.63 -14.74 12.14 39.02 65.90 92.79 119.67 146.55 173.43



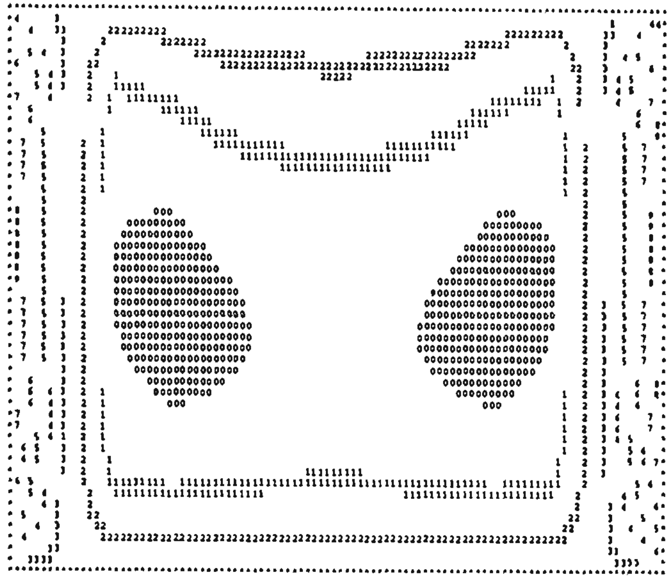
realtime= 2.488632
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-59.91 -35.82 -11.74 12.35 36.44 60.52 84.61 108.69 132.78 156.87



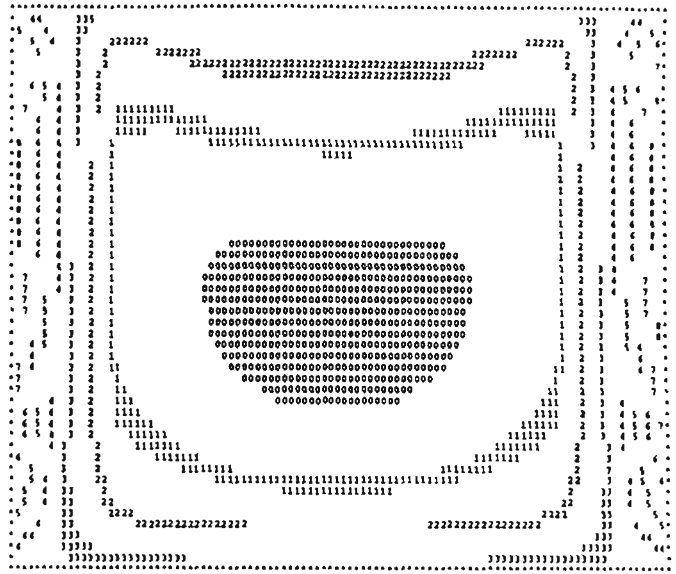
realtime= 0.6828699
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-58.95 -33.97 -8.98 16.00 40.99 65.97 90.96 115.94 140.93 165.91



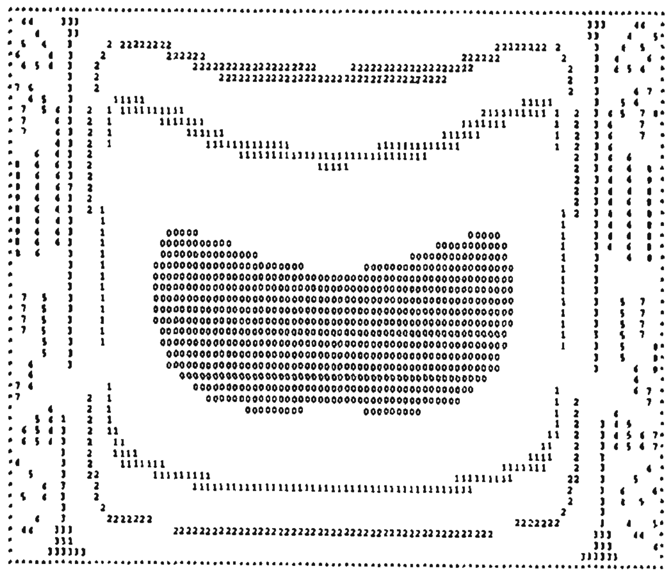
realtime= 0.9863096
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-66.89 -41.58 -16.28 9.01 34.34 59.65 84.95 110.26 135.57 160.88



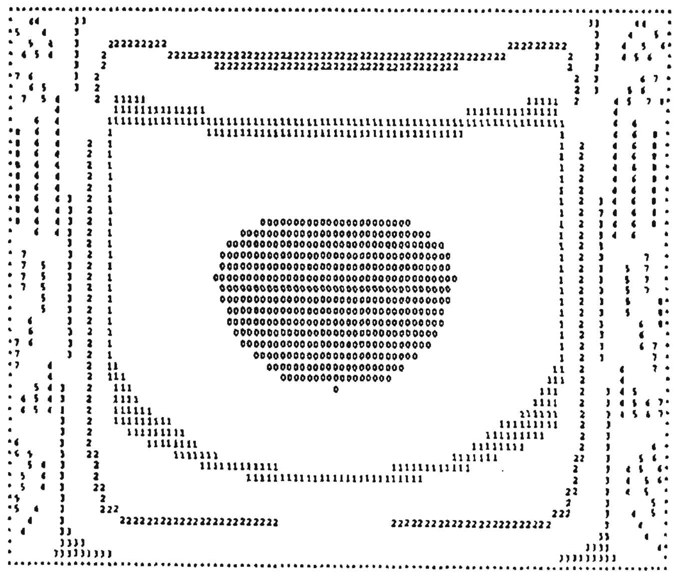
realtime= 1.987191
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-69.51 -46.63 -23.74 -0.86 22.02 44.90 67.78 90.66 113.54 136.43



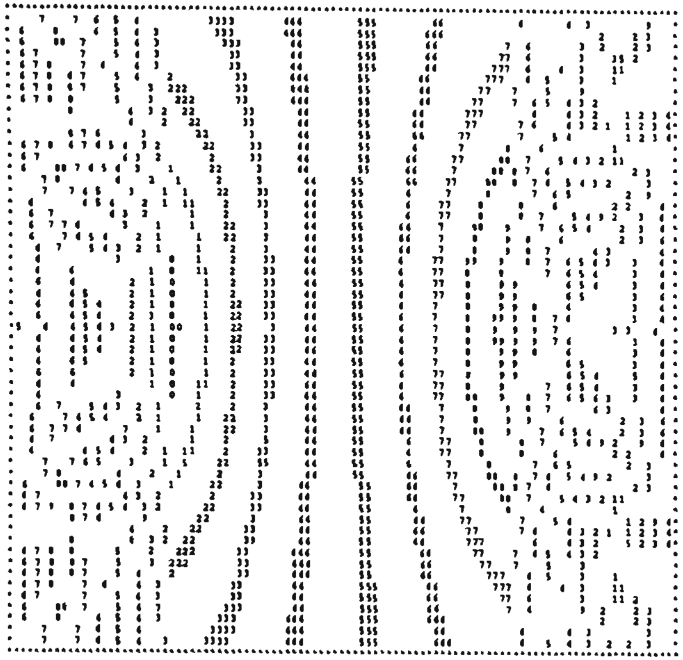
realtime= 1.495750
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-71.12 -46.49 -21.86 2.77 27.41 52.04 76.47 101.30 125.94 150.57



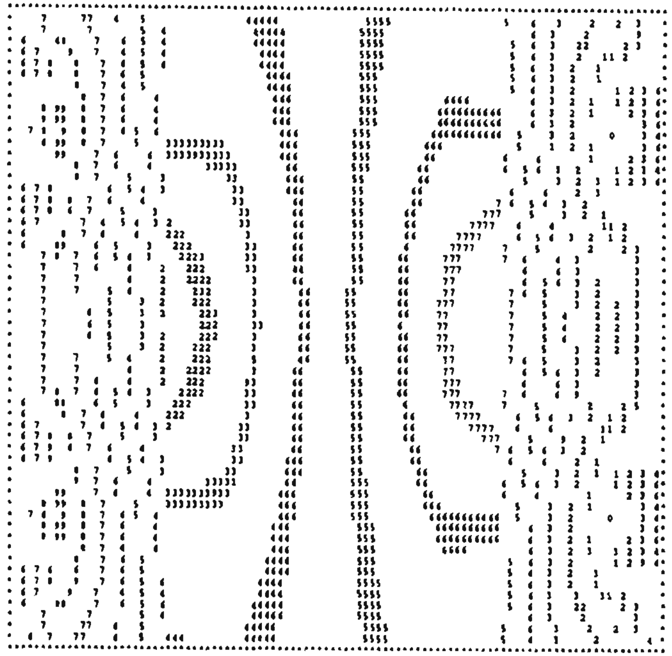
realtime= 2.489632
1 ***** CONTOUR MAP OF U-VELOCITY ***** ; K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-60.86 -40.55 -20.25 0.05 20.35 40.66 60.96 81.26 101.57 121.87



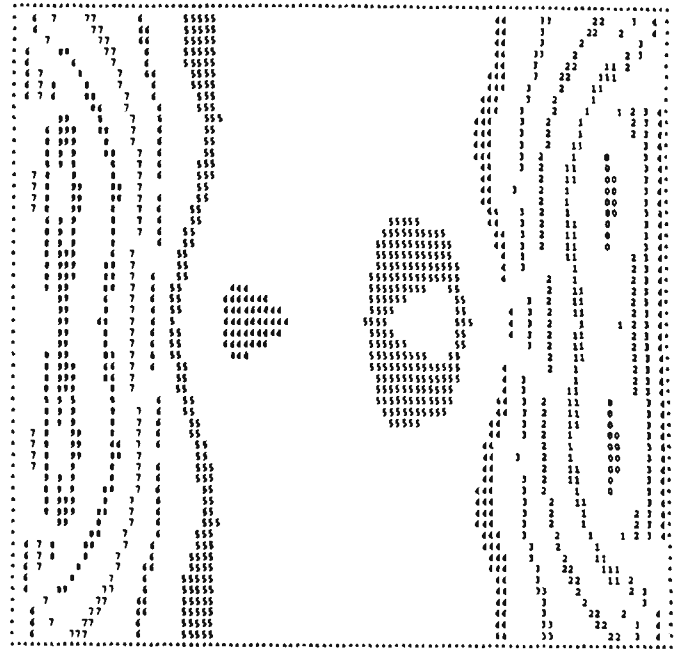
realtime= 0.4829688
5 ***** CONTOUR MAP OF V-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
-0.96 -0.74 -0.53 -0.32 -0.11 0.11 0.32 0.53 0.74 0.96



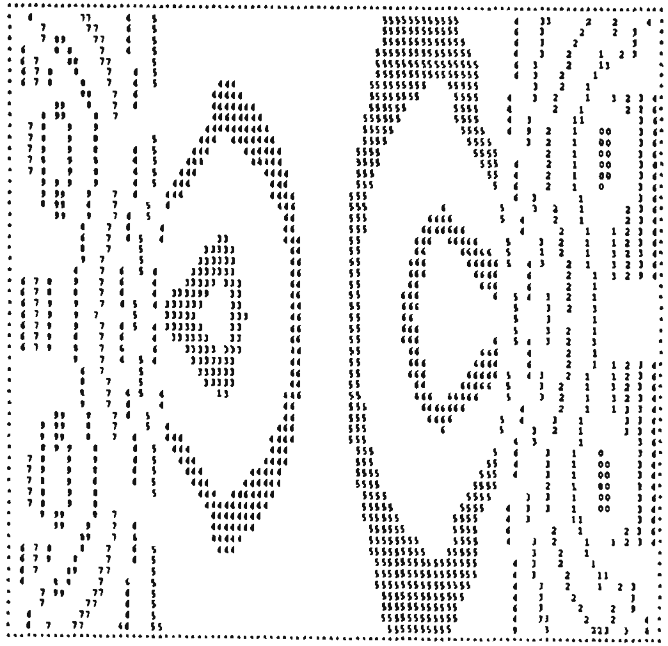
realtime= 0.9943896
1 ***** CONTOUR MAP OF V-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
-3.92 -3.05 -2.18 -1.31 -0.44 0.44 1.31 2.18 3.05 3.92



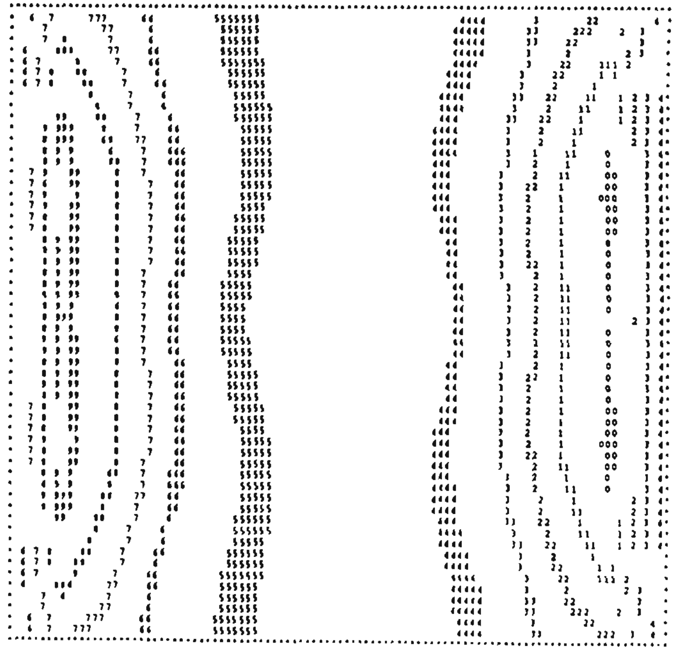
realtime= 1.987191
1 ***** CONTOUR MAP OF V-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
-8.52 -6.63 -4.74 -2.84 -0.95 0.95 2.84 4.74 6.63 8.52



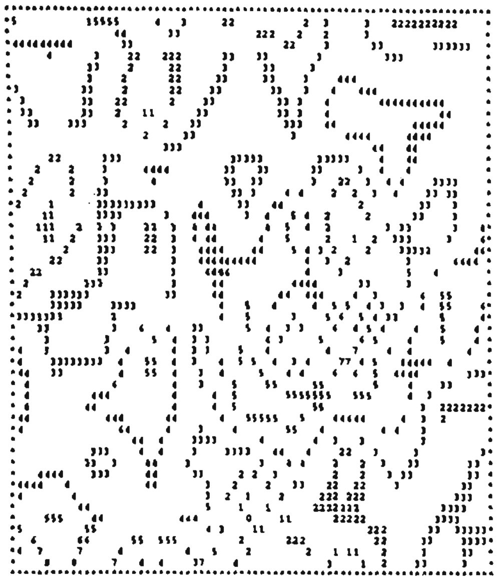
realtime= 1.485750
1 ***** CONTOUR MAP OF V-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
-6.82 -5.31 -3.79 -2.27 -0.74 0.74 2.27 3.79 5.31 6.82



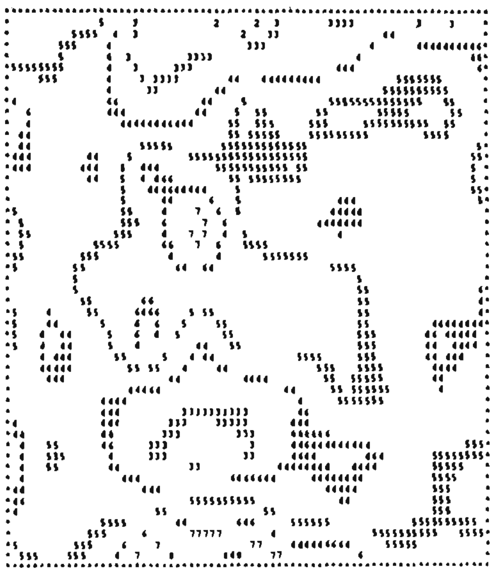
realtime= 2.486632
1 ***** CONTOUR MAP OF V-VELOCITY ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE :
-8.67 -6.74 -4.83 -2.89 -0.96 0.96 2.89 4.82 6.74 8.67



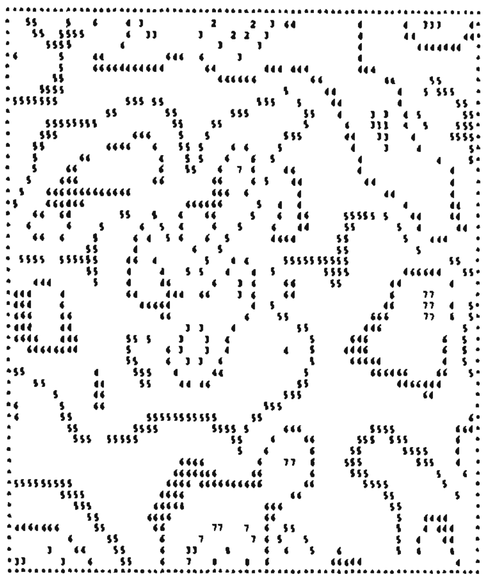
realtime= 0.4820609
1 ***** CONTOUR MAP OF V-VELOCITY a***** ; J = 3 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



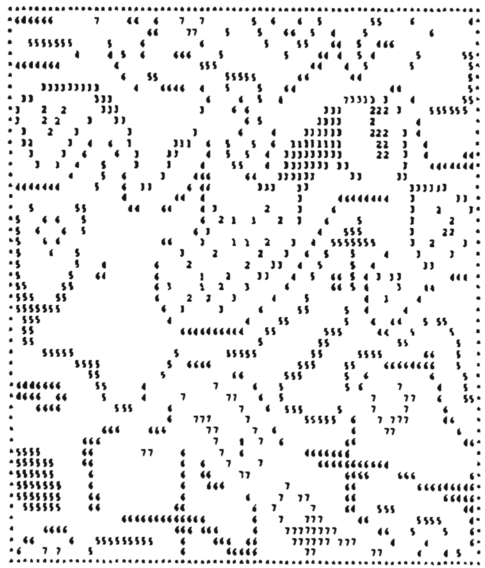
realtime= 0.9843894
1 ***** CONTOUR MAP OF V-VELOCITY a***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



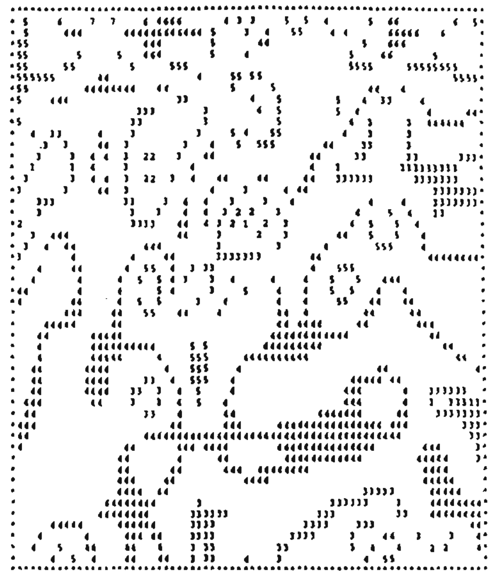
realtime= 1.485350
1 ***** CONTOUR MAP OF V-VELOCITY a***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



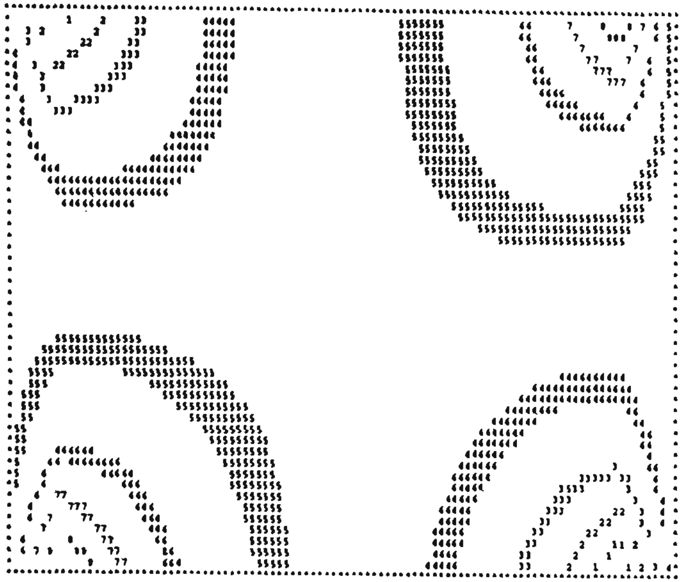
realtime= 1.987191
1 ***** CONTOUR MAP OF V-VELOCITY a***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



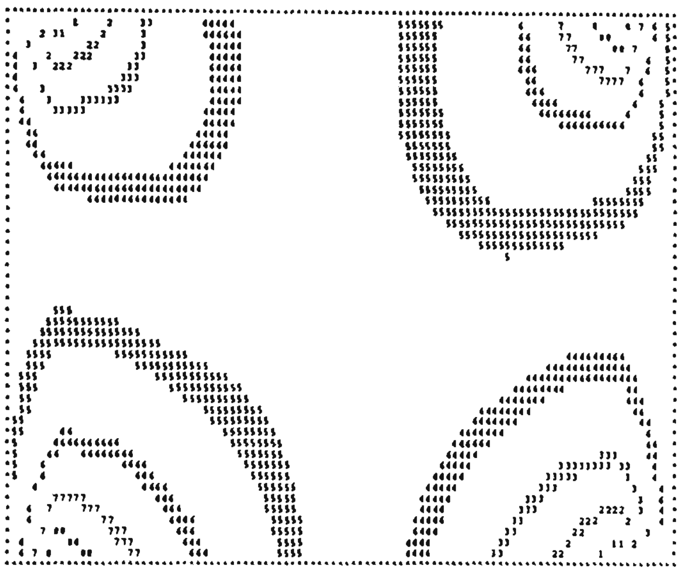
realtime= 2.488432
1 ***** CONTOUR MAP OF V-VELOCITY a***** ; J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



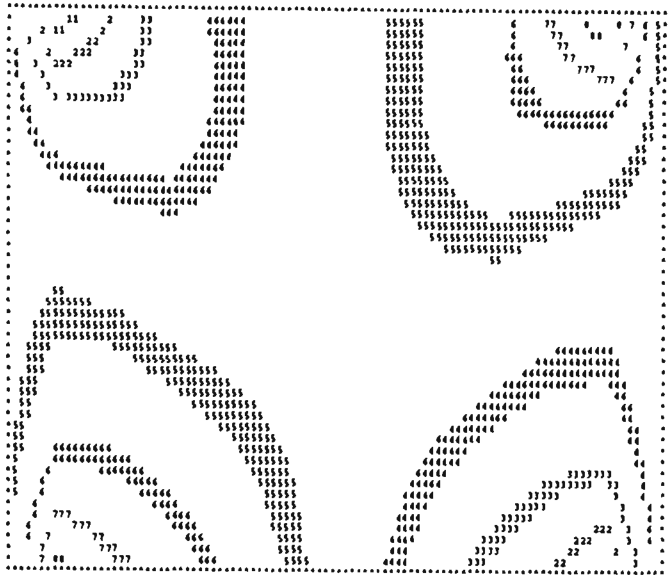
realtime= 0.4828688
1 ***** CONTOUR MAP OF V-VELOCITY ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-112.57 -87.56 -62.54 -37.52 -12.51 12.51 37.52 62.54 87.56 112.57



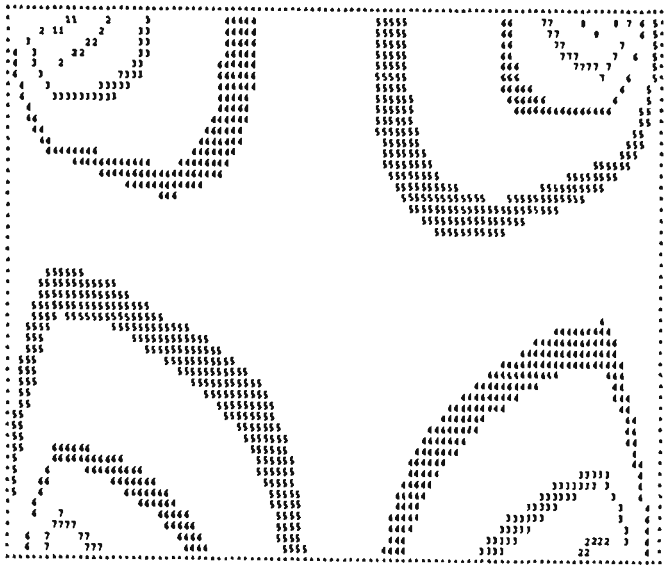
realtime= 0.9863096
1 ***** CONTOUR MAP OF V-VELOCITY ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-113.35 -88.08 -62.92 -37.75 -12.58 12.58 37.75 62.92 88.08 113.25



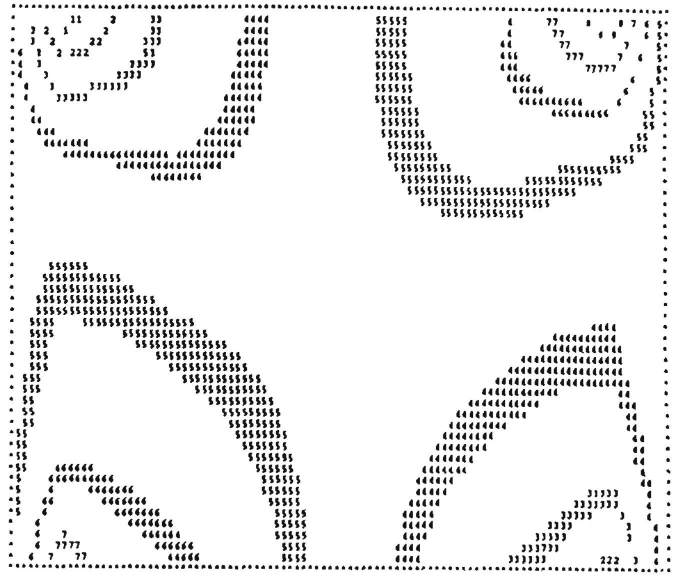
realtime= 1.485750
1 ***** CONTOUR MAP OF V-VELOCITY ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-114.78 -89.27 -63.77 -38.26 -12.75 12.75 38.26 63.77 89.27 114.76



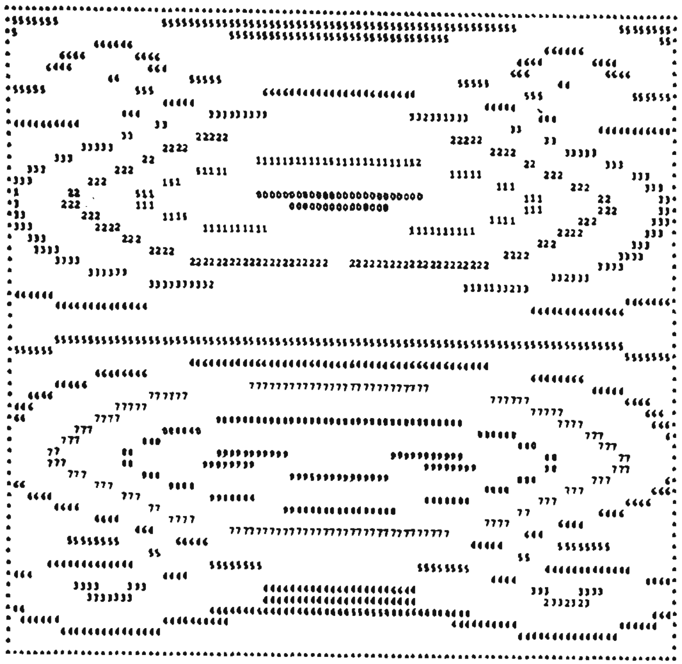
realtime= 1.987191
1 ***** CONTOUR MAP OF V-VELOCITY ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-110.99 -86.33 -61.66 -37.00 -12.33 12.33 37.00 61.66 86.33 110.99



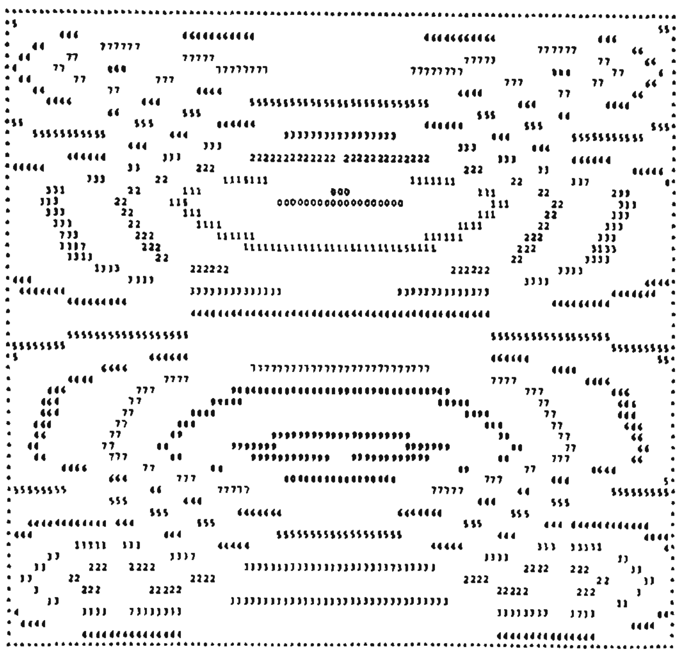
realtime= 2.488632
1 ***** CONTOUR MAP OF V-VELOCITY ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
-103.26 -80.31 -57.36 -34.42 -11.47 11.47 34.42 57.36 80.31 103.26



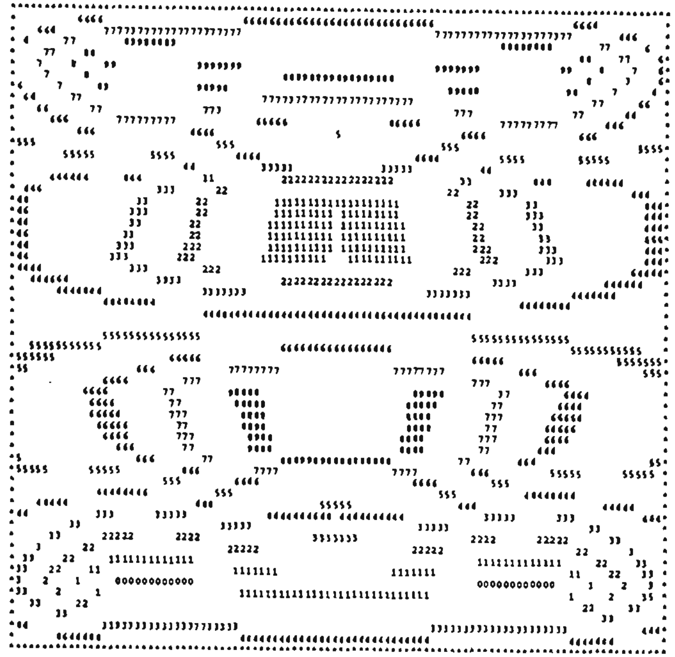
realtime= 0.4824689
1 ***** CONTOUR MAP OF W-VELOCITY m***** ; I = 4 (K = 0.50)
THE CONTOUR LEVELS ARE ;
-1.55 -1.20 -0.86 -0.52 -0.17 0.17 0.52 0.86 1.20 1.55



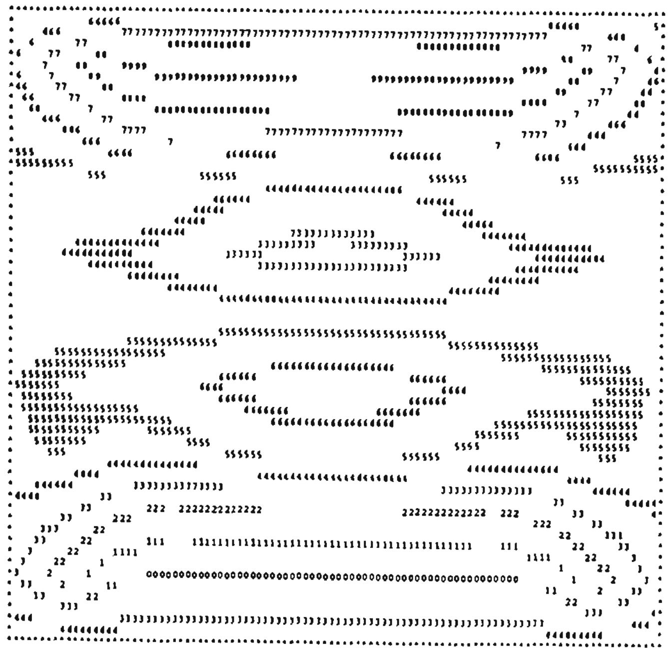
realtime= 0.9843096
1 ***** CONTOUR MAP OF W-VELOCITY m***** ; I = 4 (K = 0.50)
THE CONTOUR LEVELS ARE ;
-4.34 -3.38 -2.41 -1.45 -0.48 0.48 1.45 2.41 3.38 4.34



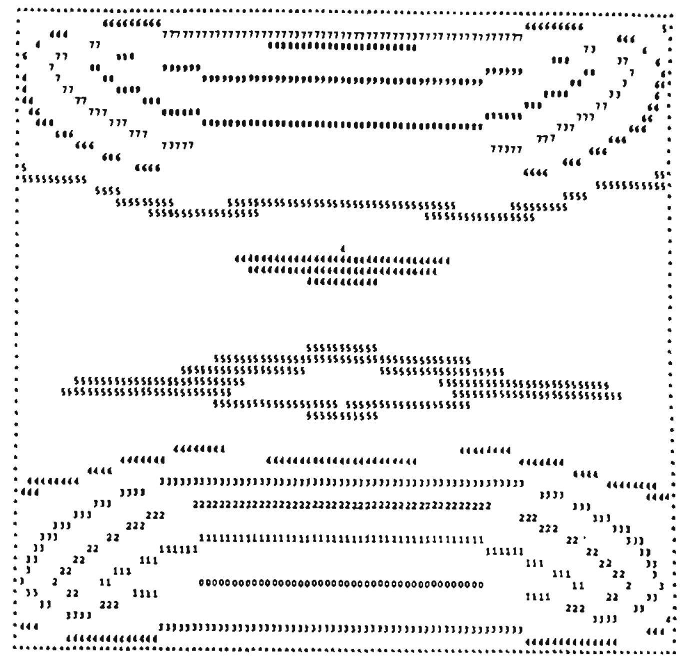
realtime= 1.485750
1 ***** CONTOUR MAP OF W-VELOCITY m***** ; I = 6 (K = 0.50)
THE CONTOUR LEVELS ARE ;
-5.85 -4.55 -3.25 -1.95 -0.65 0.65 1.95 3.25 4.55 5.85



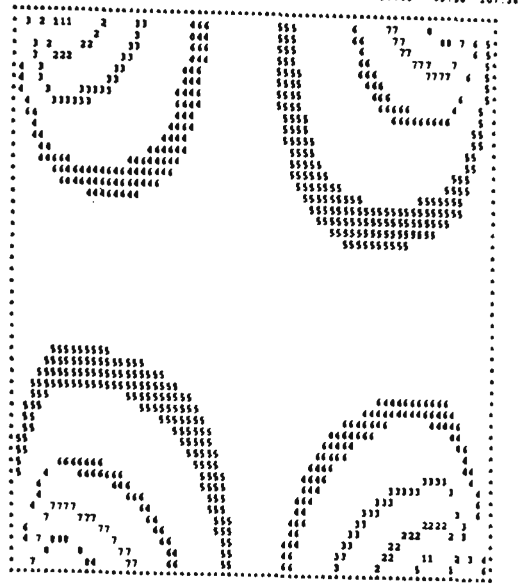
realtime= 1.987181
1 ***** CONTOUR MAP OF W-VELOCITY m***** ; I = 4 (K = 0.50)
THE CONTOUR LEVELS ARE ;
-7.78 -6.05 -4.32 -2.59 -0.86 0.86 2.59 4.32 6.05 7.78



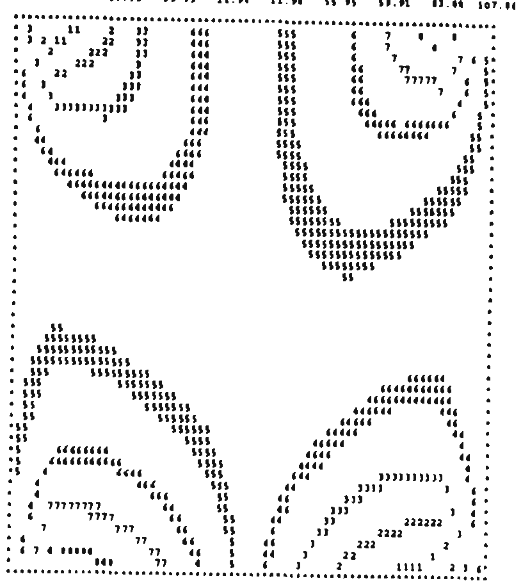
realtime= 2.488632
1 ***** CONTOUR MAP OF W-VELOCITY m***** ; I = 6 (K = 0.50)
THE CONTOUR LEVELS ARE ;
-8.46 -6.58 -4.70 -2.82 -0.94 0.94 2.82 4.70 6.58 8.46



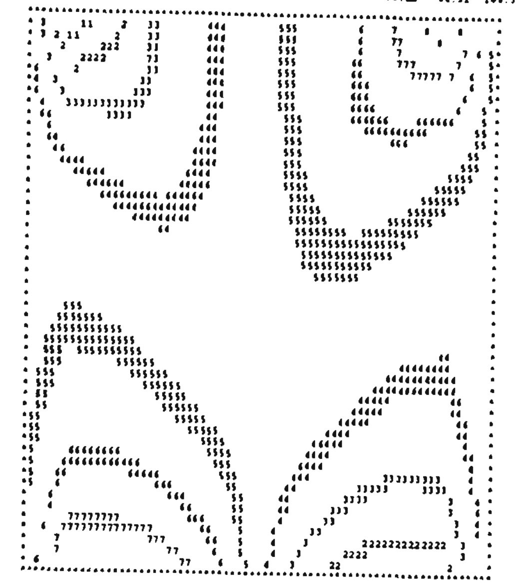
realtime= 0.4828689
1 ***** CONTOUR MAP OF W-VELOCITY ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-107.36 -83.50 -59.66 -35.79 -11.93 11.93 35.79 59.66 83.50 107.36



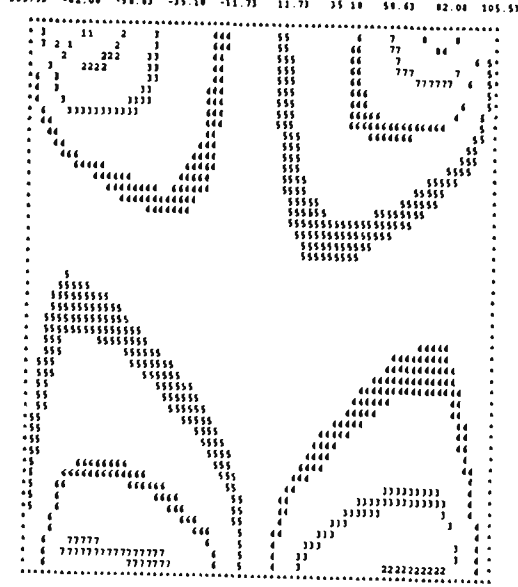
realtime= 0.9843094
1 ***** CONTOUR MAP OF W-VELOCITY ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-107.84 -83.88 -59.91 -35.95 -11.98 11.98 35.95 59.91 83.88 107.84



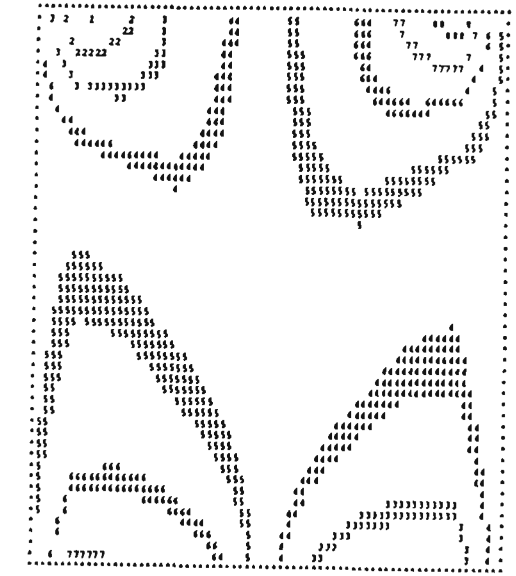
realtime= 1.485750
1 ***** CONTOUR MAP OF W-VELOCITY ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-108.39 -84.31 -60.22 -36.13 -12.04 12.04 36.13 60.22 84.31 108.39



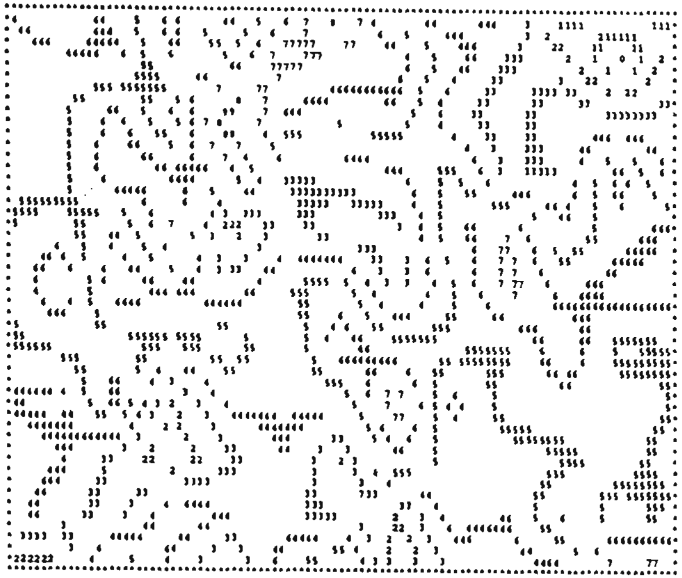
realtime= 1.987191
1 ***** CONTOUR MAP OF W-VELOCITY ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-105.53 -82.08 -58.63 -35.18 -11.73 11.73 35.18 58.63 82.08 105.53



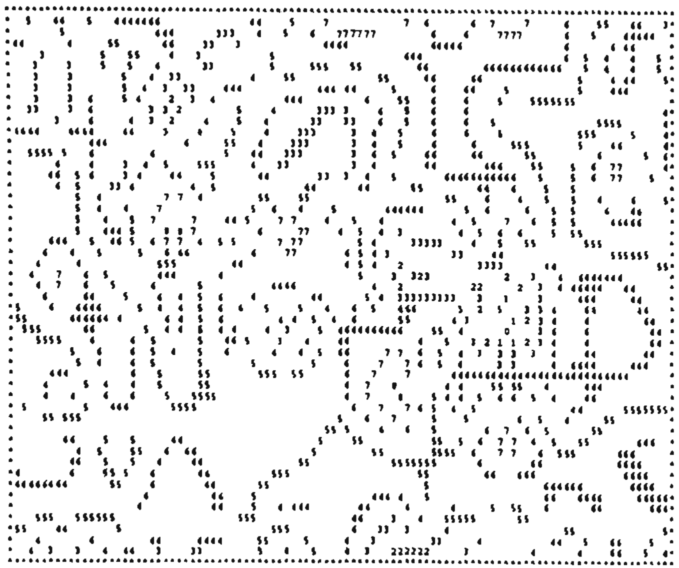
realtime= 2.488632
1 ***** CONTOUR MAP OF W-VELOCITY ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-100.31 -78.02 -55.73 -33.44 -11.15 11.15 33.44 55.73 78.02 100.31



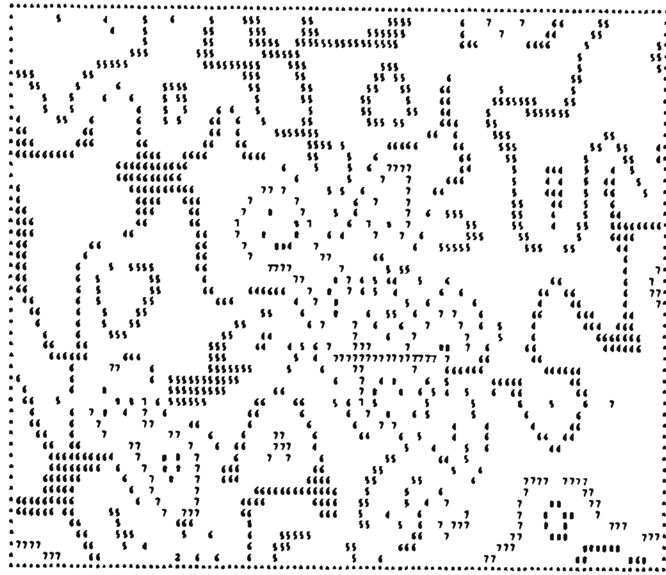
realtime= 0.4920489
***** CONTOUR MAP OF W-VELOCITY ***** , K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



