DEVELOPMENT OF AN INTERACTIVE MICROCOMPUTER SOFTWARE PACKAGE FOR THE OPTIMIZATION OF BATCH STERILIZATION PROCESSES

by

NOREEN FINNEGAN

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

Department of Agricultural Engineering
Macdonald Campus of McGill University
Montreal, Canada

Noreen Finnegan

Short Title:

A MICROCOMPUTER SOFTWARE PACKAGE TO OPTIMIZE BATCH STERILIZATION PROCESSES

UNIVERSITÉ McGILL

FACULTÉ DES ÉTUDES AVANCÉES ET DE LA RECHERCHE

Date _____

NOM DE L'AUTEUR: _____

DÉPARTEMENT: _____ GRADE: _____

TITRE DE LA THÈSE: _____

_____

_____

1. Par la présente, l'auteur accorde à l'université McGill l'autorisation de mettre cette thèse à la disposition des lecteurs dans une bibliothèque de McGill ou une autre bibliothèque, soit sous sa forme actuelle, soit sous forme d'une réproduction. L'auteur détient cependant les autres droits de publications. Il est entendu, par ailleurs, que ni la thèse, ni les longs extraits de cette thèse ne pourront être imprimés ou reproduits par d'autres moyens sans l'autorisation écrite de l'auteur.

2. La présente autorisation entre en vigueur à la date indiquée ci-dessus à moins que le Comité exécutif du conseil n'ait voté de différer cette date. Dans ce cas, la date différée sera le _____

_____

Signature de l'auteur

Adresse permanente:

_____

_____

_____

_____

Signature du doyen si une date figure à l'alinéa 2.

(English on reverse)

ABSTRACT

NOREEN FINNEGAN M.Sc.(Agr.Eng)

DEVELOPMENT OF AN INTELLIGENT INTERACTIVE MICROCOMPUTER SOFTWARE PACKAGE FOR THE OPTIMIZATION OF BATCH STERILIZATION PROCESSES

Efficient thermal processing of food renders the product microbiologically safe and shelf-stable, with minimal degradation of nutrient levels and organoleptic properties. The objective of this research project has been to develop a user-friendly software package that calculates optimal batch sterilization processing regimes for conduction-heating foods, in both cans and retortable pouches.

The package runs on a typical microcomputer, the IBM PC. It is comprised of hierarchically-arranged modules which include system initialization programs, intelligent interfaces, temperature - calculating routines, and service programs. The programs continuously store and access the intermediate results and user-supplied values which are passed between the levels of the hierarchy. Communication between user and package is limited to the interactive interfaces; the calculating routines are invisible to the user. This approach was employed to facilitate introduction to and operation of the package. The computer package was found to simplify the design and verification of thermal processes, and to be easy to use for the novice operator.

RESUME

L'optimisation des procédés de stérilisation en vrac à l'aide d'un progiciel intéractif conçu pour micro-ordinateurs

Les procédés thermiques lorsque bien employés sur les aliments, rendent les produits microbiologiquement sûrs et stables tout en minimisant la dégradation de leurs qualités nutritives et organoleptiques. Dans ce project de recherche, un progiciel qui optimise le régime des procédés de stérilisation en vrac des conserves et des poches pour aliments, à été développé.

Le progiciel a été conçu pour être exécuté par les micro-ordinateurs compatibles avec le IBM PC. Il consiste en un arrangement hiérarchique de modules dont font partis: le système d'initialisation des programmes, le système intelligent d'intéraction entre l'opérateur et le micro-ordinateur, les routines de calcul des températures, et les programmes de service. Les résultats intermédiaires et les données entrées par l'opérateur sont transférés entre les modules de la hiérarchie par des programmes qui les rangent en mémoire et les accèdent au temps requis. La communication entre l'utilisateur et le progiciel est limitée au système intelligent d'intéraction ce qui rend les routines de calcul invisibles à l'opérateur. Cette approche a été choisie pour faciliter l'introduction et l'utilisation. Il a été observé que le progiciel simplifiait la conception et la vérification des procédés thermiques, et qu'il était facile à utiliser même pour l'opérateur novice.

ACKNOWLEDGEMENTS

The author is grateful to Associate Professor and supervisor Dr. Robert Kok for his guidance and advice throughout the project. The organizational and technical skills he has imparted will be a lifelong asset.

Also thanks to Department Chairman and Professor, Dr. Edward McKyes, for his encouragement and help throughout the author's studies in engineering, both at the undergraduate and post-graduate levels.

The financial support of the Conseil des recherches des sciences agricoles du Québec (CRSAQ) is gratefully acknowledged.

The graphics portion of the computer package is the result of the creative genius of Paul Champigny, engineering student, and is a significant contribution to the overall presentation of the package (especially the clock).

Much thanks must go to Serge Tremblay and Laurent Gauthier, resident computer whizzes, who aided the author in her intrepid plunge into the world of bits and bytes.

And of course, thanks to family and friends for their much-needed and appreciated support and help.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF VARIABLES

a,b = microbial populations at time t_a and time t_b

AC=a_c = hyperbolic constant for cooling curve calculations

AH=a_h = hyperbolic constant for heating curve calculations

ALPHA= α = thermal diffusivity of the food (m**2/s)

BC=Bi_{cyl} = infinite cylinder Biot number

BCSQ=b_c^2 = hyperbolic constant for cooling curve calculations

BHSQ=b_h^2 = hyperbolic constant for heating curve calculations

BETA = Nth positive root of equation 168

BETAF1,BETAF2,BETAF3= intermediate variables for can temp calculation

BETXF1,BETXF2,BETXF3=)
BETYF1,BETYF2,BETYF3=)intermediate variables for pouch temp calculation
BETZF1,BETZF2,BETZF3=)

BP=Bi_{pl} = infinite plate Biot number

CNUT=N_0 = initial nutrient concentration in the food (g/g food)

COOLTI = time at which cooling starts (s)

CORG=C_0 = initial microorganism concentration (1/g food)

CP=C_p = specific heat of the food (kJ/kg-°C)

DTIME = duration of time increment (s)

DUMMY1 =)
DUMMY2 =) dummy (intermediate) calculating variables
DUMMYX =)

GAMM = Nth positive root of equation 202

GAMMF1 =)
GAMMF2 =) intermediate variables for can temperature calculations
GAMMF3 =)

Fo_{cyl} = Fourier Modulus for infinite cylinder

Fo_{pl} = Fourier Modulus for infinite plate

Fo_x, Fo_y, Fo_z = Fourier Modulus of x,y,z sides of brick

FC,FH=f_c, f_h = reciprocals of slopes of cooling, heating curves (s)

FLETH=F^z_{Tref} = F-value of contaminant organism (min)

FNUT=F_{nutr} = F-value of nutrient (min)

FRANUT = nutrient retention fraction

H = convective surface heat transfer coefficient (W/m**2-^oC)

HALFL=L = half-length of the can (m)

HALFX=L_x =)
HALFY=L_y =) half-lengths of pouch in X, Y, Z directions (m)
HALFZ=L_z =)

JC,JH=j_c, j_h = lag factors of the cooling, heating curves

k = thermal conductivity (W/m-^oC)

k_r = reaction rate constant

L,M,N = counter variables

NEND = summation endpoint variable

NELE = number of elements in the pouch or can

NR = number of radial nodes

NX,NY,NZ = number of X, Y, Z nodes

OBJFN = objective function

ORG0 = initial number of organisms in container (at TIME=0)

ORGE0 = initial number of organisms in element (at TIME=0)

ORGELE = number of organisms in element at end of time increment

ORGEST = number of organisms in element at start of time increment

ORGTOT = number of organisms in container at end of time increment

p = independent variable of an equation

Q = intermediate variable for hyperbolic cooling curve calculation

R = radial nodes

R_n = general variable for retention fraction of nutrient "n"

RADIUS=D/2 = radius of the can (m)

RATEL=LR = lethal rate of the organisms (1/s)

RATEN=DR = degradation rate of the nutrient (1/s)

RF1 =) intermediate variables for can temperature calculation
RF2 =)

RHO= ρ = density of the food (g/m**3)

RINNER=R_{inner} = inner radial boundary of annular element (m)

ROOT = Nth positive root of Bessel function of zero order

ROUTER=R_{outer} = outer radial boundary of annular element (m)

S = intermediate variable for hyperbolic heating curve calculation

SIGMA = summation operator

t_a, t_b = time limits for definite integral of equation 83

t_{cool} = time passed since exposure to cooling water (s)

T_0 = general initial temperature (^oC)

T_c^* = switch temperature from hyperbolic to logarithmic cooling (^oC)

T_h^* = switch temperature from hyperbolic to logarithmic heating (^oC)

T_{cool} =cooling array for cooling part for CAN, POUCH (^oC)

T_{heat} = heating array for heating part for CAN, POUCH (^oC)

T_{pi} = pseudo-initial temperature (^oC)

TAVG=T_{avg} = average of TCNT at start and end of time increment (^oC)

TBATH=T_a = temperature to which container is suddenly exposed (^oC)

TCMAX=T_{cmax} = maximum temperature reached at center of can, for SUB1(^oC)

TCNT = mean of element's nodal temperatures, at a time value (^oC)

TDT = thermal death time (min)

TEMP= T = calculated temperature value (^oC)

TERM =) summation terms
TERMN =)

TIME = time after start of processing (s)

TINIT=T_i = initial food temperature (^{o}C)

TREF=T_{ref} = reference temperature for lethality calculations (121 ^{o}C)

TRETRT=T_r = retort temperature - for SUB1 (^{o}C)

TSTARC=t_c^* = switch time for hyperbolic to logarithmic cooling (s)

TSTARH=t_h^* = switch time for hyperbolic to logarithmic heating (s)

TWATER=T_w = cooling water temperature (^{o}C)

u = general temperature ratio

u_{cyl} = temperature ratio of infinite cylinder

u_{pl} = temperature ratio of infinite plate

u_x, u_y, u_z = temperature ratio for x,y,z infinite plates for POUCH

UR,UX,UY,UZ = intermediate terms for temperature calculation

VELE = elemental volume (m**3)

VTOTC = total volume of the can (m**3)

VTOTP = total volume of the pouch (m**3)

W_n = general variable for weighting factor for nutrient "n"

X = position along X-axis of pouch (m)

XF1 =)
XF2 =) intermediate variables for pouch temperature calculation
XF3 =)

XJ0 = value of Bessel function of first kind of zero order

XJ1 = value of Bessel function of first kind of first order

XLETH = lethality imparted to the microbial population by the process

XLETHST = lethality accumulated at start of time increment (SUB1)

XNUT0 = initial amount of nutrient in food (at TIME=0) (g)

XNUTE0 = initial amount of nutrient in element (at TIME=0) (g)

XNUTE = amount of nutrient in element at end of time increment (g)

XNUTEST = amount of nutrient in element at start of time increment (g)

XNUTST = nutrient in food at start of time increment (for SUB1) (g)

XNUTOT = amount of nutrient in food at end of time increment (g)

Y = position along Y-axis of pouch (m)

Z = position along Z-axis of pouch or can (m)

ZF1 =) intermediate variables for can temperature calculation
ZF2 =)

ZLETH= z = z-value of contaminant microorganism (^{o}C)

ZNUT=z_{nutr} = z-value of nutrient (^{o}C)

I. INTRODUCTION

1.1 Background

Food preservation has historically been a major concern for humanity. The objectives of food preservation are to eliminate the deterioration of food, to lengthen its shelf-life, and to maximally retain nutrients and organoleptic properties. Thermal processing is one of the most common methods employed to attain these goals, and will be the sole preservation method discussed in this thesis. Presently, much research is being carried out to develop ways of predicting the effects of sterilization on the biological catalysts and components of the food. Mathematical models, either theoretically or empirically-derived, may be used to simulate the thermal process. By means of a model, a process can be optimized with the aid of digital computers; this is a much more time-efficient procedure than an experimental trial-and-error approach.

A model must minimally incorporate the two major concepts of thermal processing: the heat transfer in the food and the consequent lethal or inactivating effects on the contaminant organisms, and the food properties and components. Factors that must be considered when modeling the thermal process include the type of process utilized, the method of heat application, the mechanism of heat transfer, the thermal and rheological properties of the food, the degree of mixing within the food and the residence time distribution function of the food particles. These factors all affect the temperature-time distributions in the food. Furthermore, consideration of microbial death kinetics, the kinetics of inactivation of food components and properties (eg.

nutrients, enzymes, texture, colour), and initial distributions of contaminating microorganisms and food components, are necessary to develop a model which will predict the effects of thermal processing on the contaminant organisms and on the food.

In examining the effects thermal processing has on a food, the type of process utilized is of prime significance. All thermal treatments consist of a sequence of heating, cooling and packaging stages; the order in which these are performed varies according to the particular treatment selected. In the conventional batch process the food is first put in a container which is then sealed. Many such containers are then heated simultaneously in a retort and held there for a prescribed amount of time. In the semi-continuous process the food is also packed in individual containers which are consequently sealed and subjected to heat treatment, but the containers are handled sequentially rather than simultaneously. In a truly continuous process the food is first heated and partially cooled in heat exchangers, before being packed.

Very closely related to process type with regard to the extent of its effect on the processing results is the method of heat application. Heat transfer fluids commonly utilized to carry the thermal energy to the application are steam, hot water and hot combustion gases. In batch and semi-continuous processes, the outer surfaces of the containers are subjected to a temperature-time function characteristic of the given process. These functions are frequently comprised of a) a sudden increase followed by b) a period of constant temperature and c) a sudden decrease. In continuous processing, the concept of a time-varying function is not applicable since

under normal processing conditions the equipment operates practically at steady-state.

The mechanism of heat transfer in the container and its contents will be determined by the process type and the container and food properties. It is generally accepted that solids will heat by conduction, low-viscosity liquids by convection, and high-viscosity liquids and liquids containing particulate matter in suspension by a combination of the two. However, the type of process can dramatically influence the heat transfer mechanism. For example, high-viscosity liquids can be induced, by virtue of currents set up within the food, to heat convectively when processed in a continuous manner or when agitated in batch or semi-continuous equipment.

The physical, thermal and rheological properties of the food will significantly affect the heat transfer within it. Food properties such as the density, viscosity, specific heat, thermal conductivity and diffusivity must be taken into account when developing a heat transfer model. These properties will also affect mass and momentum transfer within the food and thus the degree of mixing, on the macroscopic scale (Levenspiel, 1962). During batch or semi-continuous processing of purely convectively-heating foods, all the container contents will have the same temperature at any instant. Thus, although the contents may undergo intense macromixing, the temperature-time histories of all fluid elements will be identical so that the mixing will not affect processing results. Macromixing does not occur in truly conductively-heating foods. The temperature-time histories at all locations can therefore be calculated using standard heat diffusion equations.

For intermediate-type foods the extent of macromixing determines the distribution of temperature-time histories of the fluid elements and therefore strongly influences the effects processing has on the food. Whereas for batch and semi-continuous processing the residence times for all fluid elements are equal, since they are confined to the same container, this is not true for continuous processing. In the latter situation the fluid mechanical and macromixing aspects, as well as the heat transfer mechanism, will determine the fluid element temperature-time history distribution.

In a model of a thermal process, in addition to the heat transfer phenomena, the effects of the temperature-time histories on the contaminant population and on the food composition and properties must be incorporated. To evaluate the lethal effect of a heat treatment on the contaminating organisms, an understanding of microbial death kinetics is essential. The death rate is usually calculated by means of a semi-empirical logarithmic equation. The adequacy of the process may be expressed in terms of lethality, a quantity in which is reflected the cumulative killing effect achieved throughout the container contents during the entire heat treatment. The lethality calculation is generally based on the assumption that the contaminant population is uniformly distributed throughout the food. In situations where this is not true, serious inaccuracies in the determination of process adequacy can be introduced. The effects of processing on the food components and properties can be estimated in a manner similar to that used for the contaminant population. Once the kinetics of inactivation or creation of food components such as flavour compounds, enzymes and nutrients, and properties such as texture and colour are known, the inactivation or creation rates can

be calculated, which can then be used to determine the resultant food quality after processing. Uniform distribution of components and properties is generally also assumed in this procedure.

Once the factors discussed above and their consequent effects on the processing results are sufficiently understood, an overall mathematical model in which these concepts are incorporated may be formulated. Such a model may then be used to simulate the process under a variety of conditions in order to ascertain process sufficiency or to investigate alternate operating schemes. Simulation may, for example, be used to choose the "best" thermal process, wherein "best" is determined in accordance with a set of necessarily subjectively established criteria. Such criteria might consist of the requirement that the process simultaneously impart an adequate lethal effect while maximizing an objective function representing food quality. Two techniques commonly utilized to calculate a "best" process are maximization / minimization and optimization. In the former, the independent process parameters are considered to remain constant throughout the heat treatment and the simulation is used to find the values of these which will yield the maximum or minimum value of the objective function. This technique is frequently referred to as "optimization". However, in true optimization the independent parameters are treated as time-varying so that, rather than a single value, a time history of each will be produced (Hildenbrand, 1980; Lund, 1982). More favourable objective function results can be obtained with optimization than with minimization / maximization at the expense of much greater computational effort. Optimization results will also be the more complex to implement in most processes although computer control of equipment

will simplify this.

1.2 Objectives

The objectives of this research project have been:

1) To mathematically model the heat transfer phenomena in conductively-heating foods, for the following instances:

i) Slowest heating region approach for food processed in a can

ii) Spatial distribution approach for food held in a can - with the convective surface heat transfer coefficient assumed to be infinite

iii) Spatial distribution approach for food held in a can - with the heat transfer coefficient specified by the user

iv) Spatial distribution approach for food in a retortable pouch - heat transfer coefficient assumed to be infinite

v) Spatial distribution approach for food held in a retortable pouch - heat transfer coefficient specified by the user

2) To use the temperature distributions calculated by means of the heat transfer models to predict process lethality, again for food treated in cans or retort pouches.

3) To predict the effects of the sterilization treatment on the food properties, specifically retention of nutrients.

4) To develop a user-friendly, interactive microcomputer software package that incorporates the above heat transfer, process lethality and nutrient retention models.

5) By means of this computer package, to simulate batch sterilization processes, and select the regime which results in optimal nutrient retention while concurrently satisfying process lethality requirements.

1.3 Scope

This study is limited to conductively-heating foods which are batch-processed, in either cylindrical cans or retortable pouches. The computer package chooses the most favourable processing regimes based on an "optimization" technique using time-independent rather than time-varying process parameters. According to the strict definition of optimization, this procedure is more a maximization technique.

Thermal processes are commonly optimized with respect to retention of desirable food characteristics, properties and components. In this project, the retention of nutrients is the basis upon which the optimization is performed.

II. LITERATURE REVIEW

2.1 Conduction Heat Transfer

The thermal processing of food is done to ensure commercial sterilization, that is, to reduce by a factor of 10^{12} the contaminant microbe population in the food (Charm, 1978). The rate of death of the microbes is dependent upon the temperature they are exposed to, and the duration of the exposure. Any type of bacterial spore will be destroyed if subjected to a high enough temperature for an adequate length of time. As this temperature increases, the time needed to kill the spores will decrease. It is evident that in order to predict adequate processing regimes that will ensure commercial sterility of the food, the temperature-time histories in the food must be known first. There are two major approaches by which temperature-time histories are calculated: the point temperature method and the spatial temperature distribution approach.

2.1.1 Point Temperature Approach

The point temperature method involves the calculation of the temperature-time history at the centre of the container, since it is assumed to be the coldest point or slowest heating region (Bigelow et al., 1920; Ball, 1923; Thompson, 1919). The empirical temperature-time heating and cooling curves were found to be almost straight lines when plotted on semilogarithmic paper, except for the first portion (lag) of each curve (Bigelow et al., 1920). The simple logarithmic equations were considered by Ball (1923) to be an adequate approximation for the heating curve since the early temperatures are not high enough to significantly affect the sterilizing values. However, the lag in

the initial part of the cooling curve causes deviations from the logarithmic equation at the time of maximum temperature, and therefore at the time of maximum sterilization rate. Ball (1923, 1928) developed a method to calculate the centre temperature-time histories taking into account the cooling curve lag. He used the experimental parameters "f" and "j", wherein f is the time required to reduce a given temperature difference by one log cycle, and j is the lag factor or the measure of lag in the heating or cooling rate. The cooling curve was modeled as first hyperbolic and then logarithmic, to account for the deviation caused by the cooling curve lag factor. He assumed that the cooling curve was the exact reverse of the heating curve (the same magnitude of slope but opposite in sign), an assumption which he knew to be not always correct. The equations developed were based on j values of 1.41 for both the heating and cooling curves. The j value of the cooling curve was later amended to 2.04 (Ball and Olson, 1957). Jackson and Olson (1940) further developed these equations so that the heating curves were no longer restricted to simple logarithmic curves with one constant slope. The equations were now applicable for products displaying "broken heating curves", wherein the heating curve is asymptotic to one straight line and then breaks to become asymptotic to another straight line with a different slope (Jackson, 1940). The heating and cooling curves are always asymptotic to a straight line assuming that the ambient temperature and the thermal diffusivity remain constant with respect to time (Olson and Jackson, 1942).

Carslaw and Jaeger (1947) derived solutions to the classical heat conduction equations, using Duhamel's theorem, for various object shapes including finite cylinders and bricks. Hicks (1951) modified the equations developed by

Carslaw and Jaeger (1947), and used the j parameter, to more accurately determine the centre temperatures, especially during the cooling time. As Hicks stated, an inaccuracy of only one degree Fahrenheit at the high temperatures present at the start of cooling can have a significant effect on subsequent process adequacy calculations. He used his more theoretically correct equations to ascertain whether Ball's method (Ball, 1923; Ball, 1928; Ball and Olson,1957) and assumptions were acceptable approximations. He concluded that the value of j for the cooling curve should be experimentally determined rather than set equal to 2.04, as defined by Ball and Olson (1957). Also, he determined that for large enough time values, Ball's assumption that the heating and cooling curves are the reverse of each other is acceptable.

Gillespy (1953) also analytically calculated centre temperatures in a cylindrical can, utilizing equations developed by Riedel (1947) and based on Duhamel's theorem (Carslaw and Jaeger, 1947). Gillespy's equations were used to determine the centre temperatures, with the following restrictions: the external temperature changes must be instantaneous, the initial temperature must be uniform, and the f parameter and the height/diameter ratio must be known.

In addition to the analytical approaches described above, several researchers have presented charts or tables to calculate centre temperatures. Olson and Schultz (1942) developed tables containing solutions to the classical heating equations. These tables were an improvement in terms of accuracy over those given by Newman (1936). The improved tables were based on Newman's theory that the solutions for an infinite slab and infinite cylinder could be

multiplied to obtain the solution for a finite cylinder; the theory was extended to cover the case of a brick (or pouch). The assumptions upon which the values of the tables were based included uniform initial temperature and zero surface resistivity.

A different chart approach was described by Leonhardt (1976) to estimate centre temperatures in cylindrical and rectangular cans. He used the temperature response charts developed by Hayakawa (1969) for finite cylinders which predicted the centre temperature of a conductively-heating food according to a can shape factor. By determining the shape factor d/h (diameter/height), the appropriate centre temperature response curves were plotted for numerous container shapes.

2.1.2 Spatial Temperature Distribution Approach

According to Stumbo (1948) one should not assume a process would be adequate when calculations to determine lethality are performed at the slowest heating point (or centre) of the can. He stated that probability of bacterial survival is not necessarily greatest at the point of slowest heating but rather at the point where bacterial density is likely to be greatest (Stumbo, 1949). This observation makes the determination of the temperature throughout the can more important. Williamson and Adams (1919) developed solutions to the classical transient heat conduction equations describing temperature distributions throughout various objects such as : rectangular parallelopiped (brick), long rectangular rod, infinite thin slab, finite cylinder and sphere. Separate solutions were derived for the cases when the surface is suddenly cooled or heated, and when the surface is heated at a uniform rate. The

equations are based on the assumptions of uniform initial temperature, constant thermal diffusivity and negligible surface convective heat transfer. Thompson (1919) developed solutions to the equations for foods heated in cans, and performed heat penetration tests to verify the results. His theoretically-derived solutions closely approximated the actual penetration data for water-cooled cans, but not for air-cooled ones, since his equations considered surface convective heat transfer to be negligible; this is not a reasonable assumption for air-cooling.

Many solutions to the classical transient heat conduction equations have been developed (Jackson, 1940; Olson and Schultz, 1942; Ball and Olson, 1957; Luikov, 1968) but they are based on the common assumption that surface convective heat transfer is negligible. Carslaw and Jaeger (1959) developed equations for shapes such as bricks, cylindrical containers and spheres to determine the temperature distributions when the surface resistivity cannot be assumed to be equal to zero. Ramaswamy et al.(1982) simplified these equations; these simplifications hold for conditions of sufficiently long heating times i.e. Fourier modulus greater than 0.2, and for Biot numbers between 0.02 and 200. These approximations were found to have mean errors of only 0.042 percent for an infinite slab and 0.084 percent for an infinite cylinder, and could be considered sufficiently accurate for use in process calculations.

2.2 Thermal Process Calculations

To predict adequacy of process, the effect of the temperature history in the food on the contaminant organism must be determined. There have been many

techniques published that perform process calculations. They can be subdivided into two major sections: those which determine process lethality based on the temperature history at the slowest heating point in the food, and those which base the lethality calculations on the temperatures throughout entire volumetric elements of the food. In the first section there is a further subdivision into the general methods and the formula methods. The general methods tend to be more accurate because they are based on the actual food temperature data; no assumptions are made about the temperature-time relationships. The formula methods, however, use temperature history predictions in their process calculations. These predictions can be made using either empirical or theoretical formulae (Hayakawa, 1978).

2.2.1 Slowest Heating Region Approach

2.2.1.1 General Methods

Bigelow et al. (1920) published the first systematic approach to thermal process evaluation, the General Method. Their basic concepts and methodology are still used in industry today. They combined heat penetration data and bacteriological data in a graphical method to predict process lethality. Using the temperature-time history at the slowest heating point in the container (the centre) they calculated the lethal rate associated with each time increment. By graphically integrating the lethal rate over time they were able to determine the sterilizing value of the process, F.

Schultz and Olson (1940) developed an Improved General Method wherein they made use of a special coordinate lethal rate paper. The use of this paper simplified the process calculations and reduced the chances of misplotting

the data. Another advantage of their method was the inclusion of formulae which could be used to convert heat penetration data for different initial food temperatures and retort temperatures. These developments greatly increased the applicability of the General Method. Unfortunately, the process calculations were still time-consuming. As well, specially-printed coordinate paper was required for each new thermal death time slope or z value used in the calculations.

Patashnik (1953) introduced a simplified procedure for thermal process evaluation which was similar to the General Method of Bigelow et al. (1920). In this new method the container temperature readings were taken at equal time intervals and lethal ratio values F/t were tabulated for each time interval. These ratios were totaled, and the product of the sum and the equal-time interval readings used to determine the process value. The advantage of this method over the original General Method is that lethal rate curves did not have to be plotted once the slope of the TDT curve was known; the integration under the lethality curve was done by the numerical trapezoidal integration method rather than graphically.

Hayakawa (1973) described a method to apply a lethal rate paper of fixed z value to determine the values of different z values. There are two conditions inherent in his technique: the lethal rate paper temperatue scale must be defined, and the unit area under the lethal rate curve (equivalent to F=1) must be defined for each case according to the temperature scale used. Leonhardt (1978) proposed a general lethal rate paper that would eliminate these two conditions. His general lethal rate paper can be used to represent

heat penetration data directly, whatever the z-value and reference temperature, without the necessity of defining a unit area in each case.

2.2.1.2 Formula Methods

The first analytic method to perform process calculations was proposed by Ball (1923). He wanted to reduce the time necessary to determine the process results, and consolidate the various factors which influence the calculations into a common base. He used mathematical formulations accompanied by charts relating the variables to simplify process evaluation. The Formula Method was based on a number of mathematical and empirical assumptions which restricts its applicability. For example, he assumed that the lag factor of the cooling curve was a constant of 1.41. He also assumed that the curvilinear portion of the cooling curve ends at time equal to 0.141 f_c. As well, the charts are only suitable for cases when the heating curve is unbroken, and when f_h is equal to f_c. It is evident that errors of estimation will occur if the lag factor of the cooling curve is not 1.41, if f_h is not equal to f_c, if the heating medium does not suddenly reach processing temperature (Merson et al., 1978) or if any of the other assumptions do not hold for the case under consideration. According to Hayakawa (1978), Ball's Formula Method will significantly underestimate the F value when the cooling curve lag factor is greater than 1.41. The accuracy of the Formula Method has been compared with that of the General Method for a wide range of processing conditions and in only one test was the Formula Method found to overestimate the F value, and even then by a negligible amount (Steele and Board, 1979). Process calculation problems were solved using the Formula Method and the General Method and presented in Ball (1928).

A simplification of Ball's Formula Method was published by Olson and Stevens in 1939. They developed a nomographic method which can be used for canned foods that exhibit straight-line logarithmic heating curves. With their method, the computations of Ball's (1923) method could be performed graphically. They included several worked examples which showed the application of the nomographic method.

Ball and Olson (1957) modified Ball's original Formula Method by introducing two dimensionless parameters P_h and P_c. These parameters were used to estimate the sterilizing values of the heating and cooling phases. They greatly simplified the process calculations for the case when the heating curve is broken, and when f_h and f_c are not equal. However, the assumptions that the cooling curve lag factor was a constant of 1.41 and that termination of the curvilinear section of the cooling curve occurred at a time equal to 0.141 f_c were not modified. Hicks (1958) observed that there were considerable errors in the tables of P_h values tabulated by Ball and Olson (1958) and published his recalculated values.

Stumbo and Longley (1966) published a modified Formula Method in which variable cooling curve lag factors could be used for the process calculations. They felt this would increase the reliability of estimation of the thermal process. Parametric values, which were estimated with the aid of a planimeter from temperature-time curves drawn on lethal rate paper, were tabulated. These tables are applicable only when the difference between the f_h and f_c values are less than twenty percent of the f_h value.

Hayakawa (1970, 1971) developed empirical formulae which can be used for process calculations when the cooling curve lag factor is not equal to 1.41. He derived three formulae, each of which was applicable for a range of j_c values. Together they covered the range of j_c values from 0.045 to 3.0. Using these empirical formulae he created a table of new parametric values which reduced the number of calculations required for the evaluation of a thermal process, as compared to the previous Formula Methods.

Griffon, Herndon and Ball worked together to develop one of the first computer-derived tables for calculating thermal processes. Their approaches could be used for unbroken heating curves (Herndon et al., 1968) and broken heating curves (Griffin et al., 1969), and for simple logarithmic cooling curves (Griffin et al., 1971). The procedures were similar to that described by Ball and Olson (1957).

All the Formula Methods described above are based on empirically-derived temperature histories. Other Formula Methods were developed which used a theoretical approach to determine the temperature-time data. Gillespy (1953) used an analytical solution of Duhamel's theorem (Carslaw and Jaeger, 1947) to create a table of food temperature values. Using these temperatures, his technique calculated the sterilizing values of a heat process by applying the general method. With his method, one could evaluate heat processes which involved a time-varying retort temperature.

Flambert and Deltour (1972) also used heat conduction equations to estimate food temperatures in their Formula Method. Their method is applicable for

solid foods packed and processed in cylindrical cans, assuming that the heating and cooling media temperatures remain constant with time. They prepared tables of parametric values which are easier to use than those of Gillespy (1953).