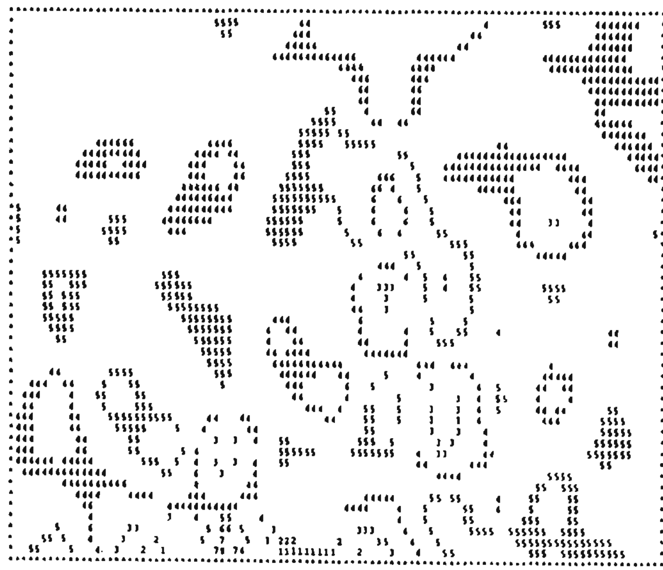
realtime= 0.9843096
***** CONTOUR MAP OF W-VELOCITY ***** , K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



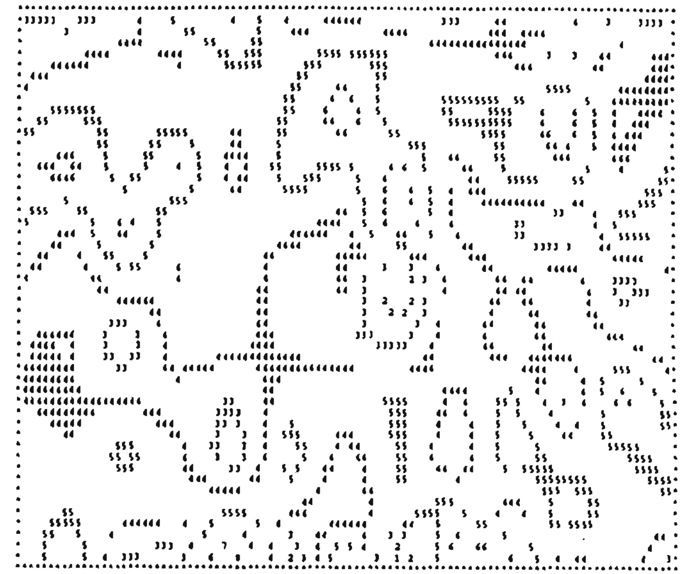
realtime= 1.485750
***** CONTOUR MAP OF W-VELOCITY ***** , K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



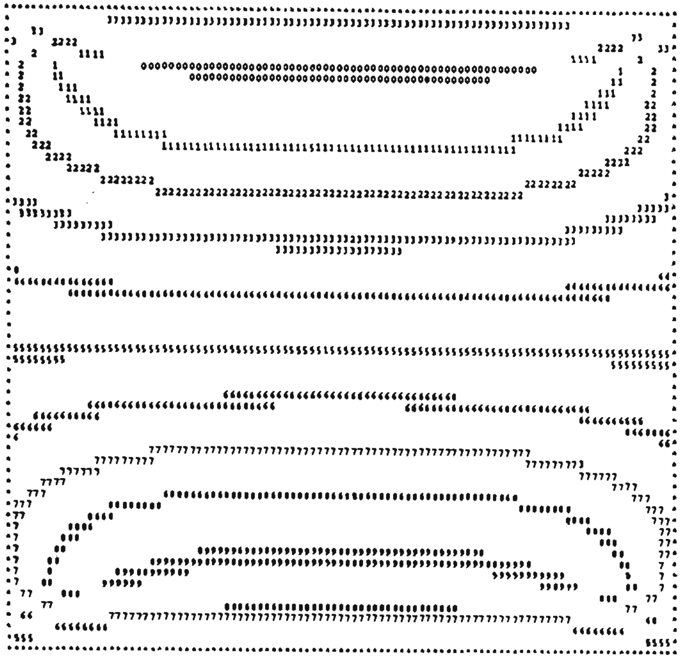
realtime= 1.987191
***** CONTOUR MAP OF W-VELOCITY ***** , K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



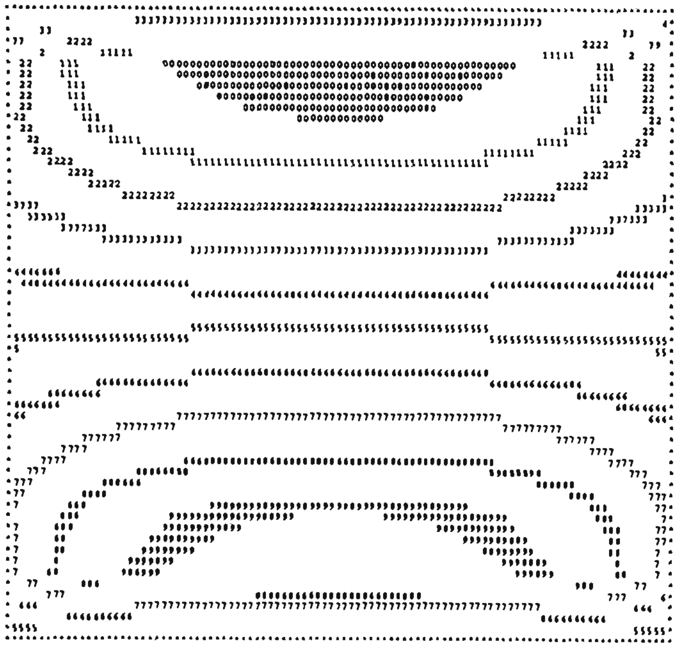
realtime= 2.488632
***** CONTOUR MAP OF W-VELOCITY ***** , K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE ;
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



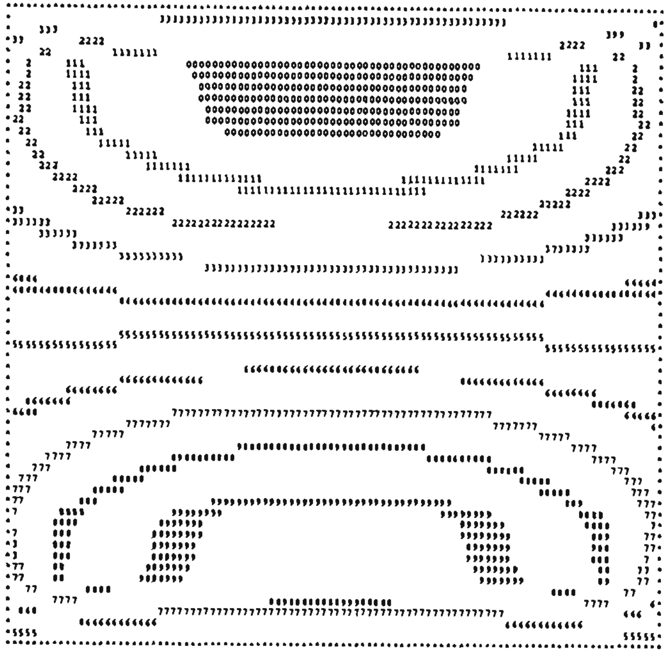
realtime= 0.6020600
1 ***** CONTOUR MAP OF PSI-2 ***** ; I = 6 (R = 0.50)
THE CONTOUR LEVELS ARE :
-17.92 -13.93 -9.95 -5.97 -1.99 1.99 5.97 9.95 13.93 17.92



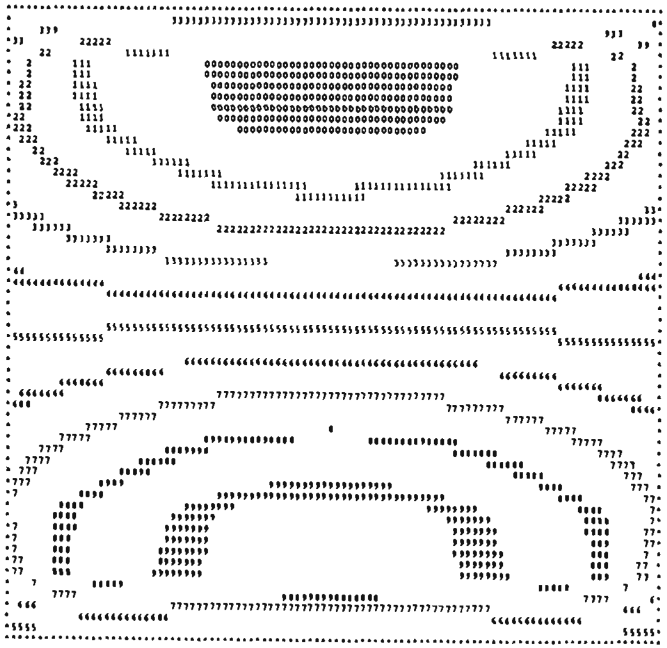
realtime= 0.9863098
1 ***** CONTOUR MAP OF PSI-2 ***** ; I = 6 (R = 0.50)
THE CONTOUR LEVELS ARE :
-18.41 -14.32 -10.23 -6.14 -2.05 2.05 6.14 10.23 14.32 18.41



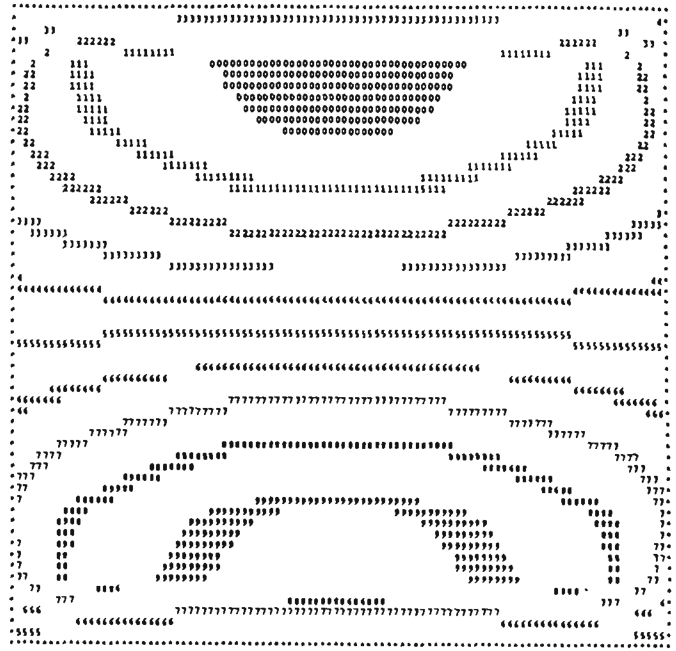
realtime= 1.485760
1 ***** CONTOUR MAP OF PSI-2 ***** ; I = 6 (R = 0.50)
THE CONTOUR LEVELS ARE :
-17.57 -13.66 -9.76 -5.86 -1.95 1.95 5.86 9.76 13.66 17.57



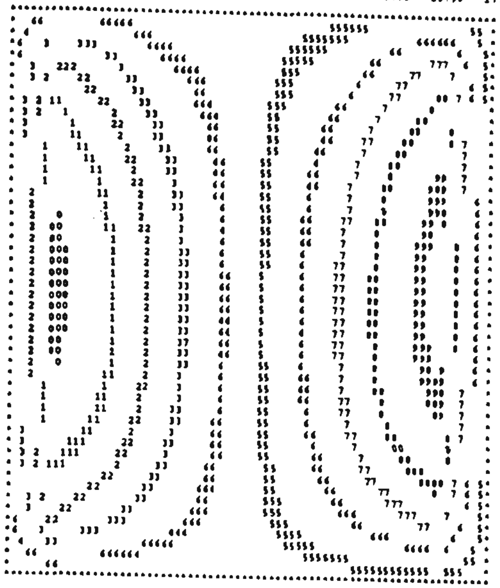
realtime= 1.987191
1 ***** CONTOUR MAP OF PSI-2 ***** ; I = 6 (R = 0.50)
THE CONTOUR LEVELS ARE :
-15.76 -12.26 -8.76 -5.25 -1.75 1.75 5.25 8.76 12.26 15.76



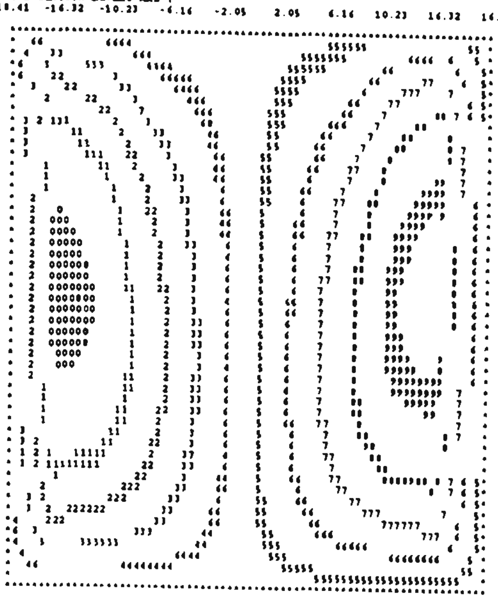
realtime= 2.446632
1 ***** CONTOUR MAP OF PSI-2 ***** ; I = 6 (R = 0.50)
THE CONTOUR LEVELS ARE :
-13.68 -10.64 -7.60 -4.56 -1.52 1.52 4.56 7.60 10.64 13.68



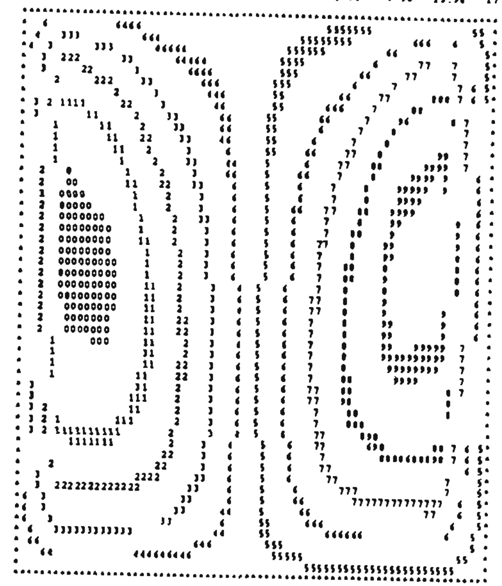
realtime= 0.4820000
1 ***** CONTOUR MAP OF PSI-2 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-17.92 -13.93 -9.95 -5.97 -1.99 1.99 5.97 9.95 13.93 17.92



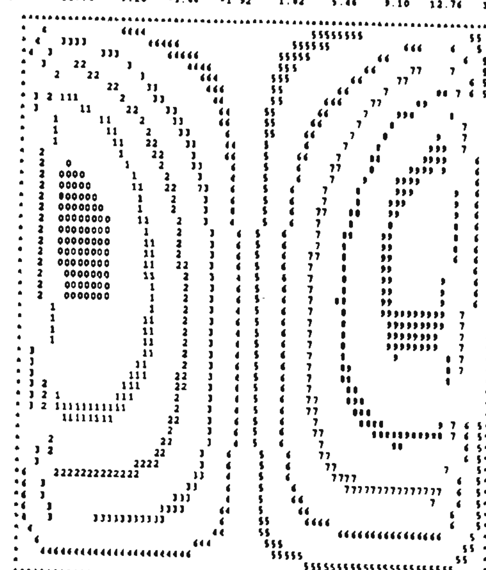
realtime= 0.8843096
1 ***** CONTOUR MAP OF PSI-2 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-18.41 -14.32 -10.23 -6.14 -2.05 2.05 6.14 10.23 14.32 18.41



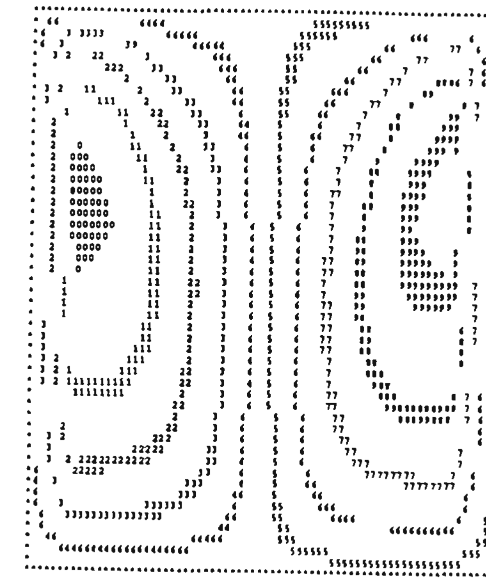
realtime= 1.485750
1 ***** CONTOUR MAP OF PSI-2 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-17.83 -13.86 -9.90 -5.94 -1.98 1.98 5.94 9.90 13.86 17.83



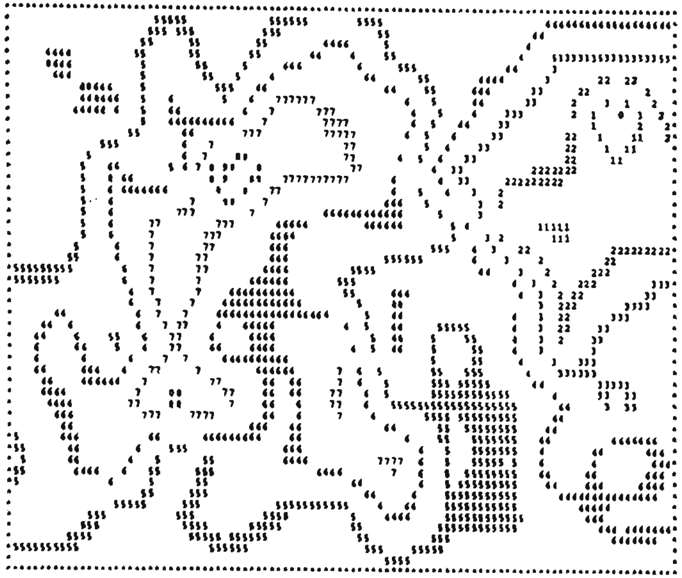
realtime= 1.967191
1 ***** CONTOUR MAP OF PSI-2 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-16.37 -12.74 -9.10 -5.46 -1.82 1.82 5.46 9.10 12.74 16.37



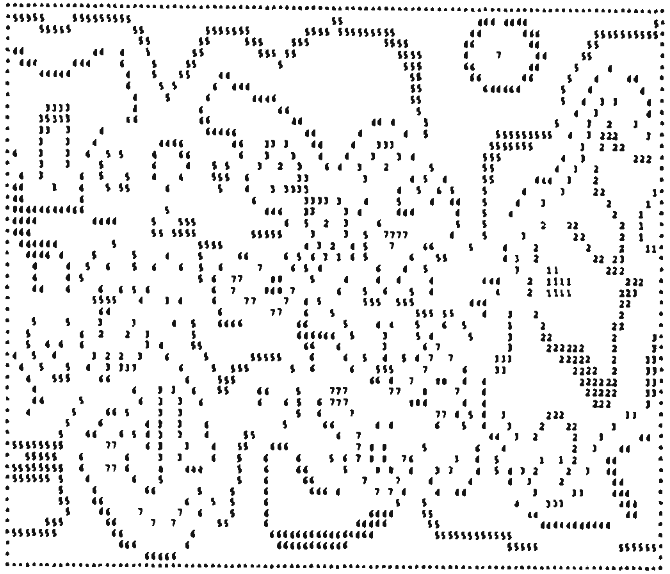
realtime= 2.448632
1 ***** CONTOUR MAP OF PSI-2 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
-14.61 -11.37 -8.12 -4.87 -1.62 1.62 4.87 8.12 11.37 14.61



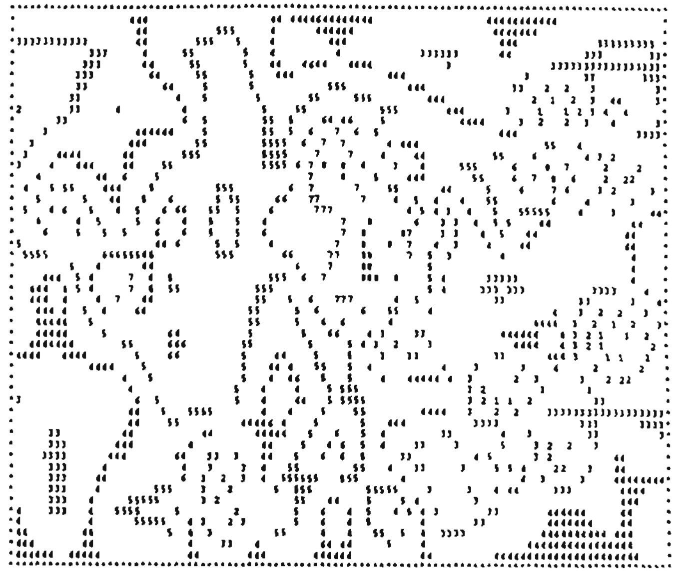
realtime= 0.4828608
1 ***** CONTOUR MAP OF PSI-2 ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



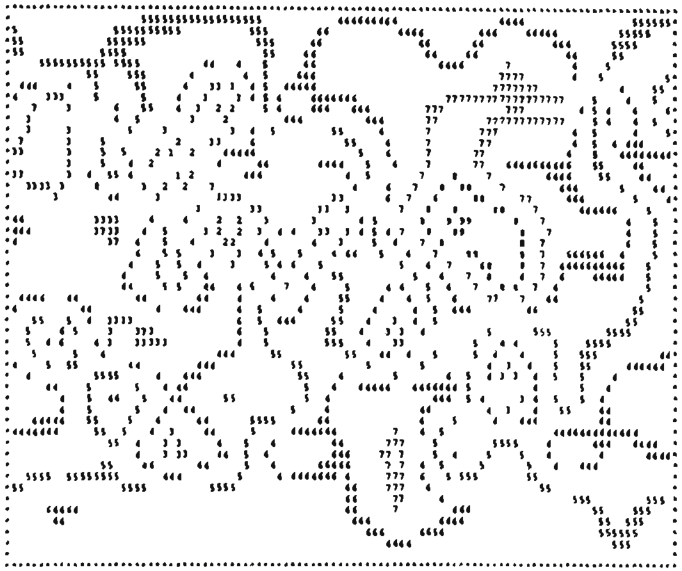
realtime= 1.485750
1 ***** CONTOUR MAP OF PSI-2 ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



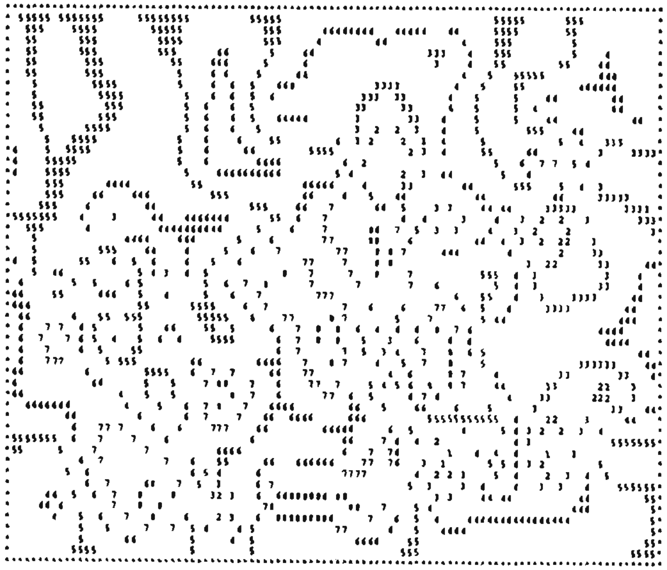
realtime= 2.484632
1 ***** CONTOUR MAP OF PSI-2 ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



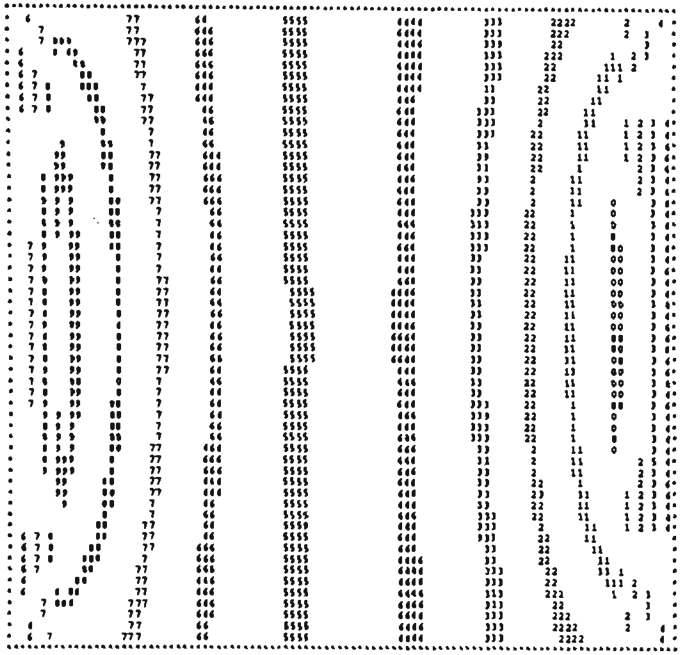
realtime= 0.9843086
1 ***** CONTOUR MAP OF PSI-2 ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



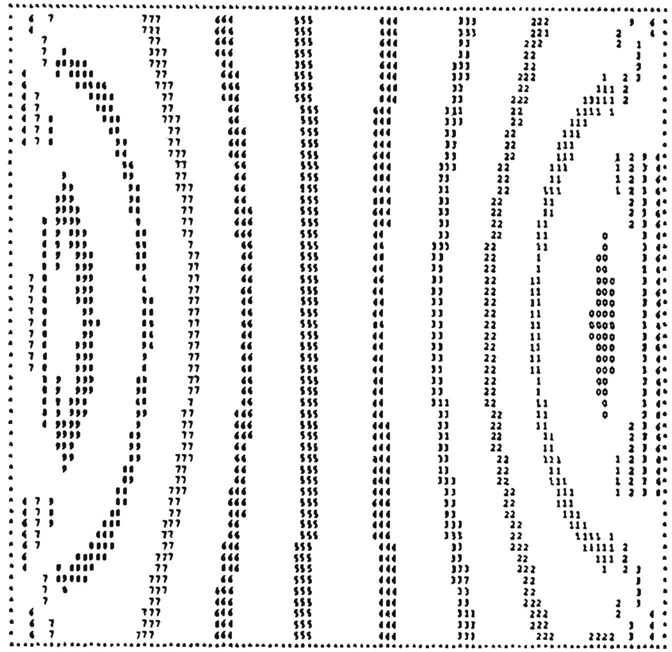
realtime= 3.887191
1 ***** CONTOUR MAP OF PSI-2 ***** , K = 4 (Z = 0.57)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



realtime= 0.4828689
1 ***** CONTOUR MAP OF PSI-3 ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-21.03 -16.36 -11.69 -7.01 -2.34 2.34 7.01 11.69 16.36 21.03

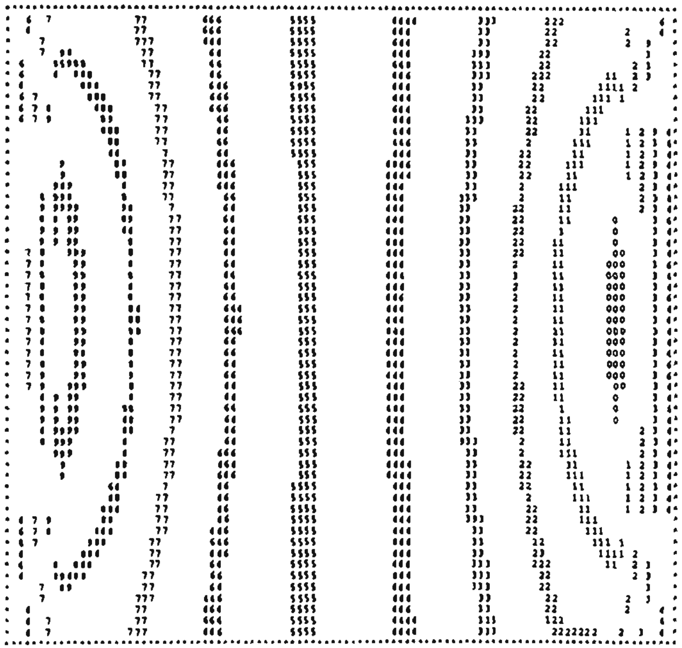


realtime= 1.495750
1 ***** CONTOUR MAP OF PSI-3 ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-19.86 -15.45 -11.03 -6.62 -2.21 2.21 6.62 11.03 15.45 19.86

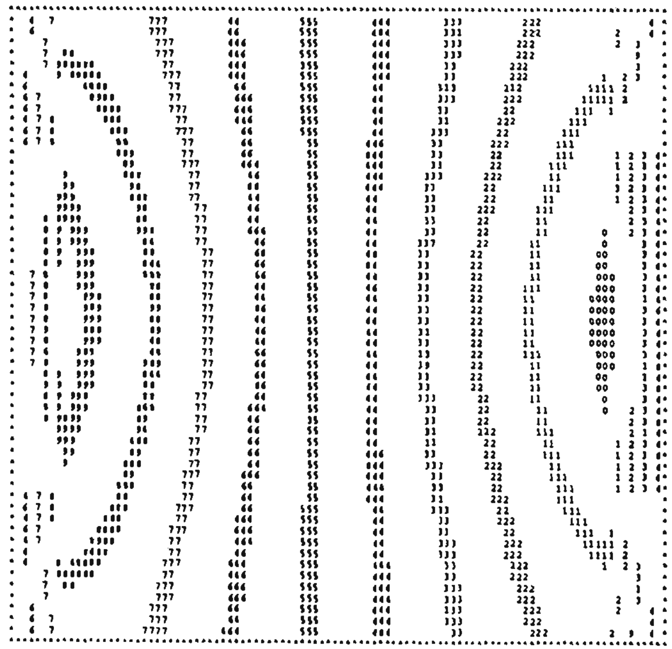


realtime= 2.488632
1 ***** CONTOUR MAP OF PSI-3 ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-15.36 -11.95 -8.53 -5.12 -1.71 1.71 5.12 8.53 11.95 15.36

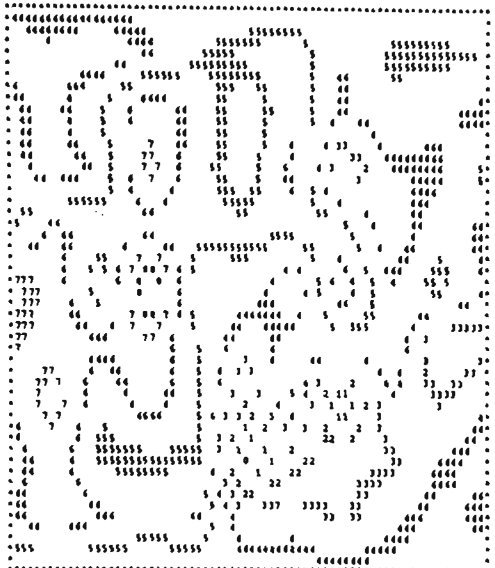
realtime= 0.9943096
1 ***** CONTOUR MAP OF PSI-3 ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-21.12 -16.42 -11.73 -7.04 -2.35 2.35 7.04 11.73 16.42 21.12



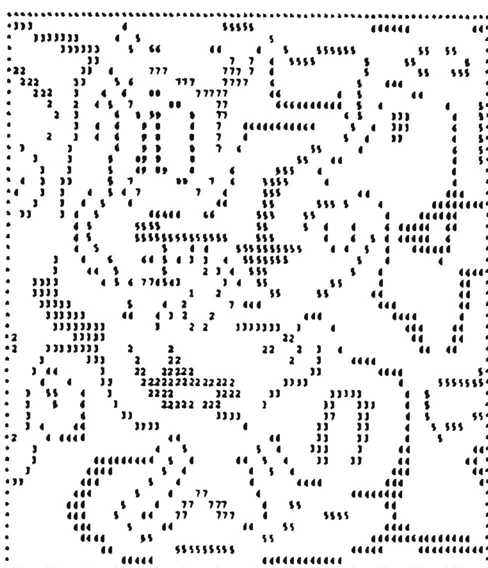
realtime= 1.987181
1 ***** CONTOUR MAP OF PSI-3 ***** ; I = 6 (X = 0.50)
THE CONTOUR LEVELS ARE ;
-17.66 -13.74 -9.81 -5.89 -1.96 1.96 5.89 9.81 13.74 17.66



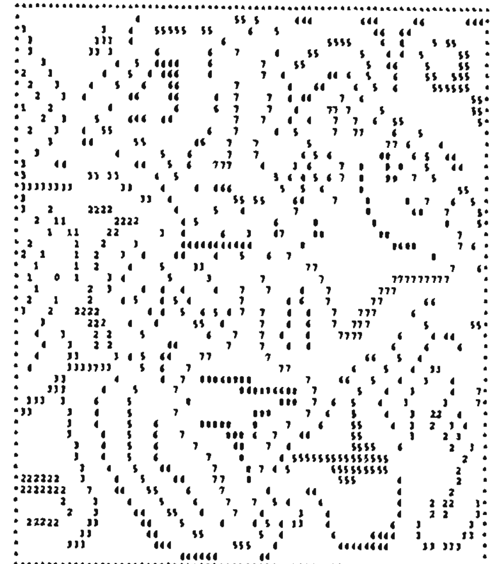
realtime= 0.4424469
1 ***** CONTOUR MAP OF PSI-3 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



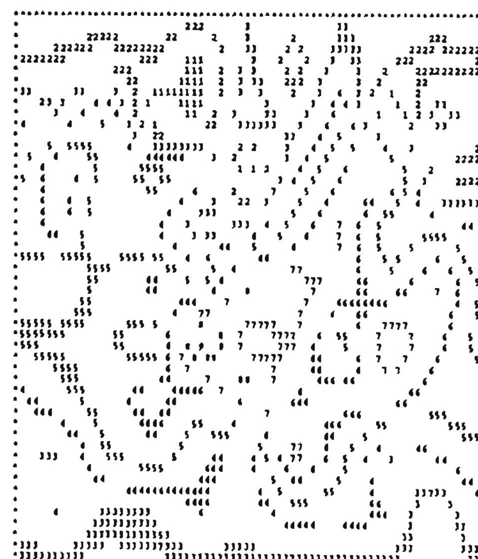
realtime= 0.5443096
1 ***** CONTOUR MAP OF PSI-3 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



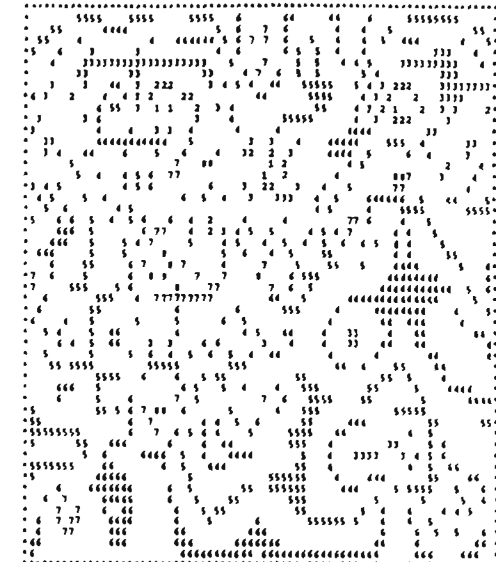
realtime= 1.405750
1 ***** CONTOUR MAP OF PSI-3 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



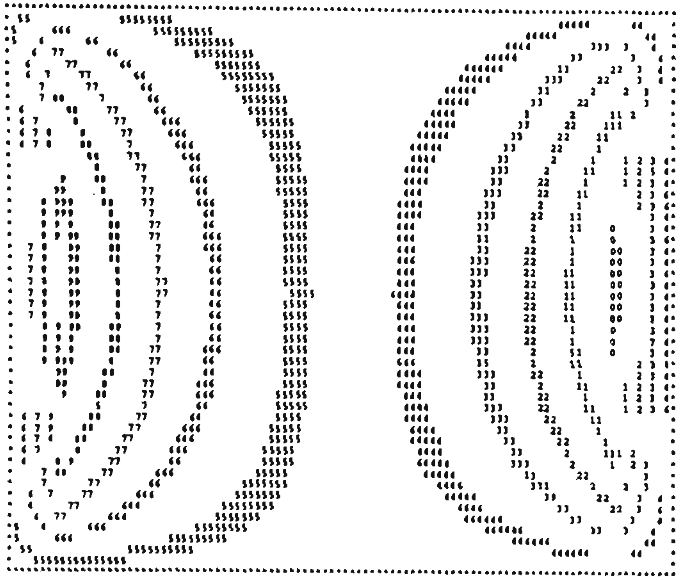
realtime= 1.807191
1 ***** CONTOUR MAP OF PSI-3 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



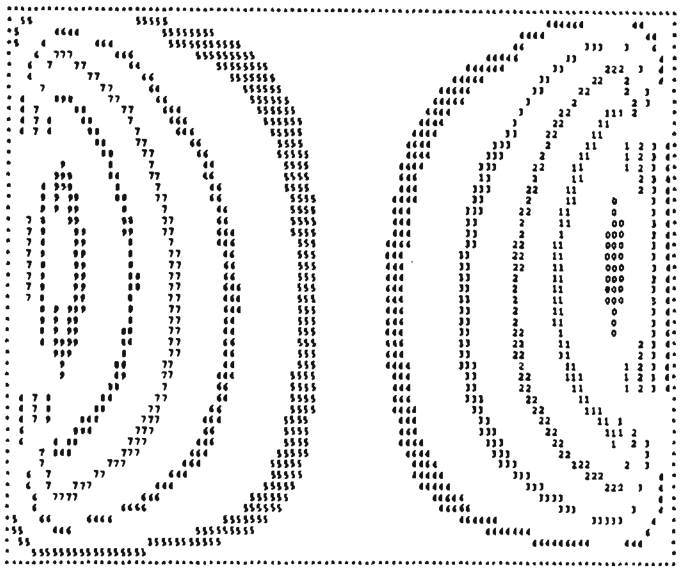
realtime= 2.494632
1 ***** CONTOUR MAP OF PSI-3 ***** , J = 7 (Y = 0.93)
THE CONTOUR LEVELS ARE :
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00



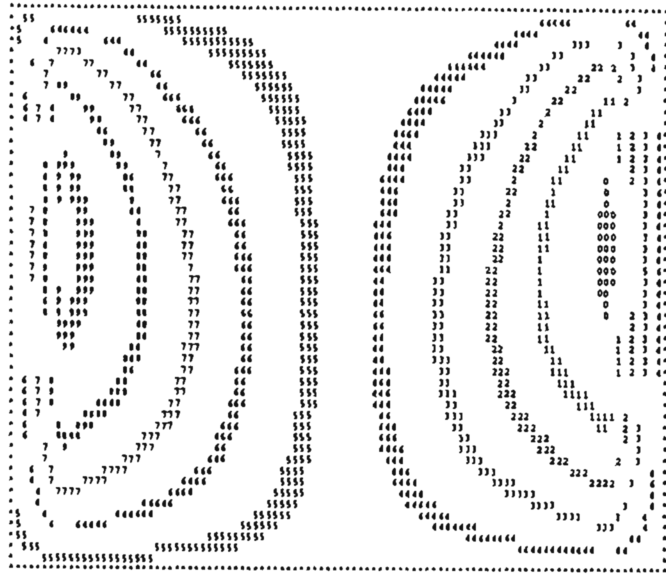
realtime= 0.6920690
1 ***** CONTOUR MAP OF PSI-3 ***** : K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
-21.03 -16.36 -11.68 -7.01 -2.34 2.34 7.01 11.68 16.36 21.03



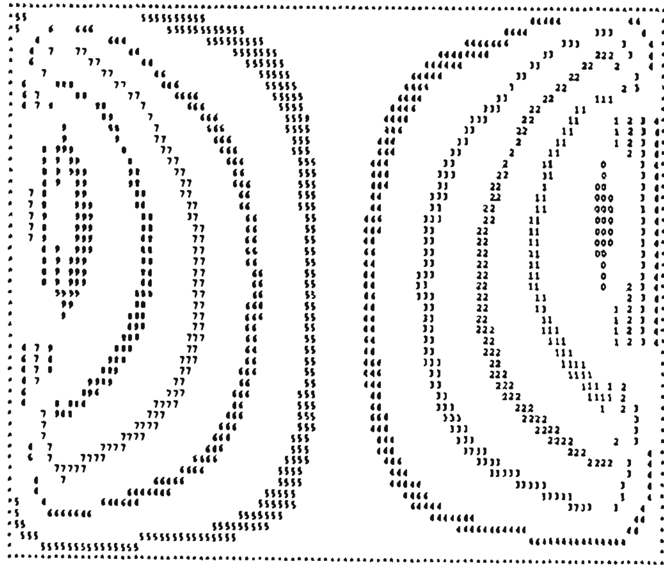
realtime= 0.9843096
1 ***** CONTOUR MAP OF PSI-3 ***** : K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
-21.12 -16.42 -11.73 -7.04 -2.35 2.35 7.04 11.73 16.42 21.12



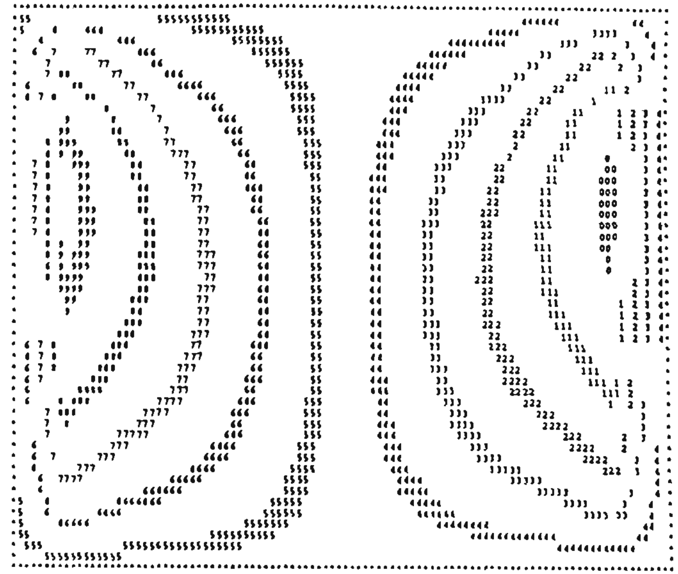
realtime= 1.495750
1 ***** CONTOUR MAP OF PSI-3 ***** : K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
-20.15 -15.67 -11.19 -6.72 -2.24 2.24 6.72 11.19 15.67 20.15



realtime= 1.987191
1 ***** CONTOUR MAP OF PSI-3 ***** : K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
-19.29 -14.23 -10.16 -6.10 -2.03 2.03 6.10 10.16 14.23 19.29



realtime= 2.490032
1 ***** CONTOUR MAP OF PSI-3 ***** : K = 6 (Z = 0.57)
THE CONTOUR LEVELS ARE :
-16.11 -12.53 -8.95 -5.37 -1.79 1.79 5.37 8.95 12.53 16.11



APPENDIX D Program AGRI_3D.FOR and its input files

Table of content

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   program agri_3d.for                                              c
c                                                                    c
c                                                                    c
c    ****************************************************************
c    *                                                              *
c    *                     Free  Convection                         *
c    *                     ----------------                         *
c    *                                                              *
c    *                            in                                *
c    *                            --                                *
c    *                                                              *
c    *           a Three-Dimensional Rectangular Box                *
c    *           -----------------------------------                *
c    *                                                              *
c    *           Containing Agricultural Produce with               *
c    *           ------------------------------------               *
c    *                Respiring Heat Generation                     *
c    *                -------------------------                     *
c    *                                                              *
c    *            a Vector Potential Formulation                    *
c    *                                                              *
c    *                         Using                                *
c    *                                                              *
c    *   Forward Time and Central (Non-Uniform) Spatial Differences *
c    *                                                              *
c    *                                                              *
c    *        a Transient Result and a Steady State                 *
c    *                                                              *
c    *              Result Will Be Obtained                         *
c    *                                                              *
c    *><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
c    *                                                              *
c    *                                                              *
c    *  Author:  Patrick X.P. Li                                    *
c    *  -------                                                     *
c    *                                                              *
c    *  Supervisors:  G. R. Thorpe                                  *
c    *  -----------   G. T. Lleonart                                *
c    *                                                              *
c    *  Address:  Victoria University of Technology                 *
c    *  -------   Department of Mechanical Engineering,             *
c    *            Footscray , Victoria, Australia.                  *
c    *                                                              *
c    *  Acknowledgements: The author is grateful to Associate       *
c    *                     Professor E. Leonardi, of University of  *
c    *                     New South Wales, for providing the author*
c    *                     with access to his model of natural      *
c    *                     convective processes in three-dimentional*
c    *                     enclosures.                              *
c    *                                                              *
c    *><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
c    *                                                              *
c     ****************************************************************
c
c     ****************************************************************
c     *------------------------------------------------------------*
c    *|                                                            |*
c    *|  Reference                                                 |*
c    *|                                                            |*
c    *|  L.P.  Goh,  E.  Leonardi and G. de Vahl Davis,"Frecon3d - |*
c    *|  Users Manual. A Program for  the  Numerical  Solution  of |*
c    *|  Mixed   Convection  in  a  Three-Dimensional  Rectangular |*
c    *|  Cavity", The University of New  South  Wales,  School  of |*
c    *|  Mech. and Ind. Engineering, Report 1988/fmt/7, (1988).    |*
c    *|                                                            |*
c    *|  and/or                                                    |*
c    *|                                                            |*
c    *|  L.P.  Goh,  E. Leonardi and G. de Vahl Davis, "Frecon3d - |*
c    *|  Programmers Manual. A Program for the Numerical  Solution |*
c    *|  of  Mixed  Convection  in a Three-Dimensional Rectangular |*
c    *|  Cavity", The University of New  South  Wales,  School  of |*
c    *|  Mech. and Ind. Engineering, Report 1988/fmt/8, (1988).    |*
c    *|                                                            |*
c     *------------------------------------------------------------*
c     ****************************************************************
c
c
c     --------------------------------------------------------------
c    |                                                              |
c    |                    ***  main program ***                     |
c    |            *** job initiation and sequence control ***       |
c    |                                                              |
c     --------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)

      character exprmnt*8
      character timestrg*11,datestrg*9

      logical blup,cgd
      logical divv,divp,dmax
      logical reed,rite,strt,dist,heat,nuss,tran
      logical rt,rp1,rp2,rp3,rmax
      logical start,runng,runend
      logical finish

      common /timestrg/timestrg,datestrg
      common /exprmnt/exprmnt
      common /logc/reed,rite,strt,dist,heat,nuss,tran
      common /logd/divv,divp,dmax
      common /logi/blup,cgd
      common /logr/rt,rp1,rp2,rp3,rmax
      common /fileno/inparam,irg,ireport,itin,itout
      common /vel/u(nxmax,nymax,nzmax),
     +            v(nxmax,nymax,nzmax),
     +            w(nxmax,nymax,nzmax)
      common /p1/p1(nxmax,nymax,nzmax)
```