2.2.2 Mass or Volume Average Approach

The second group of procedures utilizes the mass average lethality value when evaluating the thermal process. This value is estimated from the mean concentration of surviving microbes in the food. The procedures utilize solutions of transient heat conduction equations to predict the temperature distribution. One of the first mass averaging methods was published in 1951 by Gillespy. He used an equation for transient heat conduction in a finite cylinder to estimate the temperature distribution. By assuming that the contribution of the cooling phase to sterilization was equivalent to that gained when the heating curve is extended by 0.3 f_h, he greatly simplified the estimation of the sterilizing value. However, this assumption may introduce errors since the extension constant 0.3 is likely to be affected by the shape of the cylindrical can, by the length of the heating phase and by the rate of cooling (Hayakawa, 1978).

Stumbo (1953) developed a method to calculate the sterilizing value which was similar to that of Gillespy (1951). He did not, however, use Gillespy's assumption regarding the heating phase extension. Stumbo estimated the distribution of lethal values using Ball's (1923) charts. The equation he developed for the prediction of the mass average F value was based on the

assumption that the food was packed in a No.10 can. Hence, all restrictions associated with Ball's (1923) charts are applicable to Stumbo's equation. A controversy exists as to whether Stumbo's method is applicable for foods packed in cans other than size No.10.

Ball and Olson (1957) used the heat conduction equation described in Gillespy (1951) for their mass averaging process calculation method. They considered the food in a can to be divided into eleven isothermal regions, and used the tabulated values of the P_h and P_c functions to determine the sterilizing value in each isothermal region. From this sterilizing value the concentration of remaining vulnerable factor was calculated for each region. These concentrations were then used to predict the overall mass average F value.

Hayakawa (1969) developed a method to calculate mass average F values using dimensionless parameters. Six such parameters were needed to predict the thermal inactivation of a vulnerable component. A digital computer was used to calculate tables which listed the mass average F values in terms of the dimensionless parameters.

A computer was also necessary for the estimation technique of Teixeira et al. (1969a). They used a finite difference approach to numerically solve the transient heat conduction equation. The resultant temperature distribution was used to estimate the mass average concentration of a vulnerable factor in each volumetric element of a cylindrical can of solid food. The elemental concentrations were summed to yield the overall mass average F value. Manson et al. (1970) developed a similar method to estimate the mass average

sterilizing value for solid food packed and processed in a brick-shaped can.

Jen et al. (1971) utilized the method of Teixeira et al. (1969a) to refine the technique described in Stumbo (1953). To calculate the mass average sterilizing value Jen et al. (1971) used the heating and cooling curves predicted by Teixeira et al. (1969a) to develop the $f_h/U:g$ relationships rather than basing them on visual inspection. By extending the $f_h/U:g$ relationship tables for higher z vlaues, the method could be used to calculate nutrient inactivation as well as microbial destruction.

All of the mass average techniques described above use the F-z parameters to predict microbial death. Lenz and Lund (1977) proposed a method to calculate the mass average sterilizing value based on the Arrhenius equation, and using activation energies rather than z-values. The activation energies are applicable over a broader range of temperatures. They checked their method against the Formula Method of Ball and Olson (1957) and found it gave equivalent results.

2.3 Optimization of Nutrient Retention

The effect of the sterilization process on the nutritive contents of canned foods is a major concern for food processors. The microbial safety of the food is the prime criterion for determining the heat process. Simultaneously, it is desireable to retain as much of the nutrients in the food as possible. Optimization techniques have not been applied to foods until recently because the complexity of the food components and processes makes analytical treatment difficult. A lack of accurate information of a process is a hindrance to its

optimization. This is especially true for food processes, where the reactions are complicated and often cannot be quantified (Saguy, 1982).

The analytical approach to calculating and predicting the retention of nutrients during thermal processing requires a mathematical model of the process. The temperature effects on the nutrients must be incorporated in this model. Several steps need to be followed during the structuring of the model: 1) the process must be defined, 2) the theory governing the process must be determined and verified, 3) the theory must be translated into mathematical equations, 4) the algorithm which incorporates these equations must be created and 5) the results must be checked to verify the validity of the model (Saguy and Karel, 1980).

Various models to describe nutrient retention in food undergoing thermal processing have been proposed. Ball and Olson (1957) presented a method to determine the retention of a vulnerable factor such as a nutrient in a food being thermally processed in a cylindrical container. Because of the finite layer procedure used to integrate the lethal effects, and the treatment of the cooling curve lags, high accuracy in the estimation would not be expected (Jen et al., 1971).

Teixeira et al. (1969b) presented a computer technique for determining lethality and nutrient retention in foods heated in cylindrical containers. Their procedure made use of a finite difference form of the two-dimensional transient heat conduction equation to produce a temperature distribution throughout the container at any time. They then applied the rate equation

for nutrient degradation to small volume elements, over short time intervals. The final percent nutrient retention is obtained by numerically integrating the remaining nutrient amounts over the container volume and over process time. The technique does not require use of any tables, or consideration of lag factors or iso-j regions as in the method of Ball and Olson (1957).

Manson et al. (1970) developed a computer program to determine the effects of thermal processing on the nutrients of foods being heated in rectangular containers. The temperature history is provided by a finite difference numerical solution to the three-dimensional transient heat conduction equation. The effect of heat on the nutrients is calculated at all points in the container, and the results are integrated to give the final nutrient retention value.

Jen et al. (1971) extended the process lethality calculation method of Stumbo (1953), described in section 2.2.2, to make it applicable for estimating nutrient degradation. As described in the previous section, Jen et al. (1971) refined the temperature calculation method as well as extended the parameter tables over a higher range of z values.

Lenz and Lund (1977b) used a lethality-Fourier number method to calculate the mass average retention of heat-labile quality factors such as nutrients, for conduction-heating foods packaged in cylindrical containers. The lethality number was volume-averaged for different Fourier heating times and activation energies, and average retention number-Fourier heating time charts were developed.

Castillo et al. (1980) developed a model to predict the retention of nutrients based on the first order kinetics of nutrient degradation in foods processed in retortable pouches. With their model, the prediction of nutrient retention at any point in the pouch can be made. The results can be integrated over time and volume to give the final nutrient fraction retained after processing. Castillo et al. (1980) used simulated food to experimentally verify their model, and found the predictions of nutrient retention to be within the ninety percent confidence intervals of the experimental nutrient fractions.

Once a model has been formulated to describe the effects of the heat process on the vulnerable factors such as nutrients, it can be used to simulate the process. These simulations are the basis of the optimization procedures. Finding the optimum solution for a processing problem by selecting from numerous influencing factors is difficult because a large number of simulated runs must be done. To improve the efficiency in finding the optimum, several techniques have been proposed including linear programming (Bender et al., 1982) and simplex optimization (Nakai, 1984a; Nakai, 1984b; Nakai et al., 1984c). A comparison of optimization techiques and theory is presented in Nakai (1981) and Evans (1982).

Teixeira et al. (1975b) used a trial-and-error search technique to determine the best conditions for improving thiamine retention in thermally processed foods. They employed true optimization, in that they varied the retort temperature with time to find the optimum processing conditions. They used sinusoidal, ramp and step temperature input functions, and found that a ramp function resulted in optimum thiamine retention. But since only a slight

increase in retention was noted, they concluded that the results did not justify the use of time-varying retort temperatures.

Lund (1977) considered the optimization of constant retort temperature processes. He found that the optimum retention of a low z-value nutrient is obtained at a low temperature-long time process whereas a high temperature-short time process is optimum for a nutrient with a high z value. Thijssen et al.(1978) also considered the optimization of constant retort temperature processes. They developed a short-cut method using the Fourier value to calculate the optimal sterilizing conditions, in terms of process temperature and time, for a constant retort and cooling medium temperature. Thier method is applicable to all sizes and common geometries of cans and retortable pouches.

Saguy and Karel (1979) used the maximum principle theory to optimize thiamine retention during the sterilization of conduction-heating foods processed in cylindrical containers. The optimal retort temperature profile determined by the procedure improved thiamine retention by two percent. They concluded that a constant retort temperature regime was almost as good as a time-varying one, and is much easier to implement.

Ohlsson (1980) based his optimization method on the concept of the "C-value" or cook-value. This value is used to describe the changes in the nutritional and sensory properties of the food after processing. It is calculated in the same manner as the more common F-value, but is based on a reference temperature of 100 ^{o}C and a z-value of 33 ^{o}C, rather than 121 ^{o}C and 10 ^{o}C.

Optimal sterilization temperatures are chosen where the minimal quality changes occur i.e. for minimal C-values.

III. MODEL DEVELOPMENT

3.1 Heat Transfer

The heat transfer phenomena in conduction-heating foods were modeled by means of several approaches. The traditional method was included, whereby solely the temperature changes at the slowest heating region of the container are considered. The algorithms which calculate the center temperatures in a can are incorporated in BALLST, in the program module SUB1. The temperature distribution in a can as a function of both position in the container and time is also modeled. CAN1 (in program module SUB4) calculates the temperature distribution in the can, when the convective surface heat transfer coefficient is assumed to be infinite. CAN2 (module SUB5) returns the same type of temperature-time results, but it allows the specification of the heat transfer coefficient prior to the calculation. As well, the heat transfer in a retortable pouch is modeled. Again the temperature is considered to be a function of both position and time. The surface convective heat transfer coefficient can be assumed to be infinite (POUCH1 in SUB2) or can be specified by the user (POUCH2 in SUB3).

3.1.1 Slowest Heating Region Approach (can)

The temperature at the slowest heating region (SHR) of a cylindrical container undergoing a sterilization treatment is calculated by the program BALLST (module SUB1). This region is located at the geometrical center of the can for conduction-heating foods. The can is considered to be first heated, then held for a specified period of time, and finally cooled. The assumptions upon

which the model development was based are: the thermal diffusivity is isotropic and independent of temperature, the heat transfer coefficient at the container surface is infinite, and the food is initially at a uniform temperature.

Three models are available in BALLST; the first one is based directly on equations derived by Ball (1923, 1928) and Ball and Olson (1957), while the other two models are modifications of the first. Ball treated the heating curve as logarithmic and the cooling curve as first hyperbolic and then logarithmic. In the second model both the heating and the cooling curves are treated as a combination of first hyperbolic and then logarithmic sections. The third model differs from the second in the way the hyperbolic cooling equation is calculated, and in the j_c (cooling curve lag factor) value used. The various temperature-time curves generated by the three models are shown in Figure 1.

3.1.1.1 Model One

The heating curve for a conductively-heating food can be plotted on an inverted semilogarithmic axis, with the ordinate axis representing the difference between the retort temperature and the container centre temperature, and with the abscissa representing time. A typical heat penetration curve is given in Figure 2. Theoretically, the equation for the heating curve is derived from the asymptote of the curve; in practice the straight-line approximation is drawn as a tangent, one or two log cycles from the origin (Stumbo, 1973). The equation for the straight-line approximation of the heating curve is given by (Ball and Olson, 1957):

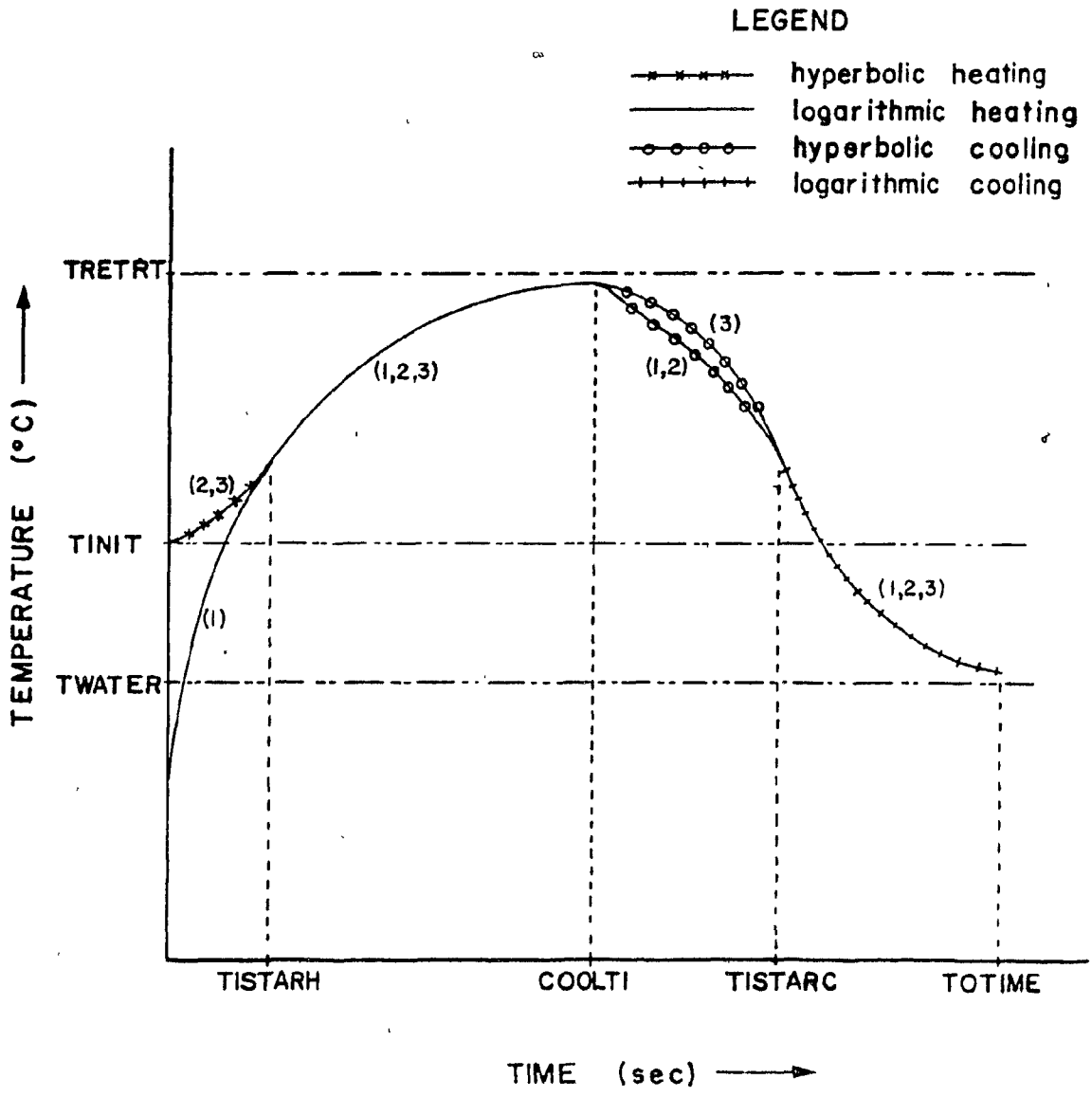


Figure 1 - Temperature-Time Histories Generated by Models in SUB1

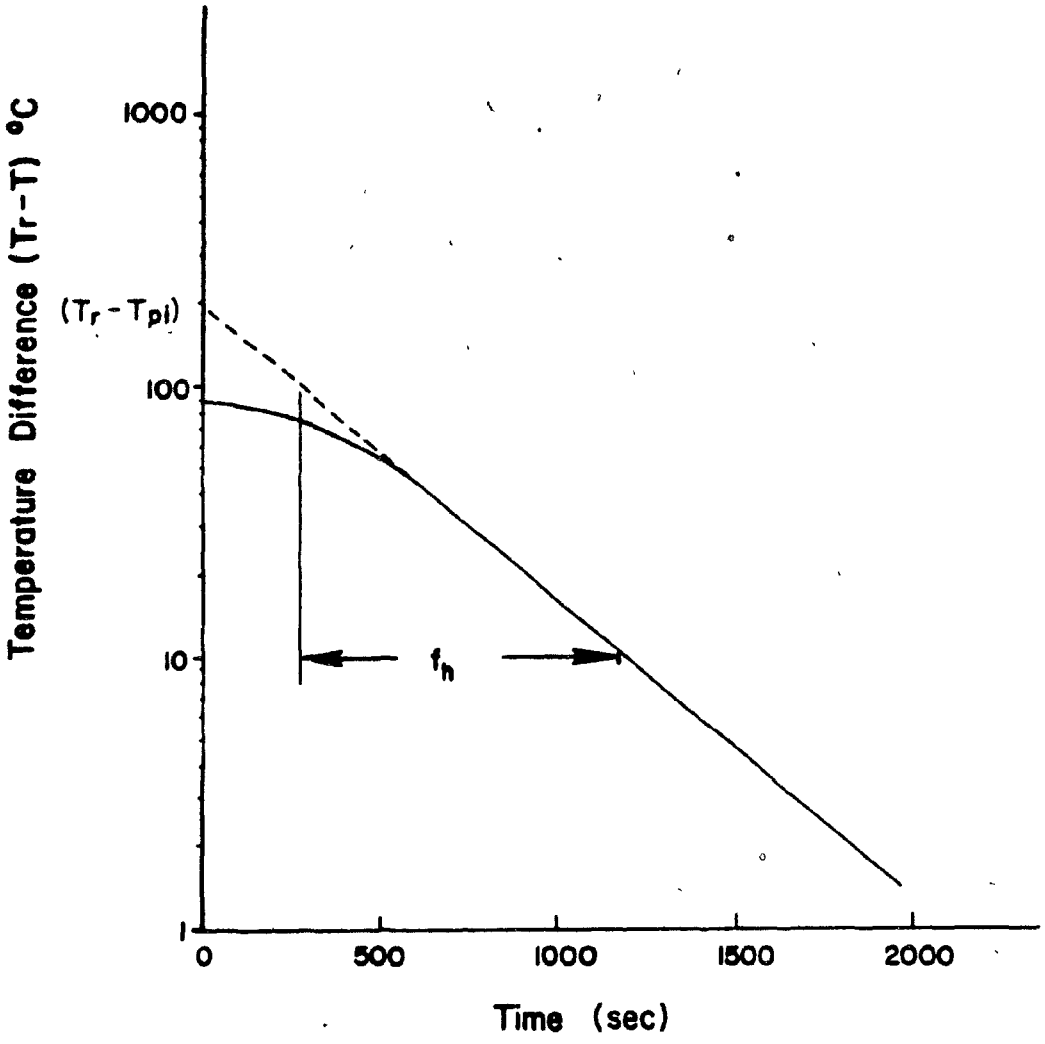


Figure 2 - Heat Penetration Curve

$$\log_{10} \frac{(T_r - T_{pi})}{(T_r - T)} = \frac{1}{f_h} \cdot t \qquad (1)$$

where T_r is the processing (or retort) temperature, T is the calculated temperature (in this instance, at the centre of the can), t is the time passed since exposure to the heat source, T_{pi} is the pseudo-initial temperature, and $1/f_h$ is the slope of the heating curve. The pseudo-initial temperature can be determined from the plot of the straight-line approximation of the heating curve, as seen in Figure 2. It represents the initial food temperature that would need to be present to yield the temperature-time curve shown, if there was no initial lag in the heating rate of the food. The value of f_h is the time necessary to reduce the temperature difference (T_r-T) by one log cycle.

Equation 1 can be rewritten as:

$$\log_{10} \frac{j_h}{u} = \frac{1}{f_h} \cdot t \qquad (2)$$

where u is the temperature ratio, and j_h is the heating curve lag factor. The general formula for the temperature ratio is given by:

$$u = \frac{(T_a - T)}{(T_a - T_0)} \qquad (3)$$

where T_a is the environmental temperature (which is T_r is this instance), T is the calculated temperature (here, equal to the centre temperature), and T_0 is equal to T_i here. The lag factor j_h is a measure of the lag in

establishing a uniform heating rate in the food upon exposure to the temperature input function. It is defined as:

$$j_h = \frac{(T_r - T_{pi})}{(T_r - T_i)} \tag{4}$$

Equations 2, 3 and 4 can be combined to yield the logarithmic heating curve equation in the form used in model one of BALLST:

$$T = T_r - (T_r - T_i) \cdot j_h \cdot 10^{(-t/f_h)} \tag{5}$$

The cooling curve of model one was considered by Ball to be comprised of first a hyperbolic section and then a logarithmic part. The hyperbolic section is given by (Ball, 1923):

$$T = T_{cmax} - a_c \cdot \left(\sqrt{1 + \frac{t_{cool}^2}{b_c^2}} - 1 \right) \tag{6}$$

where t_{cool} is the time passed since exposure to the cooling water, and a_c and b_c are fitting constants for the hyperbola. Ball (1923) determined graphically that suitable values for these constants are:

$$a_c = .3 \cdot (T_{cmax} - T_w) \tag{7}$$

$$b_c = .173 \cdot f_c \tag{8}$$

where T_w is the cooling water temperature and f_c is the reciprocal of the

slope of the cooling curve. The temperature at which the cooling curve switches from a hyperbolic shape to a logarithmic one is given by:

$$T_c^* = T_{cmax} - .343 \cdot (T_{cmax} - T_w) \tag{9}$$

This switch temperature will occur at time t_c^*, equal to:

$$t_c^* = f_c \cdot \log_{10}\left(j_c \cdot \frac{(T_{cmax} - T_w)}{(T_c - T_w)}\right) \tag{10}$$

Equation 10 can be simplified, with the use of equation 9, to give:

$$t_c^* = f_c \cdot \log_{10}(j_c/.657) \tag{11}$$

After time t_c^* has passed, the cooling curve is calculated using the general logarithmic equation:

$$T = T_a - (T_a - T_0) \cdot j_c \cdot 10^{(-t/f_c)} \tag{12}$$

where the environmental temperature T_a represents T_w in this instance, the starting temperature T_0 refers to T_{cmax}, and the general time variable t is equal to t_{cool}. This equation can also be written as:

$$T = T_w + (T_{cmax} - T_w) \cdot j_c \cdot 10^{(-t_{cool}/f_c)} \tag{13}$$

Ball (1923) determined, again graphically, that the value of 1.41 for j_c would yield the equations which most closely approximate the empirical heat

penetration data. A temperature-time curve as predicted by model one is shown in Figure 1.

3.1.1.2 Model Two

As can be seen from Figure 1, the initial portion of the heating curve generated by Ball's equations in model one does not exhibit the shape one would expect from examining actual temperature data. In model two, to more closely approximate physical heat penetration curves, the heating curve is modeled as first a hyperbolic and then a logarithmic section. This modification of the heating equation will not affect subsequent lethality calculations, since the early temperatures do not contribute significantly to the magnitude of the lethality accumulated.

The hyperbolic portion of the heating curve is calculated by:

$$T = T_i + a_h \cdot \left(\sqrt{1 + \frac{t^2}{b_h^2}} - 1 \right) \qquad (14)$$

The logarithmic heating section is calculated similarly to model one:

$$T = T_r - (T_r - T_i) \cdot j_h \cdot 10^{(-t/f_h)} \qquad (15)$$

There are two fitting constants in the hyperbolic heating equation, a_h and b_h, which can be determined by imposing the two following conditions:

1- The point of intersection of the hyperbolic and logarithmic heating curve

sections occurs at (similar to Ball's intersection point for the cooling curves of model one):

$$T_h^* = T_i + .343 \cdot (T_r - T_i) \qquad (16)$$

2- At this point of intersection the slopes of the two heating curve sections must be the same, as plotted on Cartesian coordinates. This ensures a smooth transition from hyperbolic to logarithmic heating.

From these two conditions, it is possible to determine the expression for the fitting constants a_h and b_h, in terms of the independent variables T_r, T_i, j_h and f_h. The following steps are followed (adapted from Kok, 1982):

1- The coordinates of the point of intersection of the two heating section are determined.

2- These coordinates are substituted into the hyperbolic equation 14 to obtain the first relationship between a_h and b_h.

3- The derivatives of the equations for the hyperbolic and logarithmic heating sections are found.

4- The coordinates of the point of intersection are substituted into these two differential equations.

5- By equating the two differential equations, the second relationship between

a_h and b_h is determined.

6- The equations for a_h and b_h are found from the two relationships.

The steps of the solution for a_h and b_h are now examined in detail:

1- The coordinates of the point of intersection:

At the point of intersection the temperature is equal to T_h , as defined in equation 16, and is also equal to T, from equation 15. Setting equation 16 equal to 15, one obtains the following expression:

$$T_i + .343 \cdot (T_r - T_i) = T_r - (T_r - T_i) \cdot j_h \cdot 10^{(-t_h^*/f_h)} \qquad (17)$$

The x-coordinate, time, can be determined from equation 17. The equation for the time coordinate is given by:

$$t_h^* = f_h \cdot \log_{10}(j_h/.657) \qquad (18)$$

Thus, from equations 16 and 18, the coordinates of the point of intersection (T_h^*, t_h^*) are found.

2- Substitution of (T_h^*, t_h^*) into the hyperbolic heating equation:

At the point of intersection the expression for T_h^* , from equation 16, is equal to the temperature calculated by the hyperbolic heating equation 15, at time = t_h^* :

$$T_i + .343 \cdot (T_r - T_i) = T_i + a_h \cdot \left(\sqrt{1 + \frac{t_h^{*2}}{b_h}} - 1 \right) \qquad (19)$$

From this equation, the first relationship between a_h and b_h can be obtained:

$$.343 = \frac{a_h}{(T_r - T_i)} \cdot \left(\sqrt{1 + \frac{t_h^{*2}}{b_h^2}} - 1 \right) \qquad (20)$$

3- The derivative of the hyperbolic and logarithmic heating equations:

The time derivation of the hyperbolic heating equation 14 is given by:

$$\frac{\delta T}{\delta t} = \frac{a_h}{b_h^2} \cdot \left(\frac{t}{\sqrt{1 + t^2/b_h^2}} \right) \qquad (21)$$

The time derivative of the logarithmic heating equation 15 is:

$$\frac{\delta T}{\delta t} = -(T_r - T_i) \cdot j_h \cdot \left(\frac{-\ln 10}{f_h} \right) \cdot e^{(-\ln 10 \cdot t/f_h)} \qquad (22)$$

Equation 22 can also be written as:

$$\frac{\delta T}{\delta t} = -(T_r - T_i) \cdot j_h \cdot (-2.30259/f_h) \cdot 10^{(-t/f_h)} \qquad (23)$$

4- Substitution of (T_h^*, t_h^*) into differential equations:

The differential equation of the hyperbolic heating section, at the point of intersection is as follows:

$$\frac{\delta T}{\delta t} = \frac{a_h}{b_h^2} \cdot \left(\frac{t_h^*}{\sqrt{1 + (t_h^*/b_h^2)}} \right) \qquad (24)$$

The differential equation of the logarithmic heating section, again at the point of intersection, is given by:

$$\frac{\delta T}{\delta t} = -(T_r - T_i) \cdot j_h \cdot \left(\frac{-2.30259}{f_h} \right) \cdot 10^{(-t_h^*/f_h)} \qquad (25)$$

5- Equating the two differential equations:

At the point of intersection the slopes of the two heating equations are equal. Thus, the two differential equations can be set equal to each other, to obtain the second relationship between a_h and b_h:

$$\frac{a_h}{b_h} \cdot \left(\frac{t_h}{\sqrt{1 + t_h^*/b_h^2}} \right) = -(T_r - T_i) \cdot j_h \cdot \left(\frac{-2.30259}{f_h} \right) \cdot 10^{(-t_h^*/f_h)} \qquad (26)$$

6- Determining a_h and b_h:

By mathematical manipulation of the two relationships between a_h and b_h, given in equations 20 and 26, the expressions for a_h and b_h can be derived:

$$a_h = .343 \cdot \left(\frac{S - .22673}{.45346 - S} \right) \cdot (T_r - T_i) \qquad (27)$$

where $S = \log_{10}(j_h/.657)$ (28)

$$b_h = Z(S) \cdot f_h^2 \qquad (29)$$

$$\text{where } Z(S) = \frac{1 - (.45346/S) + (.0514065/S^2)}{(.45346/S^3) - (1/S^2)} \qquad (30)$$

In model two, the cooling curve is also comprised of first a hyperbolic and then a logarithmic section. The cooling curve equations are the same as those described in the cooling section of model one, with the fitting constants a_c and b_c and the switch time and temperature being calculated similarly. A typical temperature-time curve generated by model two is shown in Figure 1.

3.1.1.3 Model Three

In model three, the heating curve is represented as being first hyperbolic and then logarithmic, as is the cooling curve. The heating curve is calculated in the same manner as that of model two, with the derived hyperbolic fitting constants a_h and b_h. The equations will not be repeated in this section. The cooling curve differs from model one and two, however, in that the hyperbolic fitting constants are calculated, the values for a_c and b_c that Ball (1923) derived graphically are not used. Also, user-supplied values of j_c are taken, instead of a constant value of 1.41.

The hyperbolic cooling curve section is given by equation 6, and the logarithmic cooling section by equation 13. The hyperbolic- to- logarithmic switch time t_c^* is found from equation 10, and the switch temperature T_c^* from equation 9. The j_c value, rather than being assumed to be 1.41, is supplied by the user. Another difference between the hyperbolic cooling curve section of model three, and those of model one and two, is in the way a_c and b_c are calculated. These hyperbolic cooling curve fitting constants are derived by

the method used to find the hyperbolic heating curve fitting constants (a_h and b_h) of model two. In brief, the method of solution is comprised of the following steps: 1) find the coordinates of the point of intersection of the hyperbolic and logarithmic cooling sections, 2) substitute these coordinates into the hyperbolic cooling equation 6 to obtain the first relationship between a_c and b_c, 3) find the derivatives of the hyperbolic and logarithmic cooling equations 6 and 13, 4) substitute the coordinates of the point of intersection into these two differential equations, 5) by equating the differential equations, derive the second relationship between a_c and b_c, 6) determine the expressions for the hyperbolic fitting constants from the relationships obtained in steps 2 and 5.

The temperature-time curve generated by model three is shown in Figure 1. The differences between the curves generated by the three models can be noted.

3.1.2 Spatial Temperature Distribution Approach (can)

In the previous section the cylindrical container centre temperature history during sterilization and cooling was discussed. In this section, the modeling of the temperature distribution in the can as a spatial as well as temporal function, is examined. The two algorithms CAN1 and CAN2 (in program modules 4 and 5 respectively) calculate the temperature at any location in a can, after exposure to a sudden sustained temperature change. The temperature distribution is thus a function of the radial and axial coordinates in the can, as well as time. The assumptions upon which the models are based are that the food temperature is initially uniform, the thermal diffusivity is considered independent of temperature and is isotropic, and heat transfer

occurs only by conduction. For the purpose of model development, the cylindrical container is considered to have its coordinate system originate at its centre. The radius of the can is 2R and the length is 2L. Only one-half of the can is modeled; the inherent symmetry of the cylinder eliminates the need to repeat the calculations for the second half. CAN1 and CAN2 differ, however, in the way the convective surface heat transfer coefficient is taken into consideration. In CAN1 the equations are based on the assumption it is infinite. The convective heat transfer at the surface of the can is therefore ignored. In CAN2, the equations utilized allow the value of the heat transfer coefficient to be specified by the user.