```
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /iplt/iplt,inu
      common  /finish/finish
c
c     ...record starting time ...
c
      call  time(timestrg)
      call  date(datestrg)
c
c     ... read in data ...
c
      call  datin
c
c     ... open output files ...
c
      call  outfiles
c
c     ... write starting time onto file...
c
      finish=.false.
      start=.true.
      runng=.false.
      runend=.false.
      call  ttime(start,runng,runend)
      start=.false.
c
c     ... read previous results ( if any) ...
c
      if(reed) call reader
c
c     ... set up constants ...
c
      call  setcon
c
c     ... print banner page ...
c
      call  banner
c
c     ... initialize arrays and/or boundary conditions ...
c
      call  inital
c
c     ... disturb stream function field to check stability ...
c
      if(dist) call distb
c
c     ... begin solution procedure ...
c
      call  iterat
c
c     ...showing the locations of thermal death points...
c
      call  location
c
c     ... write results to file ...
c
      if (rite .and. .not. blup) then
      call  writer
      endif
c
c     ... print all required results and maps ...
c
      finish=.true.
      if(.not. blup) call output
c
c     .. calculate residuals
c
      if  (rt   .or. rmax) call   resit
      if  (rp1  .or. rmax) call   resip1
      if  (rp2  .or. rmax) call   resip2
      if  (rp3  .or. rmax) call   resip3
c
c     .. calculate divergence of vector fields
c
      if(divv .or. dmax) call div(u ,v ,w ,'div of velocity',divv)
      if(divp .or. dmax) call div(p1,p2,p3,'  div of psi    ',divp)
c
      if(nuss) call htc
      call  sumary
c
      write(ireport,2010)
 2010 format(/)
c
c     ...write ending time onto file...
c
      call  time(timestrg)
      call  date(datestrg)
      runend=.true.
      call  ttime(start,runng,runend)
      write(ireport,2020)
      write(*,2020)
 2020 format(/,5x,'*** Program agri_3d.for finished ***')
      stop
      end


      subroutine banner
c
c     --------------------------------------------------------------------
c    |                                                                    |
c    | ***  writes a heading for the output in the report files   ***    |
c    |                                                                    |
c     --------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      real keff
      integer pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
```

```
     +         ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +         ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +         ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +         ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      logical  lfile
      logical  blup,cgd
      logical  reed,rite,strt,dist,heat,nuss,tran
      logical  rt,rp1,rp2,rp3,rmax
      logical  divv,divp,dmax
      logical  vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +         mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +         vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +         myp1,myp2,myp3,myt,myu,myv,myw,
     +         vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +         mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      logical  ixl,ixh,iyl,iyh,izl,izh
c
      character  table(16,132)*1,mxlci*1,mxhci*1,mylci*1,myhci*1,
     +           mzlci*1,mzhci*1,mesh*11
c
      character  title*80,otitle*80
      character  xc*1,yc*1,zc*1,vpform*1
      character  meshtp*1,exprmnt*8
      character  srcetype*1
c
      common  /title/title,otitle
     +        /tapcom/rfile(94),ifile(67),lfile(10)
      common  /exprmnt/exprmnt
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /logi/blup,cgd
      common  /logc/reed,rite,strt,dist,heat,nuss,tran
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /logd/divv,divp,dmax
      common  /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +                mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +                vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +                myp1,myp2,myp3,myt,myu,myv,myw,
     +                vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +                mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common  /logtbc/ixl,ixh,iyl,iyh,izl,izh
      common  /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,
     +        ccp,cct,tmax,pdist,rtime
      common  /alpfs/alpp,altp
      common  /intcom/itercmpo,norun,mainiter
      common  /numcom/numit
      common  /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +            ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +            zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
      common  /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +            iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +            izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
      common  /velbc/xlu,xlv,xlw,xhu,xhv,xhw,
     +              ylu,ylv,ylw,yhu,yhv,yhw,
     +              zlu,zlv,zlw,zhu,zhv,zhw
      common  /dt/dt
```

```
      common  /mesh/meshtp
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +           hxold(max),hyold(max),hzold(max)
      common  /axis/x(max),y(max),z(max)
      common  /vmpfrm/xc,yc,zc,vpform
      common  /vmpnum/npx,npy,npz
      common  /vmploc/mx(nxmax),my(nymax),mz(nzmax)
      common  /vmppos/rmx(nxmax),rmy(nymax),rmz(nzmax)
      common  /mapcom/mnx,mny,mnz
      common  /phierr/phierr
      common  /da/da
      common  /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common  /ccaa/cc0,dtemp,tcold,thot
      common  /epsonnu/epson,xll,annu,gravity,ddd,keff
      common  /param_q/qparam(20)
      common  /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common  /tstrt/tstrt
      common  /pstxyz/pstx1,psty1,pstz1
     +         ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +         ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +         ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +         ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +         ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      common  /iplt/iplt,inu
      common  /txty/tpnt(10),q(10),npt,nparam,srcetype
c
      write(ireport,2020) title,norun
 2020 format(//,a80,//,'Run number # = ',i4)
      write(ireport,2021) exprmnt,exprmnt
 2021 format(/,'Name   of   this   experiment   is ',a8,/,
     +        'File names for output results are ',a8,'.*.')
      write(ireport,2147) xll*arx, xll*ary, xll*arz
 2147 format(/,'Dimension of box(x*y*z)= ',f5.2,'m X ',
     +    f5.2,'m X ',f5.2,'m')
      write(ireport,2148) ddd
 2148 format(/,'(Effective) diameter of fruit or vegetabale= ',
     +   f6.2, 'm')
      write(ireport,2149) tcold
 2149 format(/,'Temperature of cold wall                = ',f6.1)
      write(ireport,2151) thot
 2151 format(  'Temperature of hot  wall                = ',f6.1)
      write(ireport,2154) tstrt*(thot-tcold)+tcold
 2154 format(  'Initial temperature of porous media = ',f6.1)
c
      write(ireport,2152)
 2152 format( /,'Respiration heating parameters:')
      write(ireport,2166)(tpnt(i),i=1,npt)
 2166 format ('Temperature at: ',10(f5.1,1x))
      write(ireport,2167)(q(i),i=1,npt)
 2167 format ('Heating rate   : ',10(f5.1,1x))
      if ((srcetype.eq.'c').or.(srcetype.eq.'C')) then
      write(ireport,2168)
      else
      write(ireport,2169)
```

```
      endif
 2168  format('Unit for heating rate is   mg_Co2/kg h.')
 2169  format('Unit for heating rate is   w/mmm.')
c     ... write fluid properties ...
      write(ireport,2146) epson
 2146 format(//,'Porosity                              = ',1pg10.4)
c
      write(ireport,2150) alphaf,alphap
 2150 format(   'Diffusivity of fluid                = ',1pg10.4,
     +        /,'Diffusivity of porous medium        = ',1pg10.4)
      write(ireport,2155) cpfl,cppr
 2155 format(   'Specific heat of fluid              = ',1pg10.4,
     +       / ,'Specific heat of porous medium      = ',1pg10.4)
      write(ireport,2156) cpsd,rhos
 2156 format(   'Specific heat of fruit(veget.)      = ',1pg10.4,
     +        /,'Density of fruit or veget.          = ',1pg10.4)
      write(ireport,2160) rhof,rhop
 2160 format(   'Density of fluid                    = ',1pg10.4,
     +        /,'Density of porous medium            = ',1pg10.4)
      write(ireport,2161) keff
 2161 format(   'Effective thermal conductivity      = ',1pg10.4)
c
c     ... non-dimensional parameters ...
c
      write(ireport,2120)  arx, betax
 2120 format(/,'Aspect ratio in x = ',0pf10.3,
     +         5x,'Beta_x (DEG) = ',0pf10.3)
c
      write(ireport,2130)  ary, betay
 2130 format( 'Aspect ratio in y = ',0pf10.3,
     +         5x,'Beta_y (DEG) = ',0pf10.3)
c
      write(ireport,2140)  arz, betaz
 2140 format( 'Aspect ratio in z = ',0pf10.3,
     +         5x,'Beta_z (deg) = ',0pf10.3)
c
      write(ireport,2144) darcyx
      write(ireport,2145) darcyy
      write(ireport,2143) darcyz
 2144 format( /,'Darcy number of x direction    = ',1pg10.4)
 2145 format( 'Darcy number of y direction    = ',1pg10.4)
 2143 format( 'Darcy number of z direction    = ',1pg10.4)
      write(ireport,2131) ra
 2131 format( /,'Rayleigh number = ',1pg10.4)
c
c     ...  mesh details and transient factors ...
c
      if ((meshtp.eq.'n').or.(meshtp.eq.'N')) then
       mesh = 'Non-uniform'
      else if ((meshtp.eq.'b').or.(meshtp.eq.'B')) then
       mesh = 'From binary'
      else if ((meshtp.eq.'t').or.(meshtp.eq.'T')) then
       mesh = 'Transformed'
      else
       mesh = '  Uniform  '
      endif
c
      write(ireport,2210) nx,ny,nz,mesh
 2210 format(//,'Mesh =',i3,' *',i3,' *',i3,' - ','(',a11,')')

      write(ireport,2220) alpp,altp,stabr
 2220 format(/,'False transient factors - alpha psip  = ',f8.4,
     +        /,'Alpha theta                          = ',f8.4,
     +        /,'Stability ratio                      = ',f8.4,/)

      write(ireport,2221) dt*altp
 2221  format('Main  time step for t =',1pg10.4)
      write(ireport,2222) dt*alpp
 2222  format('Inner time step for p =',1pg10.4)

c
c     ... convergence limits and pdist ...
c
      write(ireport,2165) phierr
 2165 format('PHIERR = ',1pg10.4)

      write(ireport,2410) ccp,itercmpo,cct,pdist
 2410 format(/,'Convergence limits on - psi    ccp = ',1pg10.4,
     +         4x,'Max. no. of iterations = ',i6,
     +  /24x,'Theta cct = ',1pg10.4,4x,'disturb. to p (pdist)  = '
     +        ,f6.2,/)
c
      write(ireport,2412)
 2412 format(/,'Controls: reed rite strt dist heat nuss tran',
     +          '  Residuals: rt rp1 rp2 rp3 rmax  ')
      write(ireport,2413) reed,rite,strt,dist,heat,nuss,tran,
     +                    rt,rp1,rp2,rp3,rmax
 2413 format(9x,7(4x,l1),11x,5(3x,l1))
      write(ireport,2415)
 2415 format('Divergence: divv divp dmax')
      write(ireport,2414) divv,divp,dmax
 2414 format(11x,3(4x,l1))
c
c     ... write printing controls ...
c
      write(ireport,2320)
 2320 format(//,5x,'Printing controls:  x = constant planes    { note: ',
     +            'vp1 => print p1 values, mx1 => print p1 map }')
c
      write(ireport,2321)
 2321 format(/,5x,' vp1 vp2 vp3 vt  vu  vv  vw  ',
     +        'mp1 mp2 mp3 mt  mu  mv  mw  ')
c
      write(ireport,2322) vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +                    mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw
 2322 format(4x,14(3x,l1))
c
      write(ireport,2324)
 2324 format(5x,' Values and maps are printed for the following',
     +          ' x planes:')
```

```
      write(ireport,2325) ( mx(nplane), nplane=1,npx )
 2325 format( (3x,21(2x,i3)) )
c
      write(ireport,2340)
 2340 format(//,5x,'Prting controls:  y = constant planes')
c
      write(ireport,2341)
 2341 format(/,5x,' vp1 vp2 vp3 vt  vu  vv  vw  ',
     +      'mp1 mp2 mp3 mt  mu  mv  mw  ')
      write(ireport,2342) vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +                    myp1,myp2,myp3,myt,myu,myv,myw
 2342 format(4x,14(3x,l1))
c
      write(ireport,2344)
 2344 format(5x,' Values and maps are printed for the following',
     +         ' y planes:')
      write(ireport,2345) ( my(nplane), nplane=1,npy)
 2345 format( (3x,21(2x,i3)) )
c
      write(ireport,2330)
 2330 format(//,5x,'Printing controls:  z = constant planes')
c
      write(ireport,2331)
 2331 format(/,5x,' vp1 vp2 vp3 vt  vu  vv  vw  ',
     +      'mp1 mp2 mp3 mt  mu  mv  mw  ')
c
      write(ireport,2332) vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +                    mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
 2332 format(4x,14(3x,l1))
c
      write(ireport,2334)
 2334 format(5x,' Values and maps are printed for the following',
     +         ' z planes:')
      write(ireport,2335) ( mz(nplane), nplane=1,npz)
 2335 format( (3x,21(2x,i3)) )
c
      do 5 i=1, 16
       do 5 j=1, 132
          table(i,j) = ' '
    5 continue
c
      write(ireport,2420)
 2420 format(1h1,5x,'Thermal boundary conditions (note: contour ',
     +        'levels = isothermal eg "0" implies theta=0.0; ',
     +        '"1" implies 0.0<theta<=0.11; etc')
c
      write(ireport,2430)
 2430 format(40x,' Blank surface = adiabatic, heat flux or convective ',
     +        'depending on the constants')
c
      write(ireport,2440)
 2440 format(40x,' cg, ca & cb in the equation :cg*dt/dn = ca*t + cb',/)
c
      write(ireport,2460)
 2460 format(16x,'z--->',16x,'z--->',19x,'z--->',16x,'z--->',19x,'y--->'
     +       ,16x,'y--->')
c
c     ... set boundaries of planes ...
c
c     ... vertical boundary ...
c
      do 10 i=2, 15
        table(i,3)   = '|'
        table(i,21)  = '|'
        table(i,24)  = '|'
        table(i,42)  = '|'
        table(i,48)  = '|'
        table(i,66)  = '|'
        table(i,69)  = '|'
        table(i,87)  = '|'
        table(i,93)  = '|'
        table(i,111) = '|'
        table(i,114) = '|'
        table(i,132) = '|'
   10 continue
c
c     ... horizontal boundary ...
c
      do 20 j=3,21
        table(1,j)  = '-'
        table(16,j) = '-'
   20 continue
c
      do 30 j=24, 42
        table(1,j)  = '-'
        table(16,j) = '-'
   30 continue
c
      do 40 j=48, 66
        table(1,j)  = '-'
        table(16,j) = '-'
   40 continue
c
      do 50 j=69, 87
        table(1,j)  = '-'
        table(16,j) = '-'
   50 continue
c
      do 60 j=93,111
        table(1,j)  = '-'
        table(16,j) = '-'
   60 continue
c
      do 70 j=114, 132
        table(1,j)  = '-'
        table(16,j) = '-'
   70 continue
c
c     ... set axes ...
c
```

```
      table(14,2)  = '|'
      table(13,2)  = 'y'
      table(15,2)  = '|'
      table(16,2)  = 'v'
c
      table(14,23)  = '|'
      table(13,23)  = 'y'
      table(15,23)  = '|'
      table(16,23)  = 'v'
c
      table(14,47)  = '|'
      table(13,47)  = 'x'
      table(15,47)  = '|'
      table(16,47)  = 'v'
c
      table(14,68)  = '|'
      table(13,68)  = 'x'
      table(15,68)  = '|'
      table(16,68)  = 'v'
c
      table(14,92)  = '|'
      table(13,92)  = 'x'
      table(15,92)  = '|'
      table(16,92)  = 'v'
c
      table(14,113)  = '|'
      table(13,113)  = 'x'
      table(15,113)  = '|'
      table(16,113)  = 'v'
c
c     ... set isothermal contour levels ...
c
      if (xlci.eq.0.0) then
      mxlci = '0'
      else
     +  if (xlci.le.0.11) then
         mxlci = '1'
        else
     +      if (xlci.le.0.22) then
             mxlci = '2'
            else
     +          if (xlci.le.0.33) then
                mxlci = '3'
                 else
     +              if (xlci.le.0.44) then
                    mxlci = '4'
                   else
     +                  if (xlci.le.0.55) then
                         mxlci = '5'
                       else
     +                      if (xlci.le.0.66) then
                           mxlci = '6'
                           else
     +                          if (xlci.le.0.77) then
                              mxlci = '7'
                              else
     +                              if (xlci.le.0.88) then
                                  mxlci = '8'
                              else
                                mxlci = '9'
      endif
c
      if (xhci.eq.0.0) then
      mxhci = '0'
      else
     +  if (xhci.le.0.11) then
         mxhci = '1'
        else
     +      if (xhci.le.0.22) then
             mxhci = '2'
            else
     +          if (xhci.le.0.33) then
                mxhci = '3'
                 else
     +              if (xhci.le.0.44) then
                    mxhci = '4'
                   else
     +                  if (xhci.le.0.55) then
                         mxhci = '5'
                       else
     +                      if (xhci.le.0.66) then
                           mxhci = '6'
                           else
     +                          if (xhci.le.0.77) then
                              mxhci = '7'
                           else
     +                              if (xhci.le.0.88) then
                                  mxhci = '8'
                           else
                               mxhci = '9'
      endif
c
      if (ylci.eq.0.0) then
      mylci = '0'
      else
     +  if (ylci.le.0.11) then
         mylci = '1'
        else
     +      if (ylci.le.0.22) then
             mylci = '2'
            else
     +          if (ylci.le.0.33) then
                mylci = '3'
                 else
     +              if (ylci.le.0.44) then
                    mylci = '4'
                   else
     +                  if (ylci.le.0.55) then
                         mylci = '5'
                       else
```

```
     +                    if (ylci.le.0.66) then
                        mylci = '6'
                        else
     +                     if (ylci.le.0.77) then
                          mylci = '7'
                        else
     +                       if (ylci.le.0.88) then
                            mylci = '8'
                        else
                          mylci = '9'
        endif
c
        if (yhci.eq.0.0) then
        myhci = '0'
        else
     +   if (yhci.le.0.11) then
          myhci = '1'
         else
     +      if (yhci.le.0.22) then
             myhci = '2'
            else
     +         if (yhci.le.0.33) then
               myhci = '3'
                else
     +            if (yhci.le.0.44) then
                  myhci = '4'
                 else
     +               if (yhci.le.0.55) then
                      myhci = '5'
                    else
     +                  if (yhci.le.0.66) then
                       myhci = '6'
                       else
     +                    if (yhci.le.0.77) then
                         myhci = '7'
                       else
     +                      if (yhci.le.0.88) then
                           myhci = '8'
                       else
                         myhci = '9'
        endif
c
        if (zlci.eq.0.0) then
        mzlci = '0'
        else
     +   if (zlci.le.0.11) then
          mzlci = '1'
         else
     +      if (zlci.le.0.22) then
             mzlci = '2'
            else
     +         if (zlci.le.0.33) then
               mzlci = '3'
                else
     +            if (zlci.le.0.44) then
                  mzlci = '4'
                 else
     +               if (zlci.le.0.55) then
                      mzlci = '5'
                    else
     +                  if (zlci.le.0.66) then
                       mzlci = '6'
                       else
     +                    if (zlci.le.0.77) then
                         mzlci = '7'
                       else
     +                      if (zlci.le.0.88) then
                           mzlci = '8'
                       else
                         mzlci = '9'
        endif
c
        if (zhci.eq.0.0) then
        mzhci = '0'
        else
     +   if (zhci.le.0.11) then
          mzhci = '1'
         else
     +      if (zhci.le.0.22) then
             mzhci = '2'
            else
     +         if (zhci.le.0.33) then
               mzhci = '3'
                else
     +            if (zhci.le.0.44) then
                  mzhci = '4'
                 else
     +               if (zhci.le.0.55) then
                      mzhci = '5'
                    else
     +                  if (zhci.le.0.66) then
                       mzhci = '6'
                       else
     +                    if (zhci.le.0.77) then
                         mzhci = '7'
                       else
     +                      if (zhci.le.0.88) then
                           mzhci = '8'
                       else
                         mzhci = '9'
        endif
c
        mixlyl  =  int((real(ixlyl)/real(ny))*14.0)
        mixlyh  =  int((real(ixlyh)/real(ny))*14.0)
        mixlzl  =  int((real(ixlzl)/real(nz))*17.0)
        mixlzh  =  int((real(ixlzh)/real(nz))*17.0)
c
        if (mixlyl.eq.0) mixlyl = 1
        if (mixlzl.eq.0) mixlzl = 1
c
```

```
      if (ixl) then
       do 80 j=mixlyl+1, mixlyh+1
          do 80 k=mixlzl+3, mixlzh+3
               table(j,k) = mxlci
 80     continue
      endif
c
      mixhyl  =  int((real(ixhyl)/real(ny))*14.0)
      mixhyh  =  int((real(ixhyh)/real(ny))*14.0)
      mixhzl  =  int((real(ixhzl)/real(nz))*17.0)
      mixhzh  =  int((real(ixhzh)/real(nz))*17.0)
c
      if (mixhyl.eq.0) mixhyl = 1
      if (mixhzl.eq.0) mixhzl = 1
c
      if (ixh) then
       do 90 j=mixhyl+1, mixhyh+1
          do 90 k=mixhzl+24, mixhzh+24
               table(j,k) = mxhci
 90     continue
      endif
c
      miylxl  =  int((real(iylxl)/real(nx))*14.0)
      miylxh  =  int((real(iylxh)/real(nx))*14.0)
      miylzl  =  int((real(iylzl)/real(nz))*17.0)
      miylzh  =  int((real(iylzh)/real(nz))*17.0)
c
      if (miylxl.eq.0) miylxl = 1
      if (miylzl.eq.0) miylzl = 1
c
      if (iyl) then
       do 100 i=miylxl+1, miylxh+1
          do 100 k=miylzl+48, miylzh+48
               table(i,k) = mylci
 100    continue
      endif
c
      miyhxl  =  int((real(iyhxl)/real(nx))*14.0)
      miyhxh  =  int((real(iyhxh)/real(nx))*14.0)
      miyhzl  =  int((real(iyhzl)/real(nz))*17.0)
      miyhzh  =  int((real(iyhzh)/real(nz))*17.0)
c
      if (miyhxl.eq.0) miyhxl = 1
      if (miyhzl.eq.0) miyhzl = 1
c
      if (iyh) then
       do 110 i=miyhxl+1, miyhxh+1
          do 110 k=miyhzl+69, miyhzh+69
               table(i,k) = myhci
 110    continue
      endif
c
      mizlxl  =  int((real(izlxl)/real(nx))*14.0)
      mizlxh  =  int((real(izlxh)/real(nx))*14.0)
      mizlyl  =  int((real(izlyl)/real(ny))*17.0)
      mizlyh  =  int((real(izlyh)/real(ny))*17.0)
c
      if (mizlxl.eq.0) mizlxl = 1
      if (mizlyl.eq.0) mizlyl = 1
c
      if (izl) then
       do 120 i=mizlxl+1, mizlxh+1
          do 120 j=mizlyl+93, mizlyh+93
               table(i,j) = mzlci
 120    continue
      endif
c
      mizhxl  =  int((real(izhxl)/real(nx))*14.0)
      mizhxh  =  int((real(izhxh)/real(nx))*14.0)
      mizhyl  =  int((real(izhyl)/real(ny))*17.0)
      mizhyh  =  int((real(izhyh)/real(ny))*17.0)
c
      if (mizhxl.eq.0) mizhxl = 1
      if (mizhyl.eq.0) mizhyl = 1
c
      if (izh) then
       do 130 i=mizhxl+1, mizhxh+1
          do 130 j=mizhyl+114, mizhyh+114
               table(i,j) = mzhci
 130    continue
      endif
c
      do 140 i=1, 16
       write(ireport,2470) (table(i,j), j=1, 132)
 2470 format(132a1)
 140  continue
c
      write(ireport,2480) arx, ary, arz
 2480 format(/2x,'y-z plane; x=0.0      y-z plane; x=',f5.2,
     +         6x,'x-z plane; y=0.0      x-z plane; y=',f5.2,
     +         6x,'x-y plane; z=0.0      x-y plane; z=',f5.2,/)
c
      write(ireport,2485) xlcg, xhcg,
     +                    ylcg, yhcg,
     +                    zlcg, zhcg
 2485 format(2x,'cg=',f5.2,1x,'    ',5x,
     +       4x,'cg=',f5.2,1x,'    ',5x,
     +       7x,'cg=',f5.2,1x,'    ',5x,
     +       4x,'cg=',f5.2,1x,'    ',5x,
     +       7x,'cg=',f5.2,1x,'    ',5x,
     +       4x,'cg=',f5.2,1x,'    ' )
c
      write(ireport,2490) xlca, xlcb, xhca, xhcb,
     +                    ylca, ylcb, yhca, yhcb,
     +                    zlca, zlcb, zhca, zhcb
 2490 format(2x,'ca=',f5.2,1x,'cb=',f5.2,
     +       4x,'ca=',f5.2,1x,'cb=',f5.2,
     +       7x,'ca=',f5.2,1x,'cb=',f5.2,
     +       4x,'ca=',f5.2,1x,'cb=',f5.2,
     +       7x,'ca=',f5.2,1x,'cb=',f5.2,
```

```
     +        4x,'ca=',f5.2,1x,'cb=',f5.2)
c
      write(ireport,2500)
 2500 format(//,5x,'Velocity boundary conditions',/)
      write(ireport,2501)
 2501 format(16x,'z--->',16x,'z--->',19x,'z--->',16x,'z--->',19x,'y--->'
     +        ,16x,'y--->')
c
c     ... blanck out inside boxes ...
c
        do 159 ivert=2,15
c
c         .. x = constant planes
c
            do 151 ihoriz=4, 20
                table(ivert,ihoriz) = ' '
  151        continue
            do 152 ihoriz=25, 41
                table(ivert,ihoriz) = ' '
  152        continue
c
c         .. y = constant planes
c
            do 153 ihoriz=49,65
                table(ivert,ihoriz) = ' '
  153        continue
            do 154 ihoriz=70, 86
                table(ivert,ihoriz) = ' '
  154        continue
c
c         .. z = constant planes
c
            do 155 ihoriz=94, 110
                table(ivert,ihoriz) = ' '
  155        continue
            do 156 ihoriz=115, 131
                table(ivert,ihoriz) = ' '
  156        continue
  159     continue
c
      do 200 i=1, 6
       write(ireport,2470) (table(i,j), j=1, 132)
  200 continue
c
      write(ireport,2510) ylu,yhu,zlu,zhu
 2510 format(2x,'|',4x,'u = na   ',5x,'|',2x,'|',5x,'u = na   ',4x,'|',
     +        5x,'|',5x,'u =',f5.2,4x,'|',2x,'|',5x,'u =',f5.2,4x,'|',
     +        5x,'|',5x,'u =',f5.2,4x,'|',2x,'|',5x,'u =',f5.2,4x,'|')
c
      write(ireport,2520) xlv,xhv,zlv,zhv
 2520 format(2x,'|',4x,'v =',f5.2,5x,'|',2x,'|',5x,'v =',f5.2,4x,'|',
     +        5x,'|',5x,'v = na   ',4x,'|',2x,'|',5x,'v = na   ',4x,'|',
     +        5x,'|',5x,'v =',f5.2,4x,'|',2x,'|',5x,'v =',f5.2,4x,'|')
c
      write(ireport,2530) xlw,xhw,ylw,yhw
 2530 format(2x,'|',4x,'w =',f5.2,5x,'|',2x,'|',5x,'w =',f5.2,4x,'|',
     +        5x,'|',5x,'w =',f5.2,4x,'|',2x,'|',5x,'w =',f5.2,4x,'|',
     +        5x,'|',5x,'w = na   ',4x,'|',2x,'|',5x,'w = na   ',4x,'|')
c
      do 220 i=11, 16
       write(ireport,2470) (table(i,j), j=1, 132)
  220 continue
c
      write(ireport,2580) arx, ary, arz
 2580 format(/2x,'y-z plane; x=0.0     y-z plane; x=',f5.2,
     +        6x,'x-z plane; y=0.0     x-z plane; y=',f5.2,
     +        6x,'x-y plane; z=0.0     x-y plane; z=',f5.2,////)
c
c     ... write mesh distribution ...
c
      if( (meshtp.eq.'n') .or. (meshtp.eq.'t') .or.
     +    (meshtp.eq.'N') .or. (meshtp.eq.'T') .or. tran )then
c
       write(ireport,2600)
 2600   format(1h1,25x,'** Mesh distribution for the three axes **',/)
c
c       .. x-axis
c
       write(ireport,2610)
 2610   format(//,5x,' x-axis: ')
c
       iblk11=nx/11.
       if(iblk11*11 .ne. nx) iblk11=iblk11+1
c
       do 600 kount=1,iblk11
          isrt11 = (kount-1)*10+1
          istp11 = isrt11+10
              if(iblk11 .eq. kount) istp11=nx
          isrt10 = isrt11
          istp10 = isrt10 +9
c
              write(ireport,2620) (   i    , i=isrt11,istp11 )
 2620         format(/,' i   = ',11(1x,i4,1x,5x) )
              write(ireport,2621) (   x(i), i=isrt11,istp11 )
 2621         format(   ' x   = ',11(    f6.3 ,5x) )
              write(ireport,2622) ( hx(i), i=isrt10,istp10 )
 2622         format(   ' hx = ',1x,10(5x,f6.3      ) )
  600   continue
c
c       .. y-axis
c
       write(ireport,2710)
 2710   format(//,5x,' y-axis: ')
c
       jblk11=ny/11.
       if(jblk11*11 .ne. ny) jblk11=jblk11+1
c
       do 700 kount=1,jblk11
          jsrt11 = (kount-1)*10+1
          jstp11 = jsrt11+10
```

```
            if(jblk11 .eq. kount) jstp11=ny
         jsrt10 = jsrt11
         jstp10 = jsrt10+9
c
            write(ireport,2720) (   j    ,  j=jsrt11,jstp11 )
 2720       format(/,' j   = ',11(1x,i4,1x,5x) )
            write(ireport,2721) (   y(j),  j=jsrt11,jstp11 )
 2721       format(  ' y   = ',11(    f6.3 ,5x) )
            write(ireport,2722) ( hy(j),  j=jsrt10,jstp10 )
 2722       format(  ' hy  = ',1x,10(5x,f6.3     ) )
  700    continue
c
c       .. z-axis
c
        write(ireport,2810)
 2810    format(//,5x,' z-axis: ')
c
        kblk11=nz/11.
        if(kblk11*11 .ne. nz) kblk11=kblk11+1
c
        do 800 kount=1,kblk11
           ksrt11 = (kount-1)*10+1
           kstp11 = ksrt11+10
              if(kblk11 .eq. kount) kstp11=nz
           ksrt10 = ksrt11
           kstp10 = ksrt10+9
c
              write(ireport,2820) (   k    ,  k=ksrt11,kstp11 )
 2820         format(/,' k   = ',11(1x,i4,1x,5x) )
              write(ireport,2821) (   z(k),  k=ksrt11,kstp11 )
 2821         format(  ' z   = ',11(    f6.3 ,5x) )
              write(ireport,2822) ( hz(k),  k=ksrt10,kstp10 )
 2822         format(  ' hz  = ',1x,10(5x,f6.3     ) )
  800      continue
        endif
c
        if (reed) then
c

        write(ireport,*)
        write(ireport,*) 'Information of last run:'
        write(ireport,2901) otitle,ifile(1)
 2901   format(/,'Old title   : ',/,a80,//' Run no.: ',i4)
c
        write(ireport,2902) ifile(10)
 2902   format(/'Cumulative number of iterations for last ',
     +          'run no. is = ',i7)
c
        write(ireport,2903) rfile(28)
 2903   format(/,'Cumulative cpu. time in seconds for last ',
     +          'run no. is = ',f9.2,' seconds')
c
c      ... non-dimensional parameters ...
c
        write(ireport,2905) rfile(15),rfile(18)
 2905   format( /'Aspect ratio in x = ',0pf10.3,
     +             5x,'Beta_x (DEG) = ',0pf10.3)
c
        write(ireport,2906) rfile(16),rfile(19)
 2906   format( 'Aspect ratio in y = ',0pf10.3,
     +          5x,'Beta_y (DEG) = ',0pf10.3)
c
        write(ireport,2907) rfile(17),rfile(20)
 2907   format( 'Aspect ratio in z = ',0pf10.3,
     +          5x,'Beta_z (DEG) = ',0pf10.3)
c
c      ... mesh details and transient factors ...
c
        mesh=' See below '
c
        write(ireport,2908) rfile(22),ifile(2),ifile(3),ifile(4),mesh,
     +                rfile(23),rfile(21),rfile(27)
 2908   format(//,'False transient factors - alpha psi    = ',f8.4,
     +          10x,'Mesh              =',i3,' *',i3,' *',i3,' - ',
     +          '{',a11,'}',
     +         /26x,'Alpha theta = ',f8.4,10x,'Stability ratio = ',f8.4,
     +        /26x,'dt            = ',f8.4)

c
c      ... convergence limits and pdist ...
c
        write(ireport,2909) rfile(25),rfile(26)
 2909   format(/,'Convergence limits on    - Psi    ccp    = ',1pg10.4,
     +          /,26x,'Theta cct    = ',1pg10.4)
c
        write(ireport,2910) (lfile(l),l=1,4)
 2910   format(/,'Logicals - Converged  dist   heat tran',
     +          /,15x,l1,7x,l1,3(5x,l1))
c
c      ... write mesh distribution ...
c
c

         write(ireport,2930)
 2930     format(///,30x,'*** Old mesh distribution ***',/)
c
c         .. x-axis
c
         write(ireport,2931)
 2931     format(/)
c
         iblk11=ifile(2)/11.
         if(iblk11*11 .ne. ifile(2)) iblk11=iblk11+1
c
         do 900 kount=1,iblk11
            isrt11 = (kount-1)*10+1
            istp11 = isrt11+10
               if(iblk11 .eq. kount) istp11=ifile(2)
            isrt10 = isrt11
            istp10 = isrt10 +9
c
```

```
 2932          write(ireport,2932) (   i    , i=isrt11,istp11 )
               format(/,' i   = ',11(1x,i4,1x,5x) )
 2933          write(ireport,2933) ( hxold(i), i=isrt10,istp10 )
               format(   ' hx  = ',1x,10(5x,f6.3     ) )
  900     continue
c
c         .. y-axis
c
         write(ireport,2931)
c
         jblk11=ifile(3)/11.
         if(jblk11*11 .ne. ifile(3)) jblk11=jblk11+1
c
         do 910 kount=1,jblk11
            jsrt11 = (kount-1)*10+1
            jstp11 = jsrt11+10
               if(jblk11 .eq. kount) jstp11=ifile(3)
            jsrt10 = jsrt11
            jstp10 = jsrt10+9
c
 2944          write(ireport,2944) (   j    , j=jsrt11,jstp11 )
               format(/,' j   = ',11(1x,i4,1x,5x) )
 2945          write(ireport,2945) ( hyold(j), j=jsrt10,jstp10 )
               format(   ' hy  = ',1x,10(5x,f6.3     ) )
  910     continue
c
c         .. z-axis
c
         write(ireport,2931)
c
         kblk11=ifile(4)/11.
         if(kblk11*11 .ne. ifile(4)) kblk11=kblk11+1
c
         do 920 kount=1,kblk11
            ksrt11 = (kount-1)*10+1
            kstp11 = ksrt11+10
               if(kblk11 .eq. kount) kstp11=ifile(4)
            ksrt10 = ksrt11
            kstp10 = ksrt10+9
c
 2957          write(ireport,2957) (   k    , k=ksrt11,kstp11 )
               format(/,' k   = ',11(1x,i4,1x,5x) )
 2958          write(ireport,2958) ( hzold(k), k=ksrt10,kstp10 )
               format(   ' hz  = ',1x,10(5x,f6.3     ) )
  920     continue
c
        write(ireport,2969)
 2969  format(1h1,5x,'Temperature boundary conditions',/)
        write(ireport,2970)
 2970  format(16x,'z--->',16x,'z--->',19x,'z--->',16x,'z--->',19x,'y--->'
     +       ,16x,'y--->')
c
c      ... blanck out inside boxes ...
c
         do 169 ivert=2,15
c
c          .. x = constant planes
c
            do 161 ihoriz=4, 20
               table(ivert,ihoriz) = ' '
  161       continue
            do 162 ihoriz=25, 41
               table(ivert,ihoriz) = ' '
  162       continue
c
c          .. y = constant planes
c
            do 163 ihoriz=49,65
               table(ivert,ihoriz) = ' '
  163       continue
            do 164 ihoriz=70, 86
               table(ivert,ihoriz) = ' '
  164       continue
c
c          .. z = constant planes
c
            do 165 ihoriz=94, 110
               table(ivert,ihoriz) = ' '
  165       continue
            do 166 ihoriz=115, 131
               table(ivert,ihoriz) = ' '
  166       continue
  169    continue
c
         do 965 i=1, 5
          write(ireport,2470) (table(i,j), j=1, 132)
  965 continue
c
         write(ireport,2980) lfile(5),lfile(6),
     +                 lfile(7),lfile(8),
     +                 lfile(9),lfile(10)
 2980 format(2x,'|',2x,'patch on = ',l1,3x,
     +          '|',2x,'|',3x,'patch on = ',l1,2x,'|',
     +       5x,'|',3x,'patch on = ',l1,2x,
     +          '|',2x,'|',3x,'patch on = ',l1,2x,'|',
     +       5x,'|',3x,'patch on = ',l1,2x,
     +          '|',2x,'|',3x,'patch on = ',l1,2x,'|')
c
c
        write(ireport,2981) ifile(44),ifile(45),ifile(48),ifile(49),
     +                 ifile(52),ifile(53),ifile(56),ifile(57),
     +                 ifile(60),ifile(61),ifile(64),ifile(65)
 2981 format(2x,'|',4x,' j=',i2,',',i2,5x,
     +          '|',2x,'|',5x,' j=',i2,',',i2,4x,'|',
     +       5x,'|',5x,' i=',i2,',',i2,4x,
     +          '|',2x,'|',5x,' i=',i2,',',i2,4x,'|',
     +       5x,'|',5x,' i=',i2,',',i2,4x,
     +          '|',2x,'|',5x,' i=',i2,',',i2,4x,'|')
c
        write(ireport,2982) ifile(46),ifile(47),ifile(50),ifile(51),
```

```
     +                ifile(54),ifile(55),ifile(58),ifile(59),
     +                ifile(62),ifile(63),ifile(66),ifile(67)
 2982 format(2x,'|',4x,' k=',i2,',',i2,5x,
     +          '|',2x,'|',5x,' k=',i2,',',i2,4x,'|',
     +      5x,'|',5x,' k=',i2,',',i2,4x,
     +          '|',2x,'|',5x,' k=',i2,',',i2,4x,'|',
     +      5x,'|',5x,' j=',i2,',',i2,4x,
     +          '|',2x,'|',5x,' j=',i2,',',i2,4x,'|')
c
      write(ireport,2983) rfile(53),rfile(57),rfile(61),rfile(65),
     +                rfile(69),rfile(73)
 2983 format(2x,'|',4x,'theta=',f4.2,3x,
     +          '|',2x,'|',5x,'theta=',f4.2,2x,'|',
     +      5x,'|',5x,'theta=',f4.2,2x,
     +          '|',2x,'|',5x,'theta=',f4.2,2x,'|',
     +      5x,'|',5x,'theta=',f4.2,2x,
     +          '|',2x,'|',5x,'theta=',f4.2,2x,'|')
c
      do 970 i=11, 16
       write(ireport,2470) (table(i,j), j=1, 132)
  970 continue
c
      write(ireport,2984) rfile(54),rfile(58),
     +                rfile(62),rfile(66),
     +                rfile(70),rfile(74)
 2984 format(2x,'cg=',f5.2,1x,'    ',5x,
     +      4x,'cg=',f5.2,1x,'    ',5x,
     +      7x,'cg=',f5.2,1x,'    ',5x,
     +      4x,'cg=',f5.2,1x,'    ',5x,
     +      7x,'cg=',f5.2,1x,'    ',5x,
     +      4x,'cg=',f5.2,1x,'    ' )
c
      write(ireport,2985) rfile(55),rfile(56),rfile(59),rfile(60),
     +                rfile(63),rfile(64),rfile(67),rfile(68),
     +                rfile(71),rfile(72),rfile(75),rfile(76)
 2985 format(2x,'ca=',f5.2,1x,'cb=',f5.2,
     +      4x,'ca=',f5.2,1x,'cb=',f5.2,
     +      7x,'ca=',f5.2,1x,'cb=',f5.2,
     +      4x,'ca=',f5.2,1x,'cb=',f5.2,
     +      7x,'ca=',f5.2,1x,'cb=',f5.2,
     +      4x,'ca=',f5.2,1x,'cb=',f5.2)
c
      write(ireport,2986) rfile(15),rfile(16),rfile(17)
 2986 format(/2x,'y-z plane; x=0.0      y-z plane; x=',f5.2,
     +        6x,'x-z plane; y=0.0      x-z plane; y=',f5.2,
     +        6x,'x-y plane; z=0.0      x-y plane; z=',f5.2,////)
c
      write(ireport,2990)
 2990 format(//,5x,'Velocity boundary conditions',/)
      write(ireport,2991)
 2991 format(16x,'z--->',16x,'z--->',19x,'z--->',16x,'z--->',19x,'y--->'
     +       ,16x,'y--->')
c
c     ... blanck out inside boxes ...
c
      do 179 ivert=2,15
c
c       .. x = constant planes
c
        do 171 ihoriz=4, 20
            table(ivert,ihoriz) = ' '
  171   continue
        do 172 ihoriz=25, 41
            table(ivert,ihoriz) = ' '
  172   continue
c
c       .. y = constant planes
c
        do 173 ihoriz=49,65
            table(ivert,ihoriz) = ' '
  173   continue
        do 174 ihoriz=70, 86
            table(ivert,ihoriz) = ' '
  174   continue
c
c       .. z = constant planes
c
        do 175 ihoriz=94, 110
            table(ivert,ihoriz) = ' '
  175   continue
        do 176 ihoriz=115, 131
            table(ivert,ihoriz) = ' '
  176   continue
  179 continue
c
      do 977 i=1, 6
       write(ireport,2470) (table(i,j), j=1, 132)
  977 continue
c
      write(ireport,2992) rfile(83),rfile(86),rfile(89),rfile(92)
 2992 format(2x,'|',4x,'u = na   ',5x,'|',2x,'|',5x,'u = na   ',4x,'|',
     +        5x,'|',5x,'u =',f5.2,4x,'|',2x,'|',5x,'u =',f5.2,4x,'|',
     +        5x,'|',5x,'u =',f5.2,4x,'|',2x,'|',5x,'u =',f5.2,4x,'|')
c
      write(ireport,2993) rfile(78),rfile(81),rfile(90),rfile(93)
 2993 format(2x,'|',4x,'v =',f5.2,5x,'|',2x,'|',5x,'v =',f5.2,4x,'|',
     +        5x,'|',5x,'v = na   ',4x,'|',2x,'|',5x,'v = na   ',4x,'|',
     +        5x,'|',5x,'v =',f5.2,4x,'|',2x,'|',5x,'v =',f5.2,4x,'|')
c
      write(ireport,2994) rfile(79),rfile(82),rfile(85),rfile(88)
 2994 format(2x,'|',4x,'w =',f5.2,5x,'|',2x,'|',5x,'w =',f5.2,4x,'|',
     +        5x,'|',5x,'w =',f5.2,4x,'|',2x,'|',5x,'w =',f5.2,4x,'|',
     +        5x,'|',5x,'w = na   ',4x,'|',2x,'|',5x,'w = na   ',4x,'|')
c
      do 978 i=11, 16
       write(ireport,2470) (table(i,j), j=1, 132)
  978 continue
c
      write(ireport,2995) rfile(15),rfile(16),rfile(17)
 2995 format(/2x,'y-z plane; x=0.0      y-z plane; x=',f5.2,
```