3.1.2.1 CAN1

The classical equation of conduction of heat in a finite cylinder is used (Carslaw and Jaeger, 1959):

$$\frac{\delta T}{\delta t} = \frac{k}{\rho \cdot c_p} \cdot \left(\frac{1}{r} \cdot \frac{\delta (r \cdot \delta T / \delta r)}{\delta r} + \frac{\delta^2 T}{\delta z^2} \right) \qquad (31)$$

where k is the thermal conductivity of the food, ρ is the food density, c_p is the food's specific heat, r is the radial coordinate and z is the axial coordinate. According to basic heat transfer theory (Holman, 1976):

$$\alpha = \frac{k}{\rho \cdot c_p} \qquad (32)$$

where α is the thermal diffusivity of the food, equation 31 can be rewritten as:

$$\frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{1}{r} \cdot \frac{\delta (r \cdot \delta T / \delta r)}{\delta r} + \frac{\delta^2 T}{\delta z^2} \right) \tag{33}$$

The temperature distribution caused by the heat transfer in a finite cylinder can be calculated by separately determining the temperature distributions through the cylindrical surface (infinite cylinder case) and through the ends of the can (infinite plate case). These two solutions to the differential heat transfer equations are then multiplied, using Newman's rule (Merson et al., 1978).

For the cylindrical surface, the temperature distribution is given by (Carslaw and Jaeger, 1959):

$$\frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{1}{r} \cdot \frac{\delta (r \cdot \delta T / \delta r)}{\delta r} \right) \tag{34}$$

Equation 34 can be simplified to:

$$\frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{\delta^2 T}{\delta r^2} + \frac{1}{r} \cdot \frac{\delta T}{\delta r} \right) \tag{35}$$

For the ends of the can, the temperature distribution is calculated using (Carslaw and Jaeger, 1959):

$$\frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{\delta^2 T}{\delta z^2} \right) \tag{36}$$

The solutions for the differential equations representing the temperature

distribution through both the infinite cylinder and the infinite plate are given in terms of the temperature ratio u, as defined in equation 3. The temperature ratio for the infinite cylinder is determined by (Luikov, 1968):

$$u_{cyl} = \sum_{n=1}^{\infty} A_n \cdot J_0(\mu_n \cdot r/(D/2)) \cdot e^{(-\mu_n^2 (\alpha \cdot t/(D/2)^2))} \quad (37)$$

$$\text{where } A_n = \frac{2}{\mu_n \cdot J_1(\mu_n)} \quad (38)$$

where μ_n is the n^{th} root of the Bessel function of the first kind of the zero order, and $J_0(x)$ and $J_1(x)$ are the values of the Bessel functions of the first kind of the zero and first order respectively (see Appendix 1 for the Bessel functions).

The temperature ratio for the infinite plate is (Luikov, 1968):

$$u_{pl} = \sum_{m=1}^{\infty} A_m \cdot \cos\left(\mu_m \frac{z}{L}\right) \cdot e^{(-\mu_m^2 \cdot \alpha \cdot t/L^2)} \quad (39)$$

$$\text{where } A_m = \frac{2}{\mu_m} \cdot (-1)^{m+1} \quad (40)$$

$$\text{and } \mu_m = (2m - 1) \cdot \pi/2 \quad (41)$$

The temperature ratio for the finite cylinder can be found by multiplying together the temperature ratios of the infinite cylinder and the infinite plate (Luikov, 1968):

$$u = u_{cyl} \cdot u_{pl} \tag{42}$$

$$u = \sum_{m,n=1}^{\infty} A_n \cdot A_m \cdot J_0\left(u_n \cdot \frac{r}{(D/2)}\right) \cdot \cos\left(u_m \frac{z}{L}\right) \cdot e^{(-(u_n^2 + u_m^2) \cdot \alpha t)/(LD/2)^2)} \tag{43}$$

Thus, the temperature at any position in the can, at any time, can be calculated from:

$$T = T_a - u \cdot (T_a - T_0) \tag{44}$$

Equation 44 is the general temperature equation. When the heating curve is being determined T_a represents the retort temperature T_r, and T_0 is the initial food temperature T_i. To calculate the temperature after cooling has started, equation 44 is used, with different values substituted for some of the variables. First, the heating temperatures T_{heat} are calculated until the end of the processing period, as if no cooling is to be done:

$$T_{heat} = T_a - u \cdot (T_a - T_0) \tag{45}$$

where T_a is equal to T_r, T_0 is T_i, and the time value is t. Now a set of cooling temperatures is calculated, assuming that T_r is the new initial temperature in the food and T_w is the new environmental temperature, from the time when cooling starts until the end of the processing period:

$$T_{cool} = T_w - u \cdot (T_w - T_r) \tag{46}$$

where the time value used in the expression u is t_{cool}, rather than t. The temperatures T_{cool} will deviate from the actual food temperatures because: 1) when cooling starts the temperature in the food is not the retort temperature but a value close to it and, 2) the interior food temperatures are still increasing at the start of cooling, and the subsequent effect on the predicted temperatures should be considered, which is not the case in the calculation of T_{cool}. The actual cooling curve temperatures are calculated by:

$$T = T_{heat} - (T_r - T_{cool}) \tag{47}$$

3.1.2.2 CAN2

The equations upon which CAN2 is based allow the specification of the convective surface heat transfer coefficient, by way of the Biot modulus Bi. Following the same equation development explained in the previous section for CAN1, the temperature ratio for the infinite cylinder is given by (Ramaswamy et al., 1982):

$$u_{cyl} = 2\, Bi_{cyl} \cdot \sum_{m=1}^{\infty} \frac{J_0(\gamma_m \cdot r/(D/2))}{(Bi_{cyl}^2 + \gamma_m^2) \cdot J_o(\gamma_m)} e^{(-\gamma_m^2 \cdot Fo_{cyl})} \tag{48}$$

where γ_m is the mth positive root of: $\gamma \cdot J_1(\gamma) = Bi_{cyl} \cdot J_0(\gamma)$ (49)

$$Bi_{cyl} = h \cdot (D/2) \;/\; k \tag{50}$$

$$Fo_{cyl} = \alpha \cdot t \;/\; (D/2)^2 \tag{51}$$

The temperature ratio for the infinite plate is (Ramaswamy et al., 1982):

$$u_{pl} = \sum_{n=1}^{\infty} \frac{2 \sin\beta_n}{\beta_n + \sin\beta_n \cdot \cos\beta_n} \cdot \cos(\beta_n \cdot z/L) \cdot e^{(-\beta_n^2 \cdot Fo_{pl})} \quad (52)$$

where β_n is the nth positive root of: $\beta \cdot \tan\beta = Bi_{pl}$ (53)

$$Bi_{pl} = h \cdot L / k \quad (54)$$

$$Fo_{pl} = \alpha \cdot t / L^2 \quad (55)$$

As shown in equation 42, the temperature ratio for the finite cylinder is the product of the temperature ratios for the infinite cylinder and the infinite plate. The resultant finite cylinder temperature ratio is:

$$u = 4\, Bi_{cyl} \cdot \sum_{m,n=1}^{\infty} \frac{J_0(\gamma_m \cdot r/(D/2))}{(Bi_{cyl}^2 + \gamma_m^2) \cdot J_0(\gamma_m)} \cdot \frac{\sin\beta_n}{\beta_n + \sin\beta_n \cdot \cos\beta_n} \cdot \cos(\beta_n \cdot z/L) \cdot e^{(-\alpha \cdot t\,(\gamma_m^2/(D/2)^2 + \beta_n^2/L^2))} \quad (56)$$

The temperature in the can, at any position and time, can now be calculated:

$$T = T_a - u \cdot (T_a - T_0) \quad (57)$$

The heating and cooling curves are calculated, using the same approach described in the previous section (3.1.2.1) , with equations 45, 46 and 47.

3.1.3 Spatial Temperature Distribution Approach (pouch)

The temperature distribution in a retortable pouch has been modeled as a function of position in the pouch, and time. The temperature is calculated, at any position and at any time, after the pouch has been exposed on all sides to a sudden sustained temperature change. The models are based on the following assumptions: only conduction - heating occurs, the thermal diffusivity is independent of temperature and is isotropic, and the initial temperature throughout the pouch is uniform. The pouch is modeled as a rectangular brick, or parallelopiped. It is considered to have the origin of the coordinate system at its centre. The half-lengths along the X, Y and Z-axis are given by L_x, L_y and L_z respectively. Calculations need be done for only one-eighth of the pouch, due to its symmetry.

The temperature-calculating algorithms are incorporated in POUCH1 and POUCH2 (program modules SUB2 and and SUB3 respectively). POUCH1 performs the temperature calculations based on the assumption that the convective surface heat transfer coefficient is infinite; the effect of the convective heat transfer at the surface of the pouch is ignored. The equations of POUCH2, however, allow the specification of the heat transfer coefficient by the user.

3.1.3.1 POUCH1

The general equation of heat conduction in a parallelopiped (pouch) is utilized (Carslaw and Jaeger, 1959):

$$\frac{\delta T}{\delta t} = \alpha \left(\frac{\delta^2 T}{\delta x^2} + \frac{\delta^2 T}{\delta y^2} + \frac{\delta^2 T}{\delta z^2} \right) \qquad (58)$$

The temperature profile caused by the transfer of heat through a parallelopiped can be calculated by separately determining the temperature profile through each of its representative sides, where each side is considered to be an infinite plate, and then multiplying the three solutions. The heat transfer through the X, Y and Z infinite plates are given by (Carslaw and Jaeger, 1959):

$$\frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{\delta^2 T}{\delta x^2} \right) \quad ; \quad \frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{\delta^2 T}{\delta y^2} \right) \quad ; \quad \frac{\delta T}{\delta t} = \alpha \cdot \left(\frac{\delta^2 T}{\delta z^2} \right) \qquad (59)$$

The solutions for these three differential equations can be expressed in terms of the temperature ratio, u. The temperature ratio for the infinite plate has already been given in equation 39. To better illustrate the pattern of multiplication, the temperature ratio is given here for each infinite plate:

$$u_x = \frac{4}{\pi} \sum_{\ell=1}^{\infty} \frac{\cos\left((2\ell-1) \cdot \pi \cdot x / 2L_x \right)}{(2\ell-1)\,(-1)^{(\ell+1)}} \cdot e^{\left(-\alpha \cdot t\, (2\ell-1)^2\, \pi^2 / 4L_x^2 \right)} \qquad (60)$$

$$u_y = \frac{4}{\pi} \sum_{m=1}^{\infty} \frac{\cos\left((2m-1) \cdot \pi \cdot y / 2L_y \right)}{(2m-1)\,(-1)^{(m+1)}} \cdot e^{\left(-\alpha \cdot t\, (2m-1)^2 \cdot \pi^2 / 4L_y^2 \right)} \qquad (61)$$

$$u_z = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\cos\left((2n-1) \cdot \pi \cdot x / 2L_z \right)}{(2n-1)\,(-1)^{(n+1)}} \cdot e^{\left(-\alpha\, t\, (2n-1)^2 \cdot \pi^2 / 4L_z^2 \right)} \qquad (62)$$

The temperature ratio for the finite parallelopiped is found by multiplying the temperature ratios of each infinite plate together (Williamson and Adams,

1919):

$$u = u_x \cdot u_y \cdot u_z \tag{63}$$

$$u = \frac{64}{\pi^3} \sum_{\ell,m,n=1}^{\infty} \frac{\cos((2\ell-1)\cdot\pi\cdot x/2L_x)\cdot\cos((2m-1)\cdot\pi\cdot y/2L_y)\cdot\cos((2n-1)\cdot\pi\cdot z/2L_z)}{(2\ell-1)\cdot(2m-1)\cdot(2n-1)\cdot(-1)^{\ell+m+n+1}}$$

$$\cdot e^{-\alpha t\left(((2\ell-1)^2\cdot\pi^2/4L_x^2) + ((2m-1)^2\cdot\pi^2/4L_y^2) + ((2n-1)^2\cdot\pi^2/4L_z^2)\right)} \tag{64}$$

The temperature at any position in the retort pouch can be calculated, at any time, by:

$$T = T_a - u\cdot(T_a - T_o) \tag{65}$$

As discussed in section 3.1.2.1 the heating curve is calculated by substituting variables in equation 65 in the following manner:

$$T = T_r - u\cdot(T_r - T_i) \tag{66}$$

The cooling curve is calculated by:

$$T = T_{heat} - (T_r - T_{cool}) \tag{67}$$

$$\text{where } T_{heat} = T_r - u\cdot(T_r - T_i) \tag{68}$$

and $T_{cool} = T_w - u \cdot (T_w - T_r)$ (69)

3.1.3.2 POUCH2

The model POUCH2 calculates the temperature in a retort pouch, while allowing the specification of the surface convective heat transfer coefficient, by means of the Biot modulus. The same equation development as in the previous section for POUCH1, equations 58 and 59, is used. The temperature ratios for each of the three infinite plates are (Ramaswamy et al., 1982):

$$u_x = \sum_{\ell=1}^{\infty} \frac{2 \sin \beta_\ell}{\beta_\ell + \sin \beta_\ell \cdot \cos \beta_\ell} \cdot \cos(\beta_\ell \cdot x/L_x) \cdot e^{(-\beta_\ell^2 \cdot Fo_x)} \quad (70)$$

where β_ℓ is the ℓth positive root of: $\beta \cdot \tan \beta = Bi_x$ (71)

$$Fo_x = \alpha \cdot t / L_x^2 \quad (72)$$

$$u_y = \sum_{m=1}^{\infty} \frac{2 \sin \beta_m}{\beta_m + \sin \beta_m \cdot \cos \beta_m} \cdot \cos(\beta_m \cdot y/L_y) \cdot e^{(-\beta_m^2 \cdot Fo_y)} \quad (73)$$

where β_m is the mth positive root of: $\beta \cdot \tan \beta = Bi_y$ (74)

$$Fo_y = \alpha \cdot t / L_y^2 \quad (75)$$

$$u_z = \sum_{n=1}^{\infty} \frac{2 \sin \beta_n}{\beta_n + \sin \beta_n \cdot \cos \beta_n} \cdot \cos(\beta_n \cdot z/L_z) \cdot e^{(-\beta_n^2 \cdot Fo_z)} \qquad (76)$$

where β_n is the nth positive root of: $\beta \cdot \tan \beta = Bi_z$ (77)

$$Fo_z = \alpha \cdot t / L_z^2 \qquad (78)$$

The temperature in the finite parallelopiped is found by multiplying the three temperature ratios given in equations 70, 73 and 76. The heating and cooling curves are calculated using the same method described in the previous section, in equations 66 to 69.

3.2 Process Adequacy

The objective of thermal processing is the total destruction of all contaminant microorganisms which might be responsible for microbial decomposition of the food, and food-borne infections and intoxications. The processing regimes for low-acid, hermetically-sealed foods must guarantee the complete destruction of Clostridium botulinum, and should also eliminate any other organisms that can cause health or spoilage problems (Pelczar et al., 1977). Vegetative microbial cells are very heat-sensitive, and are easily destroyed. It is the bacterial spores that are the major problem in low-acid foods, as they may be 10,000 times as resistant to heat as the vegetative cells (Charm, 1978). Both spores and cells are commonly considered to be destroyed by heat according to first order reaction rate kinetics. The

general rate equation is given by (Teixeira et al., 1969b):

$$-\frac{dC}{dt} = k_r \cdot C \tag{79}$$

where C is the concentration of live bacteria and k_r is the reaction rate constant. This equation can also be written as:

$$-\frac{d(\log_{10}C)}{dt} = \frac{1}{D} \tag{80}$$

$$\text{where } D = 2.303 / k_r \tag{81}$$

D is the decimal reduction time characteristic of the microorganism, which is the processing time necessary to reduce the microbial population by one log cycle, as shown in Figure 3. The D value is often modeled as a function of processing temperature (Stumbo, 1973; Merson et al., 1978):

$$D = D_{ref} \cdot 10^{(T_{ref}-T)/z} \tag{82}$$

where D is the D value at temperature T, D_{ref} is the D value at temperature T_{ref}, and z is a destruction parameter characteristic of the organism, and can be considered to be constant throughout processing. Integrating equation 80 from time t_a to t_b, when the microbial concentration changes from a to b respectively, yields the following expression:

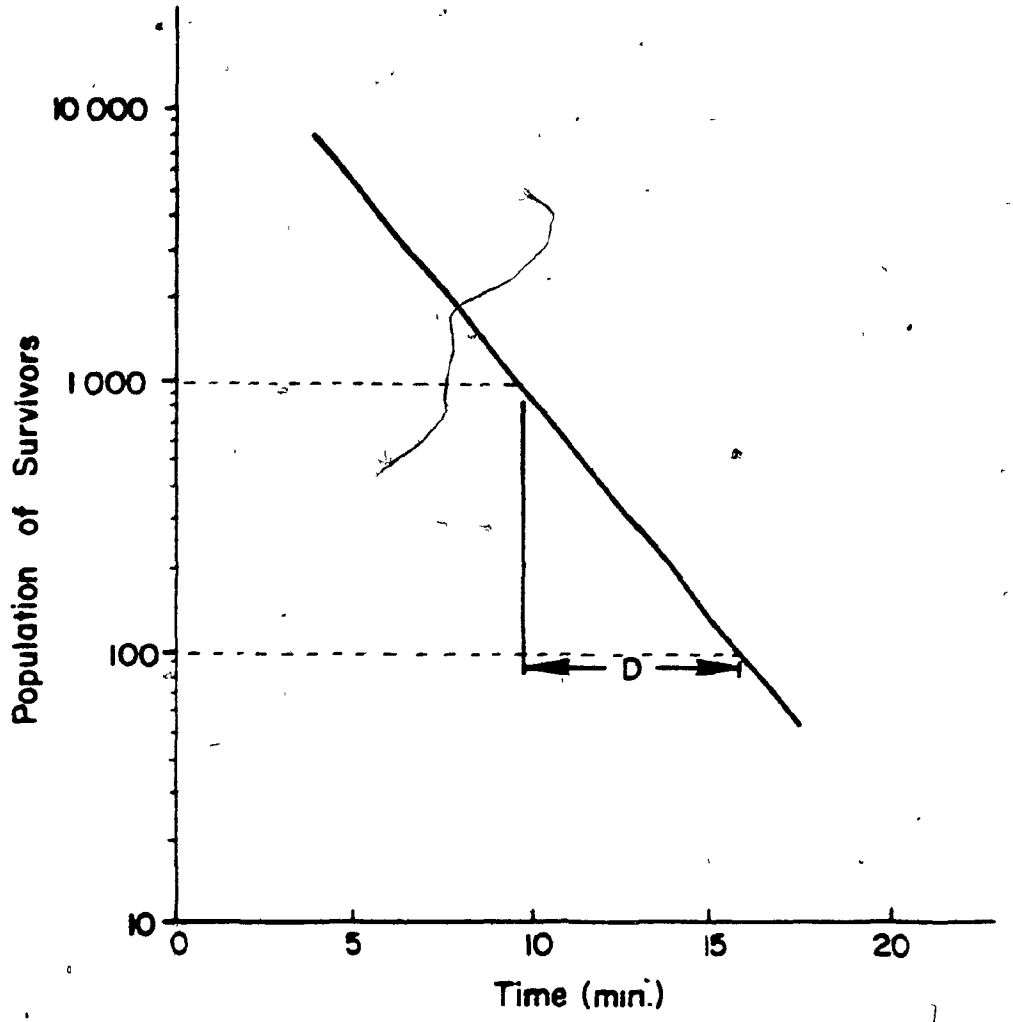


Figure 3 - Representation of the Decimal Reduction Time D

$$-\int_{a}^{b} d(\log_{10}C) = \frac{1}{D_{ref}} \cdot \int_{t_a}^{t_b} \frac{dt}{10^{(Tref-T(t))/z}} \qquad (83)$$

Solving the left hand side of the equation and setting it equal to the sterilizing value F gives (Merson et al., 1978):

$$F^{z}_{Tref} = D_{ref} \cdot (\log a - \log b) = \int_{t_a}^{t_b} \frac{dt}{10^{(Tref-T(t))/z}} \qquad (84)$$

where the F value is dependent upon the temperature and the destruction parameters associated with the particular microorganism. It represents the time required, at a reference temperature, to lower the concentration of spores by a specified factor. The 12D concept frequently used in thermal processing refers to the amount of time required - the F value - to decrease the initial microbial load by a factor of 10^{12}, or in other words, to ensure "commercial sterility" (Esty and Meyer, 1922). The reference temperature for which the F value is calculated is usually taken to be 121 degrees C for low-acid foods (Hayakawa, 1978) and 93 or 100 degrees C for acid foods (Merson et al., 1978).

The product $F^{z}_{Tref} \times 10^{(Tref-T)/z}$ is called the "thermal death time", TDT. It is the time required to reduce the population of the target contaminant microbes by a factor of 10^{12}, when the food is at temperature T. The previously-mentioned destruction parameter z can be obtained from the semilogarithmic plot of TDT vs processing temperature, and represents the temperature increase required for the TDT to be reduced by one log cycle, as shown in Figure 4.

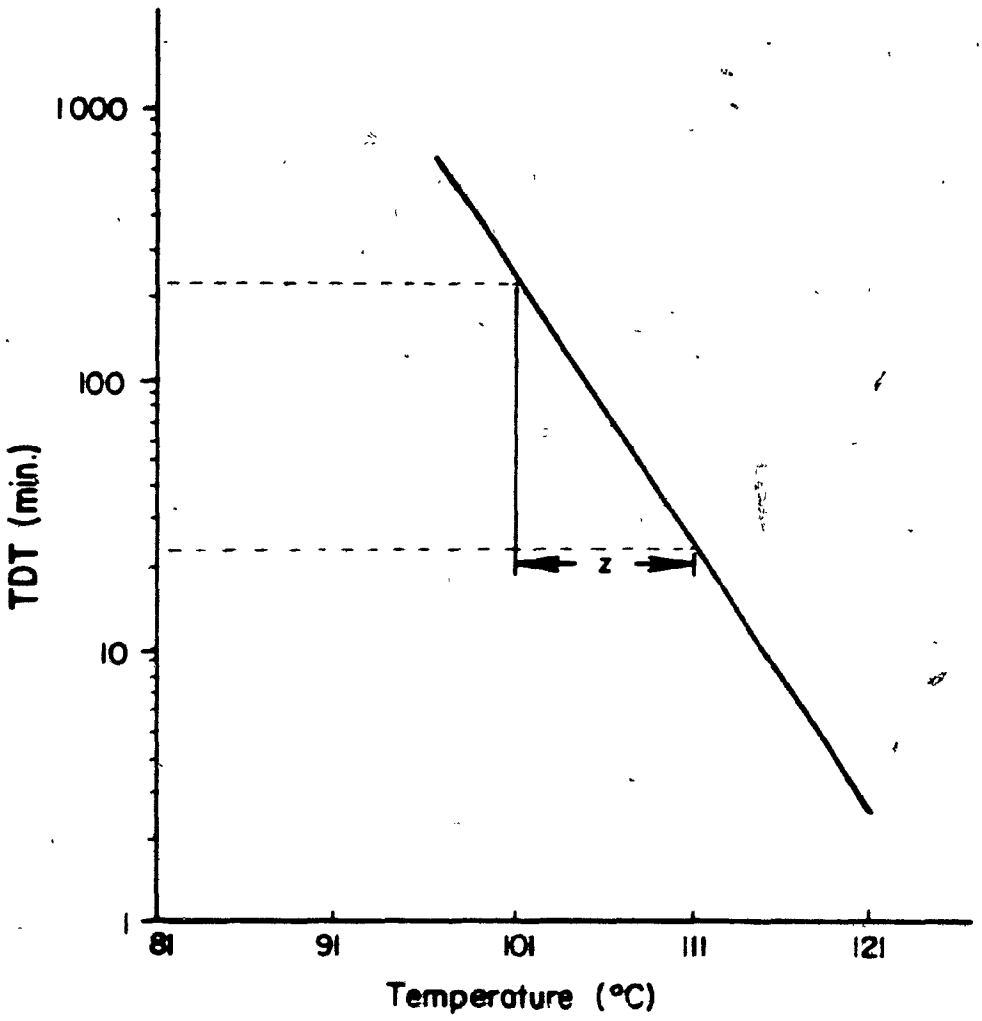


Figure 4 - The Thermal Death Time Curve for Cl botulinum (after Charm, 1978)

The integrand of equation 84 is commonly written using TDT, and is equivalent to the "lethality":

$$\text{Lethality} = \int_{t_a}^{t_b} \frac{dt}{TDT} \qquad (85)$$

The definite integral of equation 85 represents the lethality imparted to the food by the process, and must be greater than unity for sterility to be achieved. The lethality can also be represented as (Merson et al., 1978):

$$\text{Lethality} = \frac{(F_{Tref})\ \text{process}}{(F_{Tref})\ \text{required}} \qquad (86)$$

Again, the result must be greater than unity to ensure sterility in the food. The lethality calculations can be based on either the temperatures at the slowest heating region, or on the temperatures throughout the container. In this research project both approaches were utilized.

3.2.1 Slowest Heating Region Approach

In this instance, the lethality calculations are based on the SHR for a can, whereby only the temperatures at the centre of the can are used. The rationale of this approach is that if the slowest heating region of the can has reached a certain temperature, the rest of the can is sure to have attained at least the same temperature (Bigelow et al., 1920; Ball, 1923). The extra lethality accrued due to the higher outer temperatures is considered to be a safety factor. There is controversy concerning whether or not the centre of the can is actually the region where the highest number of survivors

is likely to be present (Stumbo, 1949; Teixeira et al., 1969a) but this assumption is traditionally held to be true, and will be used in this model.

To facilitate process calculations, the total processing time is considered to be subdivided into increments and the calculations are done for each increment. In this paper, the lethal rate LR during each time increment is calculated by:

$$LR = \frac{1}{TDT} = \frac{1}{F_{Tref}^{z}} \cdot 10^{(T_{avg}-T_{ref})/z} \qquad (87)$$

where T_{avg} is the arithmetic average of the can temperatures at the start and end of the time increment. The lethal rate is based on an incremental straight-line approximation of what is usually a non-linear temperature curve, so some error is introduced here whose magnitude will increase as the curvature of the temperature-time curve becomes more pronounced. As can be seen from equations 85 and 87, the lethal rate is integrated over the processing period to determine the lethality imparted by the heating regime. A trapezoidal integration scheme is used, as shown in Figure 5, to obtain the area under the lethal rate-time curve, and thus determine the lethality imparted by the process. The lethality is approximated by:

$$\text{Lethality} = \sum_{m=1}^{(ntime-1)} LR_m \cdot \Delta t \qquad (88)$$

The accuracy of the value obtained will increase with the number of time increments.

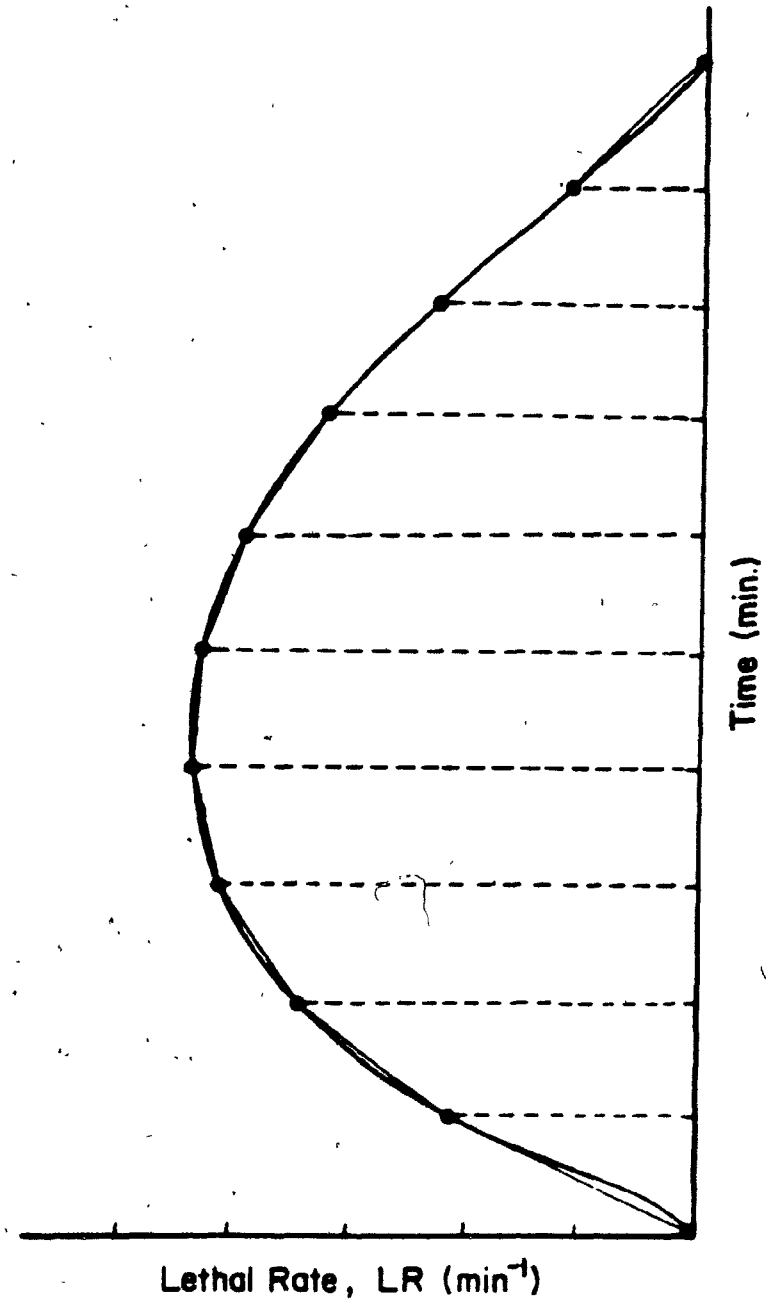


Figure 5 - Trapezoidal Integration of Lethal Rate to obtain Lethality

3.2.2 Elemental Approach

The second method used to calculate process adequacy is to determine the lethality imparted by the process, based on the temperatures attained throughout the container rather than on the centre temperatures alone. For this method, the container is considered to be subdivided into elements whose shapes are dependent upon that of the container. The pouch consists of rectangular elements (Figure 6), while the can is made up of concentric annuli (Figure 7). The elemental approach involves calculating the number of survivors in each element, by means of the lethal rate, and then summing these values over all the container, for each time increment. The lethality imparted can then be determined, for each time increment, by comparing the number of survivors at the end of the time increment with those at the beginning (Teixeira et al., 1969a). The lethality value calculated at the end of the final time increment will then represent the total lethality imparted by the process.