```
     +         6x,'x-z plane; y=0.0      x-z plane; y=',f5.2,
     +         6x,'x-y plane; z=0.0      x-y plane; z=',f5.2,////)
      endif
c
      write(iplt,*)'Central x-axis temperature_time
     + transient results:'
      write(iplt,8000) pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      write(iplt,8001) hx(1),hx(1)+hx(2),hx(1)+hx(2)+hx(3),
     + hx(1)+hx(2)+hx(3)+hx(4),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7)+hx(8),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7)+hx(8)+hx(9),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+
     + hx(7)+hx(8)+hx(9)+hx(10),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+
     + hx(7)+hx(8)+hx(9)+hx(10)+hx(11)

      write(45,*)'Central x-axis x_component_velocity-time
     + transient results:'
      write(45,8000) pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      write(45,8001) hx(1),hx(1)+hx(2),hx(1)+hx(2)+hx(3),
     + hx(1)+hx(2)+hx(3)+hx(4),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7)+hx(8),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+hx(7)+hx(8)+hx(9),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+
     + hx(7)+hx(8)+hx(9)+hx(10),
     + hx(1)+hx(2)+hx(3)+hx(4)+hx(5)+hx(6)+
     + hx(7)+hx(8)+hx(9)+hx(10)+hx(11)
8000        format(16x,11(1x,i2,i2,i2))
8001        format(16x,11(1x,f6.3))
      write(inu,*)'Nusselt numbers: nu_x_1, nu_x_nx, nu_y_1,
     + nu_y_ny, nu_z_1, nu_z_nz'
      write(inu,*)

      return
      end

      subroutine datin
c
c     ------------------------------------------------------------------
c     |
c     |     *** read in data (parameters and grid distribution) ***    |
c     |                                                                |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)

      logical  reed,rite,strt,dist,heat,nuss,tran
      logical  rt,rp1,rp2,rp3,rmax
      logical  divv,divp,dmax
      logical  vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +         mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +         vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +         myp1,myp2,myp3,myt,myu,myv,myw,
     +         vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +         mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      logical  ixl,ixh,iyl,iyh,izl,izh
c
      character  title*80,otitle*80
      character  xc*1,yc*1,zc*1,vpform*1
      character  meshtp*1,exprmnt*8,gridfile*8,srcetype*1,datype*1
c
      integer drawstep
      integer pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      real keff
c
      common  /title/title,otitle
      common  /exprmnt/exprmnt
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /logc/reed,rite,strt,dist,heat,nuss,tran
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /logd/divv,divp,dmax
      common  /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +                mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +                vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +                myp1,myp2,myp3,myt,myu,myv,myw,
     +                vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +                mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common  /logtbc/ixl,ixh,iyl,iyh,izl,izh
      common  /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +                cct,tmax,pdist,rtime
      common  /alpfs/alpp,altp
      common  /intcom/itercmpo,norun,mainiter
      common  /numcom/numit
      common  /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +             ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +             zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
      common  /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +             iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +             izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
```

```fortran
      common /velbc/xlu,xlv,xlw,xhu,xhv,xhw,
     +              ylu,ylv,ylw,yhu,yhv,yhw,
     +              zlu,zlv,zlw,zhu,zhv,zhw
      common /mesh/meshtp
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /h/hx(max),hy(max),hz(max),
     +         hxold(max),hyold(max),hzold(max)
      common /axis/x(max),y(max),z(max)
      common /vmpfrm/xc,yc,zc,vpform
      common /vmpnum/npx,npy,npz
      common /vmploc/mx(nxmax),my(nymax),mz(nzmax)
      common /vmppos/rmx(nxmax),rmy(nymax),rmz(nzmax)
      common /mapcom/mnx,mny,mnz
      common /phierr/phierr
      common /da/da
      common /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,xll,annu,gravity,ddd,keff
      common /ccaa/cc0,dtemp,tcold,thot
      common /draw/drawstep
      common /pst/pstx,psty,pstz
      common /pstxyz/pstx1,psty1,pstz1
     +       ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +       ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +       ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +       ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +       ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      common /param_q/qparam(20)
      common /numshow/numshow
      common /iterpp/iterpp
      common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common /tstrt/tstrt
      common /gridfile/gridfile
      common /txty/tpnt(10),q(10),npt,nparam,srcetype
      common /datype/datype

      inparam  = 7
      irg  = 8
c
      open(unit=inparam   ,file='inparam',status='old')
c
      tran= .false.
c
c     ... read title ...
c
      read(inparam,1005) title
 1005 format(/,a80)
c
c     ... read physical data ...
c
      read(inparam,1020) keff,alphaf,cpfl,cpsd,rhof,rhos
      read(inparam,5555) epson,xll,annu,gravity,ddd
 5555 format(/,5(g13.4,1x))
c
c      ... read reference cold and hot temperatures, initial temperature
...
c
      read(inparam,5555) tcold,thot,tstrt
c
c     ... read box geometry parameters ...
c
      read(inparam,1020) arx,ary,arz,betax,betay,betaz
 1020 format(/,6(g10.4,1x))
      if ((arx.ne.1.0).and.(ary.ne.1.0).and.(arz.ne.1.0)) then
       write(ireport,2010)
       write(*,4010)
 2010   format(//'solution halted, either arx or ary or arz must be',
     +        ' equal to 1.0 !',//)
c
 4010   format(//' solution halted, either arx or ary or arz must be',
     +        ' equal to 1.0 !'//)
c
       stop
      endif
 1025 format(/,i10)
c
c     ... read nodal information ...
c
      read(inparam,1026) meshtp, nx, ny, nz
 1026 format(/,9x,a1,1x,3(i5,1x))
c
c
      if ((nx.gt.nxmax).or.(ny.gt.nymax).or.(nz.gt.nzmax)) then
       write(ireport,2020)
       write(*,4020)
 2020   format(//'solution halted, mesh size in input file greater',
     +        ' than maximum allowable in program !',//)
c
 4020   format(//' solution halted, mesh size in input file greater',
     +        ' than maximum allowable in program !'//)
c
       stop
      endif
c
c     ... read stabitily ratios, convergence criterion ...
c
      read(inparam,1020) stabr,alpp,altp,phierr,ccp,cct
c
c ... read maximum number of iterations and contour map output interval.
c
      read(inparam,1035) itercmpo,iterpp,mainiter,drawstep
 1035 format(/,4(i10,1x))
c
c  ... read positions for output transient results ...
c
      read(inparam,1036) pstx1,psty1,pstz1
     +         ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
      read(inparam,1036) pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +         ,pstx6,psty6,pstz6
      read(inparam,1036) pstx7,psty7,pstz7,pstx8,psty8,pstz8
     +         ,pstx9,psty9,pstz9
```

```
      read(inparam,1036) pstx10,psty10,pstz10,pstx11,psty11,pstz11
 1036 format(/,3(i3,1x,i3,1x,i3,3x))
c
c     ...  read run number ...
c
      read(inparam,1010) norun
 1010 format(/,i5)
c
c     ... read input control ...
c
      read(inparam,1044) reed, rite, strt, dist, heat, nuss
 1044 format(/,6(15,1x))
c
c    ... read map dimensions ...
c
      read(inparam,1047) mnx,mny,mnz,pdist
 1047 format(/,3(i5,1x),42x,g10.4)
c
c     ... read temperature boundary conditions ...
c
      read(inparam,1050) ixl,  xlci,  ixlyl,ixlyh,ixlzl,ixlzh,
     +                   ixh,  xhci,  ixhyl,ixhyh,ixhzl,ixhzh,
     +                   xlcg,xlca,xlcb,   xhcg,xhca,xhcb
 1050 format(////,l1,11x,f5.2,4i4,   5x,   l1,11x,f5.2,4i4,
     +          //,3(f5.2,1x),15x,    5x,   3(f5.2,1x) )
      read(inparam,1051) iyl,  ylci,  iylxl,iylxh,iylzl,iylzh,
     +                   iyh,  yhci,  iyhxl,iyhxh,iyhzl,iyhzh,
     +                   ylcg,ylca,ylcb,   yhcg,yhca,yhcb
      read(inparam,1051) izl,  zlci,  izlxl,izlxh,izlyl,izlyh,
     +                   izh,  zhci,  izhxl,izhxh,izhyl,izhyh,
     +                   zlcg,zlca,zlcb,   zhcg,zhca,zhcb
 1051 format(   //,l1,11x,f5.2,4i4,   5x,   l1,11x,f5.2,4i4,
     +          //,3(f5.2,1x),15x,    5x,   3(f5.2,1x) )
c
      if( ( ixl.and.((ixlyh.gt.ny).or.(ixlzh.gt.nz)) ) .or.
     +    ( ixh.and.((ixhyh.gt.ny).or.(ixhzh.gt.nz)) ) .or.
     +    ( iyl.and.((iylxh.gt.nx).or.(iylzh.gt.nz)) ) .or.
     +    ( iyh.and.((iyhxh.gt.nx).or.(iyhzh.gt.nz)) ) .or.
     +    ( izl.and.((izlxh.gt.nx).or.(izlyh.gt.ny)) ) .or.
     +    ( izh.and.((izhxh.gt.nx).or.(izhyh.gt.ny)) ) ) then
c
       write(ireport,2030)
       write(*,4030)
 2030    format(//,'solution halted, size of isothermal patch larger',
     +             ' than the size of the wall !',//)
c
 4030    format(//' solution halted, size of isothermal patch larger',
     +             ' than the size of the wall !'//)
c
        stop
      endif
c
c     ... read velocity boundary conditions ...
c
      read(inparam,1055) xlv, xlw, xhv, xhw
 1055 format(////,5x,1x,2(f5.2,1x),20x,5x,1x,2(f5.2,1x))
       read(inparam,1056) ylu, ylw, yhu, yhw
 1056 format(//,f5.2,1x,5x,1x,f5.2,1x,20x,f5.2,1x,5x,1x,f5.2)
       read(inparam,1060) zlu, zlv, zhu, zhv
 1060 format(//,2(f5.2,1x),5x,1x,20x,2(f5.2,1x))
c
c     ... read printing form ...
c
       read(inparam,1062) vpform
 1062 format(///,4x,a1)
c
c     ... read printing controls ...
c
       read(inparam,1065) vxp1, vxp2, vxp3, vxt, vxu, vxv, vxw
 1065 format(/,7x,7i4)
       read(inparam,1065) mxp1, mxp2, mxp3, mxt, mxu, mxv, mxw
       read(inparam,1070) xc, npx
 1070 format(/,4x,a1,37x,i5)
c
       if (xc.eq.'c') then
        read(inparam,1075) (rmx(i),i=1,npx)
 1075 format( (/,11(f5.2,1x)) )
        do 50 i=1, npx
           mx(i) = int(rmx(i)*(nx-1)/arx) + 1
   50    continue
       else
        read(inparam,1080) (mx(i),i=1,npx)
 1080 format( (/,11(i5,1x)) )
       endif
c
       read(inparam,1065) vyp1, vyp2, vyp3, vyt, vyu, vyv, vyw
       read(inparam,1065) myp1, myp2, myp3, myt, myu, myv, myw
       read(inparam,1070) yc, npy
c
       if (yc.eq.'c') then
        read(inparam,1075) (rmy(j),j=1,npy)
        do 60 i=1, npy
           my(i) = int(rmy(i)*(ny-1)/ary) + 1
   60    continue
       else
        read(inparam,1080) (my(j),j=1,npy)
       endif
c
       read(inparam,1065) vzp1, vzp2, vzp3, vzt, vzu, vzv, vzw
       read(inparam,1065) mzp1, mzp2, mzp3, mzt, mzu, mzv, mzw
       read(inparam,1070) zc, npz
c
       if (zc.eq.'c') then
        read(inparam,1075) (rmz(k),k=1,npz)
        do 70 i=1, npz
           mz(i) = int(rmz(i)*(nz-1)/arz) + 1
   70    continue
       else
        read(inparam,1080) (mz(k),k=1,npz)
       endif
```

```
c
c     ... read in controls for calculation of residuals ...
c
      read(inparam,1085) rt, rp1, rp2, rp3, rmax
 1085 format(//,5(15,1x))
c
c     ... read in controls for calculation of divergence ...
c
      read(inparam,1090) divv, divp, dmax
 1090 format(//,3(15,1x))
c
c     ... read in experiment name for creating output files ...
c
      read(inparam,1006) exprmnt
 1006  format(/,a8)
c
c     ... read heating source parameters ...
c
      read(inparam,3000) npt,nparam,srcetype
 3000  format(//,i4,1x,i4,5x,a1)
      read(inparam,5555) tpnt(1),tpnt(2),tpnt(3),tpnt(4),tpnt(5)
      read(inparam,5555) tpnt(6),tpnt(7),tpnt(8),tpnt(9),tpnt(10)
      read(inparam,5555) q(1),q(2),q(3),q(4),q(5)
      read(inparam,5555) q(6),q(7),q(8),q(9),q(10)
c
c     ... read display interval ...
c
      read(inparam,6000) numshow
 6000  format(/,i3)
c
c     ...  read Darcy numbers in x, y, and z directions ...
c
      read(inparam,1008) darcyx,darcyy,darcyz,datype
 1008 format(/,g10.4,1x,g10.4,1x,g10.4,5x,a1)
c
c     ... read grid file name ...
c
      read(inparam,1007) gridfile
 1007  format(/,a8)

      if( (meshtp.eq.'n') .or. (meshtp.eq.'N')
     + .or. (meshtp.eq.'T') .or. (meshtp.eq.'t') ) then
c
c       ... read stepsizes between nodes if mesh is non uniform ...
c
      open(unit=irg   ,file=gridfile ,status='old')
      read(irg,1027) nx
 1027  format(//,26x,i3)
      read(irg,1028) (hx(i),i=1,nx-1)
 1028  format( (/,14f6.2) )
c
      read(irg,1027) ny
      read(irg,1028) (hy(j),j=1,ny-1)
c
      read(irg,1027) nz
      read(irg,1028) (hz(k),k=1,nz-1)
      close(irg)
c
      else if((meshtp.ne.'b').or.(meshtp.ne.'B')) then
c
c       ... set uniform intervals between nodes ...
c
      do 10 i=1, nx
        hx(i) = 1.0
 10    continue
      do 11 j=1, ny
        hy(j) = 1.0
 11    continue
      do 12 k=1, nz
        hz(k) = 1.0
 12    continue
      endif

      if((meshtp.eq.'t').or.(meshtp.eq.'T')) tran=.true.
c
c     ... normalise stepsizes in accordance to lengths of
c        boundaries ...
c
      if(.not.((meshtp.eq.'b').or.(meshtp.eq.'B'))) then
       sumhx = 0.0
       do 20 i=1, nx-1
             sumhx = sumhx + hx(i)
 20     continue
c
       ratiox = arx/sumhx
       do 21 i=1, nx-1
          hx(i)   = hx(i)*ratiox
 21     continue
c
       sumhy = 0.0
       do 30 j=1, ny-1
             sumhy = sumhy + hy(j)
 30     continue
c
       ratioy = ary/sumhy
       do 31 j=1, ny-1
          hy(j)   = hy(j)*ratioy
 31     continue
c
       sumhz = 0.0
       do 40 k=1, nz-1
             sumhz = sumhz + hz(k)
 40     continue
c
       ratioz = arz/sumhz
       do 41 k=1, nz-1
          hz(k)   = hz(k)*ratioz
 41     continue
      endif
      close(inparam)
```

```
      write(*,*)'Data input done. INPARAM is closed.'

      return
      end


      subroutine distb
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |       *** disturb stream function field at random ***          |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +               cct,tmax,pdist,rtime
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /p1/p1(nxmax,nymax,nzmax)
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
c
c     ... random disturbance (between -pdist to +pdist) of psi field ...
c
c
c     .. p1 field
c
      ii=1
      jj=0
      do 10  i=2,nxm1
      do 10  j=2,nym1
      do 10  k=2,nzm1
      p1(i,j,k)  = p1(i,j,k)+p1(i,j,k)*genran(ii,jj)*pdist
   10 continue
c
c     .. p2 field
c
      do 20  i=2,nxm1
      do 20  j=2,nym1
      do 20  k=2,nzm1
      p2(i,j,k)  = p2(i,j,k)+p2(i,j,k)*genran(ii,jj)*pdist
   20 continue
c
c     .. p3 field
c
      do 30  i=2,nxm1
      do 30  j=2,nym1
      do 30  k=2,nzm1
      p3(i,j,k)  = p3(i,j,k)+p3(i,j,k)*genran(ii,jj)*pdist
   30 continue
c
c     .. recalculate velocity
c
      call veloc
c
      return
      end

      subroutine div(f1,f2,f3,head,divlog)
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |         *** calculate divergence of a vector field ***         |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical divlog
c
      character*15 head
c
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +           cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +           cy4(nymax),cy5(nymax),cy6(nymax)
      common /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +           cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
      dimension f1(nxmax,nymax,nzmax),
     +          f2(nxmax,nymax,nzmax),
     +          f3(nxmax,nymax,nzmax)
c
      do 10 i=2, nxm1
       do 10 j=2, nym1
          do 10 k=2, nzm1
             aux(i,j,k)  =   (    cx1(i)*f1(i-1,j,k)
     +                          + cx2(i)*f1(i  ,j,k)
     +                          + cx3(i)*f1(i+1,j,k) )
     +                     + (    cy1(j)*f2(i,j-1,k)
     +                          + cy2(j)*f2(i,  j,k)
     +                          + cy3(j)*f2(i,j+1,k) )
     +                     + (    cz1(k)*f3(i,j,k-1)
     +                          + cz2(k)*f3(i,j,k  )
     +                          + cz3(k)*f3(i,j,k+1) )
   10 continue
c
      if(divlog) then
       do 15 i=1, nx
          do 15 j=1, ny
             aux2d(i,j)  = 0.0
   15    continue
c
       do 20 k=2, nzm1
          do 25 i=2, nxm1
             do 25 j=2, nym1
              aux2d(i,j)  = aux(i,j,k)
```

```
   25          continue
               call mprtz(aux2d,head,k)
   20     continue
        endif
c
        call maxval(aux,head)
c
        return
        end
c
c
c
        subroutine drawcurv
c      ------------------------------------------------------------
c     |                                                            |
c     |   draw the heating-temperature curve at specifed point     |
c     |                                                            |
c      ------------------------------------------------------------
        integer drawstep,position
        character  line*117,exprmnt*8
        common /exprmnt/exprmnt
        common /intcom/itercmpo,norun,mainiter
        common /draw/drawstep
        common /pst/pstx,psty,pstz
        common /point/points(200)
        common /timestep/timestep
        pointmax=-10000.
        pointmin=10000.
        write(*,*) 'heating curve of ',exprmnt
        do 20 i=1,200
        if (pointmax.lt.points(i)) pointmax=points(i)
        if (pointmin.gt.points(i)) pointmin=points(i)
   20    continue
        write(*,40) pointmin,(pointmin+pointmax)/2.,pointmax
   40     format(1x,t1,f8.2,t33,f8.2,t68,f8.2)
        write(*,50) '|','|','|'
   50     format(2x,t2,a1,t35,a1,t70,a1)
        line=' '
        do 60 i=1,70
        line(i:i)='_'
   60    continue
        write(*,*) line
        if (pointmax .eq. pointmin) goto 99
        ay=69./(pointmax-pointmin)
        by=1.-ay*pointmin
        txtemp=-2.
        do 10 i=1,76
        line=' '
        line(1:1)='|'
        line(70:70)='|'
        position=nint(ay*points(i)+by)
        line(position:position)='.'
        write(*,70)line ,txtemp
   70     format(1x,a71,t72,f5.2)
        txtemp=txtemp+0.5
   10    continue
        do 80 i=1,71
        line(i:i)='_'
   80    continue
        write(*,*) line
        return
   99    write(*,*)'heating source is not a function of temperature.'
        do 98 i=1,71
        line(i:i)='_'
   98    continue
        write(*,*) line
        return
        end

        subroutine gaussjdn(a,n,np,b,m,mp)
c
c      ------------------------------------------------------------
c     |                                                            |
c     |           ***  Gauss-Jordan elimination  ***               |
c     |                                                            |
c      ------------------------------------------------------------
c
        parameter (nmax=50)
        dimension a(np,np),b(np,mp),ipiv(nmax),indxr(nmax),indxc(nmax)
        do 11 j=1,n
        ipiv(j)=0
   11   continue
        do 22 i=1,n
        big=0.
        do 13 j=1,n
          if(ipiv(j).ne.1)then
            do 12 k=1,n
               if (ipiv(k).eq.0) then
               if (abs(a(j,k)).ge.big)then
                 big=abs(a(j,k))
                 irow=j
                 icol=k
               endif
               else if (ipiv(k).gt.1) then
               pause 'singular matrix'
               endif
   12         continue
          endif
   13     continue
        ipiv(icol)=ipiv(icol)+1
        if (irow.ne.icol) then
          do 14 l=1,n
            dum=a(irow,l)
            a(irow,l)=a(icol,l)
            a(icol,l)=dum
   14       continue
          do 15 l=1,m
            dum=b(irow,l)
            b(irow,l)=b(icol,l)
            b(icol,l)=dum
```

```
15        continue
        endif
        indxr(i)=irow
        indxc(i)=icol
        if (a(icol,icol).eq.0.) pause 'singular matrix.'
        pivinv=1./a(icol,icol)
        a(icol,icol)=1.
        do 16 l=1,n
          a(icol,l)=a(icol,l)*pivinv
16        continue
        do 17 l=1,m
          b(icol,l)=b(icol,l)*pivinv
17        continue
        do 21 ll=1,n
          if(ll.ne.icol)then
            dum=a(ll,icol)
            a(ll,icol)=0.
            do 18 l=1,n
              a(ll,l)=a(ll,l)-a(icol,l)*dum
18          continue
            do 19 l=1,m
              b(ll,l)=b(ll,l)-b(icol,l)*dum
19          continue
          endif
21        continue
22      continue
        do 24 l=n,1,-1
        if(indxr(l).ne.indxc(l))then
          do 23 k=1,n
            dum=a(k,indxr(l))
            a(k,indxr(l))=a(k,indxc(l))
            a(k,indxc(l))=dum
23          continue
        endif
24      continue
        return
        end

        function genran(ii,jj)
c
c       -----------------------------------------------------------------
c      |
c      |       *** generates random numbers between -1 and 1 ***        |
c      |                                                                |
c       -----------------------------------------------------------------
c
        integer*2 ii,jj
c
        jj=ii*899
        if(jj.lt.0) then
        jj=jj+32767+1
        endif
        rjj=jj
        genran=2.0*(rjj/32767) -1.0
c
```

```
        return
        end

        subroutine htc
c
c       -----------------------------------------------------------------
c      |                                                                |
c      | *** calculate local and average heat transfer coefficients *** |
c      |                                                                |
c       -----------------------------------------------------------------
c
        parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
        character*15 head
        character locate*3
c
        common /fileno/inparam,irg,ireport,itin,itout
        common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +                 cct,tmax,pdist,rtime
        common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
        common /h/hx(max),hy(max),hz(max),
     +             hxold(max),hyold(max),hzold(max)
        common /axis/x(max),y(max),z(max)
        common /mapcom/mnx,mny,mnz
        common /tbfdc/cxf(2,3),cyf(2,3),czf(2,3)
        common /t/theta(nxmax,nymax,nzmax)
        common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
        dimension tint(max),tt(max)
c
c     ... x = 0 and x = arx walls ...
c
        do 100 i=1,nx,nxm1
c
        if(i .eq. 1) then
        locate = ' 0 '
        elseif(i .eq. nx) then
        locate = 'arx'
        endif
c
c     .. calculate local Nusselt numbers
c
        if(i .eq. 1) then
         do 110 j=1, ny
            do 110 k=1, nz
                aux2d(j,k)  =  -cxf(1,1)*theta(1,j,k)
     +                         -cxf(1,2)*theta(2,j,k)
     +                         -cxf(1,3)*theta(3,j,k)
110      continue
c
        elseif(i .eq. nx) then
c
         do 120 j=1, ny
            do 120 k=1, nz
                aux2d(j,k)  =  +cxf(2,1)*theta(nx   ,j,k)
```

```
     +                                +cxf(2,2)*theta(nxm1,j,k)
     +                                +cxf(2,3)*theta(nxm2,j,k)
  120       continue
         endif
c
         head = '    local nu     '
         call mprtx(aux2d,head,1)
c
c       .. calculate the z-averaged Nusselt numbers, nuz(y)
c
         do 132 j=1,ny
            do 131 k=1,nz
               tint(k) = aux2d(j,k)
  131       continue
         call simpsn (tint,nz,hz,tt(j))
  132 continue
c
         write (ireport,2110) locate, (j, tt(j), j=1,ny)
 2110 format(//,' wall x = ',a3,':    average Nusselt number along',
     +        'lines parallel to the z-axis:-',//,19x,'j',9x,'nu',//,
     +        (17x,i3,6x,1p,g12.4))
c
c       . calculate the average of the z-averaged Nusselt numbers, nuzy
c
         call simpsn (tt,ny,hy,tbar)
c
         write (ireport,2120) tbar
 2120 format(//,15x,'the average of those averages is ',1p,g12.4)
c
c       .. calculate the y-averaged Nusselt numbers, nuy(z)
c
         do 142 k=1,nz
            do 141 j=1,ny
               tint(j) = aux2d(j,k)
  141       continue
         call simpsn (tint,ny,hy,tt(k))
  142 continue
c
         write (ireport,2130) (k, tt(k), k=1,nz)
 2130 format(//15x,'average Nusselt number along lines parallel',
     +' to the y-axis:-'//19x,'k',9x,'nu'//(17x,i3,6x,g12.4))
c
c       . calculate the average of the y-averaged Nusselt numbers, nuyz
c
         call simpsn (tt,nz,hz,tbar)
c
         write (ireport,2140) tbar
 2140 format(//15x,'the average of those averages is ',g12.4)
c
c       .. plot contour map only if nuyz > .1
c
         if (tbar.gt..1) then
          head = '   nu at x = '//locate
          write (ireport,2150) head
 2150      format(1h1,10x,5('*'),' contour map of ',a15,5('*'))
           call  map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
  100 continue
c
c       ... y = 0 and y = ary walls ...
c
         do 200 j=1,ny,nym1
c
         if(j .eq. 1) then
         locate = ' 0 '
         elseif(j .eq. ny) then
         locate = 'ary'
         endif
c
c       .. calculate local Nusselt numbers
c
         if(j .eq. 1) then
          do 210 i=1, nx
             do 210 k=1, nz
                aux2d(i,k)  =  -cyf(1,1)*theta(i,1,k)
     +                         -cyf(1,2)*theta(i,2,k)
     +                         -cyf(1,3)*theta(i,3,k)
  210       continue
c
         elseif(j .eq. ny) then
c
          do 220 i=1, nx
             do 220 k=1, nz
                aux2d(i,k)  =  cyf(2,1)*theta(i,ny   ,k)
     +                        +cyf(2,2)*theta(i,nym1,k)
     +                        +cyf(2,3)*theta(i,nym2,k)
  220       continue
         endif
c
         head = '    local nu     '
         call mprty(aux2d,head,1)
c
c       .. calculate the z-averaged Nusselt numbers, nuz(x)
c
         do 232 i=1,nx
            do 231 k=1,nz
               tint(k) = aux2d(i,k)
  231       continue
         call simpsn (tint,nz,hz,tt(i))
  232 continue
c
         write (ireport,2210) locate, (i, tt(i), i=1,nx)
 2210 format(//,' wall y = ',a3,':    average Nusselt number along',
     +        'lines parallel to the z-axis:-',//,19x,'i',9x,'nu',//,
     +        (17x,i3,6x,1p,g12.4))
c
c       . calculate the average of the z-averaged Nusselt numbers, nuzx
c
         call simpsn (tt,nx,hx,tbar)
```

```
c
      write (ireport,2220) tbar
 2220 format(//,15x,'the average of those averages is ',1p,g12.4)
c
c     .. calculate the x-averaged Nusselt numbers, nux(z)
c
      do 242 k=1,nz
         do 241 i=1,nx
            tint(i) = aux2d(i,k)
  241    continue
      call simpsn (tint,nx,hx,tt(k))
  242 continue
c
      write (ireport,2230) (k, tt(k), k=1,nz)
 2230 format(//15x,'average Nusselt number along lines parallel',
     +' to the x-axis:-'//19x,'k',9x,'nu'//(17x,i3,6x,g12.4))
c
c     . calculate the average of the x-averaged Nusselt numbers, nuxz
c
      call simpsn (tt,nz,hz,tbar)
c
      write (ireport,2240) tbar
 2240 format(//15x,'the average of those averages is ',g12.4)
c
c     .. plot contour map only if nuxz > .1
c
      if (tbar.gt..1) then
       head = '   nu at y = '//locate
       write (ireport,2150) head
       call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
      endif
  200 continue
c
c     ... z = 0 and z = arz walls ...
c
      do 300 k=1,nz,nzm1
c
      if(k .eq. 1) then
      locate = ' 0 '
      elseif(k .eq. nz) then
      locate = 'arz'
      endif
c
c     .. calculate local Nusselt numbers
c
      if(k .eq. 1) then
       do 310 i=1, nx
          do 310 j=1, ny
             aux2d(i,j) = -czf(1,1)*theta(i,j,1)
     +                    -czf(1,2)*theta(i,j,2)
     +                    -czf(1,3)*theta(i,j,3)
  310   continue
c
      elseif(k .eq. nz) then
c
       do 320 i=1, nx
          do 320 j=1, ny
             aux2d(i,j) = czf(2,1)*theta(i,j,nz)
     +                   +czf(2,2)*theta(i,j,nzm1)
     +                   +czf(2,3)*theta(i,j,nzm2)
  320   continue
      endif
c
      head = '   local nu    '
      call mprtz(aux2d,head,1)
c
c     .. calculate the y-averaged Nusselt numbers, nuy(x)
c
      do 332 i=1,nx
         do 331 j=1,ny
            tint(j) = aux2d(i,j)
  331    continue
      call simpsn (tint,ny,hy,tt(i))
  332 continue
c
      write (ireport,2310) locate, (i, tt(i), i=1,nx)
 2310 format(//,' wall z = ',a3,':    average Nusselt number along',
     +        'lines parallel to the z-axis:-',//,19x,'i',9x,'nu',//,
     +        (17x,i3,6x,1p,g12.4))
c
c     . calculate the average of the y-averaged Nusselt numbers, nuyx
c
      call simpsn (tt,nx,hx,tbar)
c
      write (ireport,2320) tbar
 2320 format(//,15x,'the average of those averages is ',1p,g12.4)
c
c     .. calculate the x-averaged Nusselt numbers, nux(y)
c
      do 342 j=1,ny
         do 341 i=1,nx
            tint(i) = aux2d(i,j)
  341    continue
      call simpsn (tint,nx,hx,tt(j))
  342 continue
c
      write (ireport,2330) (j, tt(j), j=1,ny)
 2330 format(//15x,'average Nusselt number along lines parallel',
     +' to the x-axis:-'//19x,'j',9x,'nu'//(17x,i3,6x,g12.4))
c
c     . calculate the average of the x-averaged Nusselt numbers, nuxy
c
      call simpsn (tt,ny,hy,tbar)
c
      write (ireport,2340) tbar
 2340 format(//15x,'the average of those averages is ',g12.4)
c
c     .. plot contour map only if nuxy > .1
c
      if (tbar.gt..1) then
```

```
      head = '  nu at z = '//locate
      write (ireport,2150) head
      call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
     endif
 300 continue
c
     return
     end

     subroutine inital
c
c    -----------------------------------------------------------------
c   |                                                                 |
c   |      *** initialise velocity, vector potential,                |
c   |                 vorticity and temperature fields ***           |
c   |                                                                 |
c    -----------------------------------------------------------------
c
     parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
     logical  reed,rite,strt,dist,heat,nuss,tran
     logical  ixl,ixh,iyl,iyh,izl,izh
c
     common  /fileno/inparam,irg,ireport,itin,itout
     common  /logc/reed,rite,strt,dist,heat,nuss,tran
     common  /logtbc/ixl,ixh,iyl,iyh,izl,izh
     common  /intcom/itercmpo,norun,mainiter
     common  /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
    +             ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
    +             zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
     common  /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
    +             iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
    +             izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
     common  /velbc/xlu,xlv,xlw,xhu,xhv,xhw,
    +              ylu,ylv,ylw,yhu,yhv,yhw,
    +              zlu,zlv,zlw,zhu,zhv,zhw
     common  /dt/dt
     common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
     common  /h/hx(max),hy(max),hz(max),
    +            hxold(max),hyold(max),hzold(max)
     common  /axis/x(max),y(max),z(max)
     common  /vel/u(nxmax,nymax,nzmax),
    +             v(nxmax,nymax,nzmax),
    +             w(nxmax,nymax,nzmax)
     common  /t/theta(nxmax,nymax,nzmax)
     common  /p1/p1(nxmax,nymax,nzmax)
     common  /p2/p2(nxmax,nymax,nzmax)
     common  /p3/p3(nxmax,nymax,nzmax)
     common  /p1old/p1old(nxmax,nymax,nzmax)
     common  /p2old/p2old(nxmax,nymax,nzmax)
     common  /p3old/p3old(nxmax,nymax,nzmax)
     common  /told/told(nxmax,nymax,nzmax)
     common  /tstrt/tstrt
c
c    ... initialize u, v, w, p1, p2, p3, if rquired ...
c
     if (reed) then
c
c    .. set velocity boundary conditions to zero
c
c
c      . on x = 0.0 and x = arx walls
c
     do 10 j=1, ny
        do 10 k=1, nz
             u( 1,j,k) = 0.0
             u(nx,j,k) = 0.0
             v( 1,j,k) = 0.0
             v(nx,j,k) = 0.0
             w( 1,j,k) = 0.0
             w(nx,j,k) = 0.0
  10  continue
c
c
c      . on y = 0.0 and y = ary walls
c
     do 20 i=1, nx
        do 20 k=1, nz
             u(i, 1,k) = 0.0
             u(i,ny,k) = 0.0
             v(i, 1,k) = 0.0
             v(i,ny,k) = 0.0
             w(i, 1,k) = 0.0
             w(i,ny,k) = 0.0
  20  continue
c
c      . on z = 0.0 and z = arz walls
c
     do 30 i=1, nx
        do 30 j=1, ny
             u(i,j, 1) = 0.0
             u(i,j,nz) = 0.0
             v(i,j, 1) = 0.0
             v(i,j,nz) = 0.0
             w(i,j, 1) = 0.0
             w(i,j,nz) = 0.0
  30  continue
     else
c
c    .. initialise all fields to zero
c
      write(ireport,2010)
2010   format(///,13x,'*** fields have been initialised ***')
c
     do 40 i=1, nx
        do 40 j=1, ny
           do 40 k=1, nz
             u(i,j,k)      = 0.0
             v(i,j,k)      = 0.0
             w(i,j,k)      = 0.0
```

```
c
            p1(i,j,k)     = 0.0
            p2(i,j,k)     = 0.0
            p3(i,j,k)     = 0.0
            p1old(i,j,k)  = 1.0
            p2old(i,j,k)  = 1.0
            p3old(i,j,k)  = 1.0
c
            theta(i,j,k)  = tstrt
            told(i,j,k)   = 3.0
 40     continue
c
      endif
c
c     ... set temperature boundary conditions (if required) ...
c
      if(heat) then
c
c       .. isothermal condition on x = 0.0 wall
c
       if (ixl) then
          do 90 j=ixlyl, ixlyh
              do 90 k=ixlzl, ixlzh
                   theta(1,j,k) = xlci
 90        continue
       endif
c
c       .. isothermal condition on x = arx wall
c
       if (ixh) then
          do 100 j=ixhyl, ixhyh
              do 100 k=ixhzl, ixhzh
                   theta(nx,j,k) = xhci
 100       continue
       endif
c
c       .. isothermal condition on y = 0.0 wall
c
       if (iyl) then
          do 110 i=iylxl, iylxh
              do 110 k=iylzl, iylzh
                   theta(i,1,k) = ylci
 110       continue
       endif
c
c       .. isothermal condition on y = ary wall
c
       if (iyh) then
          do 120 i=iyhxl, iyhxh
              do 120 k=iyhzl, iyhzh
                   theta(i,ny,k) = yhci
 120       continue
       endif
c
c       .. isothermal condition on z = 0.0 wall
c
       if (izl) then
          do 130 i=izlxl, izlxh
              do 130 j=izlyl, izlyh
                   theta(i,j,1) = zlci
 130       continue
       endif
c
c       .. isothermal condition on z = arz wall
c
       if (izh) then
          do 140 i=izhxl, izhxh
              do 140 j=izhyl, izhyh
                   theta(i,j,nz) = zhci
 140       continue
       endif
c
      else
c
c       .. no heating at all
c
       do 150 i=1, nx
          do 150 j=1, ny
              do 150 k=1, nz
               theta(i,j,k) = 0.
 150   continue
      endif
c
      return
      end

      subroutine interp(nx,ny,nz,nxnew,nynew,nznew,hxo,hyo,hzo,
     +                  hxn,hyn,hzn,t)
c
c     --------------------------------------------------------------------
c    |                                                                    |
c    |  *** interpolate t from (nx,nx,nz) to (nxnew,nynew,nznew) ***      |
c    |                                                                    |
c     --------------------------------------------------------------------
c
c--------------------------------------------------------------------------
c   * subroutine inter calculates the velocity components and            |
c     temperature at the mesh points corresponding to the new mesh       |
c     sizes. the method used is linear interpolation of the velocity     |
c     components and temperature corresponding to the old mesh sizes     |
c                                                                         |
c   * please see notes on velvec for method of interpolation             |
c--------------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      real     intt1, intt2, intt3, intt4, intt5, intt6
c
      dimension xo(max),yo(max),zo(max),xn(max),yn(max),zn(max),
     +          hxo(max),hyo(max),hzo(max),hxn(max),hyn(max),hzn(max),
```

```
     +            t(nxmax,nymax,nzmax),aux(nxmax,nymax,nzmax)
c
c     ...subroutine inter begins...
c
      nxnew1 = nxnew - 1
      nynew1 = nynew - 1
      nznew1 = nznew - 1
c
      xo(1) = 0.0
      yo(1) = 0.0
      zo(1) = 0.0
c
      do 15 i=1, nx-1
       xo(i+1) = xo(i)+hxo(i)
   15 continue
c
      do 25 j=1, ny-1
       yo(j+1) = yo(j)+hyo(j)
   25 continue
c
      do 35 k=1, nz-1
       zo(k+1) = zo(k)+hzo(k)
   35 continue
c
      xn(1) = 0.0
      yn(1) = 0.0
      zn(1) = 0.0
c
      do 45 i=1, nxnew-1
       xn(i+1) = xn(i)+hxn(i)
   45 continue
c
      do 55 j=1, nynew-1
       yn(j+1) = yn(j)+hyn(j)
   55 continue
c
      do 65 k=1, nznew-1
       zn(k+1) = zn(k)+hzn(k)
   65 continue
c
c------------------------------------------------------------------
c     calculate values of t at the planes k=1 and k=1            |
c------------------------------------------------------------------
c
      do 70 i=1, nxnew
       do 70 j=1, nynew
          xnew = xn(i)
          ynew = yn(j)
c
          call search(xnew,xo,io,nx)
          call search(ynew,yo,jo,ny)
c
          ratiox = (xnew-xo(io)) / (-xo(io)+xo(io+1))
          ratioy = (ynew-yo(jo)) / (-yo(jo)+yo(jo+1))
c
          intt1 = ((t(io+1,jo+1,1) - t(io+1,jo,1)) * ratioy)
     +                                         + t(io+1,jo,1)
          intt2 = ((t(io,jo+1,1) - t(io,jo,1)) * ratioy) +
     +              t(io,jo,1)
          aux(i,j,1) = ((intt1 - intt2) * ratiox) + intt2
c
          intt1 = ((t(io+1,jo+1,nz) - t(io+1,jo,nz)) * ratioy)
     +                                         + t(io+1,jo,nz)
          intt2 = ((t(io,jo+1,nz) - t(io,jo,nz)) * ratioy)
     +                                         + t(io,jo,nz)
          aux(i,j,nznew) = ((intt1 - intt2) * ratiox) + intt2
   70 continue
c
c------------------------------------------------------------------
c     calculate values of t at the planes i=1 and i=m            |
c------------------------------------------------------------------
c
      do 75 i=1, nxnew
       do 75 k=1, nznew
          xnew = xn(i)
          znew = zn(k)
c
          call search(xnew,xo,io,nx)
          call search(znew,zo,ko,nz)
c
          ratiox = (xnew-xo(io)) /  (-xo(io)+xo(io+1))
          ratioz = (znew-zo(ko)) /  (-zo(ko)+zo(ko+1))
c
          intt1 = ((t(io+1,1,ko+1) - t(io+1,1,ko)) * ratioz)
     +                                         + t(io+1,1,ko)
          intt2 = ((t(io,1,ko+1) - t(io,1,ko)) * ratioz) +
     +              t(io,1,ko)
          aux(i,1,k) = ((intt1 - intt2) * ratiox) + intt2
c
          intt1 = ((t(io+1,ny,ko+1) - t(io+1,ny,ko)) * ratioz)
     +                                         + t(io+1,ny,ko)
          intt2 = ((t(io,ny,ko+1) - t(io,ny,ko)) * ratioz)
     +                                         + t(io,ny,ko)
          aux(i,nynew,k) = ((intt1 - intt2) * ratioz) + intt2
   75 continue
c
c------------------------------------------------------------------
c     calculate values of t at the planes j=1 and j=n            |
c------------------------------------------------------------------
c
      do 80 j=1, nynew
       do 80 k=1, nznew
c
          ynew = yn(j)
          znew = zn(k)
c
          call search(ynew,yo,jo,ny)
          call search(znew,zo,ko,nz)
c
          ratioy = (ynew-yo(jo)) /  (-yo(jo)+yo(jo+1))
```

```
          ratioz = (znew-zo(ko)) /   (-zo(ko)+zo(ko+1))
c
          intt1 = ((t(1,jo+1,ko+1) - t(1,jo+1,ko)) * ratioz)
     +                                      + t(1,jo+1,ko)
          intt2 = ((t(1,jo,ko+1) - t(1,jo,ko)) * ratioz)
     +                                      + t(1,jo,ko)
          aux(1,j,k)= ((intt1 - intt2) * ratioy) + intt2
c
          intt1 = ((t(nx,jo+1,ko+1) - t(nx,jo+1,ko)) * ratioz)
     +                                      + t(nx,jo+1,ko)
          intt2 = ((t(nx,jo,ko+1) - t(nx,jo,ko)) * ratioz)
     +                                      + t(nx,jo,ko)
          aux(nxnew,j,k)= ((intt1 - intt2) * ratioy) + intt2
   80 continue
c
c-----------------------------------------------------------------------
c     calculate values of t for internal nodes                         |
c-----------------------------------------------------------------------
c
      do 85 i=2, nxnew1
       do 85 j=2, nynew1
          do 85 k=2, nznew1
c
c-----------------------------------------------------------------------
c             calculate position of new mesh point relative to old    |
c             mesh                                                     |
c-----------------------------------------------------------------------
c
             xnew = xn(i)
             ynew = yn(j)
             znew = zn(k)
c
             call search(xnew,xo,io,nx)
             call search(ynew,yo,jo,ny)
             call search(znew,zo,ko,nz)
c
             ratiox = (xnew-xo(io)) /   (-xo(io)+xo(io+1))
             ratioy = (ynew-yo(jo)) /   (-yo(jo)+yo(jo+1))
             ratioz = (znew-zo(ko)) /   (-zo(ko)+zo(ko+1))
c
c-----------------------------------------------------------------------
c            calculate t1 at the new mesh point corresponding to the |
c            new mesh sizes using values of t at the eight mesh points
c            corresponding to the old mesh sizes surrounding it      |
c-----------------------------------------------------------------------
c
      intt1 = ((t(io+1,jo+1,ko+1) - t(io+1,jo,ko+1)) * ratioy)
     +       + t(io+1,jo,ko+1)
      intt2 = ((t(io,jo+1,ko+1) - t(io,jo,ko+1)) * ratioy)
     +       + t(io,jo,ko+1)
      intt3 = ((t(io+1,jo+1,ko) - t(io+1,jo,ko)) * ratioy)
     +       + t(io+1,jo,ko)
      intt4 = ((t(io,jo+1,ko)-t(io,jo,ko)) * ratioy)
     +       + t(io,jo,ko)
             intt5 = ((intt3 - intt4) * ratiox) + intt4
c
                intt6 = ((intt1 - intt2) * ratiox) + intt2
c
                aux(i,j,k) = ((intt6 - intt5) * ratioz) + intt5
   85 continue
c
      do 90 i=1, nxnew
       do 90 j=1, nynew
          do 90 k=1, nznew
             t(i,j,k) = aux(i,j,k)
   90 continue
c
      return
      end

      subroutine iterat
c
c     -----------------------------------------------------------------
c    |                                                                 |
c    |                  *** solution control routine ***               |
c    |                                                                 |
c     -----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
      parameter(big=10000000000.0)

      logical  blup,cgd,trace
      logical  reed,rite,strt,dist,heat,nuss,tran
      integer  timestrt,timerun,timecal
c
      common  /itheta/itheta
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /logi/blup,cgd
      common  /logc/reed,rite,strt,dist,heat,nuss,tran
      common  /phierr/phierr
      common  /mapcom/mnx,mny,mnz
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common  /axis/x(max),y(max),z(max)
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +                cct,tmax,pdist,rtime
      common  /test/testp1,testp2,testp3,testt
      common  /intcom/itercmpo,norun,mainiter
      common  /numcom/numit
      common  /vel/u(nxmax,nymax,nzmax),
     +              v(nxmax,nymax,nzmax),
     +              w(nxmax,nymax,nzmax)
      common  /t/theta(nxmax,nymax,nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /dt/dt
      common  /alpfs/alpp,altp
      common  /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common  /epsonnu/epson,xll,annu,gravity,ddd  ,keff
```