The number of elements in the pouch is calculated by:

$$NELE = (NX-1) \cdot (NY-1) \cdot (NZ-1) \tag{89}$$

where NX, NY and NZ are the number of nodes (including the boundary values) along the X, Y and Z-axis of the pouch. The number of elements in the can is:

$$NELE = (NR-1) \cdot (NZ-1) \tag{90}$$

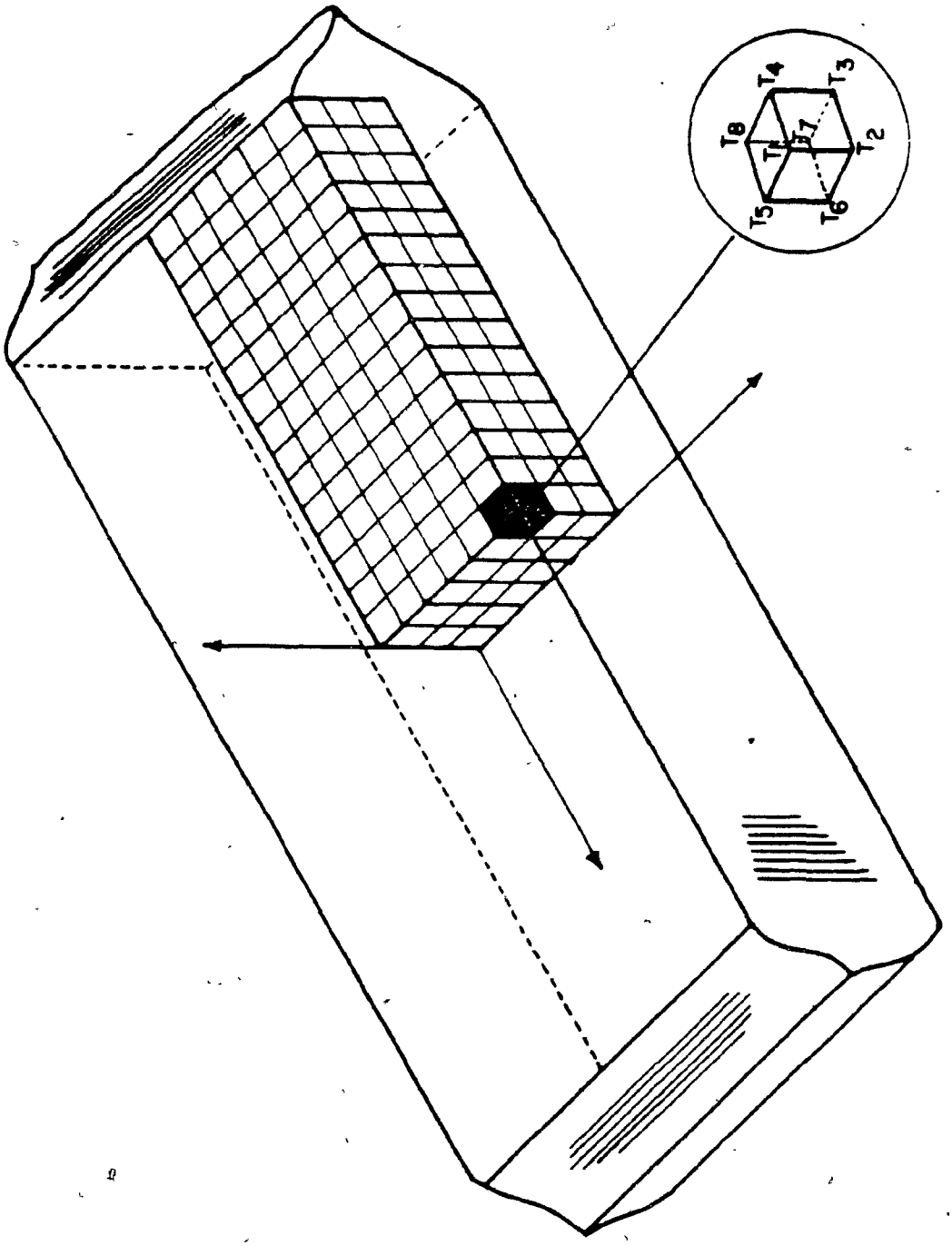


Figure 6 - Elements and Nodal Temperatures in the Pouch

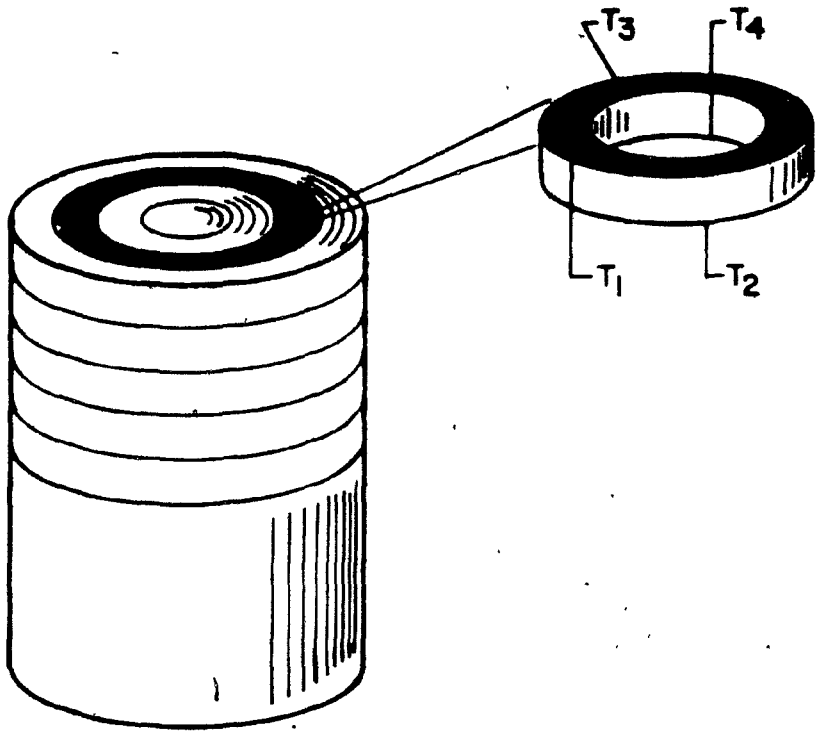


Figure 7 - Elements and Nodal Temperatures in the Can

where NR and NZ are the number of radial and axial nodes, including boundary values, of the can.

The volume of each pouch element is:

$$VELE = (L_x/(NX-1)) \cdot (L_y/(NY-1)) \cdot (L_z/(NZ-1)) \tag{91}$$

while the volume of each can element is given by:

$$VELE = \pi \cdot (L/(NZ-1)) \cdot (R_{outer}^2 - R_{inner}^2) \tag{92}$$

The temperature at the center of each element in the container, TCNT, is approximated by the arithmetic mean of the nodal temperatures for the element. In the pouch, the TCNT is found by averaging the eight corner temperatures of the rectangular element. For the can, the TCNT is comprised of the mean of the four inner and outer radial boundary temperatures of the annular element. The TCNT for a pouch element is given by:

$$TCNT = (T_{i,j,k} + T_{i,j,k+1} + T_{i,j+1,k} + T_{i,j+1,k+1} + T_{i+1,j,k} + T_{i+1,j,k+1} + T_{i+1,j+1,k} + T_{i+1,j+1,k+1}) / 8 \tag{93}$$

where i, j and k are subscripts for the temperatures in the X, Y and Z directions respectively. The TCNT for the can element is calculated by:

$$TCNT = (T_{r,z} + T_{r,z+1} + T_{r+1,z} + T_{r+1,z+1}) / 4 \tag{94}$$

where r and z are subscripts for the temperatures in the radial and axial directions of the can.

Recalling that calculations are performed for only one-eighth of the pouch and one-half of the can, the initial number of microorganisms in the pouch (equation 95) and the can (equation 96), prior to processing, are:

$$ORG0 = (VTOTP / 8) \cdot \rho \cdot C_0 \tag{95}$$

$$ORG0 = (VTOTC / 2) \cdot \rho \cdot C_0 \tag{96}$$

where C_0 is the initial concentration of organisms in the container, and VTOTP and VTOTC are the total volume of the pouch and can, respectively. The remaining model development is based on equations which are identical regardless of the type of container. The initial number of microoganisms in each element, prior to processing, is first calculated:

$$ORGE0 = VELE \cdot \rho \cdot C_0 \tag{97}$$

The lethal rate in each element, during each time increment, is then calculated using the average of the TCNT temperatures at the start and end of the time increment (T_{avg}):

$$LR = \frac{1}{F_{Tref}^{z}} \cdot 10^{(T_{avg}-T_{ref})/z} \tag{98}$$

Using the elemental lethal rate, the number of surviving organisms in the element at the end of the time increment can now be calculated:

$$ORGELE = ORGEST \cdot 10^{(-12 \cdot LR \cdot \Delta t)} \quad (99)$$

where ORGEST represents the number of organisms present in the element at the start of the time increment. The total number of organisms present in the container at the end of the time increment can now be found by summing all the elemental values:

$$ORGTOT = \sum_{i=1}^{nele} ORGELE_i \quad (100)$$

To calculate the lethality, whose magnitude will be 1.0 when the number of organisms has been reduced by a factor of 10^{12}, the ratio of the initial number of organisms in the container to the number of survivors left at the end of the time increment is determined:

$$Lethality = \log_{10}(ORG0 \; / \; ORGTOT) \; / \; 12 \quad (101)$$

At the end of the last time increment, this value will be equal to the total accumulated process lethality.

3.3 Nutrient Retention

Consequences of thermal processing include the degradation of heat-labile

nutrients and other food components, the inactivation of some food properties such as colour, and the augmentation of other food properties such as "cookedness". In order to determine the processing regime which will result in optimal food quality, one must know the effects the regime will have on the food properties and components. Once the heat transfer throughout the food can be predicted, the behaviour of the heat-labile food components can be modeled.

Several models can be used to predict the degradation of nutrients during thermal processing. A common approach is to use the Arrhenius equation and associated kinetics data, the activation energy E_a and k_r (Hill and Grieger-Block, 1980; Lenz and Lund, 1980). Another approach often used employs the D and z parameters to determine nutrient degradation (Jen et al., 1971; Teixeira et al., 1969b; Thijssen and Kochen, 1980) by means of a semilogarithmic equation as described by equation 84 above. In this project the latter approach was taken, but with the use of the F and z values of the nutrient or component under consideration, where the value of F is equal to the value of 12D. The modeling of the nutrient retention therefore closely parallels that of process adequacy, discussed in the previous section. Again, two methods are employed - both the slowest heating region and the elemental approaches.

3.3.1 Slowest Heating Region Approach

In this method the nutrient retention calculations are based on the center temperatures alone, using the same rationale as described in section 3.2.1. The nutrient destruction rate (DR) during each time increment is calculated

by:

$$DR = \frac{1}{F_{nutr}} \cdot 10^{(T_{avg}-T_{ref}) / z_{nutr}} \qquad (102)$$

where F_{nutr} and z_{nutr} are the destruction characteristics of the nutrient. This destruction rate is used to calculate the "lethality" imparted to the nutrient at the end of the time increment:

$$Lethality_{nutr} = \sum_{m=1}^{(ntime-1)} DR_m \cdot \Delta t \qquad (103)$$

This "lethality" value can be used to determine how much of the nutrient remains (in absolute amounts) at the end of the time increment. Keeping in mind that when the "lethality" is equal to 1 a reduction by a factor of 10^{12} of the amount of nutrient is effected, the nutrient remaining at the end of the time increment can be predicted by:

$$XNUTOT = XNUTST \cdot 10^{(12.0 - Lethality_{nutr})} \qquad (104)$$

where XNUTST is the amount of nutrient present at the start of the time increment. To determine the fraction of nutrient retained at the end of the time increment as compared to the amount originally present in the food, the probabilistic approach is taken wherein the original amount of nutrient is considered equal to 1 g:

$$FRANUT = XNUTOT / 1 \qquad (105)$$

At the end of the last time increment, this value FRANUT represents the overall fraction of nutrient retained after processing.

3.3.2 Elemental Approach

The second method of calculating nutrient retention uses temperatures attained throughout the container, rather than only the centre temperatures. The number of elements, the elemental volumes, and the temperatures are all calculated in the same manner as described for process adequacy in section 3.2.2. The initial amount of nutrients in the pouch (equation 106) and in the can (equation 107), before processing, is calculated by:

$$XNUT0 = (VTOTP / 8) \cdot \rho \cdot N_0 \qquad (106)$$

$$XNUT0 = (VTOTC / 2) \cdot \rho \cdot N_0 \qquad (107)$$

where N_0 is the initial concentration of nutrients in the container. The remaining model development is independent of container shape. The initial amount of nutrient in each element, prior to processing, is given by

$$XNUTE0 = VELE \cdot \rho \cdot N_0 \qquad (108)$$

The nutrient destruction rate in each element, during each time increment, is calculated by:

$$DR = \frac{1}{F_{nutr}} \cdot 10^{(T_{avg}-T_{ref}) / z_{nutr}} \qquad (109)$$

This elemental destruction rate is used to determine the amount of nutrient left in the element at the end of the time increment

$$XNUTE = XNUTEST \cdot 10^{(-12.0 \cdot DR \cdot \Delta t)} \tag{110}$$

where XNUTEST is the amount of nutrient present in the element at the start of the time increment, and the $DR \cdot \Delta t$ term is the "lethality" imparted by the process to the nutrient. The amount of nutrient remaining in the container at the end of the time increment is found by summing the elemental amounts:

$$XNUTOT_a = \sum_{i=1}^{(ne)} XNUTE_i \tag{111}$$

The fraction of nutrient retained at the end of the time increment, as compared to the amount originally present, is calculated by:

$$FRANUT = XNUTOT / XNUTO \tag{112}$$

where the original amount, XNUTO, is considered equal to 1. At the end of the last time increment, the value of FRANUT is the overall nutrient fraction retained after processing.

3.4 Optimization Method

The interest shown in maximizing food quality during thermal processing has led to the applications of optimization theory to determine optimal processing regimes. The optimization of quality retention in thermally-processed foods

is based on the disparities in the temperature-dependence of the death of contaminant microbes and the inactivation of nutrients and food properties (Ohlsson, 1980). Since the death rates of microorganisms are generally much more temperature-dependent than are the inactivation rates of nutrients and other quality factors such as colour, using a higher temperature/shorter time process may result in a better quality food (Lund, 1982). The optimization model must take these disparities into account when selecting the processing regime that will result in maximal nutrient retention, while ensuring process adequacy. An example of the temperature-dependent differences in the inactivation of Clostridium botulinum versus thiamine is shown in Figure 8 (Tung and Smith, 1980). Processing at 112.1 ^{o}C for approximately 19.1 minutes will yield a thiamine retention of 90 percent, while a regime of 115.8 ^{o}C for 8.1 minutes will result in 95 percent thiamine retained and a regime of 126.9 ^{o}C for about 0.6 minutes will yield a retention of 99 percent. All the above processes will ensure commercial sterility, but the resultant nutritive values of the food will differ. It is the task of the optimization model to choose the best heating regime.

The optimization model developed in this project is applicable to batch sterilization of food heated in a can. The centre temperatures alone are used to do the calculations (the SHR approach), and the heat transfer equations are based on Ball's method (1923) as discussed in section 3.1.1. The optimal processing regime is selected on the basis of its ability to allow maximal nutrient retention in the food while concurrently satisfying commercial sterility requirements. The choice is made by determining which one of numerous adequate processes will maximize the objective function representing

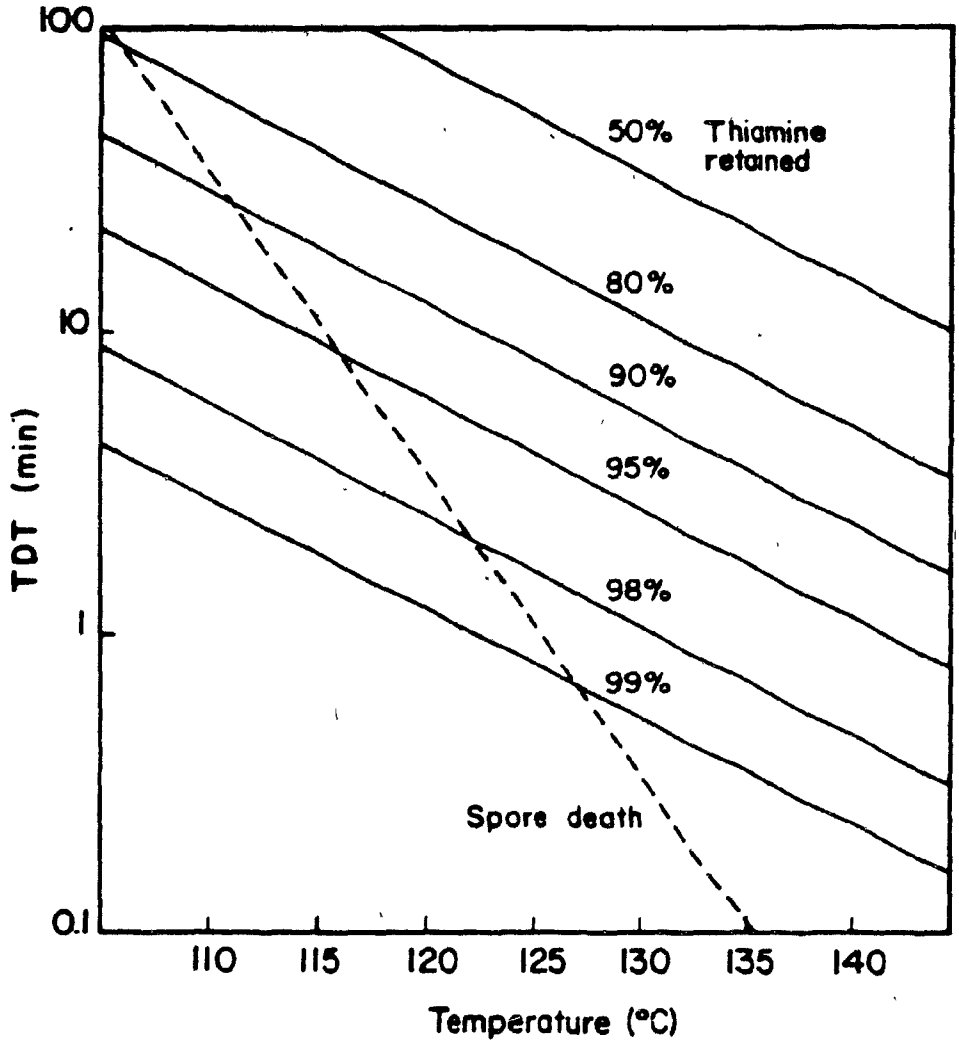


Figure 8 - Spore Inactivation vs Nutrient Degradation
(Tung and Smith, 1980)

nutrient retention. This objective function is comprised of up to four nutrient values. The nutrient retention fractions are combined with their respective weighting factors in a normalized dot product-arrangement to yield the objective function:

$$OBJFN = \frac{\sum_{n=1}^{4} W_n \cdot R_n}{\sum_{n=1}^{4} W_n} \qquad (113)$$

where R_n is the nutrient retention fraction and W_n is the corresponding weighting factor. R_n can range from 0 (none of the nutrient is retained) to 1 (all the nutrient is retained). W_n can also vary from 0 to 1, where a higher weighting factor indicates that a greater importance is attached to that particular nutrient. As well, the value of the objective function can range from 0 (no nutrient retention) to 1 (total retention). The regime which yields the highest value for the objective function is the one which will result in maximal nutrient retentions, and is therefore considered to be the optimal process.

IV. THE COMPUTER PACKAGE

4.1 Overview

The program package was designed to perform simulation, minimization / maximization and optimization calculations for a variety of thermal processes. It is arranged hierarchically, comprising several levels of modules which include system startup programs, user-friendly interfaces, temperature-calculating routines and service facilities. Using this approach the modules can be loaded and overlayed in volatile memory as they are needed, to circumvent the limitation that only 64 KB of RAM can be allocated to a BASIC program. Thus, the program package, which is considerably larger than 64 KB, can still be implemented in BASIC on a microcomputer. Also, this approach facilitates package updating and the incorporation of additional modules.

During operation, control is transferred up and down between the various program levels. The user communicates solely with intelligent, interactive interfaces, the calculating routines remain invisible. The interfaces ask for appropriate information and check the validity and allow correction of unreasonable data. Information such as system status codes, intermediate results and user-supplied values are stored in temporary files. The primary purpose of these files is to transfer the information between the program levels in the hierarchy, thus allowing inter-level communication during operation. A schematic of the package is shown in Figure 9.

The system is installed on an IBM PC having 640 KB of memory, a colour

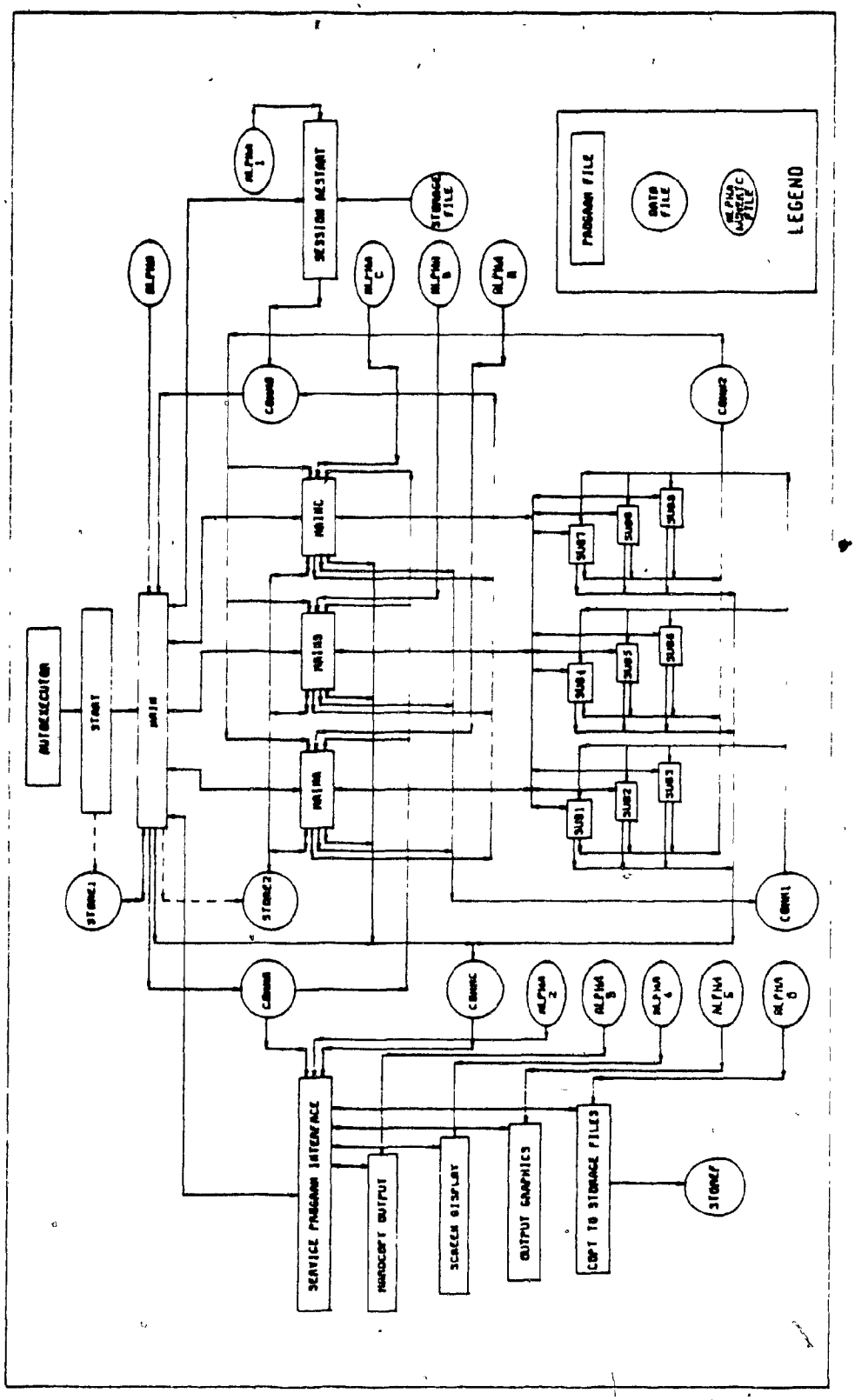


Figure 9 - Schematic of the Program Package

monitor, two double-sided disk drives and one configured virtual disk. The package resides on two 5.25 inch floppy diskettes. The first diskette contains the IBM operating system files, the initialization programs AUTOEXEC and START, all interactive, service and temperature-calculating programs in compiled BASIC form, and the ALPHA text files used by the interfaces (with the exception of ALPHAD, which was stored on the second diskette due to lack of space). The compiled versions of the programs are used during package operation as they ensure fast program execution and provide a certain degree of software security, since compiled programs are not easily decipherable. On the second diskette are stored the communication files COMMA, COMMB and COMMC and the storage files STORE1, STORE2 and STOREF, all used by the system to transfer information between the program levels, as well as the saved graphics screens used for display of program results. During system operation the first diskette remains in drive A and the second in drive B. The virtual drive C is automatically created in the RAM of the IBM PC when the system is booted up. The files on this virtual disk include copies of the ALPHA text files and the COMM1 and COMM2 scratch files. Package execution time is minimized by accessing these frequently-used files from the virtual disk rather than a physical one. A memory division map of the computer package is given in Table 1.

4.2 Operation Sequence

A sample operation sequence is described below:

1-When the microcomputer is switched on, with the first diskette in drive A and the second in B, IBM DOS automatically searches for and executes AUTOEXEC. It performs a number of tasks to initialize the system and copies

FLOPPY 1	FLOPPY 2	RAM & ROM
DRIVE A	DRIVE B	
SYSTEM FILES	STORE1	COMMAND.COM
BASRUN.EXE	STORE2	BASRUN.EXE
SUPERSPL.COM	STOREF	SUPERSPL BUFFER
SUPERDRV.COM	COMMA	SUPERDRV BUFFER
AUTOEXEC.BAT	COMMB	COMM1
START.EXE	COMMC	COMM2
MAIN.EXE	GRAPHICS	ALPHA
MAINA.EXE[a]	MACLOGO.SCR	ALPHAA
MAINB.EXE	TITLE.SCR	ALPHAB
MAINC.EXE	RETORT.SCR	ALPHAC
MAIND.EXE	CLOCK.SCR	ALPHAD
MAINE.EXE	SANDCL.SCR	ALPHAE
SESSREST.EXE	FIRE.SCN	ALPHA1
SERVINT.EXE	ALPHAD	ALPHA2
HARDOUT.EXE		ALPHA3
SCREENDP.EXE		ALPHA4
OUTPUT.EXE		ALPHA5
COPYSTFI.EXE		ALPHA6
BALLST.EXE		
POUCH1.EXE		
POUCH2.EXE		
CAN1.EXE		
CAN2.EXE		
ALPHA		
ALPHAA[a]		
ALPHAB		
ALPHAC		
ALPHAE		
ALPHA1		
ALPHA2[a]		
ALPHA3[a]		
ALPHA4[a]		
ALPHA5[a]		
ALPHA6[a]		

a - Programs to be added upon system expansion

Table 1 - System Memory Division Map

the ALPHA text files from the diskette in drive A onto the virtual disk. AUTOEXEC next instructs DOS to load and execute the START program.

2-START creates the STORE1 file, inputs a numeric system code to it, displays header information and then chains to the interactive interface MAIN.

3-When activated, MAIN uses the code in STORE1 to determine from which level control was transferred. Depending upon the information read from STORE1, MAIN will present the user with various options. Responses are also elicited from the user at this time. Data are stored in the files STORE2 and COMMA, created by MAIN, to be passed to the desired subMAINs. Before chaining to a subMAIN, MAIN writes an operating code to STORE1. This information is later made available to it again when it is reactivated, upon completion of the desired subMAIN.

4-The subMAIN chosen opens STORE2 and reads the operating code contained therein to determine from which level control was transferred to it. If the code indicates control came from MAIN, the subMAIN obtains any other relevant information from COMMA. The subMAIN then asks for the argument values needed to run the desired SUB, and stores them in COMM1. Values and operating information are written into STORE2, so that when the subMAIN is reactivated (after execution of the SUB routine) it will be able to "remember" its status before it chained to the SUB. Control is next transferred to the SUB.

5-The SUB routine executes, using the values present in COMM1 as its arguments, and writes the temperature results into COMM2. The SUB then chains

back to the calling subMAIN.

6-With the aid of the operating information in STORE2 ie. in this instance the operating code will indicate that control did not come from MAIN, the subMAIN can determine that a SUB has just been executed. It then reads the SUB results from COMM2 and using the information in STORE2, re-establishes its operating status which existed before the execution of the SUB. It can then resume its own execution, which may include doing lethality and nutrient retention calculations and process optimization. The subMAIN may also store information in COMMB to be passed up to MAIN.

7-Control is eventually tranferred back to MAIN, which determines its subsequent actions from user input as well as the information it had previously written in STORE1.

Throughout the execution of the package, results of the calculations can be stored in file COMMC. This file will be used by the service program interface SERVINT to be developed in the future, and accessible from MAIN, by which a paper copy (HARDOUT), a screen display (SCREENDP) or a graphical display (OUTPUT) of the results can be obtained. The fourth service program COPYSTFI will enable intentional system interruption. It will copy the files COMM1 and COMM2, resident on the volatile disk, into STOREF on the physical diskette in drive B. Thus, when the session is interrupted the file contents are not lost. The session restart program SESSREST, also accessible from MAIN, will regenerate the COMM1 and COMM2 files by retrieving the information from STOREF. It will then chain back to MAIN and the session will resume as

before.

4.3 Detailed Module Description

A detailed description of the package components is given in this section; the schematic shown in Figure 9 will be of use here.

4.3.1 Initialization Programs

When the microcomputer is switched on, with both package diskettes installed, the IBM Disk Operating System (DOS) searches for and automatically executes the program AUTOEXEC. This program, containing DOS commands, performs several initialization tasks, including setting up the virtual disk drive C in RAM, allocating part of the memory as a buffer for the printer spooling facility, loading the IBM Graphics program, and copying the ALPHA text files onto drive C. AUTOEXEC then transfers control to the other initialization program START.

START is a BASIC program. It first creates the file STORE1, inputs a numeric operating code to it, and then closes the file. This information will indicate to the next program in the hierarchy that control has been transferred from START. START then displays graphical title screens, and after a suitable length of time, chains to the interface MAIN.

4.3.2 Interface Modules

The interface modules include MAIN and the four subMAINs. These programs are the means by which the user and the package communicate. They display menus and options, and accept input from the user. Extensive error-checking is performed throughout the interaction; this is the lowest level in the

hierarchy where error-checking is performed. Data is examined for validity, and the user is given the opportunity to correct any unreasonable information.

The interfaces need the user-supplied data to direct the operation of the package, to activate the temperature-calculating routines, and to perform process calculations and optimizations. MAIN is primarily a task definition program, allowing the user to direct the operation of the computer package. The subMAINs perform two functions during package execution. Firstly, they act as the interfaces between the user and the temperature- calculating routines, the SUBs. They ask the user for the argument values necessary to run the appropriate SUB, perform extensive error-checking of these supplied values, and control the flow of this information to the SUBs. Secondly, the subMAINs calculate process lethality and nutrient retention, and present the overall results in both tables and graphs. The subMAINs are based on a general structure, illustrated in Figure 10. Future subMAINs may be designed according to this structure as well, whether they are used to assess batch or continuous processes, for conduction- or convection- heating foods.

4.3.2.1 MAIN

MAIN is the interactive interface which allows the user to select a program task eg., the user may wish to determine the center temperature in a cylindrical container after a certain heating period, or may want to examine the effect of lowering the retort temperature on nutrient retention in a conduction-heating food. MAIN writes operating codes and other control information in storage files. By means of these files, MAIN can transfer information down to the subMAINs as well as receive data from them, and can

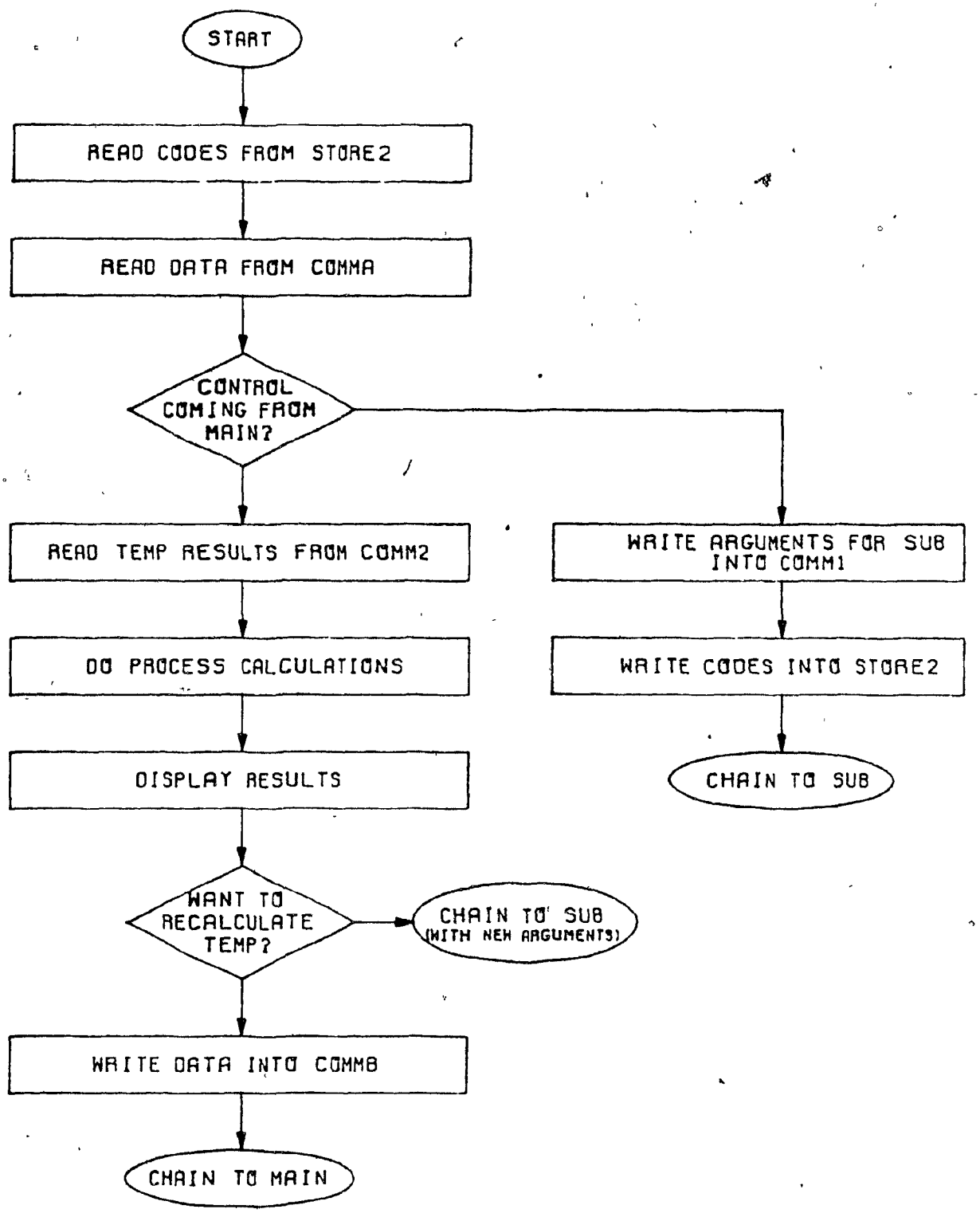


Figure 10 - General Flowchart of the SubMAINs

also retain information it may need upon later reactivation. According to directions received from the user, MAIN selects and activates the appropriate program module. At present the choices are limited to the five subMAINs. However, the system is designed to allow the future inclusion of the service programs interface and the session restart facility, both of which would be activated by MAIN.

4.3.2.2 MAINB

MAINB is selected by MAIN when the user wants to employ the slowest heating region approach (Ball's method) to calculate the lethality and nutrient retention for food in a cylindrical container. MAINB obtains information from the user, including the argument values needed to run SUB1, as well as the organism and nutrient destruction parameters required to calculate the lethality and nutrient retention. MAINB checks the data, stores the argument values in COMM1 and activates SUB1. It then retrieves from COMM2 the array of temperature-time results. Using these center temperatures, MAINB calculates the accumulated lethality and nutrient retention values, at the end of each time increment. It will then either repeat the calculations with different user-supplied values, or will return control to MAIN.

The lethal rate for the microorganism is calculated, during each time increment, by:

```
RATEL  = 1.0 / FLETH * 10.0**((TAVG-TREF)/ZLETH)          (114)
```

where TAVG is the arithmetic average of the can center temperatures at the

beginning and the end of the time increment, FLETH is the F-value of the microorganism at the reference temperature TREF (usually 121° C) and ZLETH is its z-value. The lethal rate is used to directly calculate the lethality imparted by the process at the end of each time increment, of duration DTIME:

```
XLETH  = XLETHST  +  RATEL * DTIME                    (115)
```

where XLETHST is the lethality already accumulated at the start of the time increment. The lethality calculated at the end of the final time increment is the total accumulated process lethality.

The fraction of nutrient retained in the food after processing is determined somewhat differently. It is calculated using a probabilistic approach; the amount of nutrient left at the end of a time increment is compared to that present at the start of processing, an amount taken to be equal to 1.0.

```
RATEN  = 1.0 / FNUT * 10.0**((TAVG-TREF)/ZNUT)        (116)
```

where FNUT is the F-value of the nutrient at the reference temperature and ZNUT is its z-value. The amount of nutrient left at the end of the time increment is:

```
XNUTOT = XNUTST * 10.0**(-12.0*RATEN*DTIME)           (117)
```

where XNUTST represents the nutrient amount present at the beginning of the

time increment. The fraction of nutrient retained at the end of a time increment, compared to the amount present in the food originally, is:

FRANUT = XNUTOT / 1.0 (118)

Once these lethality and nutrient calculations are completed, MAINB presents the results to the user.

4.3.2.3 MAINC

This interface is used to calculate the temperature, lethality and nutrient retention for food processed in a retort pouch. These values are treated as functions of both time and position in the container. The retort pouch is modeled as a rectangular brick, and is considered to be subdivided into elemental volumes as shown in Figure 6. In the calculations only one-eighth of the pouch needs to be considered, because of its three-dimensional symmetry.

MAINC activates the appropriate SUB which, using the argument values obtained from the user by the subMAIN, calculates the temperatures at the spatial nodes, at every time value. As can be seen in Figure 6, the nodes for the pouch are the points located at the eight corners of each element. The SUB writes the array of temperature results into COMM2 and then returns control to the subMAIN. Subsequently, for each time value MAINC determines the temperature at the center of each element by arithmetically averaging the nodal temperatures. Using these calculated average temperatures, together with destruction and kinetic parameters, MAINC then determines the lethality

and the nutrient amounts (by weight) in each element, at the end of every time increment. The total lethality and nutrient retention fractions are then found by summing the quantities in all elements. Thus, the accumulated container lethality and nutrient retention values are available at every incremental time value.

To determine the lethality imparted, a contaminant microbe concentration is postulated to be initially present in the food; the program keeps a record of the number of survivors remaining as the process continues. The processing time is divided into a number of increments and the lethality and nutrient retention results are calculated at the end of each one. The initial number of microorganisms in the food can be determined from the concentration:

$$ORGO = (1.0/8.0 * VTOTP) * RHO * CORG \quad (119)$$

where CORG is the postulated organism concentration at the start of processing, RHO is the density of the food and VTOTP is the total volume of the pouch. The number of elements the pouch is subdivided into is:

$$NELE = (NX-1) * (NY-1) * (NZ-1) \quad (120)$$

where NX, NY, and NZ are the number of nodes (including boundary values) in the X, Y, and Z directions respectively. The volume of each element can be calculated by:

$$VELE = (HALFX/(NX-1)) * (HALFY/(NY-1)) * (HALFZ/(NZ-1)) \quad (121)$$

where HALFX, HALFY and HALFZ correspond to the pouch half-lengths in the X, Y, and Z directions. The initial number of microorganisms present in each element before the start of thermal processing is:

```
ORGE0 = VELE * RHO * CORG                                    (122)
```

At each time value, the temperature throughout each element is calculated. For the pouch, this temperature TCNT is found by taking the arithmetic average of the eight nodal temperatures of each of the rectangularly-shaped elements. The average temperature, TAVG, during the time increment is found for each element by taking the mean of the TCNT values at the beginning and end of the time increment. This temperature is then used to find the lethal rate for the microorganism during the time increment, for each element:

```
RATEL.= 1.0 / FLETH * 10.0**((TAVG-TREF)/ZLETH)              (123)
```

Using the lethal rate, the number of organisms surviving in the element at the end of the time increment, of duration DTIME, is calculated by:

```
ORGELE = ORGEST * 10.0**(-12.0*RATEL*DTIME)                  (124)
```

where ORGEST is the number of organisms present in the element at the beginning of that increment. The total number of surviving organisms at the end of a time increment is the sum of the organisms present in all elements:

```
ORGTOT = SIGMA((...M=1 to NELE (ORGELE)...))                 (125)
```

where SIGMA((...M=1 to NELE (VARIABLE)...)) represents the summation operator commonly used in mathematics. The lethality at the end of the time increment is then calculated:

```
XLETH = ALOG10(ORGO/ORGTOT) / 12.0                    (126)
```

Thus, the lethality value calculated at the end of the final time increment represents the accumulated lethality imparted by the process.

The nutrient retention in the food is calculated using a similar approach. The original amount (by weight) of nutrient present in the food prior to processing is determined by:

```
XNUTO = (1.0/8.0 * VTOTP) * RHO * CNUT                (127)
```

where CNUT is the initial concentration of the nutrient in the food. The original amount of nutrient present in each element is:

```
XNUTEO = VELE * RHO * CNUT                            (128)
```

During each time increment, the degradation rate in the element is calculated by:

```
RATEN = 1.0 / FNUT * 10.0**((TAVG-TREF)/ZNUT)         (129)
```

The amount of nutrient left in each element, at the end of the time increment, is found by:

```
XNUTE = XNUTEST * 10.0**(-12.0*RATEN*DTIME)                    (130)
```

where XNUTEST is the amount of nutrient present in the element at the beginning of the time increment. The total amount of nutrient remaining in the food at the end of the time increment is found by summing the amounts in each element:

```
XNUTOT = SIGMA((...M=1 to NELE (XNUTE)...))                    (131)
```

The fraction of nutrient retained at the end of the time increment is:

```
FRANUT = XNUTOT / XNUTO                                        (132)
```

The nutrient fraction calculated at the end of the final time increment represents the fraction of nutrient retained after processing.

4.3.2.4 MAIND

MAIND is used to calculate the temperature, lethality and nutrient retention values, as a function of time and position in the container, for food processed in a cylindrical container. The can is considered to be subdivided into layers of concentric annuli, whose volumes depend on their position in the can, as shown in Figure 7. Calculations are performed for only one-half

of the can because of its inherent symmetry.

MAIND activates the appropriate SUB, which calculates the nodal temperatures, at every time value. The nodes of the can are the radial boundaries of the annular elements, as well as the center point of the can. The SUB writes the temperature results into COMM2 and transfers control back to MAIND. For each time value, MAIND calculates the mean temperature present throughout an annular element, TCNT, by averaging the four radial temperatures of the annulus. The average temperature TAVG, representing the average temperature in the element during the time increment, is calculated by taking the mean of the TCNT temperatures at the start and the end of time increment under consideration.

With these average elemental temperatures, and with destruction and kinetic parameters, MAIND determines lethality and nutrient retention fractions using algorithms similar to those of MAINC. Again the lethality and nutrient amounts are calculated in each element, at the end of every time increment. The total lethality and nutrient retention fractions are then found by summing the amounts in all elements, at the end of every time increment. Thus, the accumulated container lethality and nutrient retention fractions are available at the end of the final time increment.

The nutrient and lethality equations for the can are the same as those for the pouch, with the following exceptions:

The initial organism population is:

$$ORGO = (0.5 * VTOTC) * RHO * CORG \quad (133)$$

The number of elements in the can is:

$$NELE = (NR-1) * (NZ-1) \quad (134)$$

where NR is the number of radial nodes. The volume of each element in the can is calculated by:

$$VELE = PI * (HALFL/(NZ-1)) * (ROUTER**2 - RINNER**2) \quad (135)$$

where HALFL is the half-length of the can in the Z direction, ROUTER is the outer radius of the element and RINNER is the inner radius. The temperature throughout each element TCNT is calculated for the can by taking the mean of the four nodal temperatures. The initial amount of nutrient present is calculated by:

$$XNUTO = (0.5 * VTOTC) * RHO * CNUT \quad (136)$$

All other equations are the same as those for the pouch. Upon completion of the lethality and nutrient retention calculations, MAIND displays the results to the user.

4.3.2.5 MAINE

MAINE is the subMAIN interface selected by MAIN to determine the process temperature at which maximal nutrient retention will result during the batch processing of conduction-heating food in a can. The lethality and nutrient

retention calculations are based on the slowest heating region method wherein only the temperature at the centre of the can is considered. As with the other subMAINs, the user is instructed to enter argument values to run the temperature-calculating SUB program, in this case SUB1. MAINE differs from the previous subMAINs discussed in that an objective function is specified. It is this function, representing a combination of nutrient retention values, that is maximized. One of the main process constraints is the lethality requirement, which must be concurrently satisfied.

In brief, MAINE executes in the following manner. Process parameters needed to run SUB1 (BALLST) are supplied by the user. These parameters include the total process time and the target process lethality, but not the time at which cooling starts nor the retort (process) temperature. MAINE sequentially chooses process temperatures, from 110 to 130 ^{o}C at 2 degree intervals. The following verification is done to ensure that all processes used in further calculations will result in commercial sterility. If at the lowest temperature (110 ^{o}C) the total processing time is sufficient to result in a lethality equal to or greater than the target lethality supplied by the user, then all other processes at higher temperatures will be adequate. Otherwise, the total processing time is augmented to an extent that at 110 ^{o}C the process lethality will be at least equal to the target lethality when the total process time and the cooling start time coincide. Once the total time is verified in the above manner the cooling start time is calculated, for each temperature, which will result in the lethality requested by the user. This is done iteratively by looping between SUB1 (to calculate the temperature history at the centre of the can using the cooling start time) and MAINE (to

determine the lethality imparted to the contaminant microbes) until the earliest time at which cooling may start is found. The nutrient retention fractions are then calculated at each temperature, from 110 to 130 °C at 2 degree intervals, and are used to determine the values of the objective function. Once these eleven objective function values are calculated, they are used in a curve-fitting routine to determine the equation which relates the objective function value to the process temperature. The maximum value of the objective function is then calculated by means of this equation, for process temperatures between 110 and 130 °C at 0.05 degree intervals. When the process temperature which yields the highest objective function value is determined, the corresponding cooling start time which will satisfy lethality requirements is calculated. The optimal process temperature and times are thus known at this point. A partial flowchart of MAINE is presented in Figure 11. A more detailed description of the functioning of MAINE follows.

The value of the objective function is calculated from the dot product of two vectors, each consisting of up to four values. The first vector contains the nutrient retention fractions, at the end of the process. The second is comprised of the weighting factors assigned to the nutrients. The form of the objective function is shown in Equation 137:

$$OBJF = (W_1*R_1 + W_2*R_2 + W_3*R_3 + W_4*R_4) / (W_1 + W_2 + W_3 + W_4) \qquad (137)$$

where R_n (n=1 to 4) represents the nutrient retention fractions, and W_n the corresponding weighting factor. Both R_n and W_n can vary from 0 to 1. The value of the objective function can range from 0 (no nutrient retention) to

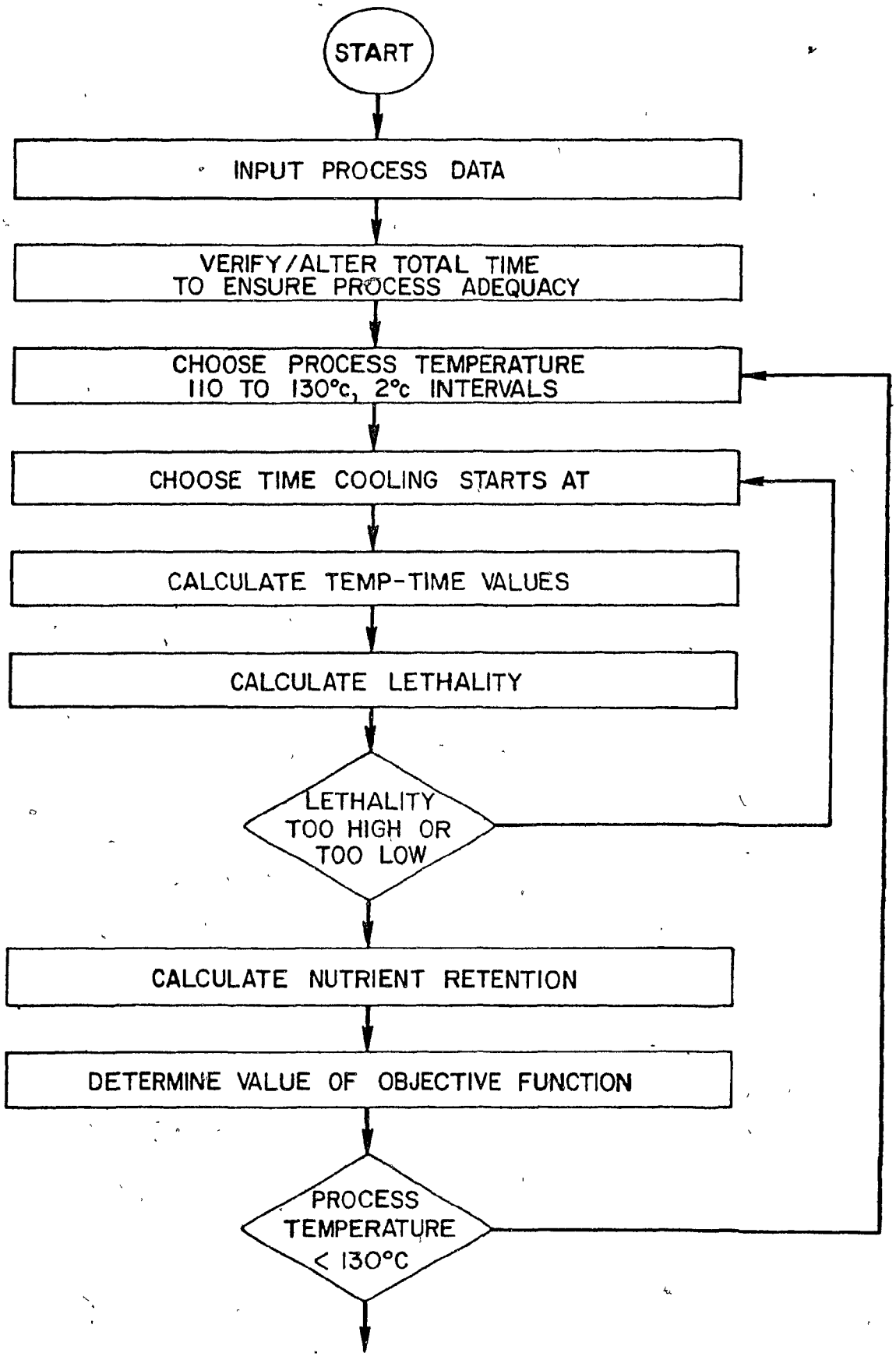


Figure 11 - Flowchart of MAINE

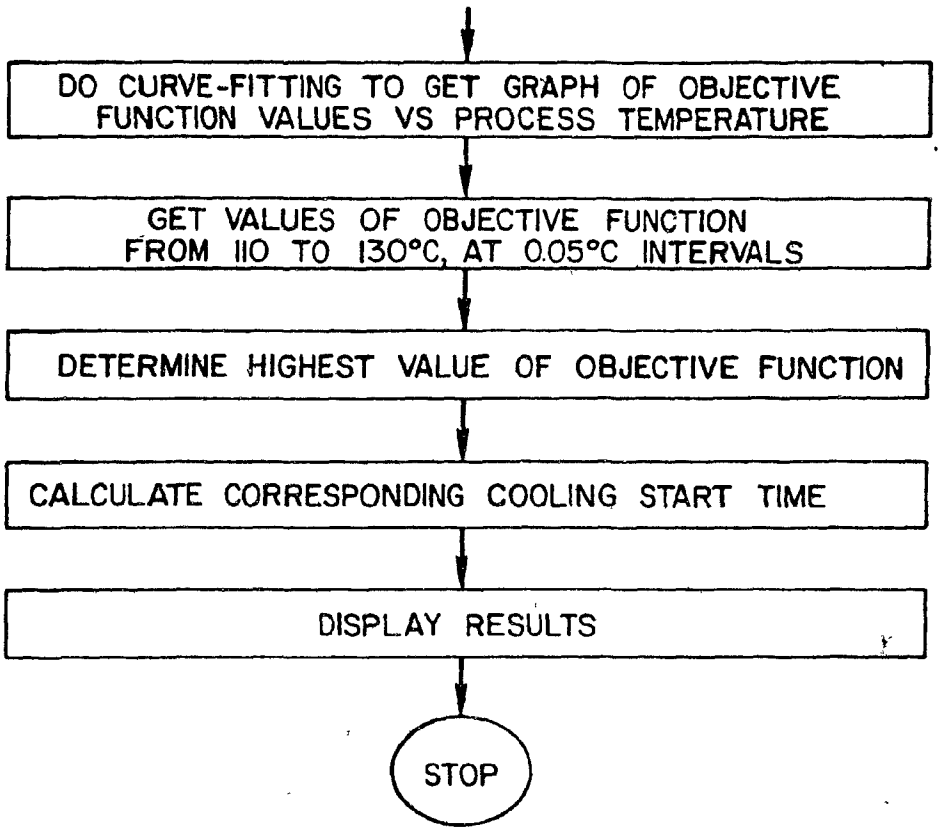


Figure 11 (Continued) - Flowchart of MAINE

1 (total retention).

First the values of the objective function are calculated for the eleven temperatures between 110 and 130 °C, at 2 degree intervals. This temperature range was selected since the optimal nutrient and sensory quality results usually occur near the midpoint temperature. For example, optimal thiamine retention has been reported at 117 to 119 °C by Teixeira et al.(1969b) and at 121.1 °C in sweet potato puree by Rizvi and Acton (1982), while the optimal process temperature is between 117 to 120 °C when surface browning is the optimization parameter used (Ohlsson, 1980).

The eleven objective function values and associated process temperatures are used to determine the equation relating these two variables. The equation is approximated by a third order polynomial. The curve-fitting routine is based on the matrix approach and utilizes the Gauss-Jordan elimination method. This section of the MAINE module was adapted from standard programs available in Miller (1981), which perform the Gauss-Jordan elimination technique of matrix inversion and the polynomial approximations.

Once the equation relating the objective function values to the process temperatures is determined, it is used to calculate the 401 values for the temperatures from 110 to 130 °C at 0.05 degree intervals. The values are taken at 0.05 degree increments since most thermocouples can typically measure up to an accuracy of 1.0 degree when uncalibrated and 0.1 degrees when calibrated (Benedict, 1981). It is thus considered that this temperature range and resolution is adequate for physical application of the theoretical

calculations.

From the 401 objective function values, calculated at 110 to 130 ^{o}C in 0.05 degree increments, the maximum objective function value is determined; the corresponding process temperature is known. The only other processing parameter remaining to be calculated is the time at which cooling starts. This is determined iteratively using the target process lethality requirement. Thus the optimal process is known, in terms of processing temperature, and total and cooling start times.

When the curve-fitting cannot be done (i.e. when the coefficient matrix is singular or the inverse matrix is zero, during the Gauss-Jordan elimination routine) the optimal process is selected from among the first eleven objective function values calculated, at 110 to 130 ^{o}C in 2.0 degree increments.

4.3.3 Temperature-Calculating Modules

The SUB programs in the package perform the detailed temperature calculations. The SUBs presently included can be used to calculate temperatures in batch processed conduction-heating foods, in either cylindrical cans or in retort pouches. The argument values necessary for their execution are stored in COMM1 by the calling subMAIN. Upon activation, the SUB executes, and then stores its results in COMM2. These are written in the form of an array, whose dimensionality is dependent upon the SUB which produces it. For a set of time values, SUB1 calculates a one-dimensional array of corresponding center temperatures in a cylindrical container. SUB2 and SUB3 calculate temperature in a retort pouch as a function of position and time. For sets of time values

and X, Y and Z coordinates, they produce four-dimensional temperature arrays. Similarly, for sets of time values and R and Z coordinates, SUB4 and SUB5 calculate three-dimensional arrays of temperatures within a cylindrical container. Once the temperature array is calculated and stored in COMM2, the SUB transfers control back to the subMAIN which activated it.

4.3.3.1 SUB1 - BALLST

SUB1 calculates the center temperatures of a cylindrical can for a set of time values. It is considered that a temperature-time function consisting of a) a step increase, followed after a specified period by b) a step decrease is applied at the can surface. The underlying assumptions for model development are: the thermal diffusivity is isotropic and is independent of temperature, the heat transfer coefficient at the surface is infinite, and the food is initially at a uniform temperature. There are three models available in SUB1. In all three, Ball's (1923) approach is utilized, whereby the temperature-time history at the can centre is modeled as consisting of two parts: a heating curve and a cooling curve. In the first model, Ball's (1923) method is followed strictly in that the heating curve is logarithmic, and the cooling curve is a combination of a hyperbolic section followed by a logarithmic one. The second model deviates slightly from Ball's procedure; in it the heating curve, like the cooling curve, consists of a hyperbolic section followed by a logarithmic one. The third model is also based on Ball's method, but is more general. As in the second model, both the heating and cooling curves are treated as combinations of hyperbolic and logarithmic parts. The difference lies in the way in which the constants of the hyperbolic cooling section are obtained. In model two they are calculated using a constant JC

value (cooling curve lag factor) of 1.41. In model three, JC values supplied by the user are utilized. The equations describing the heating and cooling curves are presented below in 'Fortran'-type format.

The hyperbolic portion of the heating curve is calculated (for models two and three) using the following equations:

```
FNZ(p)  = (1 - .453461/p + .0514065/p**2) / (.453461/p**3 - 1/p**2)   (138)
S       = ALOG10(JH/0.657)                                            (139)
AH      = 0.343 * (S-0.22673) * (TRETRT-TINIT) / (0.45346-S)          (140)
BHSQ    = FNZ(S) * FH**2                                              (141)
TEMP    = TINIT + AH*(SQRT(1+(TIME)**2/BHSQ) -1)                      (142)
```

where JH is the heating curve lag factor, TRETRT is the retort temperature, TINIT is the initial temperature of the food, 1/FH is the slope of the heating curve, TIME is the time passed since the temperature step increase was applied at the surface, and TEMP is the temperature at the centre of the can. The switch-time at which the shape of the heating curve changes from hyperbolic to logarithmic (for models 2 and 3) is given by:

```
TSTARH = FH * S                                                       (143)
```

The logarithmic portion of the heating curve is calculated (for all models) using:

```
DUMMY1 = (TRETRT - TINIT) * JH                                        (144)
```

```
TEMP  = TRETRT - DUMMY1*10**(-TIME/FH)                    (145)
```

At TIME=COOLTI, the heating curve ends and the cooling curve starts. For all models, the maximum temperature at the centre of the can is considered to occur at this instant:

```
TCMAX = TRETRT - (TRETRT-TINIT) * JH * 10**(-COOLTI/FH)   (146)
```

The hyperbolic portion of the cooling curve is calculated by means of equations 147, 148 and 152 for models 1 and 2, and equations 149 to 152 for model 3: (JC = 1.41 for models 1 and 2, and is user-supplied for model 3)

```
AC    = 0.3 * (TCMAX-TWATER)                              (147)
BCSQ  = (0.173*FC)**2                                     (148)
Q     = ALOG10(JC/0.657)                                  (149)
AC    = 0.343 * (Q-0.22673) * (TCMAX-TWATER)/(0.45346-Q)  (150)
BCSQ  = FNZ(Q) * (FC**2)                                  (151)
TEMP  = TCMAX - AC*(SQRT(1+(TIME-COOLTI)**2/BCSQ) -1)     (152)
```

where TWATER is the temperature of the cooling water and 1/FC is the slope of the cooling curve. The switch time from hyperbolic to logarithmic cooling for all models is calculated with equation 153:

```
TSTARC = FC * Q                                           (153)
```

where Q is as specified in equation 149. The logarithmic portion of the

cooling curve for all models is given by equations 154 and 155:

$$DUMMY2 = (TCMAX - TWATER) * JC \quad (154)$$

$$TEMP = TWATER + DUMMY2*10**(-(TIME*COOLTI)/FC) \quad (155)$$

Examples of the heating and cooling curves generated by the various models of SUB1 are shown in Figure 1.

4.3.3.2 SUB2 - POUCH1

SUB2 calculates the temperature at any location in a retort pouch, for each of a set of time values, after it is exposed to a sudden, sustained temperature change at its surface. The pouch is modeled as a brick, and is assumed to initially be at a uniform temperature. The thermal diffusivity of the food is considered to be isotropic, and independent of temperature. The convective heat transfer coefficient at the surface is assumed to be infinite. SUB2 returns a four-dimensional temperature array whose contents correspond to X, Y, Z and TIME argument values. The following equations are used by SUB2 to calculate the temperature at a particular location in the pouch, at a specified time (Williamson and Adams, 1919):

$$TEMP = TBATH - (TBATH-TINIT)*(64/(PI**3))*UX*UY*UZ \quad (156)$$

$$DUMMY1 = (2 * N) - 1 \qquad \text{where } N=1,2,3,\ldots,NEND \quad (157)$$

$$DUMMY2 = -ALPHA * TIME \quad (158)$$

$$XF1 = COS(DUMMY1*PI*X/2/HALFX) \quad (159)$$

XF2 = XF1 / DUMMY1 / (-1**(N+1)) (160)

XF3 = (DUMMY1*PI/2/HALFX) **2 (161)

TERM = DUMMY2 * XF3 (162)

TERMN = XF2 * EXP(TERM) (163)

UX = SIGMA((...N=1 to NEND (TERMN)...)) (164)

where TBATH is the temperature to which the pouch is suddenly exposed, ALPHA is the thermal diffusivity, and X is the coordinate along the X-axis of the pouch at which the temperature is being determined. NEND is the end point of the summation and ideally is infinity. For the purpose of calculation, it is the point at which further summing results in only a small deviation, arbitrarily set. UY and UZ are calculated in a manner similar to UX, by means of equations 157 to 164, but with the following substitutions: HALFY or HALFZ for HALFX, Y or Z for X, and M or L for N. As illustrated in Figure 6, the X, Y, Z coordinate system is considered to have its origin at the center of the pouch.