```
      common /timestep/timestep
      common /iterpp/iterpp
      common /trace/trace
      common /sumdivt/sumdivt
      common /fig/fig
      common /timestrt/timestrt,timerun,timecal
c
c     ... initialise timer, counters and flags ...
c
      rtime  = 0.0
      numit  = 0
c
      fig=0.5
      trace= .true.
      cgd  = .false.
      blup = .false.
c
      testp2 = 1.0
      testp3 = 1.0
      sumdivt= 1.0
      if (heat) then
       testt = 1.0
      else
       testt = 0.0
      endif

      write(*,9000)
      write(*,9001)
      write(*,9002)
 9000  format(/,' estimated')
 9001  format(t1,'    running',t12,' cpu',t17,'percentage')
 9002  format(t1,'        time',t12,'time',t17,' completed',t28,
     + 'itheta',t33,'   test_p2',t44,'    test_p3',
     + t56,'    test_t',t69,'    t_err',/)
c
c     ---- iteration starts ----
c
c       advance temperture field
c

      do 200 itheta=1,mainiter
c
c       .. solution converged
c
      if( cgd ) then
         write(*,2031)  itheta
         write(ireport,2031) itheta
 2031     format(/'temp solution has converged. iteration number =',
     + i7)
         numit = itheta
         call ttime(.false., .true., .false.)
         rtime = real(timerun)
         return
      endif
c
c       .. maximum iteration limit reached
c
        if(itheta.ge.mainiter) then
           write(*,2051)
           write(ireport,2051)
 2051       format(/'-- maximum temperature iterations reached.--')
c
           call ttime(.false., .true., .false.)
           rtime = real(timerun)
           numit = itheta
           return
        endif
c
c        .. solution diverged
c
        if(blup)then
           write(*,2061) itheta
           write(ireport,2061) itheta
 2061       format(/'temp solution has diverged in',i4,' its. ',/)
           return
        endif
c
c        ... update each field ...
c
        do 4000 nump2p3=1,iterpp
         call p1solp
         call p2solp
         call p3solp
        if ((testp2.lt.ccp).and.(testp3.lt.ccp)) goto 4001
 4000    continue
 4001    continue
c
c        call velocity solution subroutines
c
           call veloc
           call vbc
c
c        call temperature solution subroutines
c
           call tsolp
           call tbc

        if(sumdivt.lt.cct)
     +    cgd= .true.
        if ( (testt.gt.big).or.(testp2.gt.big).or.
     +        (testp3.gt.big) ) blup = .true.
c
c     ... transient results ...
c
        call iterout(itheta)
 200     continue

        return
        end
```

```
      subroutine iterout(itheta)
c
c     ----------------------------------------------------------------
c     |                                                              |
c     |          *** output transient iterate results ***            |
c     |                                                              |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
      parameter(big=10000000000.0)
c
      integer pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11

      integer timestrt,timerun,timecal,drawstep
      character timestrg*11,datestrg*9
c
      logical trace,start,runng,runend
      logical blup,cgd
      logical ok
      logical reed,rite,strt,dist,heat,nuss,tran
      logical vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +        mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +        vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +        myp1,myp2,myp3,myt,myu,myv,myw,
     +        vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +        mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
c
      common /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +               mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +               vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +               myp1,myp2,myp3,myt,myu,myv,myw,
     +               vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +               mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common /timestrg/timestrg,datestrg
      common /timestrt/timestrt,timerun,timecal
      common /fileno/inparam,irg,ireport,itin,itout
      common /logi/blup,cgd
      common /logc/reed,rite,strt,dist,heat,nuss,tran
      common /phierr/phierr
      common /mapcom/mnx,mny,mnz
      common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common /axis/x(max),y(max),z(max)
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +               cct,tmax,pdist,rtime
      common /test/testp1,testp2,testp3,testt
      common /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
      common /intcom/itercmpo,norun,mainiter
      common /tbfdc/cxf(2,3),cyf(2,3),czf(2,3)
      common /numcom/numit
      common /vel/u(nxmax,nymax,nzmax),
     +            v(nxmax,nymax,nzmax),
     +            w(nxmax,nymax,nzmax)
      common /t/theta(nxmax,nymax,nzmax)
      common /p1/p1(nxmax,nymax,nzmax)
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
      common /dt/dt
      common /alpfs/alpp,altp
      common /porous/alphaf,alphap,cpf1,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,x11,annu,gravity,ddd ,keff
      common /pst/pstx,psty,pstz
      common /pstxyz/pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      common /timestep/timestep
      common /draw/drawstep
      common /trace/trace
      common /numshow/numshow
      common /iterpp/iterpp
      common /iplt/iplt,inu
      common /buffer/buff
      common /sumdivt/sumdivt
      common /horitemp/horitemp(3,20,41),horiu(3,20,41),
     +                 horiv(3,20,41),horiw(3,20,41)
      common /drwstp/drwstp1,drwstp2,drwstp3,drwstp4,drwstp5,
     +drwstp6,drwstp7,drwstp8,drwstp9,drwstp10,drwstp11,drwstp12
      common /fig/fig
      common /realtime/realtime
      dimension tint1(max),tint2(max),tt1(max),tt2(max)
      dimension tint3(max),tint4(max),tt3(max),tt4(max)
      dimension tint5(max),tint6(max),tt5(max),tt6(max)
      dimension nu_x_1(nymax,nzmax),nu_x_nx(nymax,nzmax)
      dimension nu_y_1(nxmax,nzmax),nu_y_ny(nxmax,nzmax)
      dimension nu_z_1(nxmax,nymax),nu_z_nz(nxmax,nymax)

      start=.false.
      runng=.true.
      runend=.false.
      call time(timestrg)
      call date(datestrg)
      call ttime(start,runng,runend)

      if (itheta.eq.1) then
      timecal=timerun
      endif
c
c     ...display running massege ...
c
      if ( ( ((itheta.ne.1).and.(itheta.lt.11)).or.(itheta.eq.100)
```

```
     +  .or.(itheta.eq.50).or.(itheta.eq.20)
     +  .or.(mod(itheta,numshow).eq.0)) .and. trace) then

        buffer=real(timerun-timecal)*real(mainiter)/
     +  (3600.*real(itheta-1))


        write(*,3000) buffer,real(timerun)/3600.,
     +  100.*real(itheta)/real(mainiter),itheta,testp2,testp3,
     + testt,sumdivt
 3000 format(2x,f5.1,'h.',t10,f5.2,t18,'   ',f5.1,t28,i5,
     + t33,e10.2,t45,e10.2,t57,e10.2,t69,e10.2)

c         buff=buffer-buff
c         if(abs(buff) .lt. 0.05) then
c         trace=.false.
c         else
c         buff=buffer
c         endif

        else if (mod(itheta,numshow).eq.0) then

        write(*,3001) real(timerun)/3600.,
     +100.*real(itheta)/real(mainiter),itheta,testp2,testp3,
     + testt,sumdivt
 3001 format(1x,t10,f5.2,t18,'   ',f5.1,t28,i5,
     + t33,e10.2,t45,e10.2,t57,e10.2,t69,e10.2)
        endif
c
c ... collect transient values of velocity ans temperature on central axes
c

        if ( (itheta-1) .eq. int(80.*3600./timestep)) then
        n=20
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(50.*3600./timestep)) then
        n=19
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(40.*3600./timestep)) then
        n=18
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(30.*3600./timestep)) then
        n=17
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(20.*3600./timestep)) then
        n=16
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(15.*3600./timestep)) then
        n=15
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(10.*3600./timestep)) then
        n=14
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(5.*3600./timestep))  then
        n=13
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(3600./timestep))      then
        n=12
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(3300./timestep))      then
        n=11
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(3000./timestep))      then
        n=10
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(2700./timestep))      then
        n=9
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(2400./timestep))      then
        n=8
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(2100./timestep))      then
        n=7
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(1800./timestep))      then
        n=6
        ok=.true.
        endif

        if ( (itheta-1) .eq. int(1500./timestep))      then
        n=5
        ok=.true.
```

```
      endif

      if ( (itheta-1) .eq. int(1200./timestep))       then
      n=4
      ok=.true.
      endif

      if ( (itheta-1) .eq. int(900./timestep))        then
      n=3
      ok=.true.
      endif

      if ( (itheta-1) .eq. int(600./timestep))        then
      n=2
      ok=.true.
      endif

      if ( (itheta-1) .eq. int(300./timestep))        then
      n=1
      ok=.true.
      endif

      if (ok) then

      do 1000 i=1, nx
      horiu(1,n,i)  =          u(i,(ny+1)/2,(nz+1)/2)
      horiv(1,n,i)  =          v(i,(ny+1)/2,(nz+1)/2)
      horiw(1,n,i)  =          w(i,(ny+1)/2,(nz+1)/2)
 1000 horitemp(1,n,i) = theta(i,(ny+1)/2,(nz+1)/2)

      do 1001 i=1, ny
      horiu(2,n,i)  =          u((nx+1)/2,i,(nz   /2)
      horiv(2,n,i)  =          v((nx+1)/2,i,(nz+1)/2)
      horiw(2,n,i)  =          w((nx+1)/2,i,(nz+1)/2)
 1001 horitemp(2,n,i) = theta((nx+1)/2,i,(nz+1)/2)

      do 1003 i=1, nz
      horiu(3,n,i)  =          u((nx+1)/2,(ny+1)/2,i)
      horiv(3,n,i)  =          v((nx+1)/2,(ny+1)/2,i)
      horiw(3,n,i)  =          w((nx+1)/2,(ny+1)/2,i)
 1003 horitemp(3,n,i) = theta((nx+1)/2,(ny+1)/2,i)
      ok=.false.
      endif
c
c  ... output first 5 hours contour maps and values of velocity,
c  temperature and vector potential in 30 minutes interval (if required)
c
      realtime=(itheta-1)*timestep/3600.
      if (( (itheta-1) .eq. int(fig*3600./timestep))
     +  .and. (realtime.le. 5.0) ) then
c        if (vxp1 .or. vxp2 .or. vxp3 .or. vxt  .or. vxu  .or.
c    +        vxv  .or. vxw  .or. mxp1 .or. mxp2 .or. mxp3 .or.
c    +      , mxt  .or. mxu  .or. mxv  .or. mxw  .or. vyp1 .or.
c    +        vyp2 .or. vyp3 .or. vyt  .or. vyu  .or. vyv  .or.
c    +        vyw  .or. myp1 .or. myp2 .or. myp3 .or. myt  .or.
c    +        myu  .or. myv  .or. myw  .or. vzp1 .or. vzp2 .or.
c    +        vzp3 .or. vzt  .or. vzu  .or. vzv  .or. vzw  .or.
c    +        mzp1 .or. mzp2 .or. mzp3 .or. mzt  .or. mzu  .or.
c    +       mzv  .or. mzw) then
        ireport = 46
        do 4000 iireport =1 ,24
        call output
        ireport=ireport +1
 4000    continue
        ireport =9
c         endif
        fig=fig+0.5
        endif
c
c  ... output transient velocity, temperatures and Nu numbers ...
c
        if (  (mod(itheta,drawstep).eq.0).or.
     +  (itheta.eq.drwstp1).or.(itheta.eq.drwstp2).or.
     +  (itheta.eq.drwstp3).or.(itheta.eq.drwstp4).or.
     +  (itheta.eq.drwstp5).or.(itheta.eq.drwstp6).or.
     +  (itheta.eq.drwstp7).or.(itheta.eq.drwstp8).or.
     +  (itheta.eq.drwstp9).or.(itheta.eq.drwstp10).or.
     +  (itheta.eq.drwstp11).or.(itheta.eq.drwstp12)   ) then

             write(45,8001)realtime,
     +          u(pstx1,psty1,pstz1)*1000.*alphap/xl1,
     +          u(pstx2,psty2,pstz2)*1000.*alphap/xl1,
     +          u(pstx3,psty3,pstz3)*1000.*alphap/xl1,
     +          u(pstx4,psty4,pstz4)*1000.*alphap/xl1,
     +          u(pstx5,psty5,pstz5)*1000.*alphap/xl1,
     +          u(pstx6,psty6,pstz6)*1000.*alphap/xl1,
     +          u(pstx7,psty7,pstz7)*1000.*alphap/xl1,
     +          u(pstx8,psty8,pstz8)*1000.*alphap/xl1,
     +          u(pstx9,psty9,pstz9)*1000.*alphap/xl1,
     +          u(pstx10,psty10,pstz10)*1000.*alphap/xl1,
     +          u(pstx11,psty11,pstz11)*1000.*alphap/xl1

              write(iplt,8000)realtime,theta(pstx1,psty1,pstz1)
     +        ,theta(pstx2,psty2,pstz2),theta(pstx3,psty3,pstz3)
     +        ,theta(pstx4,psty4,pstz4),theta(pstx5,psty5,pstz5)
     +        ,theta(pstx6,psty6,pstz6),theta(pstx7,psty7,pstz7)
     +        ,theta(pstx8,psty8,pstz8),theta(pstx9,psty9,pstz9)
     +        ,theta(pstx10,psty10,pstz10),theta(pstx11,psty11,pstz11)

 8000         format(1x,e9.3,11(1x,f8.6))
 8001         format(1x,e9.3,11(1x,f9.3))

c
c     .. calculate average Nusselt numbers on boundary ...
c


c
c     ... x = 0 and x = arx walls ...
c
```

```
c      .. calculate locate Nusselt numbers ...
c
        do 110 j=1, ny
           do 110 k=1, nz
               nu_x_1(j,k)  =  -cxf(1,1)*theta(1,j,k)
     +                         -cxf(1,2)*theta(2,j,k)
     +                         -cxf(1,3)*theta(3,j,k)
               nu_x_nx(j,k)  = +cxf(2,1)*theta(nx   ,j,k)
     +                         +cxf(2,2)*theta(nxm1,j,k)
     +                         +cxf(2,3)*theta(nxm2,j,k)
  110    continue
c
c
c      .. calculate the z-averaged Nusselt numbers, nuz(y)
c
       do 132 j=1,ny
          do 131 k=1,nz
              tint1(k)  = nu_x_1(j,k)
              tint2(k)  = nu_x_nx(j,k)
  131       continue

       call simpsn (tint1,nz,hz,tt1(j))
       call simpsn (tint2,nz,hz,tt2(j))
  132 continue
c
c      . calculate the average of the z-averaged Nusselt numbers, nuzy
c
       call simpsn (tt1,ny,hy,tbar1)
       call simpsn (tt2,nz,hz,tbar2)
c
c
c      ... y = 0 and y = ary walls ...
c
c      .. calculate local Nusselt numbers
c
        do 210 i=1, nx
           do 210 k=1, nz
               nu_y_1(i,k)  =  -cyf(1,1)*theta(i,1,k)
     +                         -cyf(1,2)*theta(i,2,k)
     +                         -cyf(1,3)*theta(i,3,k)
               nu_y_ny(i,k)  = cyf(2,1)*theta(i,ny   ,k)
     +                         +cyf(2,2)*theta(i,nym1,k)
     +                         +cyf(2,3)*theta(i,nym2,k)
  210    continue
c
c      .. calculate the z-averaged Nusselt numbers, nuz(x)
c
       do 232 i=1,nx
          do 231 k=1,nz
              tint3(k)  = nu_y_1(i,k)
              tint4(k)  = nu_y_ny(i,k)
  231       continue
       call simpsn (tint3,nz,hz,tt3(i))
       call simpsn (tint4,nz,hz,tt4(i))
  232 continue
c
c
c      . calculate the average of the z-averaged Nusselt numbers, nuzx
c
       call simpsn (tt3,nx,hx,tbar3)
       call simpsn (tt4,nz,hz,tbar4)
c
c
c      ... z = 0 and z = arz walls ...
c
c
c      .. calculate local Nusselt numbers
c
        do 310 i=1, nx
           do 310 j=1, ny
               nu_z_1(i,j)  =  -czf(1,1)*theta(i,j,1)
     +                         -czf(1,2)*theta(i,j,2)
     +                         -czf(1,3)*theta(i,j,3)
               nu_z_nz(i,j)  = czf(2,1)*theta(i,j,nz)
     +                         +czf(2,2)*theta(i,j,nzm1)
     +                         +czf(2,3)*theta(i,j,nzm2)
  310    continue
c
c      .. calculate the y-averaged Nusselt numbers, nuy(x)
c
       do 332 i=1,nx
          do 331 j=1,ny
              tint5(j)  = nu_z_1(i,j)
              tint6(j)  = nu_z_nz(i,j)
  331       continue
       call simpsn (tint5,ny,hy,tt5(i))
       call simpsn (tint6,ny,hy,tt6(i))
  332 continue
c
c      . calculate the average of the y-averaged Nusselt numbers, nuyx
c
       call simpsn (tt5,nx,hx,tbar5)
       call simpsn (tt6,ny,hy,tbar6)
c
       write(inu,400)  realtime,tbar1,tbar2,tbar3,tbar4,tbar5,tbar6
  400  format(1x,e15.9,6(1x,e9.3))

       endif
       return
       end

       subroutine location
c
c      ---------------------------------------------------------------
c     |                                                               |
c     |     *** showing the thermal death commodities locations ***   |
c     |                                                               |
c      ---------------------------------------------------------------
c
       parameter(nxmax=51,nymax=51,nzmax=51,max=51)
```

```
      character*1 pnt(nxmax,nymax,nzmax)
      character  filename*12,exprmnt*8
      common  /dthpnt/dthpnt(nxmax,nymax,nzmax),thmldthl,thmldthh
      common  /geom/nx,ny,nz,nxml,nyml,nzml,nxm2,nym2,nzm2
      common  /exprmnt/exprmnt

      filename=exprmnt // '.loc'
      open(unit=70    ,file=filename,status='unknown')

      if (nx .eq. 5 ) assign 5 to label
      if (nx .eq. 6 ) assign 6 to label
      if (nx .eq. 7 ) assign 7 to label
      if (nx .eq. 8 ) assign 8 to label
      if (nx .eq. 9 ) assign 9 to label
      if (nx .eq. 10) assign 10 to label
      if (nx .eq. 11) assign 11 to label
      if (nx .eq. 12) assign 12 to label
      if (nx .eq. 13) assign 13 to label
      if (nx .eq. 14) assign 14 to label
      if (nx .eq. 15) assign 15 to label
      if (nx .eq. 16) assign 16 to label
      if (nx .eq. 17) assign 17 to label
      if (nx .eq. 18) assign 18 to label
      if (nx .eq. 19) assign 19 to label
      if (nx .eq. 20) assign 20 to label
      if (nx .eq. 21) assign 21 to label
      if (nx .eq. 22) assign 22 to label
      if (nx .eq. 23) assign 23 to label
      if (nx .eq. 24) assign 24 to label
      if (nx .eq. 25) assign 25 to label
      if (nx .eq. 26) assign 26 to label
      if (nx .eq. 27) assign 27 to label
      if (nx .eq. 28) assign 28 to label
      if (nx .eq. 29) assign 29 to label
      if (nx .eq. 30) assign 30 to label
      if (nx .eq. 31) assign 31 to label

 5    format(1x,3a1)
 6    format(1x,4a1)
 7    format(1x,5a1)
 8    format(1x,6a1)
 9    format(1x,7a1)
10    format(1x,8a1)
11    format(1x,9a1)
12    format(1x,10a1)
13    format(1x,11a1)
14    format(1x,12a1)
15    format(1x,13a1)
16    format(1x,14a1)
17    format(1x,15a1)
18    format(1x,16a1)
19    format(1x,17a1)
20    format(1x,18a1)
21    format(1x,19a1)
22    format(1x,20a1)
23    format(1x,21a1)
24    format(1x,22a1)
25    format(1x,23a1)
26    format(1x,24a1)
27    format(1x,25a1)
28    format(1x,26a1)
29    format(1x,27a1)
30    format(1x,28a1)
31    format(1x,29a1)

      k=(nz+1)/2
      do 100 i=2, nxml
      do 100 j=2, nyml
      if (dthpnt(i,j,k) .eq. 0.0) then
      pnt(i,j,k)='*'
      else
      pnt(i,j,k)='.'
      endif
100   continue
      write(70,*) 'Locations of thermally dead produce on
     + x,y central plate.'
      write(70,label) ((pnt(i,j,k),j=2,nyml),i=2,nxml)
      write(70,*)

      j=(ny+1)/2
      do 200 i=2, nxml
      do 200 k=2, nzml
      if (dthpnt(i,j,k) .eq. 0.0) then
      pnt(i,j,k)='*'
      else
      pnt(i,j,k)='.'
      endif
200   continue
      write(70,*) 'Locations of thermally dead produce on
     + x,z central plate.'
      write(70,label) ((pnt(i,j,k),k=2,nzml),i=2,nxml)
      write(70,*)
      close(70)

      return
      end

      subroutine map(fun,fv,fh,nv,nh,mnv,mnh,fvlen,fhlen)
c
c     ------------------------------------------------------------------
c    |                                                                  |
c    |   *** print contour map of variables on the line printer ***    |
c    |                                                                  |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
      parameter(scale=61./49.)
c
      character star*1,chars(39)*1,table(132,132)*1
c
```

```
      common  /fileno/inparam,irg,ireport,itin,itout
c
      dimension  fun(max,max),  fv(max),  fh(max),  r(39)
c
      data chars/' ','0',' ',' ',' ','1',' ',' ',' ','2',' ',' ',
     +' ','3',' ',' ',' ','4',' ',' ',' ','5',' ',' ',
     +' ','6',' ',' ',' ','7',' ',' ',' ','8',' ',' ',
     +' ','9',' '/
     + star/'*'/
c
c     ... reset size according to scale ...
c
      mmnv = mnv
      mmnh = scale*mnh + 0.5
c
c     ... set contour levels ...
c
      pmax = -1.0e20
      pmin =  1.0e20
      do 10 i=1, nv
       do 10 j=1, nh
          if(fun(i,j).gt.pmax) pmax = fun(i,j)
          if(fun(i,j).lt.pmin) pmin = fun(i,j)
   10 continue
c
      pstep = (pmax-pmin)/38.0
      do 20 l=1, 39
       r(l) = pmin + pstep*(l-1)
   20 continue
c
c     ... write contour levels ...
c
      write(ireport,2020)  r(2),r(6),r(10),r(14),r(18),
     +    r(22),r(26),r(30),r(34),r(38)
c 2020 format(5x,'the contour levels are : ',(/,10g12.4))
 2020 format(5x,'the contour levels are : ',(/,10f8.2))
      write(ireport,2025)
 2025 format(1x)
c
c     ... set boundary of map ...
c
      do 30 mi=1, mmnv
       table(mi,1)    = star
       table(mi,mmnh) = star
   30 continue
c
      do 40 mj=1, mmnh
       table(1,mj)    = star
       table(mmnv,mj) = star
   40 continue
c
c     ... assign contour levels ...
c
      do 50 mi=2, mmnv-1
       posx = (real(mi)/real(mmnv))*fvlen
       call  search(posx,fv,i,nv)
       do 50 mj=2, mmnh-1
          posy = (real(mj)/real(mmnh))*fhlen
          call  search(posy,fh,j,nh)
c
          ratiox = (posx - fv(i))/(fv(i+1)-fv(i))
          ratioy = (posy - fh(j))/(fh(j+1)-fh(j))
c
          fint1 = fun(i,j) + ratioy*(fun(i,j+1)-fun(i,j))
          fint2 = fun(i+1,j) + ratioy*(fun(i+1,j+1)-fun(i+1,j))
c
          value = fint1 + ratiox*(fint2-fint1)
c
          k=2
   45      continue
          if (value.le.r(k)) then
              table(mi,mj) = chars(k)
          else
              k = k+1
              go to 45
          endif
   50 continue
c
c     ... print contour ...
c
      do 60 mi=1, mmnv
      write(ireport,2030)  (table(mi,mj),  mj=1,  mmnh)
 2030 format(5x,128a1)
   60 continue
c
      return
      end
      subroutine  mapt(fun,fv,fh,nv,nh,mnv,mnh,fvlen,fhlen)
c
c     ------------------------------------------------------------------
c    |                                                                  |
c    |   *** print contour map of temperature on the line printer ***   |
c    |                                                                  |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
      parameter(scale=61./49.)
c
      character  star*1,chars(11)*1,table(132,132)*1
c
      common  /fileno/inparam,irg,ireport,itin,itout
c
      dimension  fun(max,max),  fv(max),  fh(max),  r(11)
c
      data chars/' ','1',' ','3',' ','5',' ','7',' ','9',' '/
     +    star/'*'/
c
c     ... reset size according to scale ...
c
      mmnv = mnv
```

```
      mmnh = scale*mnh + 0.5
c
c     ... set contour levels ...
c
      pmax = -1.0e20
      pmin =  1.0e20
      do 10 i=1, nv
       do 10 j=1, nh
          if(fun(i,j).gt.pmax) pmax = fun(i,j)
          if(fun(i,j).lt.pmin) pmin = fun(i,j)
   10 continue
c
      pstep = (pmax-pmin)/10.0
      do 20 l=1, 11
       r(l) = pmin + pstep*(l-1)
   20 continue
c
c     ... write contour levels ...
c
      write(ireport,2020) r
 2020 format(//,5x,'the contour levels are : ',/,(/,11g12.4))
      write(ireport,2025)
 2025 format(1x)
c
c     ... set boundary of map ...
c
      do 30 mi=1, mmnv
       table(mi,1)    = star
       table(mi,mmnh) = star
   30 continue
c
      do 40 mj=1, mmnh
       table(1,mj)    = star
       table(mmnv,mj) = star
   40 continue
c
c     ... assign contour levels ...
c
      do 50 mi=2, mmnv-1
       posx = (real(mi)/real(mmnv))*fvlen
       call search(posx,fv,i,nv)
       do 50 mj=2, mmnh-1
          posy = (real(mj)/real(mmnh))*fhlen
          call search(posy,fh,j,nh)
c
          ratiox = (posx - fv(i))/(fv(i+1)-fv(i))
          ratioy = (posy - fh(j))/(fh(j+1)-fh(j))
c
          fint1 = fun(i,j) + ratioy*(fun(i,j+1)-fun(i,j))
          fint2 = fun(i+1,j) + ratioy*(fun(i+1,j+1)-fun(i+1,j))
c
          value = fint1 + ratiox*(fint2-fint1)
c
          k=2
   45     continue
          if (value.le.r(k)) then
             table(mi,mj) = chars(k)
          else
             k = k+1
             go to 45
          endif
   50 continue
c
c     ... print contour ...
c
      do 60 mi=1, mmnv
       write(19,2030) (table(mi,mj), mj=1, mmnh)
 2030 format(5x,128a1)
   60 continue
c
      return
      end

      subroutine maxmin(fv,head)
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |  *** locate minimum & maximum values of fv and positions ***   |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character head*15
c
      common /fileno/inparam,irg,ireport,itin,itout
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /axis/x(max),y(max),z(max)
c
      dimension fv(nxmax,nymax,nzmax)
      data imin,jmin,kmin,imax,jmax,kmax/0,0,0,0,0,0/
c
      fvmin= 1.0e+20
      fvmax=-1.0e+20
      do 10 k=1,nz
       do 10 j=1,ny
          do 10 i=1,nx
             if(fv(i,j,k).lt.fvmin) then
              fvmin = fv(i,j,k)
              imin = i
              jmin = j
              kmin = k
             endif
            if(fv(i,j,k).gt.fvmax) then
              fvmax=fv(i,j,k)
              imax=i
              jmax=j
              kmax=k
            endif
   10 continue
```

```
c
      write(ireport,2010)  head,fvmin,imin,jmin,kmin,x(imin),
     + y(jmin),z(kmin),head,fvmax,imax,jmax,kmax,
     + x(imax),y(jmax),z(kmax)
 2010 format( /,1x,'minimum value of ',a15,' is ',1pg11.4,
     +         ' at mesh point (',i3,',',i3,',',i3,')',
     +        ' ie. position (x,y,z) = (',
     +            0p,f5.2,',',f5.2,',',f5.2,')' ,
     +          /,1x,'maximum value of ',a15,' is ',1pg11.4,
     +         ' at mesh point (',i3,',',i3,',',i3,')',
     +        ' ie. position (x,y,z) = (',
     +            0p,f5.2,',',f5.2,',',f5.2,')')
c
      return
      end
      subroutine maxval(fv,head)
c
c     -------------------------------------------------------------
c    |                                                             |
c    |      *** locate maximum value of fv and its position ***    |
c    |                                                             |
c     -------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character  head*15
c
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /axis/x(max),y(max),z(max)
c
      dimension  fv(nxmax,nymax,nzmax)
c
      fvmax=0.
      do 10  k=1,nz
       do 10  j=1,ny
          do 10  i=1,nx
                 if(  fvmax  .lt.  abs(fv(i,j,k))  )then
                  fvmax  =  abs(fv(i,j,k))
                  ipos  =  i
                  jpos  =  j
                  kpos  =  k
                 endif
   10  continue
c
      write(ireport,2010) head,fvmax,ipos,jpos,kpos,
     + x(ipos),y(jpos),z(kpos)
 2010 format(//,1x,'maximum (abs) value of ',a15,' is ',1pg11.4,
     +         ' at mesh point (',i3,',',i3,',',i3,')',
     +        ' ie. position (x,y,z) = (',
     +            0p,f5.2,',',f5.2,',',f5.2,')')
c
      return
      end
      subroutine mprtx(f,head,ipos)
c
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |  *** prints arrays in blocks of 11 columns for y-z planes ***  |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character*15  head
      character  xc*1,yc*1,zc*1,vpform*1
c
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /intcom/itercmpo,norun,mainiter
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
      common  /axis/x(max),y(max),z(max)
      common  /vmpfrm/xc,yc,zc,vpform
c
      dimension  f(max,max)


c
c     ... print heading ...
c
      if(vpform .eq. 'c' ) then
       write(ireport,2010)  head,x(ipos)
 2010   format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': x = ',
     +             f5.2,/)
      else
       write(ireport,2011)  head,ipos
 2011   format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': i = ',i3,/)
      endif
c
c     ... set constants to print arrays in blocks of 11 ...
c
      jblock=ny/11.
      if(jblock*11  .ne.  ny)  jblock=jblock+1
c
c     ... print array ...
c
      do 10  kount=1,jblock
       jstart=(kount-1)*11+1
       jstop=jstart+10
       if(jblock.eq.kount)  jstop=ny
c
       if(vpform  .eq.  'c'  ) then
          write(ireport,2020)  (y(j),j=jstart,jstop)
 2020      format(//,5x,'  y  =  ',2x,f6.3,10(5x,f6.3)  )
       else
          write(ireport,2021)  (j,j=jstart,jstop)
 2021      format(//,5x,'  j  =  ',i4,10(7x,i4))
       endif
```

```fortran
      write(ireport,2030)
 2030  format(1x)
      if(vpform .eq. 'c' ) then
        do 11 kk=1,nz
            write(ireport,2040)  z(kk),(f(kk,j),j=jstart,jstop)
 2040        format(' z=',f6.3,2x,1p,11g11.3)
   11     continue
      else
        do 12 kk=1,nz
            write(ireport,2041)  kk,(f(kk,j),j=jstart,jstop)
 2041        format(' k=',i3,2x,1p,11g11.3)
   12     continue
      endif
   10 continue
c
       return
       end
       subroutine mprty(f,head,jpos)
c
c      ------------------------------------------------------------------
c     |                                                                  |
c     |  *** prints arrays in blocks of 11 columns for x-z planes ***   |
c     |                                                                  |
c      ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character*15 head
      character xc*1,yc*1,zc*1,vpform*1
c
      common  /vmpfrm/xc,yc,zc,vpform
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /intcom/itercmpo,norun,mainiter
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +           hxold(max),hyold(max),hzold(max)
      common  /axis/x(max),y(max),z(max)
c
      dimension f(max,max)
c
c    ... print heading ...
c
      if(vpform .eq. 'c' ) then
       write(ireport,2010) head,y(jpos)
 2010   format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': y = ',
     +            f5.2,/)
      else
       write(ireport,2011) head,jpos
 2011   format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': j = ',i3,/)
      endif
c
c    ... set constants to print arrays in blocks of 11 ...
c
      kblock=nz/11.
      if(kblock*11 .ne. nz) kblock=kblock+1
c
c    ... print array ...
c
      do 10 kount=1,kblock
       kstart=(kount-1)*11+1
       kstop=kstart+10
       if(kblock.eq.kount) kstop=nz
c
       if(vpform .eq. 'c' ) then
           write(ireport,2020)  (z(k),k=kstart,kstop)
 2020       format(//,5x,' z = ',2x,f6.3,10(5x,f6.3) )
       else
           write(ireport,2021)  (k,k=kstart,kstop)
 2021       format(//,5x,' k = ',i4,10(7x,i4))
       endif
       write(ireport,2030)
 2030   format(1x)
       if(vpform .eq. 'c' ) then
         do 11 ii=1,nx
             write(ireport,2040)  x(ii),(f(ii,k),k=kstart,kstop)
 2040         format(' x=',f6.3,2x,1p,11g11.3)
   11      continue
       else
         do 12 ii=1,nx
             write(ireport,2041)  ii,(f(ii,k),k=kstart,kstop)
 2041         format(' i=',i3,2x,1p,11g11.3)
   12      continue
       endif
   10 continue
c
       return
       end
       subroutine mprtz(f,head,kpos)
c
c      ------------------------------------------------------------------
c     |                                                                  |
c     |  *** prints arrays in blocks of 11 columns for x-y planes ***   |
c     |                                                                  |
c      ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character*15 head
      character xc*1,yc*1,zc*1,vpform*1
c
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /intcom/itercmpo,norun,mainiter
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +           hxold(max),hyold(max),hzold(max)
      common  /axis/x(max),y(max),z(max)
      common  /vmpfrm/xc,yc,zc,vpform
c
      dimension f(max,max)
c
```

```
c       ... print heading ...
c
        if(vpform .eq. 'c' ) then
         write(ireport,2010) head,z(kpos)
 2010     format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': z = ',
     +            f5.2,/)
        else
         write(ireport,2011) head,kpos
 2011     format(1h1,20x,5('*'),1x,a15,' field ',5('*'),3x,': k = ',i3,/)
        endif
c
c       ... set constants to print arrays in blocks of 11 ...
c
        jblock=ny/11.
        if(jblock*11 .ne. ny) jblock=jblock+1
c
c       ... print array ...
c
        do 10 kount=1,jblock
         jstart=(kount-1)*11+1
         jstop=jstart+10
         if(jblock.eq.kount) jstop=ny
c
         if(vpform .eq. 'c' ) then
            write(ireport,2020) (y(j),j=jstart,jstop)
 2020        format(//,5x,' y = ',2x,f6.3,10(5x,f6.3) )
         else
            write(ireport,2021) (j,j=jstart,jstop)
 2021        format(//,5x,' j = ',i4,10(7x,i4))
         endif
         write(ireport,2030)
 2030     format(1x)
         if(vpform .eq. 'c' ) then
           do 11 ii=1,nx
              write(ireport,2040) x(ii),(f(ii,j),j=jstart,jstop)
 2040          format(' x=',f6.3,2x,1p,11g11.3)
   11        continue
         else
           do 12 ii=1,nx
              write(ireport,2041) ii,(f(ii,j),j=jstart,jstop)
 2041          format(' i=',i3,2x,1p,11g11.3)
   12        continue
         endif
   10   continue
c
        return
        end

        subroutine outfiles
c
c       ------------------------------------------------------------
c      |                                                            |
c      |              ***  creating output files  ***               |
c      |                                                            |
c       ------------------------------------------------------------
c
        character exprmnt*8,filename*12
        logical vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +          mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +          vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +          myp1,myp2,myp3,myt,myu,myv,myw,
     +          vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +          mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
        common /exprmnt/exprmnt
        common /iplt/iplt,inu
        common /fileno/inparam,irg,ireport,itin,itout
        common /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +                 mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +                 vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +                 myp1,myp2,myp3,myt,myu,myv,myw,
     +                 vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +                 mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
c
c       ... set logical file numbers ...
c
        inu=31
        iplt  = 30
        ireport   = 9
        itin = 11
        itout= 12
c
c       ... open output files ...
c
        filename=exprmnt // '.nu'
        open(unit=inu    ,file=filename,status='unknown')
        write(inu,*) 'Transient Nu numbers.'
        filename=exprmnt // '.plt'
        open(unit=iplt    ,file=filename,status='unknown')
        write(iplt,*) 'Transient temperatures at select points.'
        filename=exprmnt // '.prt'
        open(unit=ireport    ,file=filename,status='unknown')
        filename=exprmnt // '.plu'
        open(unit=45    ,file=filename,status='unknown')
        write(45,*) 'Velocity distribution at select points.'

        if (mxt.or.vxt) then
        filename=exprmnt // '.tx'
        open(unit=46    ,file=filename,status='unknown')
        write(46,*)'Contour maps of temperature, x=constant planes.'
        close(46)
        endif
        if (myt.or.vyt) then
        filename=exprmnt // '.ty'
        open(unit=47    ,file=filename,status='unknown')
        write(47,*)'Contour maps of temperature, y=constant planes.'
        close(47)
        endif
        if (mzt.or.vzt) then
        filename=exprmnt // '.tz'
        open(unit=48    ,file=filename,status='unknown')
```

```
      write(48,*)'Contour maps of temperature, z=constant planes.'
      close(48)
      endif
      if (mxu.or.vxu) then
      filename=exprmnt // '.uxd'
      open(unit=49    ,file=filename,status='unknown')
      write(49,*)'Contour maps of x component velocity, u, ',
     + 'x=constant planes.'
      close(49)
      endif
      if (myu.or.vyu) then
      filename=exprmnt // '.uyd'
      open(unit=50    ,file=filename,status='unknown')
      write(50,*)'Contour maps of x component velocity, u, ',
     + 'y=constant planes.'
      close(50)
      endif
      if (mzu.or.vzu) then
      filename=exprmnt // '.uzd'
      open(unit=51    ,file=filename,status='unknown')
      write(51,*)'Contour maps of x component velocity, u, ',
     + 'z=constant planes.'
      close(51)
      endif
      if (mxv.or.vxv) then
      filename=exprmnt // '.vxd'
      open(unit=52    ,file=filename,status='unknown')
      write(52,*)'Contour maps of y component velocity, v, ',
     + 'x=constant planes.'
      close(52)
      endif
      if (myv.or.vyv) then
      filename=exprmnt // '.vyd'
      open(unit=53    ,file=filename,status='unknown')
      write(53,*)'Contour maps of y component velocity, v, ',
     + 'y=constant planes.'
      close(53)
      endif
      if (mzv.or.vzv) then
      filename=exprmnt // '.vzd'
      open(unit=54    ,file=filename,status='unknown')
      write(54,*)'Contour maps of y component velocity, v, ',
     + 'z=constant planes.'
      close(54)
      endif
      if (mxw.or.vxw) then
      filename=exprmnt // '.wxd'
      open(unit=55    ,file=filename,status='unknown')
      write(55,*)'Contour maps of z component velocity, w, ',
     + 'x=constant planes.'
      close(55)
      endif
      if (myw.or.vyw) then
      filename=exprmnt // '.wyd'
      open(unit=56    ,file=filename,status='unknown')
      write(56,*)'Contour maps of z component velocity, w, ',
     + 'y=constant planes.'
      close(56)
      endif
      if (mzw.or.vzw) then
      filename=exprmnt // '.wzd'
      open(unit=57    ,file=filename,status='unknown')
      write(57,*)'Contour maps of z component velocity, w, ',
     + 'z=constant planes.'
      close(57)
      endif
      if (mxu.or.vxu) then
      filename=exprmnt // '.vwx'
      open(unit=58    ,file=filename,status='unknown')
      write(58,*)'Contour maps of square root of (v*v+w*w), ',
     + 'x=constant planes.'
      close(58)
      endif
      if (myv.or.vyv) then
      filename=exprmnt // '.uwy'
      open(unit=59    ,file=filename,status='unknown')
      write(59,*)'Contour maps of square root of (u*u+w*w), ',
     + 'y=constant planes.'
      close(59)
      endif
      if (mzw.or.vzw) then
      filename=exprmnt // '.uvz'
      open(unit=60    ,file=filename,status='unknown')
      write(60,*)'Contour maps of square root of (u*u+v*v), ',
     + 'z=constant planes.'
      close(60)
      endif
      if (mxp1.or.vxp1) then
      filename=exprmnt // '.p1x'
      open(unit=61    ,file=filename,status='unknown')
      write(61,*)'Contour maps of p1, x=constant planes.'
      close(61)
      endif
      if (myp1.or.vyp1) then
      filename=exprmnt // '.p1y'
      open(unit=62    ,file=filename,status='unknown')
      write(62,*)'Contour maps of p1, y=constant planes.'
      close(62)
      endif
      if (mzp1.or.vzp1) then
      filename=exprmnt // '.p1z'
      open(unit=63    ,file=filename,status='unknown')
      write(63,*)'Contour maps of p1, z=constant planes.'
      close(63)
      endif
      if (mxp2.or.vxp2) then
      filename=exprmnt // '.p2x'
      open(unit=64    ,file=filename,status='unknown')
      write(64,*)'Contour maps of p2, x=constant planes.'
      close(64)
```

```
      endif
      if (myp2.or.vyp2) then
      filename=exprmnt // '.p2y'
      open(unit=65    ,file=filename,status='unknown')
      write(65,*)'Contour maps of p2, y=constant planes.'
      close(65)
      endif
      if (mzp2.or.vzp2) then
      filename=exprmnt // '.p2z'
      open(unit=66    ,file=filename,status='unknown')
      write(66,*)'Contour maps of p2, z=constant planes.'
      close(66)
      endif
      if (mxp3.or.vxp3) then
      filename=exprmnt // '.p3x'
      open(unit=67    ,file=filename,status='unknown')
      write(67,*)'Contour maps of p3, x=constant planes.'
      close(67)
      endif
      if (myp3.or.vyp3) then
      filename=exprmnt // '.p3y'
      open(unit=68    ,file=filename,status='unknown')
      write(68,*)'Contour maps of p3, y=constant planes.'
      close(68)
      endif
      if (mzp3.or.vzp3) then
      filename=exprmnt // '.p3z'
      open(unit=69    ,file=filename,status='unknown')
      write(69,*)'Contour maps of p3, z=constant planes.'
      close(69)
      endif

      return
      end


      subroutine output
c
c     --------------------------------------------------------------
c    |                                                              |
c    |     *** prints theta, u, v, w, p1, p2, p3  ***               |
c    |                                                              |
c     --------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical  finish
      logical  reed,rite,strt,dist,heat,nuss,tran
      logical  vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +         mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +         vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +         myp1,myp2,myp3,myt,myu,myv,myw,
     +         vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +         mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
c
      character  exprmnt*8,filename*12
      character*25 head
      character  xc*1,yc*1,zc*1,vpform*1
c
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /logc/reed,rite,strt,dist,heat,nuss,tran
      common  /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +                 mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +                 vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +                 myp1,myp2,myp3,myt,myu,myv,myw,
     +                 vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +                 mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common  /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +                 cct,tmax,pdist,rtime
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +             hxold(max),hyold(max),hzold(max)
      common  /axis/x(max),y(max),z(max)
      common  /vmpfrm/xc,yc,zc,vpform
      common  /vmpnum/npx,npy,npz
      common  /vmploc/mx(nxmax),my(nymax),mz(nzmax)
      common  /vmppos/rmx(nxmax),rmy(nymax),rmz(nzmax)
      common  /mapcom/mnx,mny,mnz
      common  /vel/u(nxmax,nymax,nzmax),
     +             v(nxmax,nymax,nzmax),
     +             w(nxmax,nymax,nzmax)
      common  /t/theta(nxmax,nymax,nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common  /horitemp/horitemp(3,20,41),horiu(3,20,41),
     +                  horiv(3,20,41),horiw(3,20,41)
      common  /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common  /epsonnu/epson,xll,annu,gravity,ddd,keff
      common  /ccaa/cc0,dtemp,tcold,thot
      common  /finish/finish
      common  /realtime/realtime
      common  /exprmnt/exprmnt

      if (finish) then

      do 7005  i=1,3
      do 7005  j=1,20
        do 7005  k=1,41
         horiu(i,j,k)=1000.*horiu(i,j,k)*alphap/xll
         horiv(i,j,k)=1000.*horiv(i,j,k)*alphap/xll
         horiw(i,j,k)=1000.*horiw(i,j,k)*alphap/xll
7005  continue

      filename=exprmnt // '.txt'
      open(unit=33    ,file=filename,status='unknown')
      write(33,*)'Transient t distribution along x axis.'
      filename=exprmnt // '.xu'
      open(unit=36    ,file=filename,status='unknown')
```

```
      write(36,*)'Transient u distribution along x axis.'
      filename=exprmnt // '.xv'
      open(unit=37    ,file=filename,status='unknown')
      write(37,*)'Transient v distribution along x axis.'
      filename=exprmnt // '.xw'
      open(unit=38    ,file=filename,status='unknown')
      write(38,*)'Transient w distribution along x axis.'

      write(33,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(36,*)'x axis, u mm/s velocity distribution.'
      write(36,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(37,*)'x axis, v mm/s velocity distribution.'
      write(37,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(38,*)'x axis, w mm/s velocity distribution.'
      write(38,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      pstion=0.

      do 7000 i=1,nx

      write(33,8002) pstion, (horitemp(1,n,i),n=1,20)

      write(36,8001) pstion, (horiu(1,n,i), n=1,20)

      write(37,8001) pstion, (horiv(1,n,i), n=1,20)

      write(38,8001) pstion, (horiw(1,n,i), n=1,20)
      pstion=pstion+hx(i)
7000   continue
      close(33)
      close(36)
      close(37)
      close(38)

      filename=exprmnt // '.tyt'
      open(unit=34    ,file=filename,status='unknown')
      write(34,*)'Transient t distribution along y axis.'
      filename=exprmnt // '.yu'
      open(unit=39    ,file=filename,status='unknown')
      write(39,*)'Transient u distribution along y axis.'
      filename=exprmnt // '.yv'
      open(unit=40    ,file=filename,status='unknown')
      write(40,*)'Transient v distribution along y axis.'
      filename=exprmnt // '.yw'
      open(unit=41    ,file=filename,status='unknown')
      write(41,*)'Transient w distribution along y axis.'

      write(34,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(39,*)'y axis, u mm/s velocity distribution.'
      write(39,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(40,*)'y axis, v mm/s velocity distribution.'
      write(40,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(41,*)'y axis, w mm/s velocity distribution.'
      write(41,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      pstion=0.
      do 7001 i=1,ny

      write(34,8002) pstion,(horitemp(2,n,i),n=1,20)

      write(39,8001) pstion,(horiu(2,n,i),n=1,20)

      write(40,8001) pstion,(horiv(2,n,i),n=1,20)

      write(41,8001) pstion,(horiw(2,n,i),n=1,20)
      pstion=pstion+hy(i)
7001   continue
      close(34)
      close(39)
      close(40)
      close(41)

      filename=exprmnt // '.tzt'
      open(unit=35    ,file=filename,status='unknown')
      write(35,*)'Transient temperature distribution along z axis.'
      filename=exprmnt // '.zu'
      open(unit=42    ,file=filename,status='unknown')
      write(42,*)'Transient u distribution along z axis.'
      filename=exprmnt // '.zv'
      open(unit=43    ,file=filename,status='unknown')
      write(43,*)'Transient v distribution along z axis.'
      filename=exprmnt // '.zw'
      open(unit=44    ,file=filename,status='unknown')
      write(44,*)'Transient w distribution along z axis.'

      write(35,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'


      write(42,*)'z axis, u mm/s velocity distribution.'
      write(42,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'

      write(43,*)'z axis, v mm/s velocity distribution.'
      write(43,*)'at:  <1   1        5        10        15        20
     + 30       40       50       60       80 hours'
```

```
      write(44,*)'z axis, w mm/s velocity distribution.'
      write(44,*)'at:   <1  1        5       10       15       20
     +  30      40      50      60      80 hours'
      pstion =0.
      do 7002 i=1,nz

      write(35,8002)  pstion,(horitemp(3,n,i),n=1,20)

      write(42,8001)  pstion,(horiu(3,n,i),n=1,20)

      write(43,8001)  pstion,(horiv(3,n,i),n=1,20)

      write(44,8001)  pstion,(horiw(3,n,i),n=1,20)
      pstion=pstion+hz(i)
 7002  continue
 8001   format(1x,f5.3,20(1x,f8.3))
 8002   format(1x,f5.3,20(1x,f8.6))
      close(35)
      close(42)
      close(43)
      close(44)

      filename=exprmnt  //  '.dis'
      open(unit=32     ,file=filename,status='unknown')
      write(32,*)'Steady state temperature distribution in axis x,y,z'
      write(32,*)'     x         t(ix,my,mz)        y       t(mx,iy,mz)
     +    z      t(mx,my,iz)'
      mmx=(nx+1)/2
      mmy=(ny+1)/2
      mmz=(nz+1)/2
      mm=nx
      if ( mm .lt. ny) mm=ny
      if ( mm .lt. nz) mm=nz
      xpstion=0.
      ypstion=0.
      zpstion=0.
      do 1000 i=1,mm

      write(32,8000) xpstion,theta(i,mmy,mmz),
     +   ypstion, theta(mmx,i,mmz),
     +   zpstion, theta(mmx,mmy,i)
      xpstion=xpstion+hx(i)
      ypstion=ypstion+hy(i)
      zpstion=zpstion+hz(i)
 1000  continue
      close(32)
 8000         format(1x,3(f7.4,1x,f10.6,2x))

       endif

c
c     ... print variables in y-z planes (x=constant) ...
c

      if (npx.gt.0) then
c
c       .. temperature
c
       if ((ireport .eq. 46).and.(vxt.or.mxt)) then
      filename=exprmnt  //  '.tx'
      open(unit=46     ,file=filename,status='old',access='append')
       do 11 i=1, npx
          do 10 k=1, nz
              do 10 j=1, ny
               aux2d(k,j)  =   theta(mx(i),j,k)*dtemp+tcold
 10       continue
c
          head =  '   temperature   degree c'
          if (vxt) call mprtx(aux2d,head,mx(i))
          if (mxt) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
 2010         format(1h1,10x,5('*'),' contour map of ',a15,5('*'),3x,
     +                 ': i = ',i3,3x,'( x = ',f5.2,' )')
             call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
 11    continue
      close(46)
      endif
c
c       .. u-velocity
c
       if ((ireport .eq. 49).and.(vxu.or.mxu)) then
      filename=exprmnt  //  '.uxd'
      open(unit=49     ,file=filename,status='old',access='append')
       do 21 i=1, npx
          do 20 k=1, nz
              do 20 j=1, ny
               aux2d(k,j)  =u(mx(i),j,k)*1000.*alphap/xll
 20       continue
c
          head =  '    u-velocity   mm/s   '
          if (vxu) call mprtx(aux2d,head,mx(i))
          if (mxu) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
 21    continue
      close(49)
      endif
c
c       .. v and w velocity
c
       if ((ireport .eq. 58).and.(vxu.or.mxu)) then
      filename=exprmnt  //  '.vwx'
```

```
      open(unit=58    ,file=filename,status='old',access='append')
       do 531 i=1, npx
          do 530 k=1, nz
              do 530 j=1, ny
               aux2d(k,j) =  v(mx(i),j,k)* v(mx(i),j,k)+
     +    w(mx(i),j,k)* w(mx(i),j,k)
        aux2d(k,j) =sqrt(aux2d(k,j) )*1000.*alphap/xll
  530     continue
c
          head = 'v & w-velocity mm/s'
          if (vxu) call mprtx(aux2d,head,mx(i))
          if (mxu) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
  531  continue
      close(58)
      endif
c
c       .. v-velocity
c
       if ((ireport .eq. 52).and.(vxv.or.mxv)) then
      filename=exprmnt // '.vxd'
      open(unit=52    ,file=filename,status='old',access='append')
       do 31 i=1, npx
          do 30 k=1, nz
              do 30 j=1, ny
               aux2d(k,j) =  v(mx(i),j,k)*1000.*alphap/xll
   30     continue
c
          head = '   v-velocity  mm/s'
          if (vxv) call mprtx(aux2d,head,mx(i))
          if (mxv) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
   31  continue
      close(52)
      endif
c
c       .. w-velocity
c
       if ((ireport .eq. 55).and.(vxw.or.mxw)) then
      filename=exprmnt // '.wxd'
      open(unit=55    ,file=filename,status='old',access='append')
       do 41 i=1, npx
          do 40 k=1, nz
              do 40 j=1, ny
               aux2d(k,j) =  w(mx(i),j,k)*1000.*alphap/xll
   40     continue
c
          head = '   w-velocity mm/s '
          if (vxw) call mprtx(aux2d,head,mx(i))
          if (mxw) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
   41  continue
      close(55)
      endif
c
c       .. psi-1
c
       if ((ireport .eq. 61).and.(vxp1.or.mxp1)) then
      filename=exprmnt // '.p1x'
      open(unit=61    ,file=filename,status='old',access='append')
       do 51 i=1, npx
          do 50 k=1, nz
              do 50 j=1, ny
               aux2d(k,j) = p1(mx(i),j,k)
   50     continue
c
          head = '     psi-1      '
          if (vxp1) call mprtx(aux2d,head,mx(i))
          if (mxp1) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
   51  continue
      close(61)
      endif
c
c       .. psi-2
c
       if ((ireport .eq. 64).and.(vxp2.or.mxp2)) then
      filename=exprmnt // '.p2x'
      open(unit=64    ,file=filename,status='old',access='append')
       do 61 i=1, npx
          do 60 k=1, nz
              do 60 j=1, ny
               aux2d(k,j) = p2(mx(i),j,k)
   60     continue
c
          head = '     psi-2      '
          if (vxp2) call mprtx(aux2d,head,mx(i))
          if (mxp2) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
   61  continue
```

```
      close(64)
      endif
c
c       .. psi-3
c
       if ((ireport .eq. 67).and.(vxp3.or.mxp3)) then
      filename=exprmnt // '.p3x'
      open(unit=67    ,file=filename,status='old',access='append')
       do 71 i=1, npx
          do 70 k=1, nz
              do 70 j=1, ny
               aux2d(k,j) = p3(mx(i),j,k)
   70       continue
c
          head = '     psi-3      '
          if (vxp3) call mprtx(aux2d,head,mx(i))
          if (mxp3) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2010) head,mx(i),x(mx(i))
              call map(aux2d,z,y,nz,ny,mnz,mny,arz,ary)
          endif
   71   continue
c
      close(67)
      endif

      endif
c
c     ... print variables in x-z planes (y=constant) ...
c
      if (npy.gt.0) then
c
c       .. temperature
c
       if ((ireport .eq. 47).and.(vyt.or.myt)) then
      filename=exprmnt // '.ty'
      open(unit=47    ,file=filename,status='old',access='append')
       do 211 j=1, npy
          do 210 i=1, nx
              do 210 k=1, nz
               aux2d(i,k) =  theta(i,my(j),k)*dtemp+tcold
  210     continue
c
          head = '  temperature  degree c'
          if (vyt) call mprty(aux2d,head,my(j))
          if (myt) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
 2020         format(1h1,10x,5('*'),' contour map of ',a15,5('*'),3x,
     +               ': j = ',i3,3x,'( y = ',f5.2,' )')
             call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  211   continue

      close(47)
      endif
c
c       .. u-velocity
c
       if ((ireport .eq. 50).and.(vyu.or.myu)) then
      filename=exprmnt // '.uyd'
      open(unit=50    ,file=filename,status='old',access='append')
       do 221 j=1, npy
          do 220 i=1, nx
              do 220 k=1, nz
               aux2d(i,k) =  u(i,my(j),k)*1000.*alphap/xll
  220     continue
c
          head = '   u-velocity mm/s '
          if (vyu) call mprty(aux2d,head,my(j))
          if (myu) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  221   continue
      close(50)
      endif
c
c       .. v-velocity
c
       if ((ireport .eq. 53).and.(vyv.or.myv)) then
      filename=exprmnt // '.vyd'
      open(unit=53    ,file=filename,status='old',access='append')
       do 231 j=1, npy
          do 230 i=1, nx
              do 230 k=1, nz
               aux2d(i,k) =  v(i,my(j),k)*1000.*alphap/xll
  230     continue
c
          head = '   v-velocity mm/s '
          if (vyv) call mprty(aux2d,head,my(j))
          if (myv) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  231   continue
      close(53)
      endif
c
c       .. u and w -velocity
c
       if ((ireport .eq. 59).and.(vyv.or.myv)) then
      filename=exprmnt // '.uwy'
      open(unit=59    ,file=filename,status='old',access='append')
       do 631 j=1, npy
```

```
          do 630 i=1, nx
             do 630 k=1, nz
                aux2d(i,k) = u(i,my(j),k)* u(i,my(j),k)+
     + w(i,my(j),k)*w(i,my(j),k)
            aux2d(i,k)= sqrt(aux2d(i,k))*1000.*alphap/xll
  630       continue
c
          head = 'u & w velocity   mm/s'
          if (vyv) call mprty(aux2d,head,my(j))
          if (myv) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  631    continue
        close(59)
        endif
c
c        .. w-velocity
c
        if ((ireport .eq. 56).and.(vyw.or.myw)) then
       filename=exprmnt // '.wyd'
       open(unit=56    ,file=filename,status='old',access='append')
        do 241 j=1, npy
          do 240 i=1, nx
             do 240 k=1, nz
              aux2d(i,k) =  w(i,my(j),k)*1000.*alphap/xll
  240       continue
c
          head = '   w-velocity  mm/s'
          if (vyw) call mprty(aux2d,head,my(j))
          if (myw) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  241    continue
        close(56)
        endif
c
c        .. psi-1
c
        if ((ireport .eq. 62).and.(vyp1.or.myp1)) then
       filename=exprmnt // '.p1y'
       open(unit=62    ,file=filename,status='old',access='append')
        do 251 j=1, npy
          do 250 i=1, nx
             do 250 k=1, nz
              aux2d(i,k) = p1(i,my(j),k)
  250       continue
c
          head = '     psi-1     '
          if (vyp1) call mprty(aux2d,head,my(j))
          if (myp1) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  251    continue
        close(62)
        endif
c
c        .. psi-2
c
        if ((ireport .eq. 65).and.(vyp2.or.myp2)) then
       filename=exprmnt // '.p2y'
       open(unit=65    ,file=filename,status='old',access='append')
        do 261 j=1, npy
          do 260 i=1, nx
             do 260 k=1, nz
              aux2d(i,k) = p2(i,my(j),k)
  260       continue
c
          head = '     psi-2     '
          if (vyp2) call mprty(aux2d,head,my(j))
          if (myp2) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  261    continue
        close(65)
        endif
c
c        .. psi-3
c
        if ((ireport .eq. 68).and.(vyp3.or.myp3)) then
       filename=exprmnt // '.p3y'
       open(unit=68    ,file=filename,status='old',access='append')
        do 271 j=1, npy
          do 270 i=1, nx
             do 270 k=1, nz
              aux2d(i,k) = p3(i,my(j),k)
  270       continue
c
          head = '     psi-3     '
          if (vyp3) call mprty(aux2d,head,my(j))
          if (myp3) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2020) head,my(j),y(my(j))
              call map(aux2d,x,z,nx,nz,mnx,mnz,arx,arz)
          endif
  271    continue
        endif
c
```

```
      close(68)
      endif
c
c     ... print variables in x-y planes (z=constant) ...
c
      if (npz.gt.0) then
c
c       .. temperature
c
       if ((ireport .eq. 48).and.(vzt.or.mzt)) then
      filename=exprmnt // '.tz'
      open(unit=48    ,file=filename,status='old',access='append')
       do 411 k=1, npz
          do 410 i=1, nx
              do 410 j=1, ny
               aux2d(i,j) =  theta(i,j,mz(k))*dtemp+tcold
  410        continue
c
          head = '  temperature  degree c'
           if (vzt) call mprtz(aux2d,head,mz(k))
          if (mzt) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2030) head,mz(k),z(mz(k))
 2030         format(1h1,10x,5('*'),' contour map of ',a15,5('*'),3x,
     +               ': k = ',i3,3x,'( z = ',f5.2,' )')
              call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
          endif
  411   continue
       close(48)
       endif
c
c       .. u-velocity
c
       if ((ireport .eq. 51).and.(vzu.or.mzu)) then
      filename=exprmnt // '.uzd'
      open(unit=51    ,file=filename,status='old',access='append')
       do 421 k=1, npz
          do 420 i=1, nx
              do 420 j=1, ny
               aux2d(i,j) =  u(i,j,mz(k))*1000.*alphap/xll
  420        continue
c
          head = '   u-velocity  mm/s'
           if (vzu) call mprtz(aux2d,head,mz(k))
          if (mzu) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2030) head,mz(k),z(mz(k))
              call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
          endif
  421   continue
       close(51)
       endif
c
c       .. v-velocity
c
       if ((ireport .eq. 54).and.(vzv.or.mzv)) then
      filename=exprmnt // '.vzd'
      open(unit=54    ,file=filename,status='old',access='append')
       do 431 k=1, npz
          do 430 i=1, nx
              do 430 j=1, ny
               aux2d(i,j) =  v(i,j,mz(k))*1000.*alphap/xll
  430        continue
c
          head = '   v-velocity  mm/s'
           if (vzv) call mprtz(aux2d,head,mz(k))
          if (mzv) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2030) head,mz(k),z(mz(k))
              call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
          endif
  431   continue
       close(54)
       endif
c
c       .. w-velocity
c
       if ((ireport .eq. 57).and.(vzw.or.mzw)) then
      filename=exprmnt // '.wzd'
      open(unit=57    ,file=filename,status='old',access='append')
       do 441 k=1, npz
          do 440 i=1, nx
              do 440 j=1, ny
               aux2d(i,j) =  w(i,j,mz(k))*1000.*alphap/xll
  440        continue
c
          head = '   w-velocity  mm/s'
           if (vzw) call mprtz(aux2d,head,mz(k))
          if (mzw) then
             write(ireport,*)
             write(ireport,*)'realtime=',realtime
              write(ireport,2030) head,mz(k),z(mz(k))
              call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
          endif
  441   continue
       close(57)
       endif
c
c       .. u and v -velocity
c
       if ((ireport .eq. 60).and.(vzw.or.mzw)) then
      filename=exprmnt // '.uvz'
      open(unit=60    ,file=filename,status='old',access='append')
       do 741 k=1, npz
          do 740 i=1, nx
              do 740 j=1, ny
               aux2d(i,j) =  u(i,j,mz(k)) * u(i,j,mz(k))+
```

```
     +   v(i,j,mz(k))*v(i,j,mz(k))
            aux2d(i,j)= sqrt(aux2d(i,j))*1000.*alphap/xll
  740       continue
c
         head = 'u & v velocity    mm/s'
         if (vzw) call mprtz(aux2d,head,mz(k))
         if (mzw) then
            write(ireport,*)
            write(ireport,*)'realtime=',realtime
             write(ireport,2030) head,mz(k),z(mz(k))
             call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
         endif
  741    continue
        close(60)
        endif
c
c        .. psi-1
c
       if ((ireport .eq. 63).and.(vzp1.or.mzp1)) then
       filename=exprmnt // '.p1z'
       open(unit=63    ,file=filename,status='old',access='append')
        do 451 k=1, npz
           do 450 i=1, nx
               do 450 j=1, ny
                aux2d(i,j) = p1(i,j,mz(k))
  450       continue
c
         head = '      psi-1      '
         if (vzp1) call mprtz(aux2d,head,mz(k))
         if (mzp1) then
            write(ireport,*)
            write(ireport,*)'realtime=',realtime
             write(ireport,2030) head,mz(k),z(mz(k))
             call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
         endif
  451    continue
        close(63)
        endif
c
c        .. psi-2
c
       if ((ireport .eq. 66).and.(vzp2.or.mzp2)) then
       filename=exprmnt // '.p2z'
       open(unit=66    ,file=filename,status='old',access='append')
        do 461 k=1, npz
           do 460 i=1, nx
               do 460 j=1, ny
                aux2d(i,j) = p2(i,j,mz(k))
  460       continue
c
         head = '      psi-2      '
         if (vzp2) call mprtz(aux2d,head,mz(k))
         if (mzp2) then
            write(ireport,*)
            write(ireport,*)'realtime=',realtime
             write(ireport,2030) head,mz(k),z(mz(k))
             call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
         endif
  461    continue
        close(66)
        endif
c
c        .. psi-3
c
       if ((ireport .eq. 69).and.(vzp3.or.mzp3)) then
       filename=exprmnt // '.p3z'
       open(unit=69    ,file=filename,status='old',access='append')
        do 471 k=1, npz
           do 470 i=1, nx
               do 470 j=1, ny
                aux2d(i,j) = p3(i,j,mz(k))
  470       continue
c
         head = '      psi-3      '
         if (vzp3) call mprtz(aux2d,head,mz(k))
         if (mzp3) then
            write(ireport,*)
            write(ireport,*)'realtime=',realtime
             write(ireport,2030) head,mz(k),z(mz(k))
             call map(aux2d,x,y,nx,ny,mnx,mny,arx,ary)
         endif
  471    continue
        close(69)
        endif

      endif

      return
      end

      subroutine p1solp
c
c     ---------------------------------------------------------------------
c    |  *** solve for new x-component vector potential in porous ***   |
c    |
c     ---------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c

      dimension   a(max),b(max),c(max),d(max)
      common  /phierr/phierr
      common  /alpfs/alpp,altp
      common  /dt/dt
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
      common  /pconsp/cpp1
```

```
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +             cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +             cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +             cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /cxz/cxm1zm1(nxmax,nzmax),cxzm1(nxmax,nzmax),
     +         cxp1zm1(nxmax,nzmax),cxm1z(nxmax,nzmax),
     +         cxz(nxmax,nzmax),cxp1z(nxmax,nzmax),
     +         cxm1zp1(nxmax,nzmax),cxzp1(nxmax,nzmax),
     +         cxp1zp1(nxmax,nzmax)
      common  /cyz/cym1zm1(nymax,nzmax),cyzm1(nymax,nzmax),
     +         cyp1zm1(nymax,nzmax),cym1z(nymax,nzmax),
     +         cyz(nymax,nzmax),cyp1z(nymax,nzmax),
     +         cym1zp1(nymax,nzmax),cyzp1(nymax,nzmax),
     +         cyp1zp1(nymax,nzmax)
      common  /pbfdc/cx51,cx61,cx4h,cx5h,
     +             cy51,cy61,cy4h,cy5h,
     +             cz51,cz61,cz4h,cz5h
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /p1old/p1old(nxmax,nymax,nzmax)
      common  /tom/a,b,c,d
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common  /auxct/auxct(nxmax,nymax,nzmax)
      common  /t/theta(nxmax,nymax,nzmax)
      common  /zconsp/cczp4,cczp5,cczp6,cczp7
      common  /intcom/itercmpo,norun,mainiter
      common  /test/testp1,testp2,testp3,testt
      common  /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common  /niterp/niterp2,niterp3
      common  /np2p3/nnpmax,nnpmin
c
c     ... implicit in x ...
c
      do 700 iij=1,itercmpo
       do 100 j=2, nym1
        do 100 k=2, nzm1
           a(1) = 0.0
           b(1) = -cpp1*cx51 + 1.0
           c(1) = -cpp1*cx61*2.0
           d(1) = p1(2,j,k)*2.0*cx61
     +          + p1(1,j,k)*(cx51 + cy5(j) + darcy1*cz5(k))
     +          + p1(1,j-1,k)*cy4(j) + p1(1,j+1,k)*cy6(j)
     +          + darcy1*(p1(1,j,k-1)*cz4(k)+p1(1,j,k+1)*cz6(k))
         do 90 i=2, nxm1
            a(i) = -cpp1*cx4(i)
            b(i) = -cpp1*cx5(i) + 1.0
            c(i) = -cpp1*cx6(i)
            d(i) = p1(i,j,k)*(cx5(i) + cy5(j) + darcy1*cz5(k))
     +               + p1(i-1,j,k)*cx4(i) + p1(i+1,j,k)*cx6(i)
     +               + p1(i,j-1,k)*cy4(j) + p1(i,j+1,k)*cy6(j)
     +               + darcy1*(p1(i,j,k-1)*cz4(k)+p1(i,j,k+1)*cz6(k))
     +                     +(1.-darcy1)*(
     +   cxm1zm1(j,k)*p3(i,j-1,k-1)+cxzm1(j,k)*p3(i,j,k-1)+
     +   cxp1zm1(j,k)*p3(i,j+1,k-1)+cxm1z(j,k)*p3(i,j-1,k)+
     +   cxz(j,k)*p3(i,j,k)+cxp1z(j,k)*p3(i,j+1,k)+
     +   cxm1zp1(j,k)*p3(i,j-1,k+1)+cxzp1(j,k)*p3(i,j,k+1)+
     +   cxp1zp1(j,k)*p3(i,j+1,k+1))
   90    continue
c
            a(nx) = -cpp1*cx4h*2.0
            b(nx) = -cpp1*cx5h  + 1.0
            c(nx) = 0.0
            d(nx) = p1(nxm1,j,k)*2.0*cx4h
     +            + p1(nx,j,k)*(cx5h + cy5(j) + darcy1*cz5(k))
     +            + p1(nx,j-1,k)*cy4(j) + p1(nx,j+1,k)*cy6(j)
     +            + darcy1*(p1(nx,j,k-1)*cz4(k)+p1(nx,j,k+1)*cz6(k))

         call thomas(nx)
c
         do 95 i=1, nx
            aux(i,j,k) = d(i)
   95    continue
  100 continue
c
c
c     ... implicit in y ...
c
       do 200 i=1, nx
        do 200 k=2, nzm1
          do 190 j=2, nym1
              a(j-1) = -cpp1*cy4(j)
              b(j-1) = -cpp1*cy5(j) + 1.0
              c(j-1) = -cpp1*cy6(j)
              d(j-1) = aux(i,j,k)
  190     continue
c
         call thomas(nym2)
c
         do 195 j=2, nym1
            aux(i,j,k) = d(j-1)
  195    continue
  200 continue
c
c     ... implicit in z ...
c
       do 300 i=1, nx
        do 300 j=2, nym1
          do 290 k=2, nzm1
              a(k-1) = -cpp1*darcy1*cz4(k)
              b(k-1) = -cpp1*darcy1*cz5(k) + 1.0
              c(k-1) = -cpp1*darcy1*cz6(k)
              d(k-1) = aux(i,j,k)
  290     continue
c
```

```
          call thomas(nzm2)
c
          do 295 k=2, nzm1
             aux(i,j,k)    = d(k-1)
  295      continue
  300 continue
c
c     ... update p1 ...
c
      testp1 = 0.0
      p1max  = 0.0
      do 600 i=1, nx
       do 600 j=2, nym1
          do 600 k=2, nzm1
             testp1     = testp1 + abs(aux(i,j,k))
             p1(i,j,k)  = p1(i,j,k)  + dt*alpp*aux(i,j,k)
             absp1      = abs(p1(i,j,k))
             if (p1max.lt.absp1) p1max = absp1
  600 continue
      if (p1max.gt.0.001) testp1 = testp1/(nxm2*ny*nzm2*p1max)
c
c     to make vector potential steady state
c
      sumdif = 0.0
      sumall = 0.00001
      do 710 i=2,nxm1
       do 710 j=1,ny
          do 710 k=2,nzm1
           sumdif = sumdif + abs(p1old(i,j,k)-p1(i,j,k))
           sumall = sumall + abs(p1(i,j,k))
           p1old(i,j,k) = p1(i,j,k)
  710 continue
      sumdiv = sumdif/sumall
      if(sumdiv .lt.phierr) then
       go to 720
      end if
  700 continue
c
  720 if( nnpmax .lt. iij) nnpmax=iij
      if( nnpmin .gt. iij) nnpmin=iij

      return
      end


      subroutine p2solp
c
c     -------------------------------------------------------------
c    |                                                             |
c    |  *** solve for new y-component vector potential in porous ***|
c    |                                                             |
c     -------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      dimension   a(max),b(max),c(max),d(max)
      common /phierr/phierr
      common /alpfs/alpp,altp
      common /dt/dt
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /h/hx(max),hy(max),hz(max),
     +        hxold(max),hyold(max),hzold(max)
      common /pconsp/cpp1
      common /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +         cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +         cy4(nymax),cy5(nymax),cy6(nymax)
      common /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +         cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common /cyz/cym1zm1(nymax,nzmax),cyzm1(nymax,nzmax),
     +       cyp1zm1(nymax,nzmax),cym1z(nymax,nzmax),
     +       cyz(nymax,nzmax),cyp1z(nymax,nzmax),
     +       cym1zp1(nymax,nzmax),cyzp1(nymax,nzmax),
     +      cyp1zp1(nymax,nzmax)
      common /pbfdc/cx51,cx61,cx4h,cx5h,
     +         cy51,cy61,cy4h,cy5h,
     +         cz51,cz61,cz4h,cz5h
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
      common /p2old/p2old(nxmax,nymax,nzmax)
      common /tom/a,b,c,d
      common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common /auxct/auxct(nxmax,nymax,nzmax)
      common /t/theta(nxmax,nymax,nzmax)
      common /zconsp/cczp4,cczp5,cczp6,cczp7
      common /intcom/itercmpo,norun,mainiter
      common /test/testp1,testp2,testp3,testt
      common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common /niterp/niterp2,niterp3
      common /np2p3/nnpmax,nnpmin

c
c     ... implicit in x ...
c
      do 700 iij=1,itercmpo
       niterp2 = niterp2 + 1
       do 100 k=2, nzm1
        do 90 i=2, nxm1
           a(i-1) = -cpp1*cx4(i)
           b(i-1) = -cpp1*cx5(i) + 1.0
           c(i-1) = -cpp1*cx6(i)
           d(i-1) = p2(i,2,k)*2.0*cy61
     +           + p2(i,1,k)*(cx5(i) + cy51 + darcy2*cz5(k))
     +           + p2(i-1,1,k)*cx4(i) + p2(i+1,1,k)*cx6(i)
     +           + darcy2*(p2(i,1,k-1)*cz4(k)+p2(i,1,k+1)*cz6(k))
     +             - darcyz*cczp4*(theta(i,1,k-1)*cz1(k)*cczp5
     +             - theta(i-1,1,k)*cx1(i)*cczp7
     +             + theta(i,1,k+1)*cz3(k)*cczp5
     +             - theta(i+1,1,k)*cx3(i)*cczp7
     +             + theta(i,1,k)*(cz2(k)*cczp5
```

```
     +                        - cx2(i)*cczp7))
   90    continue
c
        call thomas(nxm2)
c
        do 95 i=2, nxm1
           aux(i,1,k) = d(i-1)
   95    continue
  100 continue
c
       do 200 j=2, nym1
        do 200 k=2, nzm1
           do 190 i=2, nxm1
               a(i-1) = -cpp1*cx4(i)
               b(i-1) = -cpp1*cx5(i)  + 1.0
               c(i-1) = -cpp1*cx6(i)
               d(i-1) = p2(i,j,k)*(cx5(i)  + cy5(j)  + darcy2*cz5(k))
     +                   + p2(i-1,j,k)*cx4(i)  + p2(i+1,j,k)*cx6(i)
     +                   + p2(i,j-1,k)*cy4(j)  + p2(i,j+1,k)*cy6(j)
     +                   + darcy2*(p2(i,j,k-1)*cz4(k)+p2(i,j,k+1)*cz6(k))
     +                   - darcyz*cczp4*(theta(i,j,k-1)*cz1(k)*cczp5
     +                   - theta(i-1,j,k)*cx1(i)*cczp7
     +                   + theta(i,j,k+1)*cz3(k)*cczp5
     +                   - theta(i+1,j,k)*cx3(i)*cczp7
     +                   + theta(i,j,k)*(cz2(k)*cczp5
     +                   - cx2(i)*cczp7))+(1.-darcy2)*(
     +   cym1zm1(j,k)*p3(i,j-1,k-1)+cyzm1(j,k)*p3(i,j,k-1)+
     +   cyp1zm1(j,k)*p3(i,j+1,k-1)+cym1z(j,k)*p3(i,j-1,k)+
     +   cyz(j,k)*p3(i,j,k)+cyp1z(j,k)*p3(i,j+1,k)+
     +   cym1zp1(j,k)*p3(i,j-1,k+1)+cyzp1(j,k)*p3(i,j,k+1)+
     +   cyp1zp1(j,k)*p3(i,j+1,k+1))
  190    continue
c
        call thomas(nxm2)
c
        do 195 i=2, nxm1
           aux(i,j,k) = d(i-1)
  195    continue
  200 continue
c
       do 300 k=2, nzm1
        do 290 i=2, nxm1
           a(i-1) = -cpp1*cx4(i)
           b(i-1) = -cpp1*cx5(i)  + 1.0
           c(i-1) = -cpp1*cx6(i)
           d(i-1) = p2(i,nym1,k)*2.0*cy4h
     +               + p2(i,ny,k)*(cx5(i)  + cy5h + darcy2*cz5(k))
     +               + p2(i-1,ny,k)*cx4(i)  + p2(i+1,ny,k)*cx6(i)
     +               + darcy2*(p2(i,ny,k-1)*cz4(k)+p2(i,ny,k+1)*cz6(k))
     +                 - darcyz*cczp4*(theta(i,ny,k-1)*cz1(k)*cczp5
     +                 - theta(i-1,ny,k)*cx1(i)*cczp7
     +                 + theta(i,ny,k+1)*cz3(k)*cczp5
     +                 - theta(i+1,ny,k)*cx3(i)*cczp7
     +                 + theta(i,ny,k)*(cz2(k)*cczp5
     +                 -                 cx2(i)*cczp7))
  290    continue
c
        call thomas(nxm2)
c
        do 295 i=2, nxm1
           aux(i,ny,k) = d(i-1)
  295    continue
  300 continue
c
c     ... implicit in y ...
c
       do 400 i=2, nxm1
        do 400 k=2, nzm1
           a(1) = 0.0
           b(1) = -cpp1*cy51 + 1.0
           c(1) = -cpp1*cy61*2.0
           d(1) = aux(i,1,k)
c
           do 390 j=2, nym1
               a(j) = -cpp1*cy4(j)
               b(j) = -cpp1*cy5(j)  + 1.0
               c(j) = -cpp1*cy6(j)
               d(j) = aux(i,j,k)
  390       continue
c
           a(ny) = -cpp1*cy4h*2.0
           b(ny) = -cpp1*cy51 + 1.0
           c(ny) = 0.0
           d(ny) = aux(i,ny,k)
c
        call thomas(ny)
c
        do 395 j=1, ny
           aux(i,j,k) = d(j)
  395    continue
  400 continue
c
c     ... implicit in z ...
c
       do 500 i=2, nxm1
        do 500 j=1, ny
           do 490 k=2, nzm1
               a(k-1) = -cpp1*darcy2*cz4(k)
               b(k-1) = -cpp1*darcy2*cz5(k)  + 1.0
               c(k-1) = -cpp1*darcy2*cz6(k)
               d(k-1) = aux(i,j,k)
  490       continue
c
           call thomas(nzm2)
c
           do 495 k=2, nzm1
               aux(i,j,k)    = d(k-1)
  495       continue
  500 continue
c
```

```
c       ... update p2 ...
c
        testp2 = 0.0
        p2max  = 0.0
        do 600 i=2, nxm1
         do 600 j=1, ny
            do 600 k=2, nzm1
                testp2     = testp2 + abs(aux(i,j,k))
                p2(i,j,k)  = p2(i,j,k) + dt*alpp*aux(i,j,k)
                absp2      = abs(p2(i,j,k))
                if (p2max.lt.absp2) p2max = absp2
  600 continue
       if (p2max.gt.0.001) testp2 = testp2/(nxm2*ny*nzm2*p2max)
c
c      to make vector potential steady state
c
        sumdif = 0.0
        sumall = 0.00001
        do 710 i=2,nxm1
         do 710 j=1,ny
            do 710 k=2,nzm1
            sumdif = sumdif + abs(p2old(i,j,k)-p2(i,j,k))
            sumall = sumall + abs(p2(i,j,k))
            p2old(i,j,k) = p2(i,j,k)
 710   continue
        sumdiv = sumdif/sumall
        if(sumdiv .lt.phierr) then
         go to 720
        end if
 700   continue
c
 720   if( nnpmax .lt. iij) nnpmax=iij
        if( nnpmin .gt. iij) nnpmin=iij

        return
        end

        subroutine p3solp
c
c      -----------------------------------------------------------------
c     |                                                                 |
c     |  *** solve for new z-component vector potential in porous ***  |
c     |                                                                 |
c      -----------------------------------------------------------------
c
        parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
        dimension a(max),b(max),c(max),d(max)
        common /alpfs/alpp,altp
        common /dt/dt
        common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
        common /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
        common /pconsp/cpp1
        common /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +          cx4(nxmax),cx5(nxmax),cx6(nxmax)
        common /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +          cy4(nymax),cy5(nymax),cy6(nymax)
        common /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +          cz4(nzmax),cz5(nzmax),cz6(nzmax)
        common /cyz/cym1zm1(nymax,nzmax),cyzm1(nymax,nzmax),
     +          cyp1zm1(nymax,nzmax),cym1z(nymax,nzmax),
     +          cyz(nymax,nzmax),cyp1z(nymax,nzmax),
     +          cym1zp1(nymax,nzmax),cyzp1(nymax,nzmax),
     +          cyp1zp1(nymax,nzmax)
        common /pbfdc/cx51,cx61,cx4h,cx5h,
     +          cy51,cy61,cy4h,cy5h,
     +          cz51,cz61,cz4h,cz5h
        common /p2/p2(nxmax,nymax,nzmax)
        common /p3/p3(nxmax,nymax,nzmax)
        common /p3old/p3old(nxmax,nymax,nzmax)
        common /tom/a,b,c,d
        common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
        common /auxct/auxct(nxmax,nymax,nzmax)
        common /phierr/phierr
        common /t/theta(nxmax,nymax,nzmax)
        common /zconsp/cczp4,cczp5,cczp6,cczp7
        common /intcom/itercmpo,norun,mainiter
        common /test/testp1,testp2,testp3,testt
        common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
        common /niterp/niterp2,niterp3
        common /np2p3/nnpmax,nnpmin
c
c      ... implicit in x ...
c
        do 700 iij=1,itercmpo
         niterp3 = niterp3 + 1
         do 100 j=2, nym1
          do 90 i=2, nxm1
              a(i-1) = -cpp1*cx4(i)
              b(i-1) = -cpp1*cx5(i) + 1.0
              c(i-1) = -cpp1*cx6(i)
              d(i-1) = p3(i,j,2)*2.0*cz61
     +           + p3(i,j,1)*(cx5(i) + darcy3*cy5(j) + cz51)
     +           + p3(i-1,j,1)*cx4(i) + p3(i+1,j,1)*cx6(i)
     +           + darcy3*(p3(i,j-1,1)*cy4(j)+p3(i,j+1,1)*cy6(j))
     +             - darcyy*cczp4*(theta(i-1,j,1)*cx1(i)*cczp6
     +             - theta(i,j-1,1)*cy1(j)*cczp5
     +             + theta(i+1,j,1)*cx3(i)*cczp6
     +             - theta(i,j+1,1)*cy3(j)*cczp5
     +             + theta(i,j,1)*(-cy2(j)*cczp5
     +             +              cx2(i)*cczp6))
 90      continue
c
         call thomas(nxm2)
c
         do 95 i=2, nxm1
             aux(i,j,1) = d(i-1)
```

```
   95     continue
  100 continue
c
      do 200 j=2, nym1
       do 200 k=2, nzm1
          do 190 i=2, nxm1
              a(i-1)  = -cpp1*cx4(i)
              b(i-1)  = -cpp1*cx5(i)  + 1.0
              c(i-1)  = -cpp1*cx6(i)
              d(i-1)  = p3(i,j,k)*(cx5(i)  + darcy3*cy5(j)  + cz5(k))
     +                 + p3(i-1,j,k)*cx4(i)  + p3(i+1,j,k)*cx6(i)
     +                 + darcy3*(p3(i,j-1,k)*cy4(j)+p3(i,j+1,k)*cy6(j))
     +                 + p3(i,j,k-1)*cz4(k)  + p3(i,j,k+1)*cz6(k)
     +                 - darcyy*cczp4*(theta(i-1,j,k)*cx1(i)*cczp6
     +                 - theta(i,j-1,k)*cy1(j)*cczp5
     +                 + theta(i+1,j,k)*cx3(i)*cczp6
     +                 - theta(i,j+1,k)*cy3(j)*cczp5
     +                 + theta(i,j,k)*(-cy2(j)*cczp5
     +                +                          cx2(i)*cczp6))
     +                +(darcy3 - 1.)*(
     +   cym1zm1(j,k)*p2(i,j-1,k-1)+cyzm1(j,k)*p2(i,j,k-1)+
     +   cyp1zm1(j,k)*p2(i,j+1,k-1)+cym1z(j,k)*p2(i,j-1,k)+
     +   cyz(j,k)*p2(i,j,k)+cyp1z(j,k)*p2(i,j+1,k)+
     +   cym1zp1(j,k)*p2(i,j-1,k+1)+cyzp1(j,k)*p2(i,j,k+1)+
     +   cyp1zp1(j,k)*p2(i,j+1,k+1))

  190     continue
c
      call thomas(nxm2)
c
      do 195 i=2, nxm1
         aux(i,j,k)  = d(i-1)
  195     continue
  200 continue
c
      do 300 j=2, nym1
       do 290 i=2, nxm1
          a(i-1)  = -cpp1*cx4(i)
          b(i-1)  = -cpp1*cx5(i)  + 1.0
          c(i-1)  = -cpp1*cx6(i)
          d(i-1)  = p3(i,j,nzm1)*2.0*cz4h
     +             + p3(i,j,nz)*(cx5(i)  + darcy3*cy5(j)  + cz5h)
     +             + p3(i-1,j,nz)*cx4(i)  + p3(i+1,j,nz)*cx6(i)
     +             + darcy3*(p3(i,j-1,nz)*cy4(j)+p3(i,j+1,nz)*cy6(j))
     +               - darcyy*cczp4*(theta(i-1,j,nz)*cx1(i)*cczp6
     +               - theta(i,j-1,nz)*cy1(j)*cczp5
     +               + theta(i+1,j,nz)*cx3(i)*cczp6
     +               - theta(i,j+1,nz)*cy3(j)*cczp5
     +               + theta(i,j,nz)*(-cy2(j)*cczp5
     +              +                         cx2(i)*cczp6))

  290     continue
c
      call thomas(nxm2)
c
      do 295 i=2, nxm1
         aux(i,j,nz)  = d(i-1)
  295     continue
  300 continue
c
c     ... implicit in y ...
c
      do 400 i=2, nxm1
       do 400 k=1, nz
          do 390 j=2, nym1
              a(j-1)  = -cpp1*darcy3*cy4(j)
              b(j-1)  = -cpp1*darcy3*cy5(j)  + 1.0
              c(j-1)  = -cpp1*darcy3*cy6(j)
              d(j-1)  = aux(i,j,k)
  390         continue
c
          call thomas(nym2)
c
          do 395 j=2, nym1
              aux(i,j,k)  = d(j-1)
  395         continue
  400 continue
c
c     ... implicit in z ...
c
      do 500 i=2, nxm1
       do 500 j=2, nym1
          a(1)  = 0.0
          b(1)  = -cpp1*cz51  + 1.0
          c(1)  = -cpp1*cz61*2.0
          d(1)  = aux(i,j,1)
c
          do 490 k=2, nzm1
              a(k)  = -cpp1*cz4(k)
              b(k)  = -cpp1*cz5(k)  + 1.0
              c(k)  = -cpp1*cz6(k)
              d(k)  = aux(i,j,k)
  490         continue
c
          a(nz)  = -cpp1*cz4h*2.0
          b(nz)  = -cpp1*cz51  + 1.0
          c(nz)  = 0.0
          d(nz)  = aux(i,j,nz)
c
       call thomas(nz)
c
       do 495 k=1, nz
          aux(i,j,k)    = d(k)
  495     continue
  500 continue
c
c     ... update p3 ...
c
      testp3  = 0.0
      p3max   = 0.0
```

```
      do 600 i=2, nxm1
       do 600 j=2, nym1
          do 600 k=1, nz
              testp3     = testp3 + abs(aux(i,j,k))
              p3(i,j,k)  = p3(i,j,k) + dt*alpp*aux(i,j,k)
              absp3      = abs(p3(i,j,k))
              if (p3max.lt.absp3) p3max = absp3
  600 continue
      if (p3max.gt.0.001) testp3 = testp3/(nxm2*nym2*nz*p3max)
c
c     to make vector potential steady state
c
      sumdif = 0.0
      sumall = 0.00001
      do 710 i=2,nxm1
       do 710 j=2,nym1
          do 710 k=1,nz
          sumdif = sumdif + abs(p3old(i,j,k)-p3(i,j,k))
          sumall = sumall + abs(p3(i,j,k))
          p3old(i,j,k) = p3(i,j,k)
 710  continue
      sumdiv = sumdif/sumall
      if(sumdiv.lt.phierr) then
       go to 720
      end if
 700  continue
c
 720  if( nnpmax .lt. iij) nnpmax=iij
      if( nnpmin .gt. iij) nnpmin=iij
      return
      end

      subroutine qfunctns(tpnt,afunc,ma)
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |              ***  constructing basis functions ***             |
c    |                                                                |
c     ----------------------------------------------------------------
c
      dimension afunc(ma)
      afunc(1)=1.0
      do 11 i=2,ma
             afunc(i)=tpnt*afunc(i-1)
 11    continue
      end

      subroutine qparmfit(tpnt,q,sig,ndata,a,ma,lista,mfit,
     +   covar,ncvm,chisq)
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |       ***  performing fitting for subroutine qsource   ***     |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter (mmax=50)
c        dimension covar(20,20),lista(20)
      dimension covar(ncvm,ncvm),lista(ncvm)
      dimension tpnt(ndata),q(ndata),sig(ndata),a(ma),
     +   beta(mmax),afunc(mmax)
      kk=mfit+1
      do 12 j=1,ma
      ihit=0
      do 11 k=1,mfit
        if (lista(k).eq.j) ihit=ihit+1
 11     continue
      if (ihit.eq.0) then
        lista(kk)=j
        kk=kk+1
      else if (ihit.gt.1) then
        pause 'Improper set in lista for fitting Q function.'
      endif
 12   continue
      if (kk.ne.(ma+1))
     +   pause 'Improper set in lista for fitting Q function.'
      do 14 j=1,mfit
      do 13 k=1,mfit
        covar(j,k)=0.
 13     continue
      beta(j)=0.
 14   continue
      do 18 i=1,ndata
      call qfunctns(tpnt(i),afunc,ma)
      qm=q(i)
      if(mfit.lt.ma) then
        do 15 j=mfit+1,ma
          qm=qm-a(lista(j))*afunc(lista(j))
 15       continue
      endif
      sig2i=1./sig(i)**2
      do 17 j=1,mfit
        wt=afunc(lista(j))*sig2i
        do 16 k=1,j
          covar(j,k)=covar(j,k)+wt*afunc(lista(k))
 16       continue
        beta(j)=beta(j)+qm*wt
 17     continue
 18   continue
      if (mfit.gt.1) then
      do 21 j=2,mfit
        do 19 k=1,j-1
          covar(k,j)=covar(j,k)
 19       continue
 21     continue
      endif
      call gaussjdn(covar,mfit,ncvm,beta,1,1)
      do 22 j=1,mfit
      a(lista(j))=beta(j)
 22   continue
```