4.3.3.3 SUB3 - POUCH2

SUB3 is similar to SUB2 in that it returns the same type of temperature array for a retort pouch, using similar input variables and initial assumptions. The difference between them is that SUB3 requires a user-supplied value for the convective surface heat transfer coefficient; the coefficient is not assumed to be infinite in this case. The equations used in SUB3 are based on those of Carslaw and Jaeger (1959), and are as follows:

TEMP = TBATH - (TBATH-TINIT)*8*UX*UY*UZ (165)

DUMMYX = H / (ALPHA*RHO*CP) (166)

BP = DUMMYX * HALFX (167)

BETA = the Nth positive root of: BETA * TAN(BETA) = BP (168)

DUMMY1 = SIN(BETA) (169)

DUMMY2 = -ALPHA * TIME (170)

BETXF1 = DUMMY1 / (BETA + DUMMY1*COS(BETA)) (171)

BETXF2 = BETA / HALFX (172)

BETXF3 = (BETXF2)**2 (173)

TERM = DUMMY2 * BETXF3 (174)

TERMN = BETXF1 * COS(BETXF2*X) * EXP(TERM) (175)

UX = SIGMA((...N=1 to NEND (TERMN)...)) (176)

where H is the convective surface heat transfer coefficient, RHO is the density of the food, CP is its specific heat and BP is the infinite plate Biot number in the X, Y or Z direction. UY and UZ are calculated similarly to UX with these substitutions: HALFY or HALFZ for HALFX, Y or Z for X, M or L for N, and (BETYF1, BETYF2, BETYF3) or (BETZF1, BETZF2, BETZF3) for (BETXF1, BETXF2, BETXF3). Again, the coordinate system has its origin at the centre of the pouch.

4.3.3.4 SUB4 - CAN1

SUB4 calculates the temperature at any position in a cylindrical can, after it is exposed to a sudden, sustained temperature change. The contents are

assumed to initially be at a uniform temperature, the convective surface heat transfer coefficient is treated as being infinite, and the thermal diffusivity is considered to be isotropic and independent of temperature. SUB4 returns a three-dimensional temperature array corresponding to R, Z and TIME argument values. The equations to calculate the temperature at a specific location in the can, for a particular time are (Luikov, 1968):

```
TEMP   = TBATH - (TBATH-TINIT)*4/PI*UZ*UR                          (177)

DUMMY1 = -ALPHA * TIME                                             (178)
DUMMY2 = (2*N) - 1       where N=1,2,3,...,NEND                    (179)
ZF1    = COS(DUMMY2*PI/2*Z/HALFL)                                  (180)
ZF2    = (-1**(N+1)) / DUMMY2                                      (181)
TERM   = ((DUMMY2*PI/2/HALFL)**2) * DUMMY1                         (182)
TERMN  = ZF2 * ZF1 * EXP(TERM)                                     (183)
UZ     = SIGMA((...N=1 to NEND (TERMN)...))                        (184)

XJ0(p) = Bessel function of the first kind of zero order           (185)
XJ1(p) = Bessel function of the first kind of first order          (186)
RF1    = 2 / ROOT / XJ1(ROOT)                                      (187)
RF2    = XJ0(ROOT*R/RADIUS)                                        (188)
TERM   = ((ROOT/RADIUS)**2) * DUMMY1                               (189)
TERMM  = RF1 * RF2 * EXP(TERM)                                     (190)
UR     = SIGMA((...N=1 to NEND (TERMM)...))                        (191)
```

where ROOT is the Nth positive root of the Bessel function of the first kind

of zero order, RADIUS is the physical radius of the can, HALFL is the half-length of the can, Z is the axial coordinate, and R is the radial coordinate. As shown in Figure 7, the origin of the R,Z coordinate system is considered to be at the geometrical centre of the can.

4.3.3.5 SUB5 – CAN2

SUB5 is similar to SUB4. It returns the same type of temperature array for a can, using similar input variables and assumptions. It differs from SUB4, however, in that it requires a user-supplied convective surface heat transfer coefficient. The equations of SUB5 are based on those of Carslaw and Jaeger (1959):

$$TEMP = TBATH - (TBATH-TINIT)*4*BC*UZ*UR \quad (192)$$

$$BC = H * RADIUS / ALPHA / RHO / CP \quad (193)$$

$$DUMMY1 = -ALPHA * TIME \quad (194)$$

$$DUMMY2 = SIN(BETA) \quad (195)$$

$$BETAF1 = DUMMY2 / (BETA+DUMMY2*COS(BETA)) \quad (196)$$

$$BETAF2 = BETA / HALFL \quad (197)$$

$$BETAF3 = (BETAF2) **2 \quad (198)$$

$$TERM = DUMMY1 * BETAF3 \quad (199)$$

$$TERMN = BETAF1 * COS(BETAF2*Z) * EXP(TERM) \quad (200)$$

$$UZ = SIGMA((...N=1 \text{ to } NEND\ (TERMN)...)) \quad (201)$$

$$GAMM = \text{Nth positive root of: } GAMM * XJ1(GAMM) = BC * XJ0(GAMM) \quad (202)$$

```
GAMMF1 = (BC**2 + GAMM**2) * XJO(GAMM)                          (203)
GAMMF2 = XJO(GAMM*R/RADIUS) / GAMMF1                            (204)
GAMMF3 = (GAMM/RADIUS) **2                                      (205)
TERM   = DUMMY1 * GAMMF3                                        (206)
TERMM  = GAMMF2 * EXP(TERM)                                     (207)
UR     = SIGMA((...N=1 to NEND (TERMM)...))                     (208)
```

where BP and BETA are found as in equations 167 and 168. BC is the infinite cylinder Biot number. The coordinate system is again assumed to have its origin at the centre of the can.

4.3.4 Associated Files

There are various types of temporary files within the package which ensure the smooth operation of the system. The major purpose of these files is to transfer information between the hierarchy levels, and so allow inter-module communication. This information will consist of system operation codes, user-supplied values, and intermediate results. Using a modular approach, together with these storage and communication files, allows the emulation of FORTRAN subroutine and program control features. Thus, two major disadvantages of utilizing compiled BASIC as the programming language are avoided; that is, the inability to have true subroutines with local variable names, and the inability to transfer control to a specified line in a target program during chaining. These associated files will also be the basis of the package's ability to restart an intentionally interrupted session. As well, the functioning of future report writing, graphics and other service programs will depend on them.

4.3.4.1 Storage Files

The STORE files are used to store operation codes and values generated by one level of the package, during the execution of a program in a lower level. For example, values generated by the system or entered by the user during the execution of MAIN are stored in STORE1, and are retained during the execution of a lower-level subMAIN program. When control is transferred back to MAIN, once the execution of the subMAIN program is completed, the values in STORE1 are recovered by MAIN. Thus, MAIN remembers the status it had before it chained to the lower-level program.

STORE2 is associated with the subMAIN programs in a similar manner. Values and codes are stored in STORE2 before control is transferred to a SUB program. Upon the execution of the SUB, control is retransferred to the subMAIN, the values associated with the subMAIN are recovered from STORE2, and execution of the system continues.

The file STOREF is a storage file that will be utilized by the session restart facility (to be added upon package expansion). When the user indicates that a system interrupt is desired, the volatile files COMM1 and COMM2 will be copied into STOREF on the diskette in drive B. This will allow the computer to be powered down without loss of data. The session will be restarted with the SESSREST program, the data in STOREF will be used to recreate the volatile files, and thus the system will be restored to its pre-interruption status.

4.3.4.2 Text Files

Associated with each of the interactive programs are text information files, the ALPHA files. All text needed by the interfaces are stored on the virtual disk C, in these random access files. Text lines are retrieved from the ALPHA file, when needed, by the interactive program and displayed on the screen. This approach to alphanumeric information greatly facilitates the creation of multilingual software packages. It also enables easier updating or alteration of the interactive interfaces. The ALPHA files and interfaces correspond in the following manner: MAIN-ALPHA, MAINB-ALPHAB, MAINC-ALPHAC, SESSREST-ALPHA1, etc. as can be seen in Figure 9.

4.3.4.3 Communication Files

The COMM files are used to transfer arguments, results and system operating data between the different program levels. COMM1 transmits arguments from the subMAIN programs to the calculating SUB modules. COMM2 returns the results from the calculating SUB program to the invoking subMAIN. Since COMM1 and COMM2 are in frequent use, they are stored on the virtual disk to minimize system operating time. COMMA and COMMB transmit any necessary supplementary information from MAIN to the subMAINs and from the subMAINs back to MAIN, respectively.

The other communication file COMMC performs a different function. It can be used to store the pertinent results from calculations performed throughout the system run, and thus acts as a record of the session. The information is stored in the file by the subMAIN interfaces. COMMC will be accessed by the service programs, to be developed in the future, and will be used to produce

hardcopy, screen and graphical displays of the session results.

4.3.5 Service Programs

The package was designed to allow for the addition of a service program interface SERVINT, accessed by MAIN. It will be a menu-driven interface from which the user can choose an application program. These programs are envisioned to include HARDOUT - which will produce a paper copy of the session results, SCREENDP - for screen display of the results, and OUTPUTGR - for graphical display. The final program in the SERVINT interface is COPYSTFI; it will copy the volatile data files into STOREF, and will be used in the functionning of the session restart facility SESSREST. All service programs will use the information stored in file COMMC in order to operate.

4.3.6 Session Restart Facility

Another program to be added in the future is the session restart program SESSREST, which will allow the user to restart the session from a cold start, after an intentional interruption. The program will retrieve the information from the STOREF file, which represents the volatile communication files existing before the interruption. SESSREST will use this information to recreate these volatile files. The program will then chain back to MAIN, and the session will resume as if it had not been interrupted.

V. RESULTS AND DISCUSSION

5.1 Package Design

The software package was designed to perform the heat transfer and process calculations necessary for the optimization of batch sterilization processes, in terms of the retention of nutritional components. The package is arranged in a hierarchical manner, comprising five levels of modules including: system initialization programs, user-friendly interactive interfaces and temperature-calculating routines. The modular system design was used to enable the programmer to circumvent the major limitations of the BASIC language, and to readily allow future expansion of the package. The primary advantage of this design is that the size limitation of 64 KB for a BASIC program does not restrict the size of the package. The modules are overlaid in volatile memory as they are needed and thus only the individual program modules are restricted to 64 KB - the package itself is over 200 KB.

The modules are activated (loaded into volatile memory) when needed through a "chaining" command. The use of the chaining command helps the package emulate many of the Fortran language subroutine capabilities with respect to local variable names. In BASIC subroutines variable names remain active throughout the execution of a program; the names are not local to the subroutine. Thus during the writing of BASIC programs, the programmer must take into account all variable names used. But in BASIC, when chaining to another program the variable names are not "carried" to the next program, and only those names within each program must be accounted for. In a package as large as the one under consideration, having local variable names greatly

simplifies the task of the programmer and helps eliminate many potential sources of programming errors.

Variable values needed in several programs are passed between the modules by means of data files. These data files perform two major functions. First, they can pass values between programs during package execution and in this way imitate the argument/parameter approach used in Fortran subroutines. Secondly, they can contain operating codes which direct the transfer of control within the program. Normally when a compiled program is chained to, the calling program cannot specify to which line within the next program control should be transferred. The operating codes stored in the data files can cause the target program to direct control within itself, where these codes are used to specify the appropriate line number.

As discussed above, the modular approach used in the system enables the package to emulate many of the Fortran-language capabilites of data and control transfer. The system was designed in this way for other reasons as well. The package can be easily expanded since the temperature-calculating and interface modules are independent of each other. The modules are based on a standard system interaction design which easily allows the creation and incorporation of additional program modules. Ideally the package can contain various calculating routines that are not related i.e. routines to calculate temperatures in conduction- and convection-heating foods, and that are accessed by their particular subMAIN. Thus, the expansion and calculation capabilities of the package should be nearly unlimited. However, the expansion of the package in its present implementation is limited by the

amount of volatile memory available (and of course, by the physical data storage devices). At present, the package stores the printer spooler buffer, all the ALPHA text files needed for the interfaces, and the data files COMM1 and COMM2 on the virtual disk. Every interface module added would require another ALPHA text file. It is evident that the available volatile memory restricts the number of interface modules which can be incorporated in the package. By simply modifying the DOS-level system initialization program AUTOEXEC so that drive C is assigned to a 5 MB hard disk rather than a virtual RAM disk this limitation would be eliminated. The hard disk access time is almost as short as for a virtual drive, an important consideration for frequently-accessed text and data files, and the hard disk has much larger memory available - thus increasing potential package expandability.

The expandability of the package was a major consideration during the system design. There are several features not yet fully utilized in the present system, the communication files COMMA and COMMB. These files are used to transfer information between MAIN and the subMAINs. They could become an important component in future package development.

5.2 Sample Package Executions

Several runs were performed during the testing phase of this project, and the output results are included. The process calculations were done for a contaminant population of Clostridium botulinum having an F-value of 2.45 minutes and a z-value of 10 ^{o}C. The food components whose retentions were optimized are: thiamine, anthocyanin, a general enzyme and a general vitamin. Thiamine was used for the calculations of MAINB, MAINC and MAIND (in which

only one nutrient is considered). All four components were used in MAINE, the optimization module. The D and z values for thiamine as reported in the literature vary: D_{121} = 154 minutes, z = 34 °C (Feliciotti and Esselen, 1957); D_{121} = 165.6 minutes, z = 25.6 °C (Teixeira et al., 1975a); D_{121} = 210 minutes, z = 26.7 °C (Mulley et al., 1975). Representative values of F_{121} = 1980 minutes (D_{121} = 165 minutes) and z = 27 °C were taken for thiamine. For anthocyanin values of 213.6 minutes for F_{121} and 45.4 °C for z were calculated from activation energy information given by Lund (1977). For the general enzyme, F_{121} = 66 minutes and z = 31.1 °C were taken from the ranges of 12 to 120 minutes for F_{121} and 6.7 to 55.6 °C for z as reported by Lund (1977). The F_{121} and z values for a general vitamin were taken to be 1200 minutes and 28 °C repectively, based on information in Lund (1977). Different process temperatures and times were used to generate the sample runs, but only a few are included in this section. A full printout of the results is included in Appendix 2.

5.2.1 MAINB

The slowest heating region approach was used in the MAINB process calculations. Food held in a 303 x 406 can was processed at 120 °C for 3600 seconds and then cooled for 1400 seconds. The initial food temperature was 40 °C and the cooling water was 60 °C. The fh and fc values were taken to be 2460 seconds each, as reported by Teixeira et al. (1975a) for a 303 x 406 can. The sample executions were done using the first two models of BALLST. In Figures 12, 13 and 14 the process calculation results based on model 1 of BALLST are shown. The temperature curve of Figure 15 is generated using model 2 of BALLST, and illustrates the differences in the initial part of the

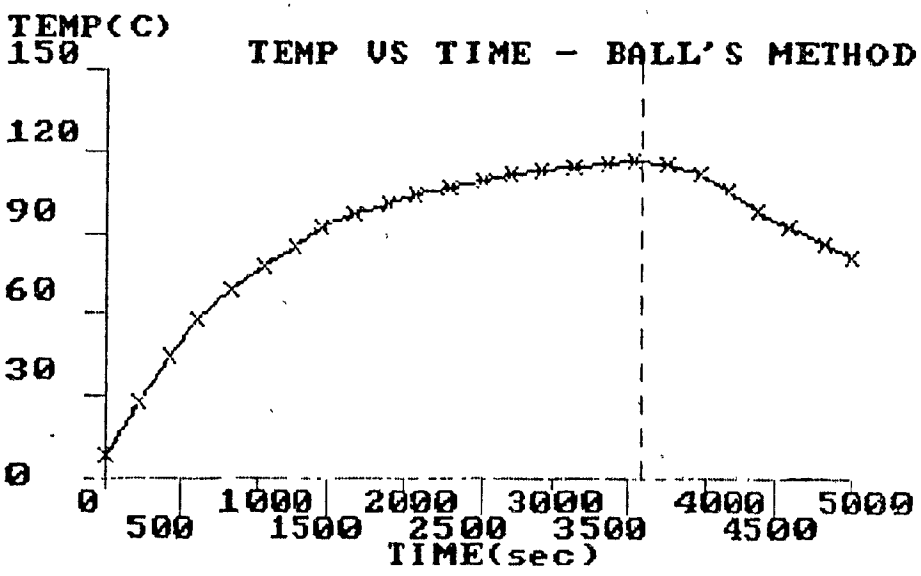


Figure 12 - Temperature-Time History - Model 1 (BALLST)

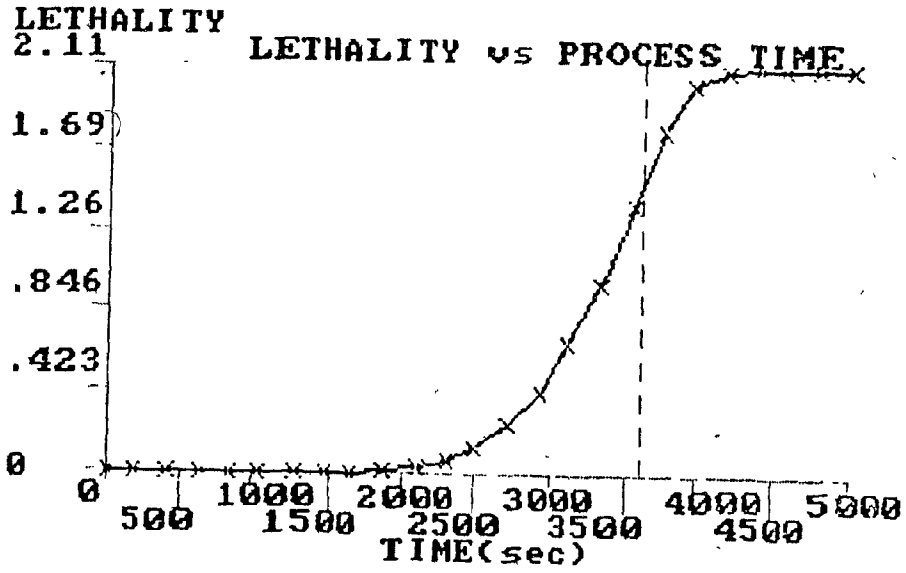


Figure 13 - Lethality vs Time - Model 1

(BALLST)

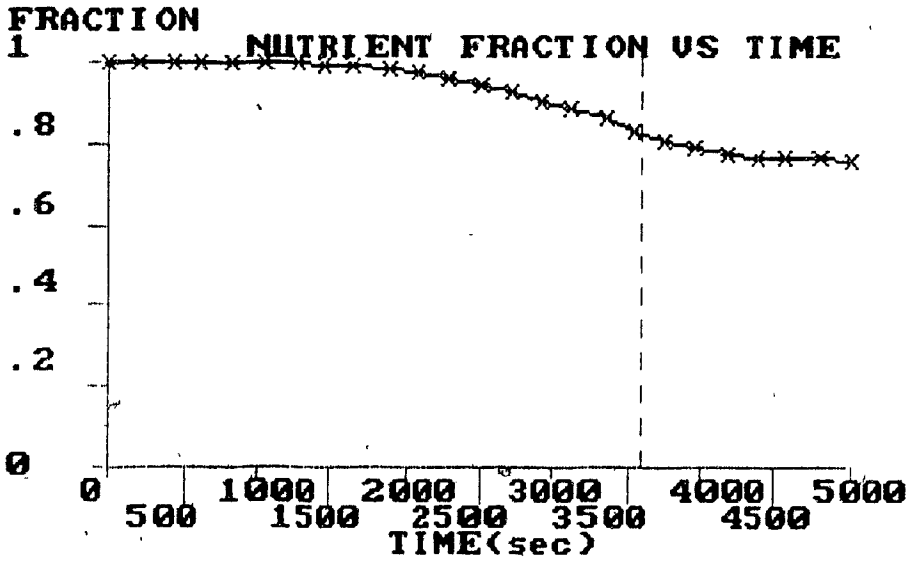


Figure 14 - Nutrient Retention Fraction vs Time - Model 1 (BALLST)

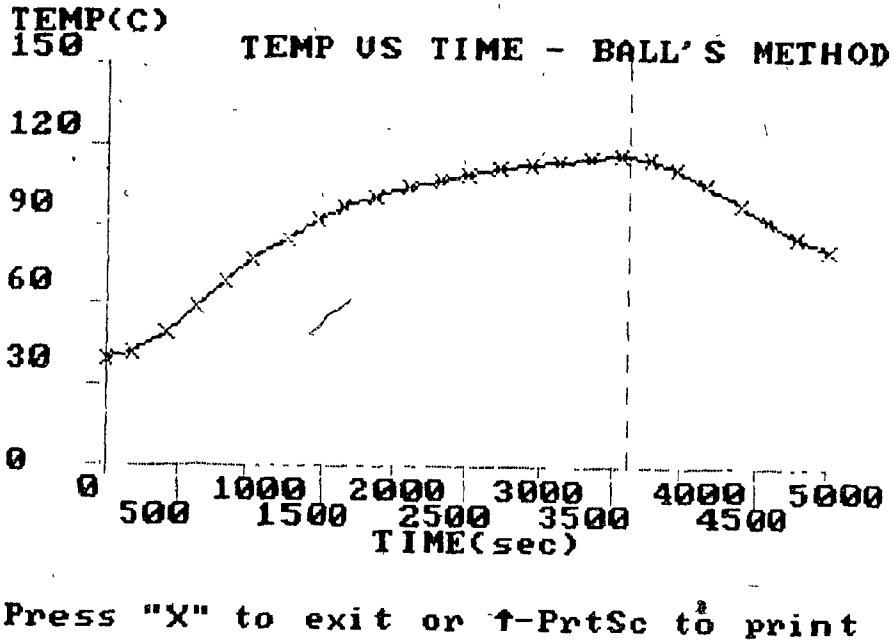


Figure 15 - Temperature-Time History - Model 2

(BALLST)

heating curve produced by models 1 and 2, due to their particular ways of calculating temperatures i.e. logarithmic heating curve (model 1) versus hyperbolic-logarithmic combination heating curve (model 2). Complete printouts are given in Appendix 2.1.

5.2.2 MAINC

The elemental approach was used to calculate temperature distributions as a function of time and position in a retort pouch. Both POUCH1, with the surface convective heat transfer coefficient assumed infinite, and POUCH2, with a specified h value, were used in the sample runs. The pouch was considered to be 0.130 m in length, 0.095 m in width and 0.015 m in thickness. It is initially at 40 ^{o}C, is processed in a retort at 120 ^{o}C for 3000 seconds and cooled with water at 60 ^{o}C for 1000 seconds. The food has a thermal diffusivity of 1.6 x 10^{-7} m^2/sec and a specific heat of 4000 kJ/kg-^{o}C. The h value is assumed equal to infinity for the run utilizing POUCH1, and 200 and 800 W/m^2-^{o}C for the two runs with POUCH2. Only part of the results are included here; the complete sample output can be seen in Appendix 2.2. Figures 16, 17 and 18 show the temperature histories at the centre, at 0.032 m, 0.023 m, 0.003 m in the X, Y and Z directions, and at the outer surface of the pouch - for h assumed infinite (POUCH1). Figures 19, 20 and 21 give the temperature distributions at the same locations but with an h value of 800 W/m^2-^{o}C. With this high h value there is not much difference in the temperature curves produced by POUCH1 and POUCH2. However, as can be seen in Figures 22, 23 and 24 (again at the same positions in the pouch) the effects of a lower h value (h=200 W/m^2-^{o}C) on the pouch temperatures become noticeable. This is especially evident in Figure 24, where the temperature

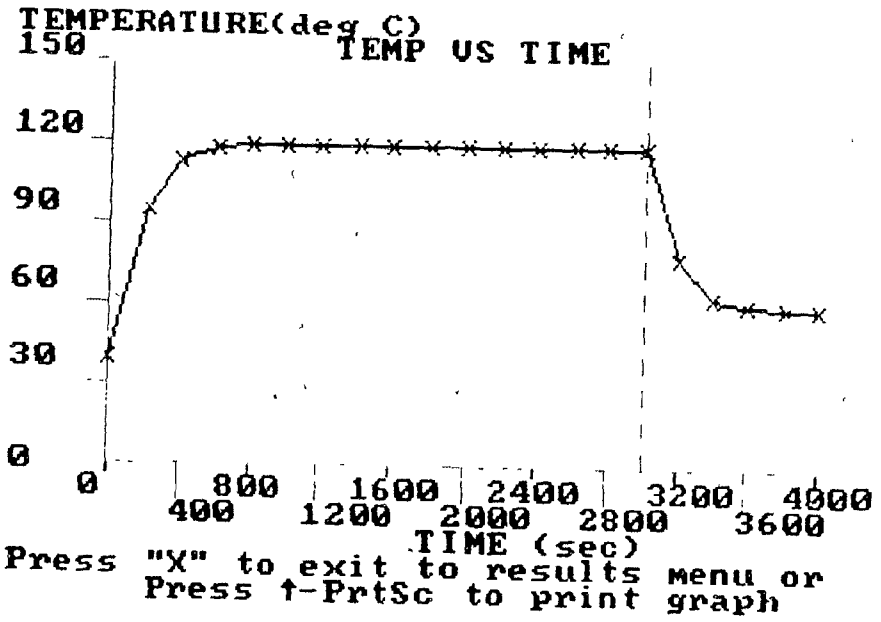


Figure 16 - Temperature History at the Centre of the Pouch
h = infinite (POUCH1)

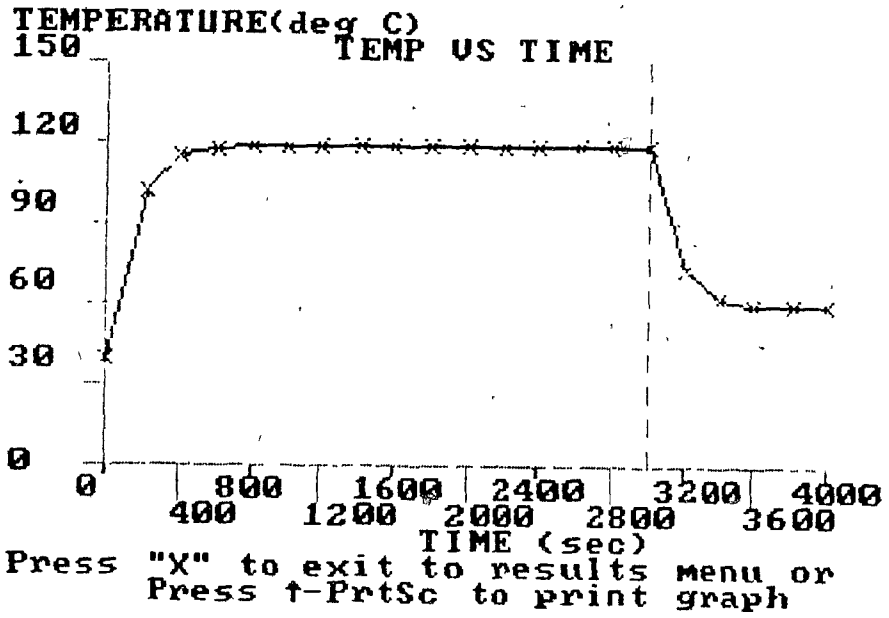


Figure 17 - Temperature History at (.032m,.023m,.003m)
h=infinite (POUCH1)

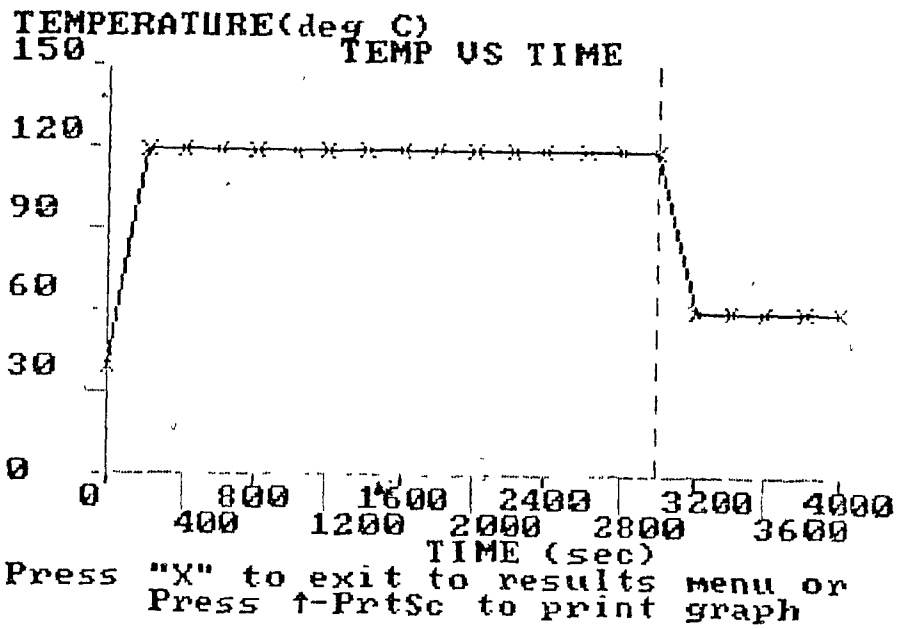


Figure 18 - Temperature History at the Outer Surface of Pouch h=infinite (POUCH1)

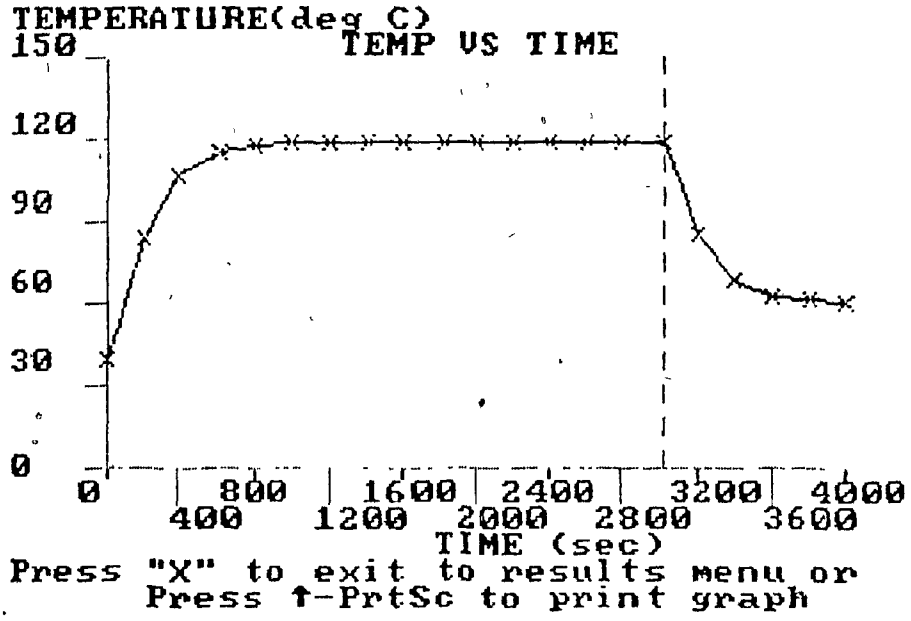


Figure 19 - Temperature History at the Centre of the Pouch
h=800 W/m^2-·C (POUCH2)