```
      chisq=0.
      do 24 i=1,ndata
      call qfunctns(tpnt(i),afunc,ma)
      sum=0.
      do 23 j=1,ma
        sum=sum+a(j)*afunc(j)
23      continue
      chisq=chisq+((q(i)-sum)/sig(i))**2
24    continue
      return
      end

      subroutine qsource(tpnt,q,npt,nbest,abest)
c
c    -----------------------------------------------------------------
c   |
c   | ***  fitting produce respiration heat source to a function *** |
c   |                                                                |
c    -----------------------------------------------------------------
c

      dimension tpnt(50),q(50),sig(50),a(10),covar(20,20),lista(20)
      dimension abest(10),afunc(50)
      common /point/points(200)
      common /ccaa/cc0,dtemp,tcold,thot
      common /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,xll,annu,gravity,ddd,keff

      sumerr= 9.E30
      nparam = 4
c
c  ... find the best fitting for the function ...
c
      do 1000 ntry=1,7
             do 6 i=1,10
             a(i)=0.0
6              continue
             do 9 i=1,npt
             sig(i)=0.1
9              continue
             mfit=nparam
             do 13 i=1,mfit
13             lista(i)=i
      call qparmfit(tpnt,q,sig,npt,a,nparam,lista,mfit,
     +   covar,nparam,chisq)

      if (sumerr .gt. chisq) then
      nbest=nparam
      do 8 i=1, 10
      abest(i)=a(i)
8      continue
      endif

      nparam = nparam +1
1000   continue
c
c
c    ... draw a heating source-temperature curve for confirmation ...

      txtemp=-2.
      do 91 i=1, 76
      if ( (txtemp .ge. 35.) .or. (txtemp .le. -1.5)) then
      points(i)=0.0
      else
              call qfunctns(txtemp,afunc,nbest)
              points(i)=0.0
              do 500 nterm=1,nbest
              points(i)=points(i)+abest(nterm)*afunc(nterm)
500             continue
      endif
               points(i)=points(i)*cc0
     +          *(epson*rhof*cpfl+(1.-epson)*rhos*cpsd)/(rhof*cpfl)
      points(i)=points(i)*(1000.*dtemp*keff)/(xll*xll)
               if( points(i).lt. 0.0) points(i)=0.0

      txtemp=txtemp + 0.5
91     continue
      call drawcurv
      return
      end

      subroutine reader
c
c    -----------------------------------------------------------------
c   |
c   |           *** read results from binary input file ***          |
c   |                                                                |
c    -----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical  lfile
      logical  blup,cgd
      logical  reed,rite,strt,dist,heat,nuss,tran
      logical  rt,rp1,rp2,rp3,rmax
      logical  divv,divp,dmax
      logical  vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +         mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +         vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +         myp1,myp2,myp3,myt,myu,myv,myw,
     +         vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +         mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      logical  ixl,ixh,iyl,iyh,izl,izh
c
      character  filename*12
      character  title*80,otitle*80
      character  xc*1,yc*1,zc*1,vpform*1
      character  meshtp*1,exprmnt*8,gridfile*8,srcetype*1,datype*1
      character  omeshtp*1,oexprmnt*8,ogridfil*8,osrcetyp*1,odatype*1
      integer  drawstep
      integer  pstxl,pstyl,pstzl
```

```
     +      ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +      ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +      ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +      ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +      ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      real keff
c
      common /title/title,otitle
      common /tapcom/rfile(94),ifile(67),lfile(10)
      common /exprmnt/exprmnt
      common /fileno/inparam,irg,ireport,itin,itout
      common /logc/reed,rite,strt,dist,heat,nuss,tran
      common /logi/blup,cgd
      common /logr/rt,rp1,rp2,rp3,rmax
      common /logd/divv,divp,dmax
      common /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +               mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +               vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +               myp1,myp2,myp3,myt,myu,myv,myw,
     +               vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +               mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common /logtbc/ixl,ixh,iyl,iyh,izl,izh
      common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +               cct,tmax,pdist,rtime
      common /alpfs/alpp,altp
      common /intcom/itercmpo,norun,mainiter
      common /numcom/numit
      common /t/theta(nxmax,nymax,nzmax)
      common /p1/p1(nxmax,nymax,nzmax)
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
      common /p1old/p1old(nxmax,nymax,nzmax)
      common /p2old/p2old(nxmax,nymax,nzmax)
      common /p3old/p3old(nxmax,nymax,nzmax)
      common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +           ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +           zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
      common /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +           iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +           izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
      common /velbc/xlu,xlv,xlw,xhu,xhv,xhw,
     +             ylu,ylv,ylw,yhu,yhv,yhw,
     +             zlu,zlv,zlw,zhu,zhv,zhw
      common /mesh/meshtp
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
      common /axis/x(max),y(max),z(max)
      common /vmpfrm/xc,yc,zc,vpform
      common /vmpnum/npx,npy,npz
      common /vmploc/mx(nxmax),my(nymax),mz(nzmax)
      common /vmppos/rmx(nxmax),rmy(nymax),rmz(nzmax)
      common /mapcom/mnx,mny,mnz
      common /phierr/phierr
      common /da/da
      common /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,xll,annu,gravity,ddd,keff
      common /ccaa/cc0,dtemp,tcold,thot
      common /draw/drawstep
      common /pst/pstx,psty,pstz
      common /pstxyz/pstx1,psty1,pstz1
     +      ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +      ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +      ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +      ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +      ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      common /param_q/qparam(20)
      common /numshow/numshow
      common /iterpp/iterpp
      common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common /tstrt/tstrt
      common /gridfile/gridfile
      common /txty/tpnt(10),q(10),npt,nparam,srcetype
      common /datype/datype
      common /vel/u(nxmax,nymax,nzmax),
     +          v(nxmax,nymax,nzmax),
     +          w(nxmax,nymax,nzmax)
      common /dt/dt

      filename=exprmnt // '.bin'
      open(unit=itin ,file=filename   ,status='unknown',
     +                                 form='unformatted')
c
c ... read header from binary file ...
c
      rewind itin
      read(itin) otitle,omeshtp,oexprmnt,osrcetyp,odatype,ogridfil,
     +           rfile,ifile,lfile
c
c ... set old mesh size ...
c
      nxold=ifile(2)
      nyold=ifile(3)
      nzold=ifile(4)
c
c ... read mesh from binary file ...
c
      read(itin) ( hxold(i),  i=1,  nxold-1)
      read(itin) ( hyold(j),  j=1,  nyold-1)
      read(itin) ( hzold(k),  k=1,  nzold-1)
c
c ... read from binary file ...
c
      read(itin)(((     p1(i,j,k),i=1,nxold),j=1,nyold),k=1,nzold)
      read(itin)(((     p2(i,j,k),i=1,nxold),j=1,nyold),k=1,nzold)
      read(itin)(((     p3(i,j,k),i=1,nxold),j=1,nyold),k=1,nzold)
      read(itin)(((theta(i,j,k),i=1,nxold),j=1,nyold),k=1,nzold)
c
```

```
      endfile itin
      close(itin)
c
      if(title .ne. otitle) then
      write(*,1000)
      write(ireport,1000)
 1000  format(' Warning! New title is not the same as old title.')
      write(*,1010)     title
      write(ireport,1010)  title
 1010  format(1x,'Old title:',/,a80)
      write(*,1020)     otitle
      write(ireport,1020) otitle
 1020  format(1x,'New title:',/,a80)
      endif
      if(meshtp .ne. omeshtp) then
      write(*,1030)
      write(ireport,1030)
 1030  format(' Warning! New mesh type is not the same as',
     +  ' the old type.')
      write(*,1040)     meshtp,omeshtp
      write(ireport,1040) meshtp,omeshtp
 1040  format(1x,/,'Old mesh type:    ',a1,
     +          /,'New mesh type:    ',a1)
      endif
      if(srcetype .ne. osrcetyp) then
      write(*,1050)
      write(ireport,1050)
 1050  format(' Warning! New heating source type is not the same as',
     +  ' the old type.')
      write(*,1060) srcetype, osrcetyp
      write(ireport,1060) srcetype, osrcetyp
 1060  format(1x,/,'Old heating type:    ',a1,
     +          /,'New heating type:    ',a1)
      endif
      if(datype .ne. odatype) then
      write(*,1070)
      write(ireport,1070)
 1070  format(' Warning! New Darcy numbers may be different.')
      write(*,1080) datype, odatype
      write(ireport,1080) datype, odatype
 1080  format(1x,/,'Old DATYPE :    ',a1,
     +          /,'New DATYPE:    ',a1)
      endif
      if(gridfile .ne. ogridfil) then
      write(*,1090)
      write(ireport,1090)
 1090  format(' Warning! New grid file is not the same as',
     +  ' old grid file.')
      write(*,1100) gridfile, ogridfil
      write(ireport,1100) gridfile, ogridfil
 1100  format(1x,/,'Old grid file :    ',a8,
     +          /,'New grid file:    ',a8)
      endif
      if(exprmnt .ne. oexprmnt) then
      write(*,1110)
      write(ireport,1110)
 1110  format(' Warning! New experiment name for output files is',
     +  ' not the same as the old name.')
      write(*,1120) exprmnt, oexprmnt
      write(ireport,1120) exprmnt, oexprmnt
 1120  format(1x,/,'Old experiment name :    ',a8,
     +          /,'New experiment name:    ',a8)
      endif
c
      if((meshtp.eq.'b').or.(meshtp.eq.'B')) then
c
c      ... set stepsizes between nodes as in binary file ...
c
      nx = nxold
      ny = nyold
      nz = nzold
c
      do 10 i=1, nx
         hx(i) = hxold(i)
   10  continue
      do 11 j=1, ny
         hy(j) = hyold(j)
   11  continue
      do 12 k=1, nz
         hz(k) = hzold(k)
   12  continue

      endif
c
c
c      ... transform mesh if not same number of mesh points ...
c
      if((nxold.ne.nx) .or. (nyold.ne.ny) .or. (nzold.ne.nz) ) then
      write(*,4020) nxold,nyold,nzold,nx,ny,nz
      write(ireport,4020) nxold,nyold,nzold,nx,ny,nz
 4020  format(' Warning: old(nx,ny,nz) =',i3,' ,',i3,' ,',i3,
     +          ' .ne. to new(nx,ny,nz) =',i3,' ,',i3,' ,',i3 )
      tran = .true.
       else
c
c      ... transform mesh if not same step size ...
c
         do 20 i=1,nx-1
            if(hxold(i).ne.hx(i)) tran= .true.
   20     continue
         do 21 j=1,ny-1
            if(hyold(j).ne.hy(j)) tran= .true.
   21     continue
         do 22 k=1,nz-1
            if(hzold(k).ne.hz(k)) tran= .true.
   22     continue
      if(tran) write(*,4030)
      if(tran) write(ireport,4030)
 4030  format(' Warning: old mesh .ne. to new mesh ')
```

```
      endif
c
      if(tran) then
      write(*,4040)
      write(ireport,4040)
 4040  format(' Warning: mesh interpolated')
c
      call  interp(nxold,nyold,nzold,nx,ny,nz,hxold,hyold,hzold,
     +             hx,hy,hz,p1)
      call  interp(nxold,nyold,nzold,nx,ny,nz,hxold,hyold,hzold,
     +             hx,hy,hz,p2)
      call  interp(nxold,nyold,nzold,nx,ny,nz,hxold,hyold,hzold,
     +             hx,hy,hz,p3)
      call  interp(nxold,nyold,nzold,nx,ny,nz,hxold,hyold,hzold,
     +             hx,hy,hz,theta)
c
      end if
c
c     ... initialize velocities from
c         previously stored vector potential ...
c
      call veloc
      call vbc
c
c     ... initialise old vector potential ...
c
      do 5000 i=1,nx
       do 5000 j=1,ny
          do 5000 k=1,nz
              p1old(i,j,k)  =  p1(i,j,k)
              p2old(i,j,k)  =  p2(i,j,k)
              p3old(i,j,k)  =  p3(i,j,k)
 5000 continue
c
      return
      end

      subroutine resip1
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |             *** calculate residual for p1 ***                  |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical  rt,rp1,rp2,rp3,rmax
c
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +            cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +            cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +            cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
      p1min= 1000000000.
      p1max=-1000000000.
      do 10 i=2, nxm1
       do 10 j=2, nym1
          do 10 k=2, nzm1
              aux(i,j,k)  =  cx4(i)*p1(i-1,j,k)
     +                      +cx5(i)*p1(i  ,j,k)
     +                      +cx6(i)*p1(i+1,j,k)
     +                      +cy4(j)*p1(i,j-1,k)
     +                      +cy5(j)*p1(i,j  ,k)
     +                      +cy6(j)*p1(i,j+1,k)
     +                      +cz4(k)*p1(i,j,k-1)
     +                      +cz5(k)*p1(i  ,j,k)
     +                      +cz6(k)*p1(i,j,k+1)
      if( p1max .lt. p1(i,j,k)) p1max=p1(i,j,k)
      if( p1min .gt. p1(i,j,k)) p1min=p1(i,j,k)
 10 continue
c
      if(rp1) then
       do 15 i=1, nx
          do 15 j=1, ny
              aux2d(i,j)  =  0.0
 15    continue
c
      do 20 k=2, nzm1
         do 25 i=2, nxm1
             do 25 j=2, nym1
              aux2d(i,j)  =  aux(i,j,k)
 25          continue
             call mprtz(aux2d,' p1 residuals   ',k)
 20    continue
      endif
c
      call maxval(aux,' p1 residuals   ')
c
      return
      end

      subroutine resip2
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |             *** calculate residual for p2 ***                  |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
```

```
c
      logical  rt,rp1,rp2,rp3,rmax
c
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +              cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +              cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +              cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
      p2min= 1000000000.
      p2max=-1000000000.
      do 10 i=2, nxm1
       do 10 j=2, nym1
          do 10 k=2, nzm1
               aux(i,j,k)  = cx4(i)*p2(i-1,j,k)
     +                       +cx5(i)*p2(i   ,j,k)
     +                       +cx6(i)*p2(i+1,j,k)
     +                       +cy4(j)*p2(i,j-1,k)
     +                       +cy5(j)*p2(i,j   ,k)
     +                       +cy6(j)*p2(i,j+1,k)
     +                       +cz4(k)*p2(i,j,k-1)
     +                       +cz5(k)*p2(i   ,j,k)
     +                       +cz6(k)*p2(i,j,k+1)
      if( p2max  .lt.  p2(i,j,k))  p2max=p2(i,j,k)
      if( p2min  .gt.  p2(i,j,k))  p2min=p2(i,j,k)
   10 continue
c
      if(rp2) then
       do 15 i=1, nx
          do 15 j=1, ny
               aux2d(i,j)  = 0.0
   15   continue
c
       do 20 k=2, nzm1
          do 25 i=2, nxm1
               do 25 j=2, nym1
                aux2d(i,j)  = aux(i,j,k)
   25          continue
               call mprtz(aux2d,' p2 residuals   ',k)
   20   continue
      endif
c
      call maxval(aux,' p2 residuals   ')
c
      return
      end

      subroutine  resip3
c
c     ------------------------------------------------------------------
c     |                                                                |
c     |                *** calculate residual for p3 ***               |
c     |                                                                |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical  rt,rp1,rp2,rp3,rmax
c
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +              cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +              cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +              cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
      p3min  =   100000000.
      p3max  =  -100000000.
      do 10 i=2, nxm1
       do 10 j=2, nym1
          do 10 k=2, nzm1
               aux(i,j,k)  = cx4(i)*p3(i-1,j,k)
     +                       +cx5(i)*p3(i   ,j,k)
     +                       +cx6(i)*p3(i+1,j,k)
     +                       +cy4(j)*p3(i,j-1,k)
     +                       +cy5(j)*p3(i,j   ,k)
     +                       +cy6(j)*p3(i,j+1,k)
     +                       +cz4(k)*p3(i,j,k-1)
     +                       +cz5(k)*p3(i   ,j,k)
     +                       +cz6(k)*p3(i,j,k+1)
      if( p3min  .gt.  p3(i,j,k))  p3min=p3(i,j,k)
      if( p3max  .lt.  p3(i,j,k))  p3max=p3(i,j,k)
   10 continue
c
      if(rp3) then
       do 15 i=1, nx
          do 15 j=1, ny
               aux2d(i,j)  = 0.0
   15   continue
c
       do 20 k=2, nzm1
          do 25 i=2, nxm1
               do 25 j=2, nym1
                aux2d(i,j)  = aux(i,j,k)
   25          continue
               call mprtz(aux2d,' p3 residuals   ',k)
```

```
 20     continue
      endif
c
      call maxval(aux,' p3 residuals   ')
c
      return

      end
      subroutine resit
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |              *** calculate residual for theta ***              |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical  rt,rp1,rp2,rp3,rmax
c
      common  /logr/rt,rp1,rp2,rp3,rmax
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +            cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +            cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +            cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /vel/u(nxmax,nymax,nzmax),
     +          v(nxmax,nymax,nzmax),
     +          w(nxmax,nymax,nzmax)
      common  /t/theta(nxmax,nymax,nzmax)
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
c
      do 10 i=2, nxm1
       do 10 j=2, nym1
         do 10 k=2, nzm1
          aux(i,j,k) = -(cx1(i)*u(i-1,j,k)*theta(i-1,j,k)
     +                  + cx2(i)*u(i   ,j,k)*theta(i   ,j,k)
     +                  + cx3(i)*u(i+1,j,k)*theta(i+1,j,k))
     +              -(cy1(j)*v(i,j-1,k)*theta(i,j-1,k)
     +                  + cy2(j)*v(i   ,j,k)*theta(i   ,j,k)
     +                  + cy3(j)*v(i,j+1,k)*theta(i,j+1,k))
     +              -(cz1(k)*w(i,j,k-1)*theta(i,j,k-1)
     +                  + cz2(k)*w(i   ,j,k)*theta(i   ,j,k)
     +                  + cz3(k)*w(i,j,k+1)*theta(i,j,k+1))
     +              +cx4(i)*theta(i-1,j,k)
     +              +cx5(i)*theta(i   ,j,k)
     +              +cx6(i)*theta(i+1,j,k)
     +                     +cy4(j)*theta(i,j-1,k)
     +                     +cy5(j)*theta(i,j   ,k)
     +                     +cy6(j)*theta(i,j+1,k)
     +                     +cz4(k)*theta(i,j,k-1)
     +                     +cz5(k)*theta(i,j   ,k)
     +                     +cz6(k)*theta(i,j,k+1)
 10   continue
c
       if(rt) then
        do 15 i=1, nx
          do 15 j=1, ny
              aux2d(i,j) = 0.0
 15     continue
c
        do 20 k=2, nzm1
          do 25 i=2, nxm1
              do 25 j=2, nym1
               aux2d(i,j) = aux(i,j,k)
 25           continue
              call mprtz(aux2d,'theta residuals',k)
 20     continue
       endif
c
      call maxval(aux,'theta residuals')
c
      return
      end

      subroutine search(pos,f,index,m)
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |                                                                |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      dimension f(max)
c
      do 10 i=1, m-1
       if ((pos.ge.f(i)).and.(pos.lt.f(i+1))) then
         index = i
         go to 20
       endif
 10   continue
c
 20   return
      end

      subroutine setcon
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |                *** setting of constants ***                    |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)

      integer drawstep
```

```
      real keff
      real sbetax, sbetay, sbetaz, cbetax, cbetay, cbetaz,
     +     hl, h2, sh1, sh2, h1h2, sh1h2, hmin, denom
      character srcetype*1,datype*1

      common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +              cct,tmax,pdist,rtime
      common /alpfs/alpp,altp
      common /intcom/itercmpo,norun,mainiter
      common /fileno/inparam,irg,ireport,itin,itout
      common /dt/dt
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /h/hx(max),hy(max),hz(max),
     +            hxold(max),hyold(max),hzold(max)
      common /axis/x(max),y(max),z(max)
      common /mapcom/mnx,mny,mnz
      common /pconsp/cpp1
      common /zconsp/cczp4,cczp5,cczp6,cczp7
      common /tconsp/ctp1,ctp2,ctp3,ctp4
      common /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +            cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +            cy4(nymax),cy5(nymax),cy6(nymax)
      common /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +            cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common /cxz/cxm1zm1(nxmax,nzmax),cxzm1(nxmax,nzmax),
     +        cxp1zm1(nxmax,nzmax),cxm1z(nxmax,nzmax),
     +        cxz(nxmax,nzmax),cxp1z(nxmax,nzmax),
     +        cxm1zp1(nxmax,nzmax),cxzp1(nxmax,nzmax),
     +       cxp1zp1(nxmax,nzmax)
      common /cyz/cym1zm1(nymax,nzmax),cyzm1(nymax,nzmax),
     +        cyp1zm1(nymax,nzmax),cym1z(nymax,nzmax),
     +        cyz(nymax,nzmax),cyp1z(nymax,nzmax),
     +        cym1zp1(nymax,nzmax),cyzp1(nymax,nzmax),
     +       cyp1zp1(nymax,nzmax)
      common /tbfdc/cxf(2,3),cyf(2,3),czf(2,3)
      common /pbfdc/cx5l,cx6l,cx4h,cx5h,
     +            cy5l,cy6l,cy4h,cy5h,
     +            cz5l,cz6l,cz4h,cz5h
      common /da/da
      common /ccaa/cc0,dtemp,tcold,thot
      common /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,xll,annu,gravity,ddd,keff
      common /cxv/cx1lv,cx2lv,cx3lv,cx1hv,cx2hv,cx3hv
      common /cyv/cy1lv,cy2lv,cy3lv,cy1hv,cy2hv,cy3hv
      common /czv/cz1lv,cz2lv,cz3lv,cz1hv,cz2hv,cz3hv
      common /draw/drawstep
      common /timestep/timestep
      common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common /tstrt/tstrt
      common /numshow/numshow
      common /txty/tpnt(10),q(10),npt,nparam,srcetype
      common /param_q/qparam(20)
      common /drwstp/drwstp1,drwstp2,drwstp3,drwstp4,drwstp5,
     + drwstp6,drwstp7,drwstp8,drwstp9,drwstp10,drwstp11,drwstp12
      common /dthpnt/dthpnt(nxmax,nymax,nzmax),thmldthl,thmldthh
      common /datype/datype
c
c     ... intermediate variables ...
c
      rhop=epson*rhof+(1-epson)*rhos
      alphap=keff/(rhof*cpfl)
      cppr=cpfl
      dtemp=thot-tcold
      cc0=rhof*cpfl/(epson*rhof*cpfl+(1.-epson)*rhos*cpsd)
      tstrt=(tstrt-tcold)/dtemp
      thmldthl=-1.
      thmldthh=35.
c
c    produce initially alive
c
      do 89 i=1,nxmax
      do 89 j=1,nymax
      do 89 k=1,nzmax
 89    dthpnt(i,j,k) = 1.
c
c    heat source unit converting
c
      if ((srcetype.eq.'C') .or. (srcetype.eq.'c')) then
      do 90 i=1,10
 90      q(i)=q(i)*2.965083333

      cc0=cc0*(1.-epson)*xll*xll*rhos/(1000000.*dtemp*keff)

      else
      if ((srcetype.eq.'W').or.(srcetype.eq.'w')) then

      cc0=cc0*xll*xll/(1000.*dtemp*keff)

      else
      write(*,*) 'Unknown heating source. Program stopped.'
      stop
      endif
      endif
c
c    formulate heat source into a function of temperature
c
      call qsource(tpnt,q,npt,nparam,qparam)
c
c      determine constants
c
      ra=2.*gravity*xll*xll*xll*(thot-tcold)*rhof*rhof
      ra=ra*cpfl/(keff*annu*(thot+tcold+273+273))
      da=ddd*ddd*epson*epson*epson
      da=da/(175.*(1.-epson)*(1.-epson)*xll*xll)
c
c   find Darcy numbers
c
      if ( (datype.eq.'r').or.(datype.eq.'R')) then
      darcyx=darcyx*da
```

```
      darcyy=darcyy*da
      darcyz=darcyz*da
      endif

      darcy1=darcyz/darcyy
      darcy2=darcyz/darcyx
      darcy3=darcyy/darcyx
c
c     ... set integer constants ...
c
      nxm1 = nx-1
      nym1 = ny-1
      nzm1 = nz-1
      nxm2 = nx-2
      nym2 = ny-2
      nzm2 = nz-2
c
c     ... and calculate nodal positions and find hmin ...
c
      x(1) = 0.0
      y(1) = 0.0
      z(1) = 0.0
      hmin = 1000.0
c
      do 10 i=1, nxm1
       x(i+1) = x(i)+hx(i)
       if (hx(i).lt.hmin) hmin=hx(i)
   10 continue
c
      do 20 j=1, nym1
       y(j+1) = y(j)+hy(j)
       if (hy(j).lt.hmin) hmin=hy(j)
   20 continue
c
      do 30 k=1, nzm1
       z(k+1) = z(k)+hz(k)
       if (hz(k).lt.hmin) hmin=hz(k)
   30 continue
c
c     ... set time step ...
c
      hmin=arx/nxm1
      if (ary/nym1 .lt. hmin) hmin=ary/nym1
      if (arz/nzm1 .lt. hmin) hmin=arz/nzm1
      dt = hmin*hmin*stabr
c
c     set draw_time_step
c
      timestep=altp*dt*xll*xll/alphap
      if (drawstep.eq.0) drawstep=1
      drawstep=int(real(drawstep)*60./timestep+1)
      drwstp1  =int( 5.*60./timestep+1)
      drwstp2  =int(10.*60./timestep+1)
      drwstp3  =int(15.*60./timestep+1)
      drwstp4  =int(20.*60./timestep+1)
      drwstp5  =int(25.*60./timestep+1)
      drwstp6  =int(30.*60./timestep+1)
      drwstp7  =int(35.*60./timestep+1)
      drwstp8  =int(40.*60./timestep+1)
      drwstp9  =int(45.*60./timestep+1)
      drwstp10=int(50.*60./timestep+1)
      drwstp11=int(55.*60./timestep+1)
      drwstp12=int(60.*60./timestep+1)

      mainiter=int(real(mainiter)*3600./timestep)
      write(*,*)'max_itheta=',mainiter

      if(mainiter.gt.100000) then
      write(*,*)
      write(*,*)
      write(*,*)'Warning!'
      write(*,*)
      write(*,*)'Iteration-number is very large.'
      write(*,*)
      endif
c
c     ... set massege display internal ...
c
      if (numshow.eq.0) then
      if (mainiter.lt.100) then
      numshow=1
        else if (mainiter.lt.500) then
        numshow=10
        else if (mainiter.lt.1000)`then
        numshow=20
             else if (mainiter.lt.3000) then
             numshow=60
        else if (mainiter.lt.5000) then
        numshow=100
             else if (mainiter.lt.10000) then
             numshow=200
             else
             numshow=500
      endif
      endif

c
c     ... set default map size and lower and upper limits ...
c
      if( mnx .eq. 0 ) mnx = 51
       if( mnx .lt. 10) then
          mnx = 10
       else if( mnx .gt. 80 ) then
          mnx =80
       endif
      if( mny .eq. 0 ) mny = mnx*ary/arx +0.5
       if( mny .lt. 10) then
          mny = 10
       else if( mny .gt. 80 ) then
          mny =80
```

```
      endif
      if( mnz .eq. 0 ) mnz = mnx*arz/arx +0.5
       if( mnz .lt. 10) then
          mnz = 10
       else if( mnz .gt. 80 ) then
          mnz =80
       endif
c
c     ... calculate fda constants ...
c
      do 40 i=2, nxm1
       h1     = hx(i-1)
       h2     = hx(i)
       sh1    = h1*h1
       sh2    = h2*h2
       denom  = h1*sh2 + h2*sh1
c
       cx1(i)  = -sh2/denom
       cx2(i)  = (-sh1+sh2)/denom
       cx3(i)  = sh1/denom
       cx4(i)  = (2.0*h2)/denom
       cx5(i)  = (2.0*(-h1-h2))/denom
       cx6(i)  = (2.0*h1)/denom
   40 continue
c
      do 50 j=2, nym1
       h1     = hy(j-1)
       h2     = hy(j)
       sh1    = h1*h1
       sh2    = h2*h2
       denom  = h1*sh2 + h2*sh1
c
       cy1(j)  = -sh2/denom
       cy2(j)  = (-sh1+sh2)/denom
       cy3(j)  = sh1/denom
       cy4(j)  = (2.0*h2)/denom
       cy5(j)  = (2.0*(-h1-h2))/denom
       cy6(j)  = (2.0*h1)/denom
   50 continue
c
      do 60 k=2, nzm1
       h1     = hz(k-1)
       h2     = hz(k)
       sh1    = h1*h1
       sh2    = h2*h2
       denom  = h1*sh2 + h2*sh1
c
       cz1(k)  = -sh2/denom
       cz2(k)  = (-sh1+sh2)/denom
       cz3(k)  = sh1/denom
       cz4(k)  = (2.0*h2)/denom
       cz5(k)  = (2.0*(-h1-h2))/denom
       cz6(k)  = (2.0*h1)/denom
   60 continue
c
      do 70 i=2,nxm1
          do 70 k=2,nzm1
             hx1      = hx(i-1)
             hx2      = hx(i)
             shx1     = hx1*hx1
             shx2     = hx2*hx2
c
             hz1      = hz(k-1)
             hz2      = hz(k)
             shz1     = hz1*hz1
             shz2     = hz2*hz2
c
             denomx   = shx1*hx2+shx2*hx1
             denomz   = shz1*hz2+shz2*hz1
             denom    = denomx*denomz*denomz
             denom2   = denomz*denomz*(shx1-shx2)
c
             cxm1zm1(i,k)     = shx2*shz2*denomz/denom
             cxzm1(i,k)       = shz2*denom2/(denom*denomz)
             cxp1zm1(i,k)     =-shx1*shz2*denomz/denom
             cxm1z(i,k)       = shx2*(shz1-shz2)*denomz/denom
             cxz(i,k)         =-denom2*(shz2-shz1)/(denomz*denom)
             cxp1z(i,k)       =-shx1*(shz1-shz2)*denomz/denom
             cxm1zp1(i,k)     =-shx2*shz1*denomz/denom
             cxzp1(i,k)       =-shz1*denom2/(denomz*denom)
             cxp1zp1(i,k)     = shx1*shz1*denomz/denom
c
   70 continue
c
      do 80 j=2,nym1
          do 80 k=2,nzm1
             hy1      = hy(j-1)
             hy2      = hy(j)
             shy1     = hy1*hy1
             shy2     = hy2*hy2
c
             hz1      = hz(k-1)
             hz2      = hz(k)
             shz1     = hz1*hz1
             shz2     = hz2*hz2
c
             denomy   = shy1*hy2+shy2*hy1
             denomz   = shz1*hz2+shz2*hz1
             denom    = denomy*denomz*denomz
             denom2   = denomz*denomz*(shy1-shy2)
c
             cym1zm1(j,k)     = shy2*shz2*denomz/denom
             cyzm1(j,k)       = shz2*denom2/(denom*denomz)
             cyp1zm1(j,k)     =-shy1*shz2*denomz/denom
             cym1z(j,k)       = shy2*(shz1-shz2)*denomz/denom
             cyz(j,k)         =-denom2*(shz2-shz1)/(denomz*denom)
             cyp1z(j,k)       =-shy1*(shz1-shz2)*denomz/denom
             cym1zp1(j,k)     =-shy2*shz1*denomz/denom
             cyzp1(j,k)       =-shz1*denom2/(denomz*denom)
             cyp1zp1(j,k)     = shy1*shz1*denomz/denom
```

```
c
 80     continue
c
c     ... calculate fda's for temperature boundary conditions ...
c
      h1     = hx(1)
      h2     = hx(2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cxf(1,3)  = -sh1/denom
      cxf(1,2)  = sh1h2/denom
      cxf(1,1)  = (-sh1h2+sh1)/denom
c
      h1     = hx(nxm1)
      h2     = hx(nxm2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cxf(2,3)  = -sh1/denom
      cxf(2,2)  = sh1h2/denom
      cxf(2,1)  = (-sh1h2+sh1)/denom
c
      h1     = hy(1)
      h2     = hy(2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cyf(1,3)  = -sh1/denom
      cyf(1,2)  = sh1h2/denom
      cyf(1,1)  = (-sh1h2+sh1)/denom
c
      h1     = hy(nym1)
      h2     = hy(nym2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cyf(2,3)  = -sh1/denom
      cyf(2,2)  = sh1h2/denom
      cyf(2,1)  = (-sh1h2+sh1)/denom
c
      h1     = hz(1)
      h2     = hz(2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      czf(1,3)  = -sh1/denom
      czf(1,2)  = sh1h2/denom
      czf(1,1)  = (-sh1h2+sh1)/denom
c
      h1     = hz(nzm1)
      h2     = hz(nzm2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      czf(2,3)  = -sh1/denom
      czf(2,2)  = sh1h2/denom
      czf(2,1)  = (-sh1h2+sh1)/denom
c
c     ... calculate fda's for velocity boundary conditions ...
c
      h1     = hx(1)
      h2     = hx(2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cx3lv  = -sh1/denom
      cx2lv  = sh1h2/denom
      cx1lv  = (-sh1h2+sh1)/denom
c
      h1     = hx(nxm1)
      h2     = hx(nxm2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cx1hv  = sh1/denom
      cx2hv  = -sh1h2/denom
      cx3hv  = (sh1h2-sh1)/denom
c
      h1     = hy(1)
      h2     = hy(2)
      sh1    = h1*h1
      h1h2   = h1+h2
      sh1h2  = h1h2*h1h2
      denom  = h1*sh1h2 - h1h2*sh1
c
      cy3lv  = -sh1/denom
      cy2lv  = sh1h2/denom
      cy1lv  = (-sh1h2+sh1)/denom
c
      h1     = hy(nym1)
      h2     = hy(nym2)
      sh1    = h1*h1
```

```
      h1h2  = h1+h2
      sh1h2 = h1h2*h1h2
      denom = h1*sh1h2 - h1h2*sh1
c
      cy1hv = sh1/denom
      cy2hv = -sh1h2/denom
      cy3hv = (sh1h2-sh1)/denom
c
      h1    = hz(1)
      h2    = hz(2)
      sh1   = h1*h1
      h1h2  = h1+h2
      sh1h2 = h1h2*h1h2
      denom = h1*sh1h2 - h1h2*sh1
c
      cz3lv = -sh1/denom
      cz2lv = sh1h2/denom
      cz1lv = (-sh1h2+sh1)/denom
c
      h1    = hz(nzm1)
      h2    = hz(nzm2)
      sh1   = h1*h1
      h1h2  = h1+h2
      sh1h2 = h1h2*h1h2
      denom = h1*sh1h2 - h1h2*sh1
c
      cz1hv = sh1/denom
      cz2hv = -sh1h2/denom
      cz3hv = (sh1h2-sh1)/denom

c
c     ... calculate fda's for vector potential boundary conditions ...
c
      cx5l  = -2.0/(hx(1)*hx(1))
      cx6l  =  1.0/(hx(1)*hx(1))
      cx4h  =  1.0/(hx(nxm1)*hx(nxm1))
      cx5h  = -2.0/(hx(nxm1)*hx(nxm1))
c
      cy5l  = -2.0/(hy(1)*hy(1))
      cy6l  =  1.0/(hy(1)*hy(1))
      cy4h  =  1.0/(hy(nym1)*hy(nym1))
      cy5h  = -2.0/(hy(nym1)*hy(nym1))
c
      cz5l  = -2.0/(hz(1)*hz(1))
      cz6l  =  1.0/(hz(1)*hz(1))
      cz4h  =  1.0/(hz(nzm1)*hz(nzm1))
      cz5h  = -2.0/(hz(nzm1)*hz(nzm1))
c
c     ... constant for vector potential ...
c
      cpp1  = alpp*0.5*dt
c
      sbetax  = sin(0.0174533*betax)
      sbetay  = sin(0.0174533*betay)
      sbetaz  = sin(0.0174533*betaz)
c
      cbetax  = cos(0.0174533*betax)
      cbetay  = cos(0.0174533*betay)
      cbetaz  = cos(0.0174533*betaz)
c
      cczp7  = sbetay*cbetaz + sbetax*cbetay*sbetaz
      cczp6  = -cbetax*sbetaz
      cczp5  = cbetay*cbetaz - sbetay*sbetax*sbetaz
      cczp4  = ra
c
c     ... constants for temperature ...
c
      ctp4=rhof*cpfl/(epson*rhof*cpfl+(1.-epson)*rhos*cpsd)
      ctp3=rhof*cpfl/(epson*rhof*cpfl+(1.-epson)*rhos*cpsd)
      ctp2=altp*0.5*dt*rhof*cpfl/(epson*rhof*cpfl+
     +            (1.-epson)*rhos*cpsd)
      ctp1=altp*0.5*dt*rhof*cpfl/(epson*rhof*cpfl+
     +            (1.-epson)*rhos*cpsd)

      write(*,*) 'Data setting done.'
      return
      end

      subroutine simpsn(fun,nx,hx,answer)
c
c     -----------------------------------------------------------
c    |                                                           |
c    |          *** integration with simpson's rule ***         |
c    |                                                           |
c     -----------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical modd
c
      dimension fun(max),hx(max)
c
      nxm1=nx-1
      nxm2=nx-2
      modd=.true.
      if(2*(nx/2).eq.nx) modd=.false.
      if(modd)then
       mx=nx
      else
       mx=nxm1
      endif
      mx1=mx-1
      mx2=mx-2
      answer=0.
      do 40 i=2,mx1,2
       answer=answer+4.*fun(i)*hx(i-1)
 40   continue
      do 50 i=3,mx2,2
       answer=answer+2.*fun(i)*hx(i-1)
 50   continue
```

```
      answer=(answer+fun(1)*hx(1)+fun(mx)*hx(mx1))/3.
      if(.not. modd)then
       answer=answer+(fun(nxm1)*hx(nxm2)+fun(nx)*hx(nxm1))/2.
      endif
c
      return
      end
      subroutine sumary
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |      *** calculates summary details for flow field ***         |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      common /fileno/inparam,irg,ireport,itin,itout
      common /vel/u(nxmax,nymax,nzmax),
     +            v(nxmax,nymax,nzmax),
     +            w(nxmax,nymax,nzmax)
      common /t/theta(nxmax,nymax,nzmax)
      common /p1/p1(nxmax,nymax,nzmax)
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
c
      write(ireport,2001)
2001  format(1h1,/,
     +  40x,'================================',/,
     +  40x,'summary of flow field information',/,
     +  40x,'================================',/,
     +  ///)
c
      write(ireport,2010)
2010  format(/)
c
      call maxmin(p2,' potential p2  ')
      call maxmin(p3,' potential p3  ')
      write(ireport,2010)
c
      call maxmin(u,'   u velocity  ')
      call maxmin(v,'   v velocity  ')
      call maxmin(w,'   w velocity  ')
c
      return
      end
      subroutine tbc
c
c     ----------------------------------------------------------------
c    |                                                                |
c    |            *** thermal boundary conditions ***                 |
c    |                                                                |
c     ----------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical ixl,ixh,iyl,iyh,izl,izh
c
      common /logtbc/ixl,ixh,iyl,iyh,izl,izh
      common /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +            ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +            zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
      common /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +            iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +            izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /tbfdc/cxf(2,3),cyf(2,3),czf(2,3)
      common /t/theta(nxmax,nymax,nzmax)
c
c     ... apply gradient conditions using: cg(dt/dn)=ca*t+cb
c         resulting in adiabatic / heat flux / convective
c         or isothermal conditions on all boundaries ...
c
c
c      .. on x=0.0 and x=arx walls
c
      do 10 j=1, ny
       do 10 k=1, nz
         theta(1,j,k)  = ( xlcb - xlcg*cxf(1,2)*theta(2,j,k)
     +                          - xlcg*cxf(1,3)*theta(3,j,k) )
     +                  /(xlcg*cxf(1,1) - xlca)

         theta(nx,j,k) = (-xhcb - xhcg*cxf(2,2)*theta(nxm1,j,k)
     +                          - xhcg*cxf(2,3)*theta(nxm2,j,k) )
     +                  /(xhcg*cxf(2,1) + xhca)
10    continue
c
c      .. on y=0.0 and y=ary walls
c
      do 20 i=1, nx
       do 20 k=1, nz
         theta(i,1,k)  = ( ylcb - ylcg*cyf(1,2)*theta(i,2,k)
     +                          - ylcg*cyf(1,3)*theta(i,3,k) )
     +                  /(ylcg*cyf(1,1) - ylca)
c
         theta(i,ny,k) = (-yhcb - yhcg*cyf(2,2)*theta(i,nym1,k)
     +                          - yhcg*cyf(2,3)*theta(i,nym2,k) )
     +                  /(yhcg*cyf(2,1) + yhca)
20    continue
c
c      .. on z=0.0 and z=arz walls
c
      do 30 i=1, nx
       do 30 j=1, ny
         theta(i,j,1)  = ( zlcb - zlcg*czf(1,2)*theta(i,j,2)
     +                          - zlcg*czf(1,3)*theta(i,j,3) )
     +                  /(zlcg*czf(1,1) - zlca)

         theta(i,j,nz) = (-zhcb - zhcg*czf(2,2)*thera(i,j,nzm1)
     +                          - zhcg*czf(2,3)*theta(i,j,nzm2) )
```

```
     +                        /(zhcg*czf(2,1) + zhca)
   30 continue
c
c     ... isothermal condition on x = 0.0 wall ...
c
      if (ixl) then
       do 60 j=ixlyl, ixlyh
          do 60 k=ixlzl, ixlzh
               theta(1,j,k) = xlci
   60    continue
      endif
c
c     ... isothermal condition on x = arx wall ...
c
      if (ixh) then
       do 70 j=ixhyl, ixhyh
          do 70 k=ixhzl, ixhzh
               theta(nx,j,k) = xhci
   70    continue
      endif
c
c     ... isothermal condition on y = 0.0 wall ...
c
      if (iyl) then
       do 80 i=iylxl, iylxh
          do 80 k=iylzl, iylzh
               theta(i,1,k) = ylci
   80    continue
      endif
c
c     ... isothermal condition on y = ary wall ...
c
      if (iyh) then
       do 90 i=iyhxl, iyhxh
          do 90 k=iyhzl, iyhzh
               theta(i,ny,k) = yhci
   90    continue
      endif
c
c     ... isothermal condition on z = 0.0 wall ...
c
      if (izl) then
       do 100 i=izlxl, izlxh
          do 100 j=izlyl, izlyh
               theta(i,j,1) = zlci
  100    continue
      endif
c
c     ... isothermal condition on z = arz wall ...
c
      if (izh) then
       do 110 i=izhxl, izhxh
          do 110 j=izhyl, izhyh
               theta(i,j,nz) = zhci
  110    continue
      endif
c
      return
      end
      subroutine thomas(nn)
c
c     ------------------------------------------------------------------
c     |                                                                |
c     |           *** for solving a tridiagonal system ***             |
c     |                                                                |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      dimension  a(max),b(max),c(max),d(max)
      common  /tom/a,b,c,d
      common  /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common  /itheta/itheta
      dimension  olda(51),oldb(51),oldc(51),oldd(51)
      do 1  i=1,nn
      olda(i)=a(i)
      oldb(i)=b(i)
      oldc(i)=c(i)
    1 oldd(i)=d(i)

      if(b(1)  .eq.  0.)  goto 2
      b(1)=1.0/b(1)
c
      do 10  i=2,nn
      aa=  b(i)-a(i)*b(i-1)*c(i-1)
       if  (abs(aa).eq.0.)  goto 2
      b(i)=1.0/aa
      d(i)=d(i)-a(i)*b(i-1)*d(i-1)
   10 continue
c
      d(nn)=d(nn)*b(nn)
c
      do 20  i=nn-1,1,-1
      d(i)=(  d(i)-d(i+1)*c(i)   )*b(i)
   20 continue
c
      return
    2 write(*,*)  'thomas failed!    itheta =',itheta
      write(*,*)  'part of new a,b,c,d are:'
      do 4  i=1,nn
    4 write(*,*)  a(i),b(i),c(i),d(i)

      write(*,*)  ' old a,b,c,d are as followings.'
      do 3  i=1,nn
    3 write(*,*)  olda(i),oldb(i),oldc(i),oldd(i)
c
      return
      stop
      end
```

```
      subroutine tsolp
c
c     ------------------------------------------------------------------
c    |                                                                  |
c    |        *** solve for new temperature field ***                   |
c    |                                                                  |
c     ------------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      character srcetype*1
      dimension a(max),b(max),c(max),d(max)
      common /alpfs/alpp,altp
      common /dt/dt
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /tconsp/ctp1,ctp2,ctp3,ctp4
      common /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +           cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +           cy4(nymax),cy5(nymax),cy6(nymax)
      common /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +           cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common /vel/u(nxmax,nymax,nzmax),
     +          v(nxmax,nymax,nzmax),
     +          w(nxmax,nymax,nzmax)
      common /t/theta(nxmax,nymax,nzmax)
      common /tom/a,b,c,d
      common /aux/aux(nxmax,nymax,nzmax),aux2d(max,max)
      common /auxct/auxct(nxmax,nymax,nzmax)
      common /test/testp1,testp2,testp3,testt
      common /ccaa/cc0,dtemp,tcold,thot
      common /param_q/qparam(20)
      common /told/told(nxmax,nymax,nzmax)
      common /sumdivt/sumdivt
      common /txty/tpnt(10),q(10),npt,nparam,srcetype
      common /dthpnt/dthpnt(nxmax,nymax,nzmax),thmldthl,thmldthh
c
c     ... implicit in x ...
c
      do 100 j=2, nym1
       do 100 k=2, nzm1
          do 90 i=2, nxm1
c
c      heat source
c
      tfruit=dtemp*theta(i,j,k)+tcold

      if ( (tfruit .ge. thmldthh) .or. (tfruit .le. thmldthl))
     + dthpnt(i,j,k)=0.

            qijk=0.0
            auxtt=1.
            do 500 nterm=1,nparam
            qijk=qijk+qparam(nterm)*auxtt
            auxtt=auxtt*tfruit
500         continue

            if(qijk .lt. 0.0) qijk=0.0
            qijk=qijk*cc0*dthpnt(i,j,k)
c
c
c
            a(i-1)  = ctp1*cx1(i)*u(i-1,j,k)  - ctp2*cx4(i)
            b(i-1)  = ctp1*cx2(i)*u(i,j,k)    - ctp2*cx5(i)  + 1.0
            c(i-1)  = ctp1*cx3(i)*u(i+1,j,k)  - ctp2*cx6(i)
            d(i-1)  = theta(i-1,j,k)*(-ctp3*cx1(i)
     +              * u(i-1,j,k)  + ctp4*cx4(i))
     +              + theta(i,j-1,k)*(-ctp3*cy1(j)
     +              * v(i,j-1,k)  + ctp4*cy4(j))
     +              + theta(i,j,k-1)*(-ctp3*cz1(k)
     +              * w(i,j,k-1)  + ctp4*cz4(k))
     +              + theta(i+1,j,k)*(-ctp3*cx3(i)
     +              * u(i+1,j,k)  + ctp4*cx6(i))
     +              + theta(i,j+1,k)*(-ctp3*cy3(j)
     +              * v(i,j+1,k)  +ctp4*cy6(j))
     +              + theta(i,j,k+1)*(-ctp3*cz3(k)
     +              * w(i,j,k+1)  + ctp4*cz6(k))
     +              + theta(i,j,k)*(-ctp3*cx2(i)*u(i,j,k)
     +                                       + ctp4*cx5(i)
     +              -  ctp3*cy2(j)*v(i,j,k)  + ctp4*cy5(j)
     +              -  ctp3*cz2(k)*w(i,j,k)  + ctp4*cz5(k))
     +       +qijk
c
c      heat source
c
90        continue
c
          call thomas(nxm2)
c
          do 95 i=2, nxm1
             aux(i,j,k)  = d(i-1)
95        continue
100   continue
c
c     ... implicit in y ...
c
      do 200 i=2,nxm1
       do 200 k=2, nzm1
          do 190 j=2, nym1
             a(j-1)  = ctp1*cy1(j)*v(i,j-1,k)  - ctp2*cy4(j)
             b(j-1)  = ctp1*cy2(j)*v(i,j,k)    - ctp2*cy5(j)  + 1.0
             c(j-1)  = ctp1*cy3(j)*v(i,j+1,k)  - ctp2*cy6(j)
             d(j-1)  = aux(i,j,k)
190       continue
c
          call thomas(nym2)
c
          do 195 j=2, nym1
             aux(i,j,k)  = d(j-1)
195       continue
```

```
  200 continue
c
c     ... implicit in z ...
c
       do 300 i=2, nxm1
        do 300 j=2, nym1
           do 290 k=2, nzm1
               a(k-1) = ctp1*cz1(k)*w(i,j,k-1) - ctp2*cz4(k)
               b(k-1) = ctp1*cz2(k)*w(i,j,k)   - ctp2*cz5(k) + 1.0
               c(k-1) = ctp1*cz3(k)*w(i,j,k+1) - ctp2*cz6(k)
               d(k-1) = aux(i,j,k)
  290        continue
c
           call thomas(nzm2)
c
           do 295 k=2, nzm1
               aux(i,j,k) = d(k-1)
  295        continue
  300 continue
c
c     ... update theta ...
c
       testt = 0.0
       tmax  = 0.0
       do 400 i=2, nxm1
        do 400 j=2, nym1
           do 400 k=2, nzm1
               testt      = testt + abs(aux(i,j,k))
               theta(i,j,k) = theta(i,j,k) + dt*altp*aux(i,j,k)
              abst        = abs(theta(i,j,k))
               if (tmax.lt.abst) tmax = abst
  400 continue
       if (tmax.gt.0.001) testt = testt/(nxm2*nym2*nzm2*tmax)
c
c
c     to make temperature steady state
c
       sumdif = 0.0
       sumall = 0.00001
       do 710 i=2,nxm1
        do 710 j=2,nym1
           do 710 k=2,nzm1
           sumdif = sumdif + abs(told(i,j,k)-theta(i,j,k))
           sumall = sumall + abs(theta(i,j,k))
           told(i,j,k) = theta(i,j,k)
  710 continue
       sumdivt = sumdif/sumall
c
c         write (*,730) ((dthpnt(i,j,((nz+1)/2)),i=2,nxm1),j=2,nym1)
c 730  format (1x, 9f4.1)

       return
       end
       subroutine ttime(start,runng,runend)
c
c      -----------------------------------------------------------
c     |                                                           |
c     |   *** program starting, running and ending time ***       |
c     |                                                           |
c      -----------------------------------------------------------
c
      logical  start,runng,runend
      character   day*8,colon*1,year*10,exprmnt*8
      character  timestrg*11,datestrg*9
      integer  second10,second1,second
      integer  date,hour,timestrt,timerun,timecal,timeend
      common  /fileno/inparam,irg,ireport,itin,itout
      common  /time/date,hour,minute10,minute1,second10,second1
      common  /day/day,colon,year
      common  /timestrt/timestrt,timerun,timecal
      common  /exprmnt/exprmnt
      common  /timestrg/timestrg,datestrg


      second1  =ichar(timestrg(8:8))  - 48
      second10=ichar(timestrg(7:7))  - 48
      minute1  =ichar(timestrg(5:5))  - 48
      minute10=ichar(timestrg(4:4))  - 48
      khour1   =ichar(timestrg(2:2))  - 48
      khour10  =ichar(timestrg(1:1))  - 48
      kdate1   =ichar(datestrg(2:2))  - 48
      kdate10  =ichar(datestrg(1:2))  - 48

      date = 10*kdate10  + kdate1
      hour = 10*khour10  + khour1

      if (start) then
         timestrt=10*second10+second1+60*(minute10*10+minute1)
     +           +3600*(hour+24*date)

         write(*,30) timestrg,datestrg
         write(ireport,30) timestrg,datestrg
30        format(1x,'program is started at ',a11,' ',a9)


        else if (runng) then
           timerun=10*second10+second1+60*(minute10*10+minute1)
     +          +3600*(hour+24*date)
          timerun=timerun-timestrt
             else if (runend) then

             write(ireport,40)timestrg,datestrg
40              format('program is terminated at ',a11,' ',a9)

             timeend=10*second10+second1+60*(minute10*10+minute1)
     +          +3600*(hour+24*date)
             timeend=timeend-timestrt
             hour=nint(real(timeend)/3600.-0.5)
             if (hour .lt. 0) hour=0
             minute=nint(real(timeend-3600*hour)/60.-0.5)
```

```
            if (minute .lt. 0) minute=0
            minute10=nint(real(minute)/10.-0.5)
            if (minute10 .lt. 0) minute10=0
            minute1=minute-minute10*10
            second=timeend-3600*hour-60*minute
            second10=nint(real(second)/10.-0.5)
            if (second10 .lt. 0) second10=0
            second1=second-10*second10
      write(ireport,10)hour,minute10,minute1,second10,second1
      write(*,*)
      write(*,10)hour,minute10,minute1,second10,second1
 10         format(1x,'running time =',i4,':',i1,i1,':',i1,i1)
      endif
      return
      end

      subroutine vbc
c
c     -------------------------------------------------------------
c    |                                                             |
c    |        *** calculate velocity at boundary points ***        |
c    |                                                             |
c     -------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +             cx4(nxmax),cx5(nxmax),cx6(nxmax)
      common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +             cy4(nymax),cy5(nymax),cy6(nymax)
      common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +             cz4(nzmax),cz5(nzmax),cz6(nzmax)
      common  /vel/u(nxmax,nymax,nzmax),
     +            v(nxmax,nymax,nzmax),
     +            w(nxmax,nymax,nzmax)
      common  /p1/p1(nxmax,nymax,nzmax)
      common  /p2/p2(nxmax,nymax,nzmax)
      common  /p3/p3(nxmax,nymax,nzmax)
      common  /cxv/cx1lv,cx2lv,cx3lv,cx1hv,cx2hv,cx3hv
      common  /cyv/cy1lv,cy2lv,cy3lv,cy1hv,cy2hv,cy3hv
      common  /czv/cz1lv,cz2lv,cz3lv,cz1hv,cz2hv,cz3hv
c
c   x=0 and x=1 planes, ie, i=1 and i=nx
c
       do 10 j=2, nym1
          do 10 k=2, nzm1
c
            v(1,j,k) = - cx1lv*p3(1,j,k)
     +                 - cx2lv*p3(2,j,k)
     +                 - cx3lv*p3(3,j,k)
     +                 + cz1(k)*p1(1,j,k-1)
     +                 + cz2(k)*p1(1,j,k)
     +                 + cz3(k)*p1(1,j,k+1)
c
            w(1,j,k) = cx1lv*p2(1,j,k)
     +               + cx2lv*p2(2,j,k)
     +               + cx3lv*p2(3,j,k)
     +               - cy1(j)*p1(1,j-1,k)
     +               - cy2(j)*p1(1,j,k)
     +               - cy3(j)*p1(1,j+1,k)
c
            v(nx,j,k) = - cx1hv*p3(nxm2,j,k)
     +                  - cx2hv*p3(nxm1,j,k)
     +                  - cx3hv*p3(nx  ,j,k)
     +                 + cz1(k)*p1(nx,j,k-1)
     +                 + cz2(k)*p1(nx,j,k)
     +                 + cz3(k)*p1(nx,j,k+1)
c
            w(nx,j,k) = cx1hv*p2(nxm2,j,k)
     +                + cx2hv*p2(nxm1,j,k)
     +                + cx3hv*p2(nx  ,j,k)
     +                - cy1(j)*p1(nx,j-1,k)
     +                - cy2(j)*p1(nx,j,k)
     +                - cy3(j)*p1(nx,j+1,k)
c
 10 continue
c
c
c   y=0 and y=1, ie, j=1 and j=ny planes
c
c
       do 20 i=2, nxm1
        do 20 k=2, nzm1
            u(i,1,k) = cy1lv*p3(i,1,k)
     +               + cy2lv*p3(i,2,k)
     +               + cy3lv*p3(i,3,k)
     +               - cz1(k)*p2(i,1,k-1)
     +               - cz2(k)*p2(i,1,k  )
     +               - cz3(k)*p2(i,1,k+1)
c
            w(i,1,k) = cx1(i)*p2(i-1,1,k)
     +               + cx2(i)*p2(i  ,1,k)
     +               + cx3(i)*p2(i+1,1,k)
     +               - cy1lv*p1(i,1,k)
     +               - cy2lv*p1(i,2,k)
     +               - cy3lv*p1(i,3,k)
c
            u(i,ny,k) = cy1hv*p3(i,nym2,k)
     +                + cy2hv*p3(i,nym1,k)
     +                + cy3hv*p3(i,ny  ,k)
     +               - cz1(k)*p2(i,ny,k-1)
     +               - cz2(k)*p2(i,ny,k  )
     +               - cz3(k)*p2(i,ny,k+1)
c
            w(i,ny,k) = cx1(i)*p2(i-1,ny,k)
     +                + cx2(i)*p2(i  ,ny,k)
     +                + cx3(i)*p2(i+1,ny,k)
     +               - cy1hv*p1(i,nym2,k)
     +               - cy2hv*p1(i,nym1,k)
```

```
     +                     - cy3hv*p1(i,ny   ,k)
c
   20 continue
c
c
c  z=0 and z=1 planes, ie, k=1 and k=nz
c
c
c
         do 30 i=2, nxm1
          do 30 j=2, nym1
                u(i,j,1)  = cy1(j)*p3(i,j-1,1)
     +                    + cy2(j)*p3(i,j   ,1)
     +                    + cy3(j)*p3(i,j+1,1)
     +                    - cz1lv*p2(i,j,1)
     +                    - cz2lv*p2(i,j,2)
     +                    - cz3lv*p2(i,j,3)
c
                v(i,j,1)  = - cx1(i)*p3(i-1,j,1)
     +                      - cx2(i)*p3(i   ,j,1)
     +                      - cx3(i)*p3(i+1,j,1)
     +                      + cz1lv*p1(i,j,1)
     +                      + cz2lv*p1(i,j,2)
     +                      + cz3lv*p1(i,j,3)
c
                u(i,j,nz)  = cy1(j)*p3(i,j-1,nz)
     +                     + cy2(j)*p3(i,j   ,nz)
     +                     + cy3(j)*p3(i,j+1,nz)
     +                     - cz1hv*p2(i,j,nzm2)
     +                     - cz2hv*p2(i,j,nzm1)
     +                     - cz3hv*p2(i,j,nz   )
c
                v(i,j,nz)  = - cx1(i)*p3(i-1,j,nz)
     +                       - cx2(i)*p3(i   ,j,nz)
     +                       - cx3(i)*p3(i+1,j,nz)
     +                     + cz1hv*p1(i,j,nzm2)
     +                     + cz2hv*p1(i,j,nzm1)
     +                     + cz3hv*p1(i,j,nz   )
   30 continue
c
c
c     along z axis conjuctions
c
c
          do 40 k=2, nzm1
c
c     i=1 and j=1, u=v=0
c
                w(1,1,k)  = cx1lv*p2(1,1,k)
     +                    + cx2lv*p2(2,1,k)
     +                    + cx3lv*p2(3,1,k)
     +                    - cy1lv*p1(1,1,k)
     +                    - cy2lv*p1(1,2,k)
     +                    - cy3lv*p1(1,3,k)
c
c     i=1 and j=ny u=v=0
c
c
                w(1,ny,k)  = cx1lv*p2(1,ny,k)
     +                     + cx2lv*p2(2,ny,k)
     +                     + cx3lv*p2(3,ny,k)
     +                     - cy1hv*p1(1,nym2,k)
     +                     - cy2hv*p1(1,nym1,k)
     +                     - cy3hv*p1(1,ny   ,k)
c
c
c     i=nx and j=1  u=v=0
c
                w(nx,1,k)  = cx1hv*p2(nxm2,1,k)
     +                     + cx2hv*p2(nxm1,1,k)
     +                     + cx3hv*p2(nx   ,1,k)
     +                     - cy1lv*p1(nx,1,k)
     +                     - cy2lv*p1(nx,2,k)
     +                     - cy3lv*p1(nx,3,k)
c
c     i=nx and j=ny u=v=0
c
                w(nx,ny,k)  =   cx1hv*p2(nxm2,ny,k)
     +                      + cx2hv*p2(nxm1,ny,k)
     +                      + cx3hv*p2(nx   ,ny,k)
     +                      - cy1hv*p1(nx,nym2,k)
     +                      - cy2hv*p1(nx,nym1,k)
     +                      - cy3hv*p1(nx,ny   ,k)
   40    continue
c
c
c    along x axis conjunctions
c
c
          do 50 i=2, nxm1
c
c
c
c    j=1 and k=1,  v=w=0
c
c
                u(i,1,1)  = cy1lv*p3(i,1,1)
     +                    + cy2lv*p3(i,2,1)
     +                    + cy3lv*p3(i,3,1)
     +                    - cz1lv*p2(i,1,1)
     +                    - cz2lv*p2(i,1,2)
     +                    - cz3lv*p2(i,1,3)
c
c
c    j=1 and k=nz  v=w=0
c
c
c
                u(i,1,nz)  = cy1lv*p3(i,1,nz)
     +                     + cy2lv*p3(i,2,nz)
```

```
     +                       + cy3lv*p3(i,3,nz)
     +                       - cz1hv*p2(i,1,nzm2)
     +                       - cz2hv*p2(i,1,nzm1)
     +                       - cz3hv*p2(i,1,nz  )
c
c
c
c    j=ny and k=1   v=w=0
c
c
c
               u(i,ny,1)  = cy1hv*p3(i,nym2,1)
     +                      + cy2hv*p3(i,nym1,1)
     +                      + cy3hv*p3(i,ny  ,1)
     +                      - cz1lv*p2(i,ny,1)
     +                      - cz2lv*p2(i,ny,2)
     +                      - cz3lv*p2(i,ny,3)
c
c
c    j=ny and k=nz   v=w=0
c
c
c
               u(i,ny,nz)  = cy1hv*p3(i,nym2,nz)
     +                       + cy2hv*p3(i,nym1,nz)
     +                       + cy3hv*p3(i,ny  ,nz)
     +                       - cz1hv*p2(i,ny,nzm2)
     +                       - cz2hv*p2(i,ny,nzm1)
     +                       - cz3hv*p2(i,ny,nz  )
 50     continue
c
c
c    along  y axis
c
c
c
        do 60 j=2, nym1
c
c
c    i=1 and k=1    u=w=0
c
c
               v(1,j,1)  = - cx1lv*p3(1,j,1)
     +                     - cx2lv*p3(2,j,1)
     +                     - cx3lv*p3(3,j,1)
     +                     + cz1lv*p1(1,j,1)
     +                     + cz2lv*p1(1,j,2)
     +                     + cz3lv*p1(1,j,3)
c
c
c    i=1 and k=nz   u=w=0
c
c
               v(1,j,nz)  = - cx1lv*p3(1,j,nz)
     +                      - cx2lv*p3(2,j,nz)
     +                      - cx3lv*p3(3,j,nz)
     +                      + cz1hv*p1(1,j,nzm2)
     +                      + cz2hv*p1(1,j,nzm1)
     +                      + cz3hv*p1(1,j,nz)
c
c
c    i=nx and k=1    u=w=0
c
c
               v(nx,j,1)  = - cx1hv*p3(nxm2,j,1)
     +                      - cx2hv*p3(nxm1,j,1)
     +                      - cx3hv*p3(nx  ,j,1)
     +                      + cz1lv*p1(nx,j,1)
     +                      + cz2lv*p1(nx,j,2)
     +                      + cz3lv*p1(nx,j,3)
c
c
c
c    i=nx and k=nz     u=w=0
c
c
               v(nx,j,nz)  = - cx1hv*p3(nxm2,j,nz)
     +                       - cx2hv*p3(nxm1,j,nz)
     +                       - cx3hv*p3(nx  ,j,nz)
     +                       + cz1hv*p1(nx,j,nzm2)
     +                       + cz2hv*p1(nx,j,nzm1)
     +                       + cz3hv*p1(nx,j,  nz)
 60    continue
       return
       end
       subroutine veloc
c
c      -----------------------------------------------------------------
c     |                                                                 |
c     |         *** calculate velocity at internal points ***           |
c     |                                                                 |
c      -----------------------------------------------------------------
c
       parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
       common  /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
       common  /cx/cx1(nxmax),cx2(nxmax),cx3(nxmax),
     +            cx4(nxmax),cx5(nxmax),cx6(nxmax)
       common  /cy/cy1(nymax),cy2(nymax),cy3(nymax),
     +            cy4(nymax),cy5(nymax),cy6(nymax)
       common  /cz/cz1(nzmax),cz2(nzmax),cz3(nzmax),
     +            cz4(nzmax),cz5(nzmax),cz6(nzmax)
       common  /vel/u(nxmax,nymax,nzmax),
     +          v(nxmax,nymax,nzmax),
     +          w(nxmax,nymax,nzmax)
       common  /p1/p1(nxmax,nymax,nzmax)
       common  /p2/p2(nxmax,nymax,nzmax)
       common  /p3/p3(nxmax,nymax,nzmax)
c
       do 10 i=2, nxm1
```

```
      do 10 j=2, nym1
        do 10 k=2, nzm1
           u(i,j,k)  = cy1(j)*p3(i,j-1,k)
     +                + cy2(j)*p3(i,j  ,k)
     +                + cy3(j)*p3(i,j+1,k)
     +                - cz1(k)*p2(i,j,k-1)
     +                - cz2(k)*p2(i,j,k  )
     +                - cz3(k)*p2(i,j,k+1)
c
           v(i,j,k)  =- cx1(i)*p3(i-1,j,k)
     +                 - cx2(i)*p3(i  ,j,k)
     +                 - cx3(i)*p3(i+1,j,k)
     +                 + cz1(k)*p1(i,j,k-1)
     +                 + cz2(k)*p1(i,j,k  )
     +                 + cz3(k)*p1(i,j,k+1)
c
           w(i,j,k)  = cx1(i)*p2(i-1,j,k)
     +                + cx2(i)*p2(i  ,j,k)
     +                + cx3(i)*p2(i+1,j,k)
     +                - cy1(j)*p1(i,j-1,k)
     +                - cy2(j)*p1(i,j  ,k)
     +                - cy3(j)*p1(i,j+1,k)
   10 continue
c
      return
      end
      subroutine writer
c
c     ---------------------------------------------------------------
c    |                                                               |
c    |           *** write results to binary files ***               |
c    |                                                               |
c     ---------------------------------------------------------------
c
      parameter(nxmax=51,nymax=51,nzmax=51,max=51)
c
      logical lfile
      logical blup,cgd
      logical reed,rite,strt,dist,heat,nuss,tran
      logical rt,rp1,rp2,rp3,rmax
      logical divv,divp,dmax
      logical vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +        mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +        vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +        myp1,myp2,myp3,myt,myu,myv,myw,
     +        vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +        mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      logical ixl,ixh,iyl,iyh,izl,izh
c
      character title*80,otitle*80,filename*12
      character xc*1,yc*1,zc*1,vpform*1
      character meshtp*1,exprmnt*8,gridfile*8,srcetype*1,datype*1
c
      integer drawstep
      integer pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
     +        ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +        ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +        ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +        ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      real keff
c
      common /title/title,otitle
      common /tapcom/rfile(94),ifile(67),lfile(10)
      common /exprmnt/exprmnt
      common /fileno/inparam,irg,ireport,itin,itout
      common /logi/blup,cgd
      common /logc/reed,rite,strt,dist,heat,nuss,tran
      common /logr/rt,rp1,rp2,rp3,rmax
      common /logd/divv,divp,dmax
      common /logpvm/vxp1,vxp2,vxp3,vxt,vxu,vxv,vxw,
     +               mxp1,mxp2,mxp3,mxt,mxu,mxv,mxw,
     +               vyp1,vyp2,vyp3,vyt,vyu,vyv,vyw,
     +               myp1,myp2,myp3,myt,myu,myv,myw,
     +               vzp1,vzp2,vzp3,vzt,vzu,vzv,vzw,
     +               mzp1,mzp2,mzp3,mzt,mzu,mzv,mzw
      common /logtbc/ixl,ixh,iyl,iyh,izl,izh
      common /reacom/re,ra,pr,arx,ary,arz,betax,betay,betaz,stabr,ccp,
     +               cct,tmax,pdist,rtime
      common /alpfs/alpp,altp
      common /intcom/itercmpo,norun,mainiter
      common /numcom/numit
      common /tbr/xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +            ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +            zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb
      common /tbi/ixlyl,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +            iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +            izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh
      common /velbc/xlu,xlv,xlw,xhu,xhv,xhw,
     +              ylu,ylv,ylw,yhu,yhv,yhw,
     +              zlu,zlv,zlw,zhu,zhv,zhw
      common /mesh/meshtp
      common /geom/nx,ny,nz,nxm1,nym1,nzm1,nxm2,nym2,nzm2
      common /h/hx(max),hy(max),hz(max),
     +          hxold(max),hyold(max),hzold(max)
      common /axis/x(max),y(max),z(max)
      common /vmpfrm/xc,yc,zc,vpform
      common /vmpnum/npx,npy,npz
      common /vmploc/mx(nxmax),my(nymax),mz(nzmax)
      common /vmppos/rmx(nxmax),rmy(nymax),rmz(nzmax)
      common /mapcom/mnx,mny,mnz
      common /phierr/phierr
      common /da/da
      common /porous/alphaf,alphap,cpfl,cppr,cpsd,rhof,rhop,rhos
      common /epsonnu/epson,xll,annu,gravity,ddd,keff
      common /ccaa/cc0,dtemp,tcold,thot
      common /draw/drawstep
      common /pst/pstx,psty,pstz
      common /pstxyz/pstx1,psty1,pstz1
     +        ,pstx2,psty2,pstz2,pstx3,psty3,pstz3
```

```
     +         ,pstx4,psty4,pstz4,pstx5,psty5,pstz5
     +         ,pstx6,psty6,pstz6,pstx7,psty7,pstz7
     +         ,pstx8,psty8,pstz8,pstx9,psty9,pstz9
     +         ,pstx10,psty10,pstz10,pstx11,psty11,pstz11
      common /param_q/qparam(20)
      common /numshow/numshow
      common /iterpp/iterpp
      common /darcy/darcyx,darcyy,darcyz,darcy1,darcy2,darcy3
      common /tstrt/tstrt
      common /gridfile/gridfile
      common /txty/tpnt(10),q(10),npt,nparam,srcetype
      common /datype/datype
      common /vel/u(nxmax,nymax,nzmax),
     +            v(nxmax,nymax,nzmax),
     +            w(nxmax,nymax,nzmax)
      common /t/theta(nxmax,nymax,nzmax)
      common /p1/p1(nxmax,nymax,nzmax)
      common /p2/p2(nxmax,nymax,nzmax)
      common /p3/p3(nxmax,nymax,nzmax)
      common /dt/dt
c
c     ... record total number of iterations and total cpu time ...
c
      itnum = numit
      seccpu= rtime
      if( norun .eq. ifile(1) ) then
       itnum =itnum  + ifile(10)
       seccpu=seccpu + rfile(28)
      endif
c
c     ... write to binary file ...
c
      filename=exprmnt // '.out'
      open(unit=itout,file=filename ,status='unknown',
     +                                  form='unformatted')
      rewind itout
      write(itout) title,meshtp,exprmnt,srcetype,datype,gridfile,
c  ... write rfile ....
     +                 keff,alphaf,cpf1,cpsd,rhof,rhos,epson,
     +                 xl1,annu,gravity,ddd,tcold,thot,tstrt,
     +                 arx,ary,arz,betax,betay,betaz,
     +               stabr,alpp,altp,phierr,ccp,cct,dt,seccpu,ra,
     +               (tpnt(i),i=1,10),(q(i),i=1,10),
     +              darcyx,darcyy,darcyz,
     +               xlci,xlcg,xlca,xlcb,xhci,xhcg,xhca,xhcb,
     +               ylci,ylcg,ylca,ylcb,yhci,yhcg,yhca,yhcb,
     +               zlci,zlcg,zlca,zlcb,zhci,zhcg,zhca,zhcb,
     +              xlu,xlv,xlw,xhu,xhv,xhw,
     +              ylu,ylv,ylw,yhu,yhv,yhw,
     +              zlu,zlv,zlw,zhu,zhv,zhw,
c  ... write ifile ....
     +              norun,nx,ny,nz,itercmpo,iterpp,mainiter,
     +              npt,nparam,itnum,pstx1,psty1,pstz1,
     +              pstx2,psty2,pstz2,pstx3,psty3,pstz3,
     +              pstx4,psty4,pstz4,pstx5,psty5,pstz5,
     +              pstx6,psty6,pstz6,pstx7,psty7,pstz7,
     +              pstx8,psty8,pstz8,pstx9,psty9,pstz9,
     +              pstx10,psty10,pstz10,pstx11,psty11,pstz11,
     +              ixly1,ixlyh,ixlzl,ixlzh,ixhyl,ixhyh,ixhzl,ixhzh,
     +              iylxl,iylxh,iylzl,iylzh,iyhxl,iyhxh,iyhzl,iyhzh,
     +              izlxl,izlxh,izlyl,izlyh,izhxl,izhxh,izhyl,izhyh,
c  ... write lfile ....
     +             cgd,dist,heat,tran,
     +             ixl,ixh,iyl,iyh,izl,izh
c
c     ... write mesh to binary file ...
c
      write(itout) ( hx(i),  i=1,  nx-1)
      write(itout) ( hy(j),  j=1,  ny-1)
      write(itout) ( hz(k),  k=1,  nz-1)
c
c     ... write to binary file ...
c
      write(itout) (((    p1(i,j,k),  i=1,  nx),  j=1,  ny),  k=1,  nz)
      write(itout) (((    p2(i,j,k),  i=1,  nx),  j=1,  ny),  k=1,  nz)
      write(itout) (((    p3(i,j,k),  i=1,  nx),  j=1,  ny),  k=1,  nz)
      write(itout) (((theta(i,j,k),  i=1,  nx),  j=1,  ny),  k=1,  nz)
c
      endfile itout
c
      close(itout)
      return
      end
CCCCCC End of program AGRI_3D.FOR (7785 lines) CCCCCCCC
```

Input file INPARAM {SAMPLE}

```
* --- {Descriptions of this job. Up to 80 letters.} --- *
Apples, size.28h.281.28w, adiabatic floor, cooling from 30.
*--KEFF--* *-ALPHAF-* *--CPFL--* *--CPSD--* *--RHOF--* *--RHOS--*
 0.291E-03 2.0000E-05 1.0056E-00 3.7240E+00 1.2339E-00 1.0660E+03
*---EPSON---* *----XLL----* *---ANNU----* *--GRAVITY--* *----DDD----*
  0.383208200   0.280000000   0.000014194   9.800000000   0.067000000
*---TCOLD---* *---THOT----* *---TSTRT---* *COLD, HOT WALL, INITIAL TEMP.*
 0.0000000000 30.000000000 30.000000000
*--ARX---* *--ARY---* *--ARZ---* *-BETAX--* *-BETAY--* *-BETAZ--*{GEOM}
      1.0        1.0        1.0        0.0        0.0        0.0
*MESHTYPE* *-NX* *-NY* *-NZ* {MESH: U=UNI, N=NON-UNI, B=BINARY, T=TRANSF}
        U    11    11    11
*--STABR-* *--ALPP--* *--ALTP--* *-PHIERR-* *---CCP--* *---CCT--*
    20.00        3.0        5.0 1.0000E-03  1.000E-04  1.000E-06
*ITERCMPO* *-ITERPP-* *MAINITER* *DRAWSTEP*
         2          2         60         30
*X* *Y* *Z*    *X* *Y* *Z*    *X* *Y* *Z* {POSITIONS FOR OUTPUT TRANSIENT
  1   7   6      2   7   6      3   7   6              RESULTS, 11 POINTS}
*X* *Y* *Z*    *X* *Y* *Z*    *X* *Y* *Z*
  4   7   6      5   7   6      6   7   6
*X* *Y* *Z*    *X* *Y* *Z*    *X* *Y* *Z*
  7   7   6      8   7   6      9   7   6
*X* *Y* *Z*    *X* *Y* *Z*
 10   7   6     11   7   6
*---* {RUN NUMBER}
    1
REED* RITE* STRT* DIST* HEAT* NUSS*
    F     T     F     F     T     T
*MNX* *MNY* *MNZ* {IF 0, SCALE USING AR'S: DEF'LT MNX=51}    *-PDIST--*
    0     0     0                                                  0.2
C
TEMPERATURE BOUNDARY CONDITIONS
I=1 (IE. X=0.0)                          I=NX (IE. X=ARX)
*PATCH<T/F> *-CI* YL* YH* ZL* ZH*        *PATCH<T/F> *-CI* YL* YH* ZL* ZH*
T             0.00   1  11   1  11       F             0.00   1  11   1  11
*-G-* *-A-* *-B-*{G(DT/DX)=A*T+B}        *-G-* *-A-* *-B-*{G(DT/DX)=A*T+B}
 0.00  1.00  0.00                         1.00  0.00  0.00
J=1 (IE. Y=0.0)                          J=NY (IE. Y=ARY)
*PATCH<T/F> *-CI* XL* XH* ZL* ZH*        *PATCH<T/F> *-CI* XL* XH* ZL* ZH*
T             0.00   1  11   1  11       T             0.00   1  11   1  11
*-G-* *-A-* *-B-*{G(DT/DY)=A*T+B}        *-G-* *-A-* *-B-*{G(DT/DY)=A*T+B}
 0.00  1.00  0.00                         0.00  1.00  0.00
K=1 (IE. Z=0.0)                          K=NZ (IE. Z=ARZ)
*PATCH<T/F> *-CI* XL* XH* YL* YH*        *PATCH<T/F> *-CI* XL* XH* YL* YH*
T             0.00   1  11   1  11       T             0.00   1  11   1  11
*-G-* *-A-* *-B-*{G(DT/DZ)=A*T+B}        *-G-* *-A-* *-B-*{G(DT/DZ)=A*T+B}
 0.00  1.00  0.00                         0.00  1.00  0.00
C
VELOCITY BOUNDARY CONDITIONS
I=1                                      I=NX
*-U-* *-V-* *-W-*                        *-U-* *-V-* *-W-*
   NA   0.0   0.0                           NA   0.0   0.0
J=1                                      J=NY
*-U-* *-V-* *-W-*                        *-U-* *-V-* *-W-*
  0.0    NA   0.0                          0.0    NA   0.0
K=1                                      K=NZ
*-U-* *-V-* *-W-*                        *-U-* *-V-* *-W-*
  0.0   0.0    NA                          0.0   0.0    NA
C
PRINTING FORM AND PRINTING CONTROLS :
*---* {<I>NTEGER NUMBERING OR <C>OORDINATE NUMBERING}
    I
VAR-X:  P1* P2* P3* T-* U-* V-* W-*   X=CONSTANT (Y-Z) PLANES
          T   T   T   T   T   T   T
MAP-X:  P1* P2* P3* T-* U-* V-* W-*
          T   T   T   T   T   T   T
*---*{<P>LANE NOS. OR <C>OORD. GIVEN}     *---* {NUMBER OF PLANES}
    P                                         3
*X1-* *X2-* *X3-* *X4-* *X5-* *X6-* *X7-* *X8-* *X9-* *X10* *X11*
  03     6     9   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
VAR-Y:  P1* P2* P3* T-* U-* V-* W-*   Y=CONSTANT (X-Z) PLANES
          T   T   T   T   T   T   T
MAP-Y:  P1* P2* P3* T-* U-* V-* W-*
          T   T   T   T   T   T   T
*---*{<P>LANE NOS. OR <C>OORD. GIVEN}     *---* {NUMBER OF PLANES}
    P                                        01
*Y1-* *Y2-* *Y3-* *Y4-* *Y5-* *Y6-* *Y7-* *Y8-* *Y9-* *Y10* *Y11*
   06     0     0     0     0     0     0     0     0     0     0
VAR-Z:  P1* P2* P3* T-* U-* V-* W-*   Z=CONSTANT (X-Y) PLANES
          T   T   T   T   T   T   T
MAP-Z:  P1* P2* P3* T-* U-* V-* W-*
          T   T   T   T   T   T   T
*---*{<P>LANE NOS. OR <C>OORD. GIVEN}     *---* {NUMBER OF PLANES}
    P                                        01
*Z1-* *Z2-* *Z3-* *Z4-* *Z5-* *Z6-* *Z7-* *Z8-* *Z9-* *Z10* *Z11*
   06     0     0     4     5     6     7     8     9    10    11
C
*RT-* *RP1* *RP2* *RP3* RMAX*  <-- {RESIDUALS}
    T     T     T     T     T
C
DIVV* DIVP* DMAX*  <-- {DIVERGENCE}
    T     T     T
*------*{EXPRMNT. There must be no space and ONLY 8 characters.}
sample__
{npt,nparam, & srcetype. Number of points,parameters & type of source}
*--* *--*     *  {c or C for mg_Co2/Kg h & w or W for w/mmm }
   8    7     c
*--tpnt(1)--* *--tpnt(2)--* *--tpnt(3)--* *--tpnt(4)--* *--tpnt(5)--*
         -1.0           0.0           5.0          10.0          15.0
*--tpnt(6)--* *--tpnt(7)--* *--tpnt(8)--* *--tpnt(9)--* *-tpnt(10)--*
         20.0          25.0          34.0          40.0          45.0
*----Q(1)---* *----Q(2)---* *----Q(3)---* *----Q(4)---* *----Q(5)---*
          0.0           4.5           8.0          17.0          30.0
*----Q(6)---* *----Q(7)---* *----Q(8)---* *----Q(9)---* *---Q(10)---*
         41.0          50.0          57.0           0.0           0.0
*-*{numshow: 0=auto }
  0
*-DARCYX-* *-DARCYY-* *-DARCYZ-*     * {r or R = ratio, v or V = values}
       1.0        1.0        1.0     r
*------* {gridfile name, no space and ONLY 8 characters.}
gridfile
-------------- END OF DATA SECTION --------------
```

```
Input variables for file 'INPARAM' are described below.

variable                     remarks
--------                     -------

title           To be printed at the top of the job report.
keff            Effective thermal conductity of packed fruits
                or vegetables.
alphaf          Thermal diffusivity of air.
cpfl            Specific heat of air.
cpsd            Specific heat of fruits or vegetables.
rhof            Density of air.
rhos            Density of fruits or vegetables.
epson           Void fraction of porous media.
xll             Height of the box.
annu            Viscosity of air.
gravity         Gravity force.
ddd             Diameter of fruit.
tcold           Cold wall temperature.
thot            Hot wall temperature.
tstrt           Initial temperature in porous media.

arx,ary,arz     Aspect ratios in x-, y- and z- directions.
                 {One of these aspect ratios must be = 1}
betax,betay,    Rotation of box about x-axis , y-axis and
betaz           z-axis respectively (using a right hand rule).

meshtp          Type of mesh used. options include:
                  u: Uniform mesh
                  n: Non-uniform (specified in file 'GRIDFILE')
                  b: Read from previously written binary file
                  t: Transform (using grid in file 'GRIDFILE')
nx,ny,nz        Number of mesh points in x, y and z directions.
                 {not used for non-uniform or previous mesh.}

stabr           Stability ratio (delta t / delta h(min)**2).
alpp,altp       False transient factors.
phierr          Make steady-state to vector potential components
                in every iteration.
ccp,cct         Convergence critera for psi and theta.

itercmpo        Maximum number of iterations for each
                component of vector potential.
iterpp          Maximum number of iterations for all components
                to reach their steady state.
mainiter        Maximum number for main iterations.
drawstep        Time interal for output results.
pstx1 to pstz11 Positions in the container for transient
                temperature and velosity results.

norun           Run number - {change it to reset accumulated
                              iteration count and CPU usage}

reed            t: Read the starting values from  binary file.
                f: Fields initialised in subroutine INITAL.
rite            t: Write the solution to binary file at end.
strt            t: Print the starting values used.

dist            t: Disturb vector potential to check stability.
heat            t: Energy equation solved.
                f: Temperature set to zero. Energy equation disabled.
nuss            t: Calculate and print Nusselt numbers.

mnx,mny,mnz     Map dimensions (x, y and z respectively).
                 {Map dimensions are reset to a minimum of 10
                  and a maximum of 80.}
                 {Maps scaled for a printer with a pitch ratio
                  of 61 horiz./49 vert. - see subroutine map.}
                 {If mnz=0 then mnx*(arz/arx) is used,
                  If mny=0 then mnx*(ary/arx) is used, and
                  If mnx=0 then mnx is set to 51.}
pdist           If dist is true, then all vector potentials
                (except on the boundaries) are disturbed by
                multiplying each value by 1+random[-1,1]*pdist.

Temperature Boundary Conditions
-------------------------------
These variables are repeated for all the boundaries, viz.
            x=0, x=arx ; y=0, y=ary ; z=0, z=arz.
Only the x=0 wall variables are presented.

ixl             Set isothermal patch at x=0 <t>rue or <f>alse
xlci            Theta of isothermal patch, on x=0 wall [0, 1]
ixlyl,ixlyh     Start & end of patch in y-direction    [1,ny]
ixlzl,ixlzh     Start & end of patch in z-direction    [1,nz]

xlcg,xlca,xlcb  Specifies heat flux condition, on the x=0 wall
                using  xlcg*( dt/dx ) = xlca*t  +  xlcb.
                This condition applies on all the wall, except
                On the isothermal patch (if defined - see ixl)
                 {eg adiabatic       : xlcg=1, xlca=0, xlcb= 0
                     heat flux       : xlcg=1, xlca=0, xlcb= 1
                     convective      : xlcg=1, xlca=1, xlcb= 1
                     isothermal(t=1): xlcg=0, xlca=1, xlcb=-1}
                 {Note: This permits patch to be set to say
                        t=0  while rest of wall is at say t=1}

Velocity Boundary Conditions
----------------------------
These variables are repeated for all the boundaries, viz.
            x=0, x=arx ; y=0, y=ary ; z=0, z=arz.
Only the x=0 variables are presented.

xlv,xlw         Tangential velocities (v & w) on the x=0 wall.

Printing Controls
-----------------

vpform          Variable printing form, for labelling of axes.
                  i: integers used for axes, viz. (i,j,k)
                  c: coordinates used for axes, viz. (x,y,z)

The printing controls which follow are repeated for the x, y
and z direction. Only the x direction variables are presented.

vxp1,vxp2,vxp3, Logicals for printing 'values' for p1,p2,p3,
```

```
vxz1,vxz2,vxz3,          z1,z2,z3,t,u,v and w at the locations
vxt,vxu,vxv,vxw          specified below.

mxp1,mxp2,mxp3,  Logicals for printing  'maps'  for p1,p2,p3,
mxz1,mxz2,mxz3,          z1,z2,z3,t,u,v and w at the locations
mxt,mxu,mxv,mxw          specified below.

xc               Form of the x- printing planes specification.
                   p: integer plane location      [1,nx]
                   c: coordinate plane location   [0,arx]
npx              Number of printing planes to required.

mx(npx) or       Print planes for values and maps (integer or
rmx(npx)         real depending on xc). Values and maps are
                 produced if logicals vx?? and mx?? = true.
                  (Line may be repeated 'comment & data cards'
                   if more than 11 planes are required.)
                  (Note: If real location does not exist as a
                        plane, next rounded up plane is used.)

residuals
---------
rt,rp1,rp2,rp3,  Logicals for printing residuals of t,p1,p2,p3.
rmax             Logical for printing out only the maximum
                        value and location of residuals.

divergences
-----------
divv,divp        Logicals for printing divergence of the velocity
                 and vector potential fields.
dmax             Logical  for printing out only the maximum
                        value and location of the divergences.

exprmnt          Name of the experiment, also used as file names for
                 results output.

Heating Source
--------------

npt              Number of points of heating source data:
                 Maximum value is 10, if npt < 10, say npt=6, tpnt(1)
                 to tpnt(6) represent the temperature data and Q(1) to
                 Q(6) represent respiration rates corresponding to each
                 temperature point.
nparam           Initial number of parameters for fitting heating
                 function.
srcetype         Indication of heating source unit, mg_Co2/Kg h
                 or w/mmm
tpnt(1)          Temperature points for heating source data.
to tpnt(10)

Q(1) to Q(10)    Respiration rates at different temperature points.

numshow          Step for showing running results on monitor.

darcyx           Darcy number in x direction.
darcyy           Darcy number in y direction.
darcyz           Darcy number in z direction.

datype           Indication of values of Darcy number or ratios in
                 x,y,z, directions:
                 V or v:  values
                 R or r:  ratio
                 If datype=R or r, darcy is determine in subroutine
                 SETCON and darcyx = darcyx * darcy,
                            darcyy = darcyy * darcy,
                            darcyz = darcyz * darcy.

gridfile         File name for non-uniform grid.

                 END OF DATA FILE  'inparam'
```

gridfile (SAMPLE)

```
Sample 'gridfile' Grid Distrib File
NODAL DISTRIBUTION  IN X-DIRECTION (STEPSIZE BETWEEN NODES (NX-1))
NUMBER OF MESH POINTS-NX =011    (REPEAT NEXT 2 LINES IF (NX-1) > 14)
*-1--**-2--**-3--**-4--**-5--**-6--**-7--**-8--**-9--**10--**11--**12--**13--**
14--*
 20.00 20.00 28.00 36.00 36.00 36.00 36.00 28.00 20.00 20.00
C
NODAL DISTRIBUTION  IN Y-DIRECTION (STEPSIZE BETWEEN NODES (NY-1))
NUMBER OF MESH POINTS-NY =013    (REPEAT NEXT 2 LINES IF (NY-1) > 14)
*-1--**-2--**-3--**-4--**-5--**-6--**-7--**-8--**-9--**10--**11--**12--**13--**
14--*
 20.00 52.00 52.00 52.00 68.00 68.00 68.00 68.00 52.00 52.00 52.00 20.00
C
NODAL DISTRIBUTION  IN Z-DIRECTION (STEPSIZE BETWEEN NODES (NZ-1))
NUMBER OF MESH POINTS-NZ =011    (REPEAT NEXT 2 LINES IF (NZ-1) > 14)
*-1--**-2--**-3--**-4--**-5--**-6--**-7--**-8--**-9--**10--**11--**12--**13--**
14--*
 20.00 20.00 32.00 44.00 44.00 44.00 44.00 32.00 20.00 20.00
$    END OF SAMPLE DATA FILE 'app_353' (contains 16 lines)   $
```