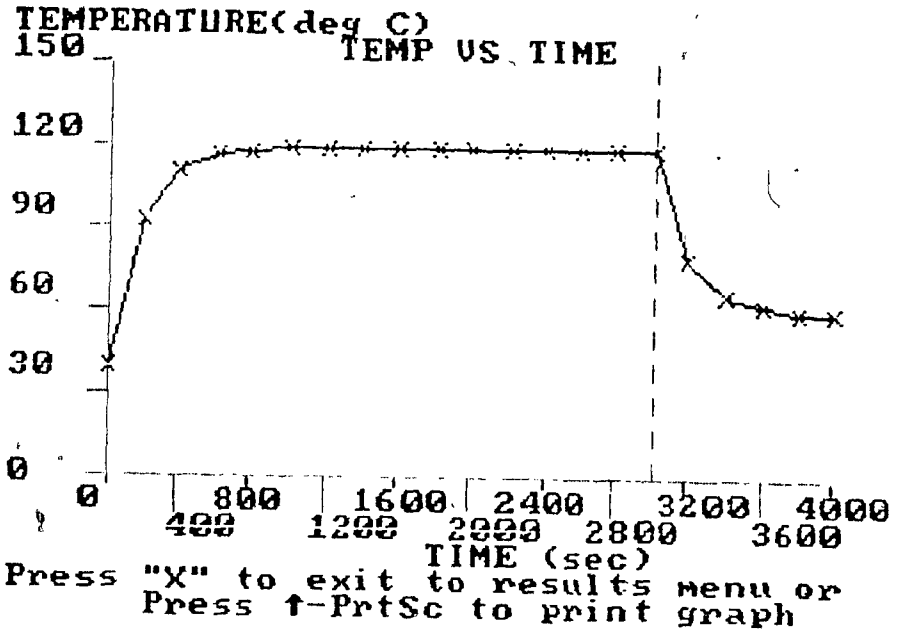


Figure 20 - Temperature History at (.032m,.023m,.003m) of Pouch h=800 W/m^2-°C (POUCH2)

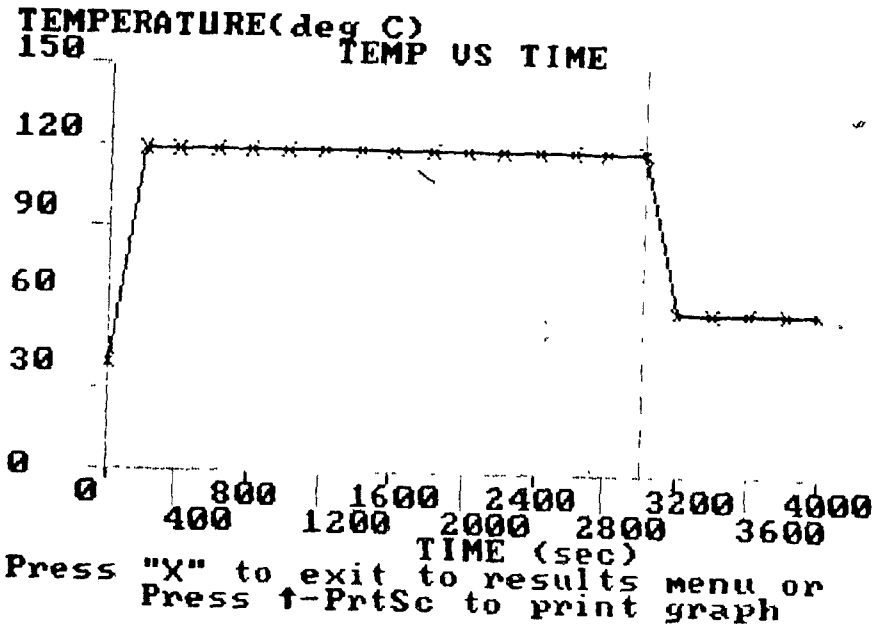


Figure 21 - Temperature History at the Outer Surface of Pouch
$h=800\ W/m^2\text{-}^\circ C$ (POUCH2)

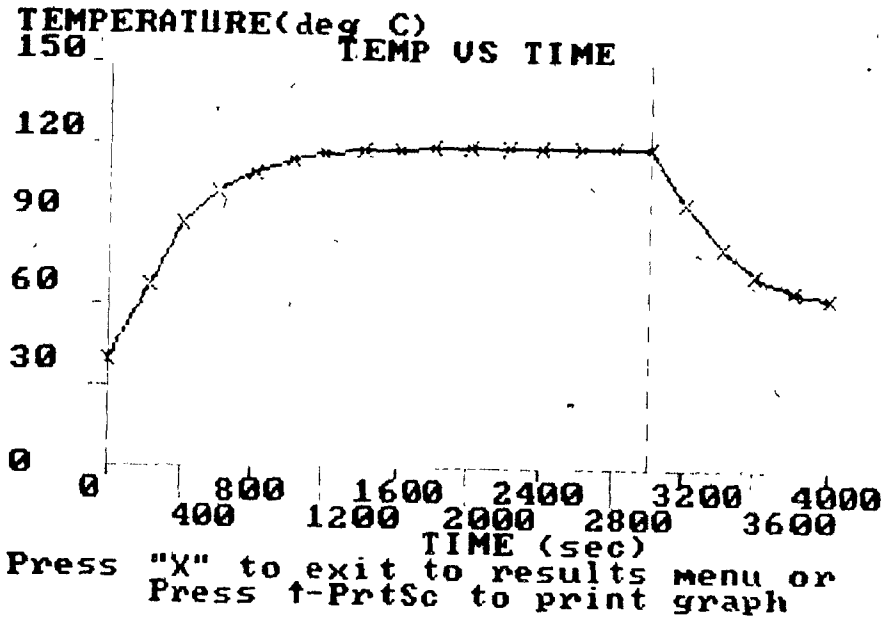


Figure 22 - Temperature History at the Centre of the Pouch
h=200 W/m^2-°C (POUCH2)

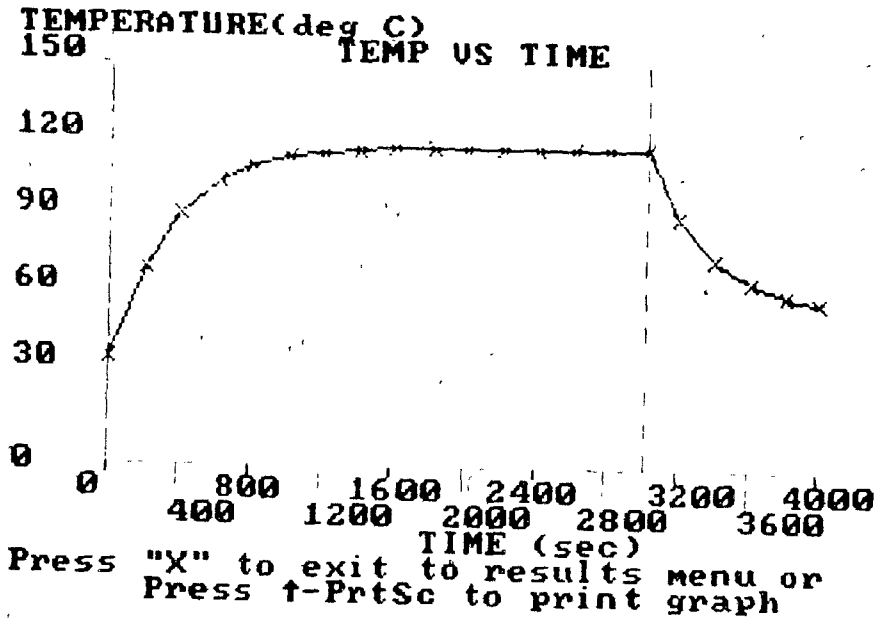


Figure 23 - Temperature History at (.032m,.023m,.003m) in Pouch
h=200 W/m^2-°C (POUCH2)

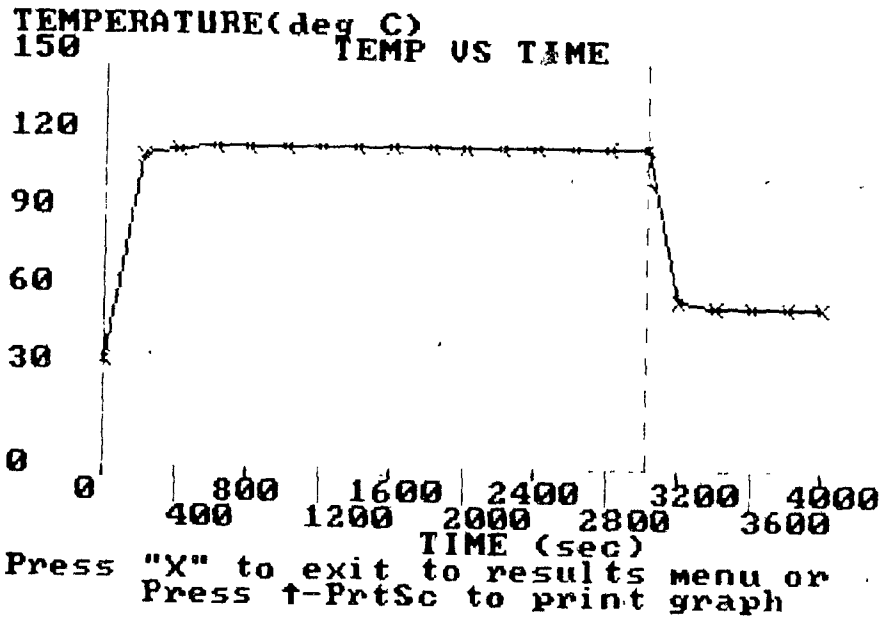


Figure 24 - Temperature History at Outer Surface of the Pouch
h=200 W/m^2-°C (POUCH2)

curve deviates slightly from the step input temperature function as a result of the surface convective heat transfer. The other results from the run, including the lethality and nutrient retention calculations, can be seen in Appendix 2.2.

5.2.3 MAIND

Calculations were performed using an elemental approach for food processed in a 303 x 406 can. The initial food, retort and cooling water temperatures were 40, 120 and 60 ^{o}C respectively. In Figure 25 is shown the temperature profile in the centre of the can according to CAN1 calculations, where the surface convective heat transfer coefficient h is assumed infinite. In Figure 26 the temperature curve at the centre of the can is depicted, but with h specified equal to 500 W/m^2-^{o}C (CAN2). The temperature curves do not indicate to a great extent the differences in the results yielded by the two calculation methods, but can be used in a comparison with the temperature curve generated by the slowest heating region approach from BALLST (Figures 12 and 15). The differences due to surface convective heat transfer are more apparent in Tables 2 and 3, the tables of accumulated lethality based on the temperature results of CAN1 and CAN2 respectively, where a final process lethality of 0.788 is calculated by CAN1 versus 0.529 by CAN2. The difference in calculation methods is also apparent for the calculation of the thiamine retention, where the retention fraction is 0.695 when based on the temperatures from CAN1 (Table 4) versus 0.736 when based on CAN2 results (Table 5).

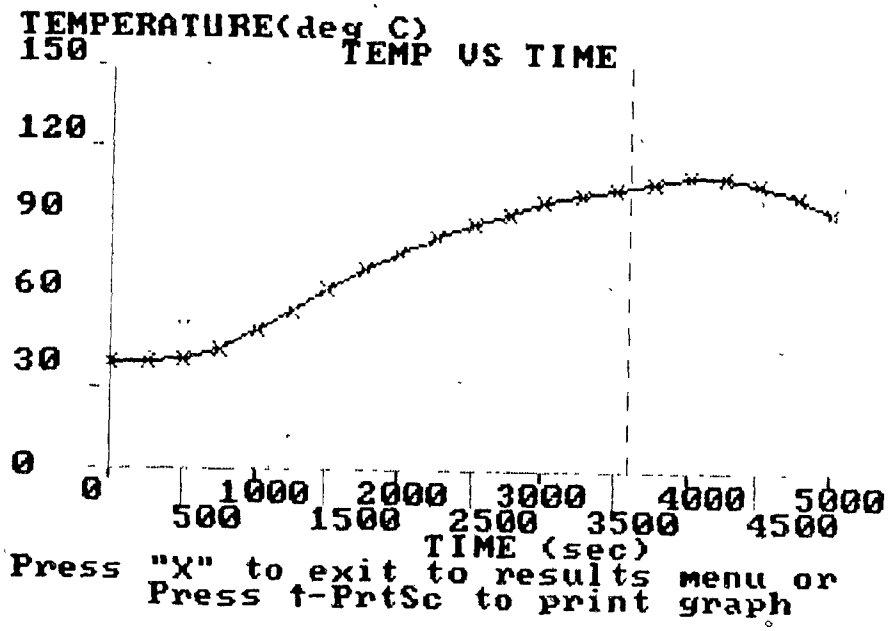


Figure 25 - Temperature History at the Centre of the Can h=infinite (CAN1)

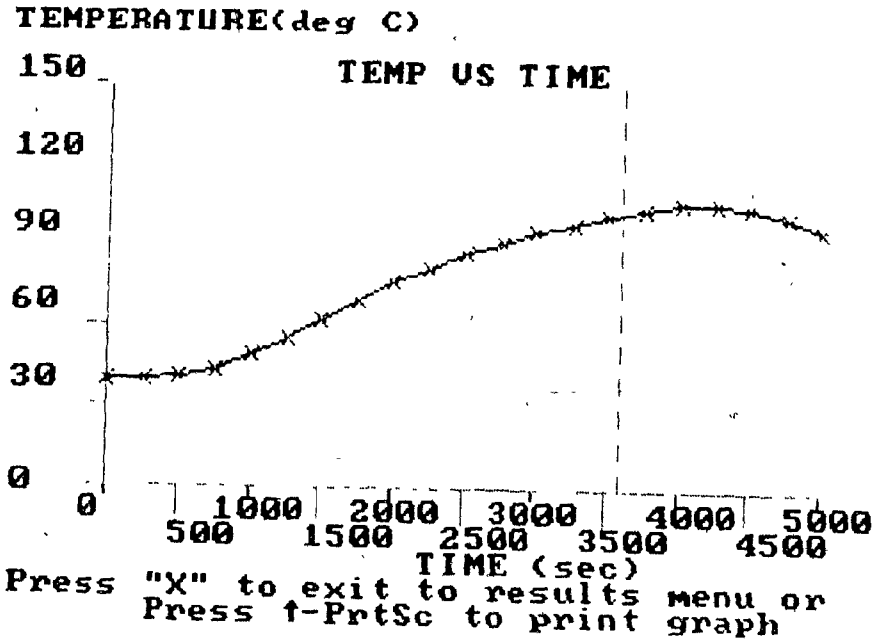


Figure 26 - Temperature History at the Centre of the Can
$h = 500\ W/m^2 \cdot °C$ (CAN2)

RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
250.00	77.304E-07
500.00	97.449E-04
750.00	16.714E-03
1000.00	25.477E-03
1250.00	33.051E-03
1500.00	39.490E-03
1750.00	47.297E-03
2000.00	57.924E-03
2250.00	71.085E-03
2500.00	.5 [illegible]
2750.00	1. 752E-0
3000.00	1. [illegible]
3250.00	1. [illegible]
3500.00	2.5 1E-02
3750.00	3. 905E-02
4000.00	54.051E-02
4250.00	69.569E-02
4500.00	5.516E-02
4750.00	8. 36E-02
5000.00	3 . [illegible]

Table 2 - Lethality Values when h=infinite (CAN1)

RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
250.00	31.[illegible]E-04
500.00	37.735E-04
750.00	98.879E-04
1000.00	15.617E-03
1250.00	23.555E-03
1500.00	31.958E-03
1750.00	39.071E-03
2000.00	46.736E-03
2250.00	54.904E-03
2500.00	63.785E-03
2750.00	85.01eE-03
3000.00	10.435E-02
3250.00	12.796E-02
3500.00	15.[illegible]E-02
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
4500.00	51.54 E-02
4750.00	52.782E-02
5000.00	52.890E-02

Table 3 - Lethality Values when h=500 W/m^2-°C (CAN2)

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	1.000E-01
250.00	99.954E-02
500.00	99.287E-02
750.00	98.334E-02
1000.00	97.093E-02
1250.00	95.612E-02
1500.00	93.903E-02
1750.00	91.983E-02
2000.00	89.868E-02
2250.00	87.574E-02
2500.00	85.117E-02
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
3750.00	71.414E-02
4000.00	70.447E-02
4250.00	69.533E-02
4500.00	68.684E-02
4750.00	[illegible]
5000.00	[illegible]

Table 4 - Nutrient Retention Fractions when h=infinite (CAN1)

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	1.0000E+01
250.00	99.[illegible]7E-02
500.00	99.591E-02
750.00	98.970E-02
1000.00	98.109E-02
1250.00	97.013E-02
1500.00	95.691E-02
1750.00	94.150E-02
2000.00	92.401E-02
2250.00	90.457E-02
2500.00	88.332E-02
2750.00	86.045E-02
3000.00	83.616E-02
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
4250.00	73.728E-02
4500.00	72.601E-02

Table 5 - Nutrient Retention Fractions when h=500 W/m^2-°C (CAN2)

A second run has been included which illustrates the temperature distributions throughout the can. In this run the can has been processed for 3600 seconds and cooled for 2000 seconds, at the same process temperatures described in the first run above. The temperature calculations have been done assuming that h is infinite (using CAN1). Figures 27, 28 and 29 show the temperature curves at the centre, at a radial value of 0.0304 m and an axial value of 0.0278 m, and at the top outer edge of the can, respectively. As can be seen from these figures, the temperature curves more closely approximate the step input temperature function as the position under consideration approaches the outer surfaces of the can. The associated lethality and nutrient retention output can be seen in Appendix 2.3.

5.2.4 MAINE

MAINE was used to determine optimal batch sterilization processes for conduction-heating foods in a can. The calculations are based on the slowest heating region approach wherein only the centre temperatures were used. The optimization was done with respect to retention of four nutrients. In the first execution the nutrients optimized were thiamine, anthocyanin, a typical enzyme and another typical vitamin. Figure 30 shows the objective function value versus retort temperature graph for the temperature range of 110 to 130 ^{o}C. The plotted symbols represent the first ten objective function values calculated for the process temperatures from 110 to 130 ^{o}C, at 2 degree intervals (refer to Chapter 4, section 4.3.2.5 for full details). The curve is the least-square third degree polynomial fitted to the objective function values. The legend gives the associated cooling start times for each retort temperature plotted. The values of the objective function values are given

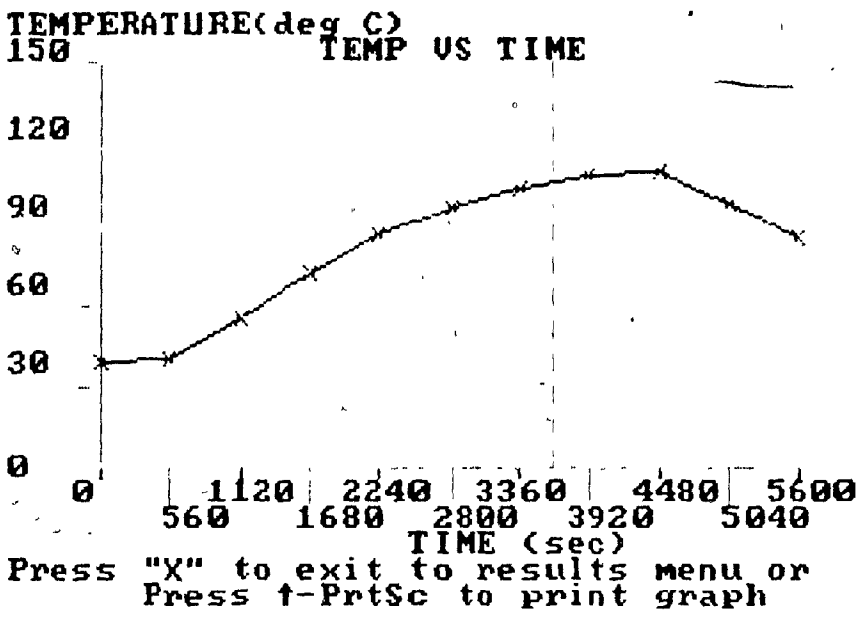


Figure 27 - Temperature History at the Centre of the Can
h=infinite (CAN1)

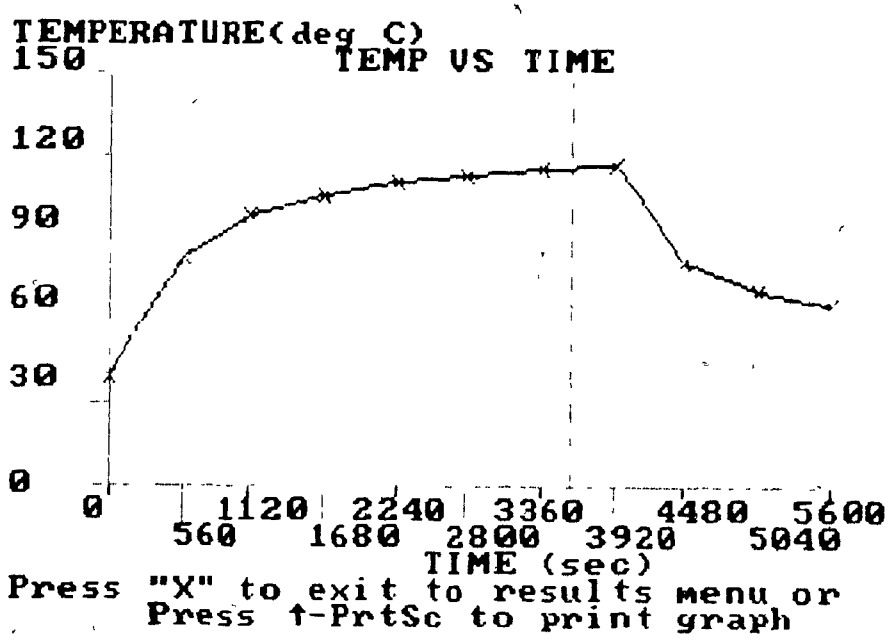


Figure 28 - Temperature History at(.0304m,.0278m) in Can
h=infinite (CAN1)

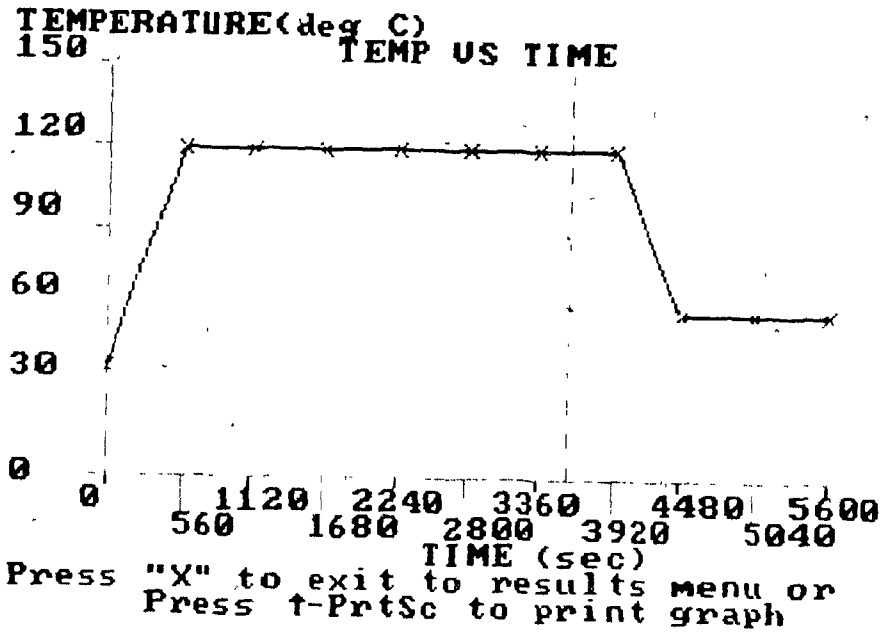


Figure 29 - Temperature History at the Outer Surface of the Can
h=Infinite (CAN1)

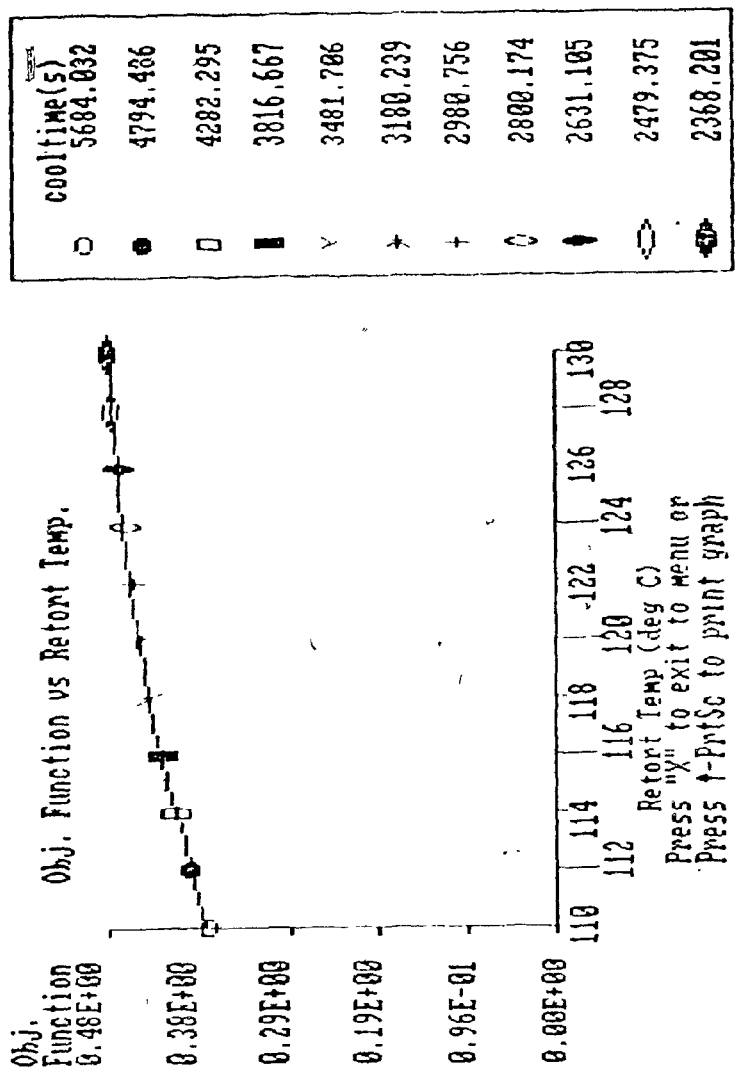


Figure 30 - Objective Function Value vs Retort Temperature
(Thiamine, Anthocyanin, General Enzyme, General Vitamin)

Results

Retort Temp (C)	Cooling starts at (s)	Objective Function value
110.000	5684.032	0.3783E+00
112.000	4794.486	0.3937E+00
114.000	4282.795	0.4079E+00
116.000	3816.667	0.4210E+00
118.000	3461.706	0.4329E+00
120.000	3180.239	0.4436E+00
122.000	2980.756	0.4531E+00
124.000	2800.174	0.4617E+00
126.000	2631.105	0.4681E+00
128.000	2479.375	0.4737E+00
130.000	2368.201	0.4778E+00
130.001	2368.201	0.4778E+00

Table 6 - Objective Function Values

(Thiamine, Anthocyanin, General Enzyme, General Vitamin)

in Table 6; the optimal value is the last line in the table. As can be surmised from the graph the optimal value is at the highest process temperature, 130 ^{o}C. At this temperature the nutrient retention fractions are: thiamine - .86, anthocyanin - .09, a typical enzyme - .005, a typical vitamin - .77. The process parameters and complete printout of this run can be seen in Appendix 2.4.

Two other runs were included in this chapter to illustrate the types of objective function values that can result from various processing regimes. In Figure 31, a run using four general nutrient kinetic data is shown. It can be seen that the shape of the curve is quite different from that of Figure 30. The reader may note a negative objective function value of -.68E-06 at the lowest retort temperature 110 ^{o}C. Previously, it was stated that the minimum allowable value of the objective function is 0. The apparent anomaly can be explained by the fact that the curve is a polynomial approximation of the actual objective function values; the value of -.68E-06 is very close to 0 and represents a slight deviation from the actual value.

The objective function value versus process temperature graph obtained from the third run is shown in Figure 32. The shape of the curve differs from those of the first two runs by its concaveness. The maximum objective function value is attained at a retort temperature of 125.55 ^{o}C, as shown in the last line of Table 7. This run is also for four general nutrients with randomly-chosen F and z data, and the process parameters and other sample output are included in Appendix 2.4. From Figure 32 one can see that the cooling start time is equal to the total processing time (4000 seconds) for

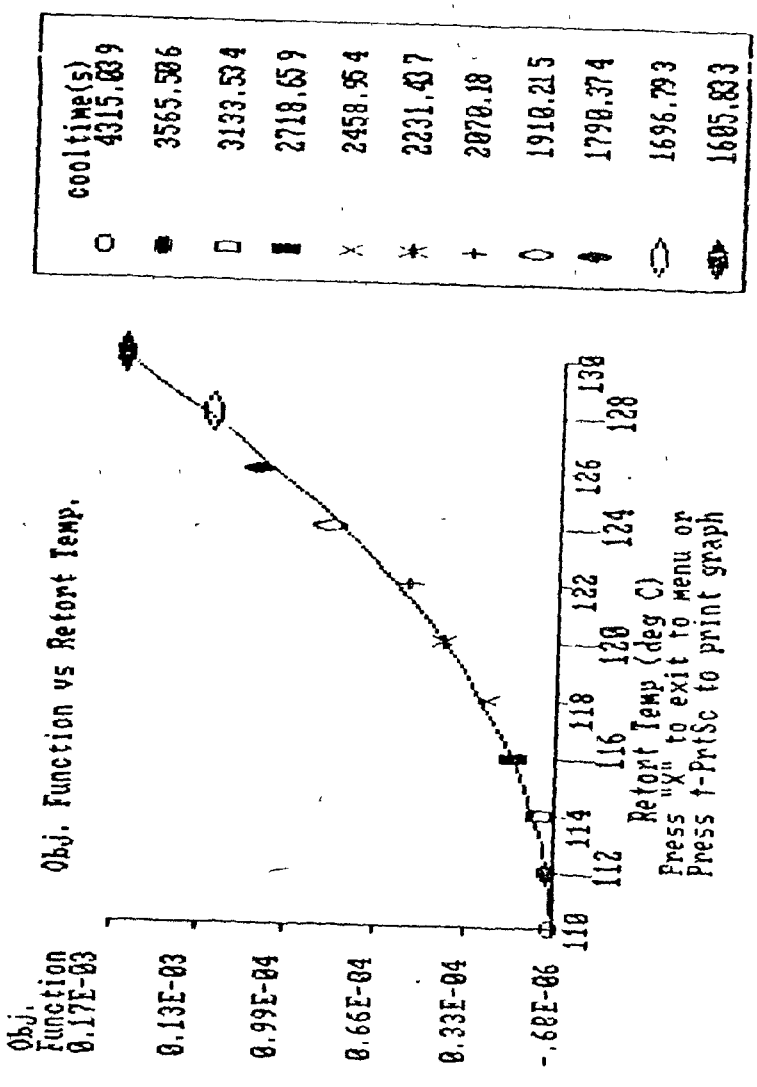


Figure 31 - Objective Function Value vs Retort Temperature
(Four General Nutrients)

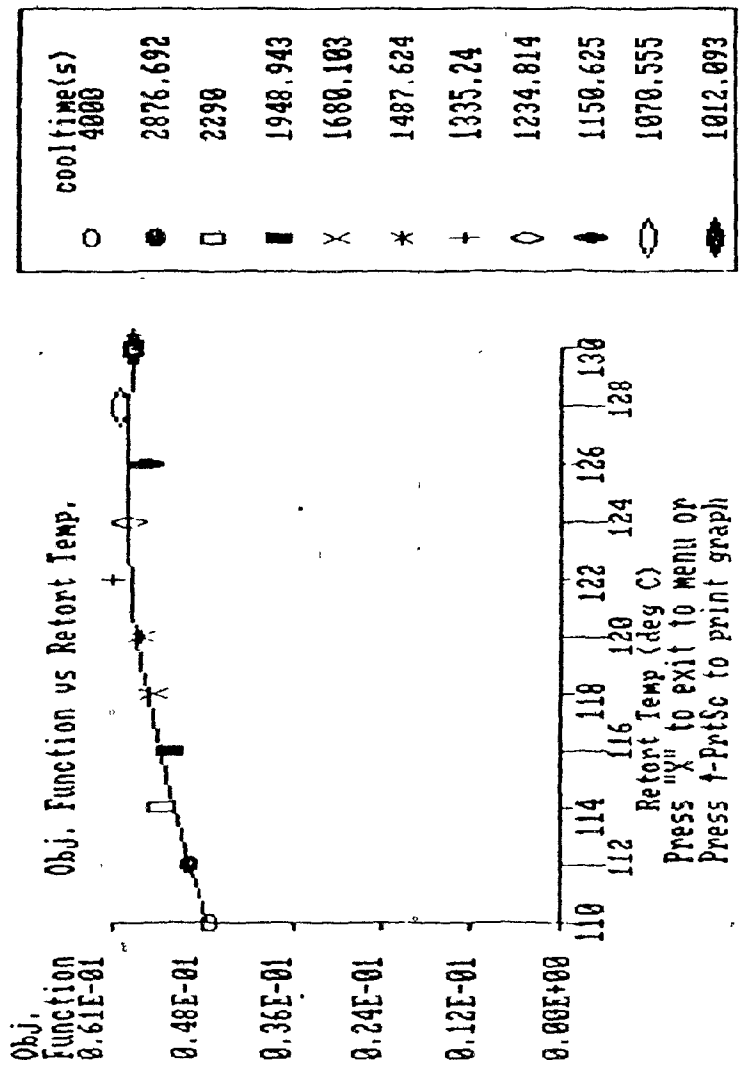


Figure 32 - Objective Function Value vs Retort Temperature
(Four General Nutrients)

Results

Retort Temp(C)	Cooling starts at (s)	Objective Function value
110.000	4000.000	0.4782E-01
112.000	2876.692	0.5036E-01
114.000	2290.000	0.5257E-01
116.000	1948.943	0.5445E-01
118.000	1680.103	0.5599E-01
120.000	1487.624	0.5718E-01
122.000	1335.240	0.5801E-01
124.000	1234.814	0.5849E-01
126.000	1150.625	0.5859E-01
128.000	1070.555	0.5832E-01
130.000	1012.093	0.5766E-01
125.551	1160.748	0.5860E-01

Table 7 - Objective Function Values
(Four General Nutrients)

a retort temperature of 110 °C. This occurs because of the way the cooling start times are calculated; they are determined through an iterative procedure. The cooling start time is first set equal to the total time and the lethality that would be produced by this process is calculated. The cooling start time is then decreased by a factor, and the new lethality is calculated. This procedure is repeated until the process lethality is within 0.05 of the target lethality specified by the user. However, if the process lethality calculated is within 0.05 of the user lethality when the cooling start time is initially set equal to the total process time, the iteration is not performed, and thus the cooling start time at 110 °C will be equal to the total processing time. A similar situation could occur when the cooling start time is being calculated for the optimal retort temperature. Again the optimal start time is selected so that the processing regime will produce a lethality within 0.05 of the user-specified lethality. It is possible that the cooling start time calculated will be equal to one chosen by the program for another retort temperature that is close to the optimal retort temperature. It is felt that the inaccuracy induced in the cooling start time value by the 0.05 lethality difference is not sufficient to be a cause of concern, being in the order of only 20 seconds in the author's experience with the package.

5.3 Package Limitations

There are several limitations associated with this package, all of which are directly related to BASIC programming language restrictions, and which cannot be corrected at the present time using currently-available software and hardware. One problem involves the accuracy of the process calculations when

they are based on the elemental approaches e.g. interfaces MAINC and MAIND. The number of spatial and temporal increments are limited because BASIC provides for the allocation of only 64 KB for the variable workspace. Thus, MAINC allows up to 125 spatial elements and 21 temporal increments while in MAIND up to 100 spatial elements and 21 time increments are permitted. The accuracy of the process calculations could be affected if the magnitudes of the time or spatial increments are excessive. This could be especially significant for the time increment since lethality and nutrient retention values are quite sensitive to temperature changes, particularly at the elevated temperatures. For a processing time of 4000 seconds, choosing the maximum number of time increments allowed would still yield a sizeable incremental time value of 200 seconds. This can be compared to the incremental size of 3 seconds suggested by Smith and Tung (1982), and 60 seconds by Tung and Garland (1978). Mathematical analysis should be done in the future to determine if, and to what extent, accuracy is affected. If the size of the spatial and/or temporal increments is found to significantly affect process calculation accuracy then the MAINC and MAIND modules should be modified in some way to allow the specification of a larger number of spatial and/or temporal increments.

A second limitation of the package is the speed of execution, especially in the case of MAINE. This module requires approximately 25 minutes to run. This is partially because it must do numerous calculations during the iterative process. In the future, a floating point processor may become available for use with programs written in compiled BASIC. Until this hardware is installed the program execution speed will be limited by the speed

of the BASIC language. Another factor in the execution time of MAINE is the frequency with which it interacts with the physical disk drive A to access the module BALLST. If the package is modified so the C drive is assigned to a 5 MB hard disk rather than a virtual RAM disk, BALLST could be stored there as well as on the physical drive A, and thus the access time would be decreased significantly.

VI. SUMMARY AND CONCLUSIONS

6.1 Summary

Thermal processing is one of the most common food preservation methods employed. Processing food in this manner reduces the amount of deterioration and lengthens the food's shelf-life, but also lowers the nutrient levels in the food. It is therefore desirable to subject food only to as much thermal processing as is necessary to destroy the contaminant, so as to retain as much of the nutrients as possible. To predict an adequate processing regime both the heat transfer through the food and the consequent effects on the contaminant population and on the food components must be known. The calculations which must be performed to determine the processing regime are complex, but with the greater availability of computing power they are possible.

To make the design and verification of thermal processes easier for the food engineer, the necessary process computations were incorporated in a user-friendly software package. The package runs on a typical microcomputer, the IBM PC, and requires very little computer experience on the part of the user. The package can determine the heat transfer in conduction - heating foods processed batchwise, for food held in cans or retortable pouches. The user decides the manner in which the temperature distribution is calculated: either utilizing the slowest heating region (SHR) approach (for the can), or treating the temperature as a function of position (for the can or pouch) - with or without consideration of the surface convective heat transfer coefficient. After calculating the heat transfer distribution in the food,

the package can then predict the consequent effects on the contaminant population and on the nutrients in the food. Either an SHR approach or an element approach can be choosen by the user for these process calculations. Finally, the package can be used to optimize the batch sterilization process with respect to retention of up to four nutrients. It will chose and display the best heating regime, in terms of processing temperature and time, corresponding to the process parameters supplied by the user.

6.2 Conclusions

The following conclusions were drawn from the work:

1- The modular system design proved to be successful. It is able to circumvent the major BASIC-language limitations of program size and subroutine capabilities in the following ways:

a) The program modules are overlaid in volatile memory as needed, and so the program package is not limited to 64 KB - it is over 200 KB. Only the individual modules are affected by this size restriction.

b) By chaining to various program modules and storing variables in separate data files, the subroutine capabilities of Fortran are emulated. Variable names remain local to each program module, and "arguments" and "parameters" are passed between programs during package execution.

c) Storing operating codes in the data files allows the package to avoid the program-chaining restrictions of the BASIC compiler e.g. not being able to chain to a specific line in the next program.

2- The package is very easily expanded, since the calculating routines and the interfaces are completely independent. The calculating routines have

minimal system interactions, and so additional routines are easily incorporated. The interfaces interact extensively with the other system programs and the data and communication files, but each interface is based on a standard system interaction design and additional modules can be incorporated with little difficulty.

3- The execution of the package required more volatile memory than was originally envisioned i.e. 640 KB rather than 256 KB. The extra memory is required to enable the installation of the C virtual drive files - the ALPHA text files - which are all copied onto the C drive during the system initialization. This could restrict the expansion of the package in terms of the number of ALPHA files that can be held on the C drive (and therefore the number of interface modules which can be added to the package). This can be avoided by slightly modifying the system initialization programs to allow the package to run on an IBM microcomputer equipped with a Winchester removable cartridge (hard disk) of 5 MB, and considering the hard disk as the C drive. The hard disk access time is almost as short as for the virtual drive, and the hard disk has much larger memory capabilities.

4- The system design proved to be powerful enough that not all elements of the design are as yet fully utilized. For example, the files responsible for communication between MAIN and the subMAINs, files COMMA and COMMB, have still to be used. These features may become especially important in the event of future expansion of the system.

5- The modules in which the process calculations are done by means of the

element approach, modules MAINC and MAIND, require memory for the variables to such an extent that the number of time increments and spatial elements are restricted. In MAINC (for the pouch) up to 21 time increments and 125 spatial elements are allowed while in MAIND (for the can) there can be 21 time increments and 125 spatial elements. These restrictions may decrease the accuracy of the process calculations if the sizes of the time or spatial increments are excessively large. The size of the time increment may be of particular significance, since the lethality values are quite temperature-sensitive at the higher food temperatures i.e. near processing temperature.

6- The optimization module performs its functions correctly but could be modified to improve its execution. It accepts user-supplied parameters, and determines values of the objective function for retort temperatures between 110 and 130 ^{o}C. With these values, an equation is found which relates the processing temperatures and the values of the objective function. The optimal processing temperature is then determined from the equation, and the required processing time is calculated. There is a drawback to the module, however, in that the execution time is quite long for an interactive program (about 25 minutes). This is because of the extensive calculations required for the optimization and frequent access of a module stored on the physical drive A. Modification of the optimization module and improvement of the hardware components may decrease execution time.

6.3 Recommendations for Further Study

The followng areas are considered to be important for further investigation:

1- The package initialization programs could be modified to allow the use of a Winchester removable cartridge (hard disk) with 5 MB of memory. This could be designated as the C drive, and so enable the addition of a greater number of interface modules, whose associated text files must be stored on the C drive.

2- Future work on the system should involve the further development of the service program interface. At present, the results display facilites are incorporated in each subMAIN module. Full utilization of the system capabilities dictates the eventual segregation of the interactive interfaces and the display programs. As well, the session restart facility should be developed.

3- A mathematical analysis should be performed to determine the extent to which the accuracies of the process and nutrient retention calculations are dependent upon the sizes of the time and spatial increments. If the incremental sizes are found to be significant, as is suspected, the programs MAINC and MAIND (which use the elemental approach) should be modified in such a way as to allow the use of a greater number of time increments and spatial elements, therefore decreasing the differential size.

4- The present optimization module performs correctly but requires about 25 minutes execution time. It could be modified to decrease the execution

time in several ways. The calculating routine, BALLST, is accessed extensively during the optimization procedure. If it was stored on the virtual drive C as well as on the physical drive A, the access time could be decrease significantly. Also, the use of a floating-point processor would help speed the number-crunching operations. There is such a processor available for use with a BASIC compiler; there may be one available from IBM for the IBM BASIC compiler in the near future.

5- Additional optimization modules could be developed which would base process and nutrient calculations on the elemental approach, for food held in both cans and retortable pouches.

6- Laboratory experimentation could be used to verify the models used in the package, including the heat transfer, process adequacy and nutrient retention algorithms.

LITERATURE CITED

Abramowitz M.A., I. Stegun. 1970. Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables, 3rd ed., National Bureau of Standards, Applied Mathematics Series, No.55.

Ball C.O. 1923. Thermal Process Time for Canned Food, Bulletin of the National Research Council, Vol.7, No.37, Part I, Washington,DC.

Ball C.O. 1928. Mathematical Solutions of Problems on Thermal Processing of Canned Food. Univ. Calif. (Berkeley) Publ. Public Health 1(2): 15.

Ball C.O. and F.C.W. Olson. 1957. Sterilization in Food Technology, 1st ed., McGraw-Hill Book Company, New York.

Bender F.E., A. Kramer, G. Kahan. 1982. Linear Programming and its Applications in the Food Industry, Food Tech., July, p.94-96.

Benedict R. 1981. Manual on the Use of Thermocouples in Temperature Measurement, American Society for Testing and Materials, Philadelphia. ASTM PCN 04-470020-40.

Beyer W.H., editor. 1981. CRC Standard Mathematical Tables, 26th ed., CRC Press, Boca Raton, Florida.

Bigelow W.D., G.S. Bohart, A.C. Richarson, C.O. Ball. 1920. Heat Penetration in Processing Canned Foods, Natl. Canners' Assn. Bul., 16L, 128 p.

Carslaw H.S. and J.C. Jaeger. 1947. Conduction of Heat in Solids, Oxford University Press, Oxford.

Carslaw H.S. and J.C. Jaeger. 1959. Conduction of Heat in Solids, 2nd ed., Oxford University Press, Oxford.

Castillo P.F., J.A. Barreiro, G.R. Salas. 1980. Prediction of Nutrient Retention in Thermally Processed Heat Conduction Food Packaged in Retortable Pouches, J. Food Sci., Vol.45.

Charm S.E. 1978. The Fundamentals of Food Engineering, 3rd ed., Avi Publishing Company, Westport, Connecticut.

Esty J.R. and K.F. Meyer. 1922. The Heat Resistance of the Spores of Bacillus botulinus and Allied Anaerobes. XI. Journal of Infectious Disease. 31(6): 650-663.

Evans L.B. 1982. Optimization Theory and its Application in Food Processing, Food Tech., July, p.80-93,96.

Felliciotti E., W.B. Esselen. 1957. Thermal Destruction Rates of Thiamine in Pureed Meats and Vegetables, Food Tech. 11(2):77.

Flambert C.M.F. and J. Deltour. 1972. Exact Lethality Calculation for Sterilizing Process. I. Principles of the Method. Lebensm.-Wiss. u Technol. 5:72.

Griffin, R.C., D.H. Herndon and C.O. Ball. 1969. Use of Computer-Derived Tables to Calculate Sterilizing Processes for Packaged Foods. II, Food Tech., Vol.23(4): 519-524.

Griffin, R.C., D.H. Herndon and C.O. Ball. 1971. Use of Computer-Derived Tables to Calculate Sterilizing Processes for Packed Foods. III, Food Tech., Vol.25(2): 134-143.

Gillespy T.C. 1951. Estimation of Sterilizing Values of Processes as Applied to Canned Foods.1. Packs Heating by Conduction. J.Sci. Food Agric. 2:107.

Gillespy T.C. 1953. Estimation of Sterilization Values of Processes as Applied to Canned Food.2. Pack Heating by Conduction. J. Sci. Food Agric. 4:553.

Hayakawa K-I. 1969. New Parameters for Calculating Mass Average Sterilizing Value to Estimate Nutrients in Thermally Processed Food. Can. Inst. Food Technol. J. 2:165.

Hayakawa K-I. 1971. Estimating Food Temperature During Various Processing or Handling Treatments, Journal of Food Science. Vol. 36(3): 378-385.

Hayakawa K-I. 1973. Modified Lethal Rate Paper Technique for Thermal Process Evaluation. Can. Inst. Food Technol.J. 6:295.

Hayakawa K-I. 1978. A Critical Review of Mathematical Procedures for Determining Proper Heat Sterilization Processes,Food Tech.,32(3):59-65

Herndon D.H., R.C. Griffin, C.O. Ball. 1968. Use of Computer-Derived Tables to Calculate Sterilizing Processes for Packaged Foods. I, Food Tech., Vol.22(4): 473-484.

Hicks E.W. 1951. On the Evaluation of Canning Processes. Food Tech., 5(4):134-142.

Hicks, E.W. 1958. A Revised Table of The P_h Function of Ball and Olson. Food Research. 23(4): 396-400.

Hildenbrand P. 1980. An Approach to Solve the Optimal Temperature Control Problem for Sterilization of Conduction-Heating Foods. J. Food Proc. Eng. (3):123.

Hill C.G., R.A. Grieger-Block. 1980. Kinetic Data: Generation, Interpretation, and Use, Food Tech., 34(2): 56.

Holman J.P. 1976. Heat Transfer, 4th ed., McGraw-Hill Book Company, New York.

Jackson, J.M. 1940. Mechanisms of Heat Transfer in Canned Foods During Thermal

Processing. p.39-50. Proceedings of The First Food Conference of The Institute of Food Technology.

Jackson, J.M. and F.C.W. Olson. 1940. Thermal Processing of Canned Foods in Tin Containers. IV. Studies of The Mechanisms of Heat Transfer Within The Contanier. Food Research. 5(4): 409-421.

Jen Y., J.E. Manson, C.R. Stumbo. 1971. A Procedure for Estimating Sterilization of and Quality Factor Degradation in Thermally Processed Foods, J. Food Sci., Vol.36, p.692-698.

Kok R. 1982. Unpublished Notes.

Lenz M.K., D.B. Lund. 1977a. The Lethality-Fourier Number Method: Experimental Verification of a Model for Calculating Temperature Profiles and Lethality in Conduction-Heating Canned Foods, J. Food Sci., Vol.42(4):989-1001.

Lenz M.K., D.B. Lund. 1977b. The Lethality-Fourier Number Method: Experimental Verification of a Model for Calculating Average Quality Factor Retention in Conduction-Heating Canned Foods, J. Food Sci, 42(4):997-1001.

Lenz M.K., D.B. Lund. 1980. Experimental Procedures for Determining Destruction Kinetics of Food Components,Food Tech.,February,p.51-55.

Leonhardt, G.F. 1976. Estimation of The Central Temperature of Thermally Conductive Food in Cylindrical and Rectangular Cans During Heat Processing. Journal of Food Science. Vol.41(3): 685-690.

Leonhardt, G.F. 1978. A General Lethal-Rate Paper for The Graphical Calculation of Processing Times. Journal of Food Science. Vol.43(2): 660.

Levenspiel O. 1962. Chemical Reaction Engineering, An Introduction to the Design of Chemical Reactors, John Wiley and Sons, New York.

Luikov A.V. 1968. Analytical Heat Diffusion Theory, Academic Press, New York.

Lund D.B. 1977. Design of Thermal Processes for Maximizing Nutrient Retention. Food Tech. 31(2):71-78.

Lund D.B. 1982. Applications of Optimization in Heat Processing, Food Tech., July, p.97-100.

Manson J.E., J.W. Zahradnik, C.R. Stumbo. 1970. Evaluation of Lethality and Nutrient Retentions of Conduction Heating Foods in Rectangular Containers, Food Tech., Vol.24, No.1297, p.109-113.

Merson R.L., R.P. Singh, P.A. Carroad. 1978. An Evaluation of Ball's Formula Method of Thermal Process Calculations, Food Technology. 32(3): 66-72,75.

Miller A.R. 1981. BASIC Programs for Scientists and Engineers, Sybex, Berkley,

California.

Mulley E.A., C.R. Stumbo, W.M Hunting. 1975. Thiamine: A Chemical Index of the Sterilization Efficacy of Thermal Processing. J. of Food Sci. 40(5):993-996.

Nakai S. 1981. Comparison of Optimization Techniques for Application to Food Product and Process Development,J.Food Sci.,Vol.47,p.144-152.

Nakai S. 1984. Standarization of Mapping Simplex Optimization, personal communication, to be published as a research note.

Nakai S. 1984. Simplex Optimization BASIC Program, personal communication.

Nakai S., K. Koide, K. Eugster. 1984. A New Mapping Super-Simplex Optimization for Food Product and Process Development, J. Food Sci. Vol.49, No.4, p.101-117.

Navankasattusas, D.B. Lund. 1978. Monitoring and Controlling Thermal Processes by On-Line Measurement of Accomplished Lethality, Food Tech., March, p.79-83.

Newman A.B. 1936. Heating and Cooling Rectangular Solids. Ind.Eng.Chem. 28:545.

Ohlsson T. 1980. Optimal Sterilization Temperatures for Sensory Quality in Cylindrical Containers, J. Food Sci., Vol.45 :1517.

Olson, F.C.W., and H.P. Stevens. 1939. Thermal Processing of Canned Foods in Tin Containers. II. Nomograms for Graphic Calculation of Thermal Processes for Non-Acid Canned Foods Exhibiting Straight-Line Semi-Logarithmic Heating Curves. Food Research. Vol.4(1): 1-20.

Olson, F.C.W. and J.M. Jackson. 1942. Heating Curves, Theory and Practical Application. Industrial and Engineering Chemistry. Vol.34(3): 337-341

Olson, F.C.W. and O.T. Schultz. 1942. Temperatures in Solids During Heating or Cooling. Industrial and Engineering Chemistry. Vol.34(7): 874-877.

Patashnik, M. 1953. A Simplified Procedure for Thermal Process Evaluation. Food Technology. Vol.7(1): 1-6.

Pelczar M.J., R.D. Reid, E.C.S. Chan. 1977. Microbiology, 4th ed., McGraw-Hill Book Company, New York.

Ramaswamy H.S., K.V. Lo, M.A. Tung. 1982. Simplified Equations to Predict Unsteady Temperatures in Regular Conductive Solids, J. Food Sci., Vol.47, p.293-309.

Riedel L. 1947. Mitt. Kaltetech Inst. Karlsruhe, Nr.1.

Rizvi S.S.H. J.C. Acton. 1982. Nutrient Enhancement of Thermostabilized Foods in Retort Pouches, Food Tech., April, p.105-109.

Saguy I., 1982. Optimization Theory, Techniques, and Their Implementation in the Food Industry: Introduction, Food Tech., July, p.87.

Saguy I. and M. Karel. 1979. Optimal Retort Temperature Profile in Optimizing Thiamin Retention in Conduction Type Heating of Canned Foods, J. Food Sci., Vol.44, p.1485-1490.

Saguy I. and M. Karel. 1980. Modeling of Quality Deterioration During Food Processing and Storage, Food Tech., February, p.78-85.

Schultz, O.T. and F.C.W. Olson. 1940. Thermal Processing of Canned Foods in Tin Containers. III. Recent Improvements in The General Method of Thermal Process Calculations- A Special Coordinate Paper And Methods of Converting Initial and Retort Temperatures. Food Research. Vol.5(4): 399-407.

Smith T., M.A. Tung. 1982. Comparison of Formula Methods for Calculating Thermal Process Lethality, J. Food Sci., Vol.47, p.626-630.

Steele, R.J. and P.W. Board. 1979. Amendments to Ball's Formula Method for Calculating the Lethal Value of Thermal Processes. Journal of Food Science. Vol.44(1): 292-293.

Stumbo C.R. 1948. Bacteriological Considerations Relating to Process Evaluation. Food Tech., 2(2):115-132.

Stumbo C.R. 1949. Further Considerations Relating to Evaluation of Thermal Processes for Foods, Food Tech., No.3, p.126-131.

Stumbo C.R. 1953. New Procedures for Evaluating Thermal Processes for Foods in Cylindrical Containers. Food Tech. 7. 309.

Stumbo, C.R. and R.E. Longley. 1966. New Parameters for Process Calculation. Food Technology. Vol. 20(3): 341-345.

Stumbo C.R. 1973. Thermobacteriology in Food Processing, 2nd ed., Academic Press, New York.

Teixeira A.A., J.R. Dixon, J.W. Zahradnik, G.E. Zinsmeister. 1969a. Computer Determination of Spore Survival Distributions in Thermally Processed Conduction Heated Foods, Food Technology. Vol.23(3): 78-80.

Teixeira A.A., J.R. Dixon, J.W. Zahradnik, G.E. Zinsmeister. 1969b. Computer Optimization of Nutrient Retention in the Thermal Processing of Conduction Heated Foods, Food Technology, Vol.23, No.845.

Teixeira A.A., C.R. Stumbo, J.W. Zahradnik. 1975. Experimental Evaluation of Mathematical and Computer Models for Thermal Process Evaluation, J. Food

Sci., Vol.40, p.653-655.

Teixeira A.A., G.E. Zinsmeister, J.W. Zahradnik. 1975. Computer Simulation of Variable Retort Control and Container Geometry as a Possible Means of Improving Thiamine Retention in Thermally Processed Foods, J. Food Sci., Vol.40, p.656-659.

Thijssen H.A.C., P.J.A. Kerkhof, A.A.A. Liefkens. 1978. Short-Cut Method for the Calculation of Sterilization Conditions Yielding Optimum Quality Retention for Conduction-Type Heating of Packaged Foods, J. Food Sci. Vol 43: 1096-1101.

Thijssen H.A.C., L.H. Kochen. 1980. Calculation of Optimum Sterilization Conditions for Packed Conduction-Type Foods, J. Food Sci., Vol.45, p.1267-1272.

Thompson, G.E. 1919. Temperature-Time Relations in Canned Foods During Sterilization. Industrial and Engineering Chemistry. Vol.11(7): 657-664.

Tung M.A., T.D. Garland. 1978. Computer Calculation of Thermal Processes, J. Food Sci., Vol.43, p.365-369.

Tung M.A., T. Smith. 1980. Innovations in Thermal Processing, Agriculture 2000, 75th Proceedings, Macdonald College,Quebec, p.103-120.

Williamson E.D. and L.H. Adams. 1919. Temperature Distribution in Solids During Heating or Cooling, Physical Reviews, 2nd series, 14(2): 99-114.

APPENDIX 1

Bessel Functions of the First Kind of the Zero and First Order

The Bessel function of the first kind of the zero order is given by (Beyer, 1981.):

$$(1)\quad J_0(x) = 1 - \frac{x^2}{2^2(1!)^2} + \frac{x^4}{2^4(2!)^2} - \frac{x^6}{2^6(3!)^2} + \frac{x^8}{2^8(4!)^2} - \ldots$$

The polynomial approximations to the equation $J_0(x)$ are given by (Abramowitz and Stegun, 1970):

$$(2)\quad J_0(x) = 1 - 2.2499997\,(x/3)^2 + 1.2656208\,(x/3)^4 - .3163866\,(x/3)^6 + .444479\,(x/3)^8 - .0039444\,(x/3)^{10} + .0002100\,(x/3)^{12}$$

(for $-3 <= x <= 3$)

$$(3)\quad J_0(x) = x^{-1/2} \cdot d0 \cdot \cos(w0)$$

$$(4)\quad \text{where } d0 = .79788456 - .00000077\,(3/x) - .00552740\,(3/x)^2 - .00009512\,(3/x)^3 + .00137237\,(3/x)^4 - .00072805\,(3/x)^5 + .00014476\,(3/x)^6$$

$$(5)\quad \text{and } w0 = x - .78539816 - .04166397\,(3/x) - .00003954\,(3/x)^2 + .00262573\,(3/x)^3 - .00054125\,(3/x)^4 - .00029333\,(3/x)^5 + .00013558\,(3/x)^6$$

(for $3 <= x < \infty$)

The Bessel function of the first kind of the zero order is given by (Beyer, 1981):

$$(6)\quad J_1(x) = \frac{x}{2} - \frac{x^3}{2^3 \cdot 1!\,2!} + \frac{x^5}{2^5 \cdot 2!\,3!} - \frac{x^7}{2^7 \cdot 3!\,4!} + \frac{x^9}{2^9 \cdot 4!\,5!} - \ldots$$

This equation can be approximated by the following polynomials ((Abromowitz and Stegun, 1981):

$$(7)\quad J_1(x) = x\,((\ (1/2) - .56249985\,(x/3)^2 + .21093573\,(x/3)^4 - .03954289\,(x/3)^6 + .00443319\,(x/3)^8 - .00031761\,(x/3)^{10} + .00001109\,(x/3)^{12} \ ))$$

(for $-3 <= x <= 3$)

$$(8)\quad J_1(x) = x^{-1/2} \cdot d1 \cdot \cos(w1)$$

$$(9)\quad \text{where } d1 = .79788456 + .00000156\,(3/x) + .01659667\,(3/x)^2 + .00017105\,(3/x)^3 - .00249511\,(3/x)^4 + .00113653\,(3/x)^5 - .00020033\,(3/x)^6$$

$$(10)\quad \text{and } w1 = x - 2.35619449\,(3/x) + .12499612\,(3/x) + .0005650\,(3/x)^2 - .00637879\,(3/x)^3 + .00074348\,(3/x)^4 + .00079824\,(3/x)^5 - .00029166\,(3/x)^6$$

(for $3 <= x < \infty$)

APPENDIX 2

Sample Package Executions

APPENDIX 2.1

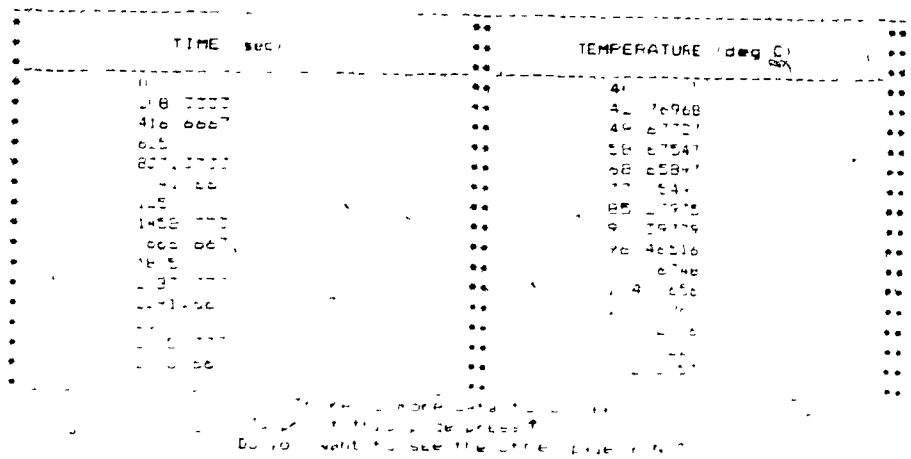
Sample Runs from the Execution of MAINB

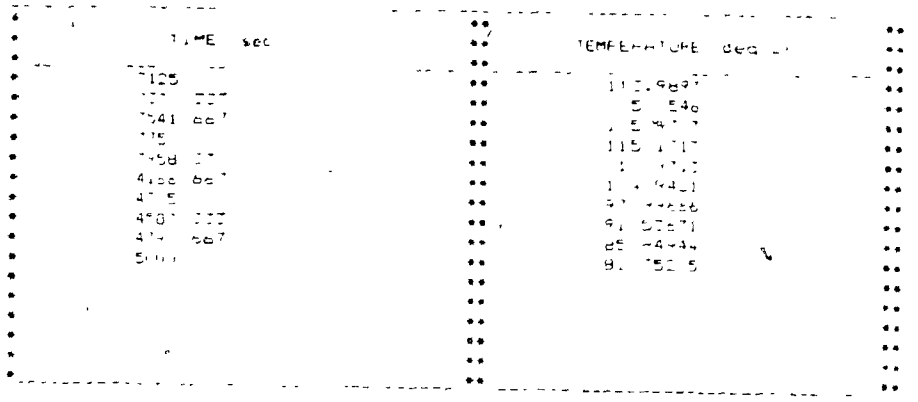
PROCESS PARAMETER TABLE

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Slope of Heating Curve(sec)	2460
Slope of Cooling Curve(sec)	2460
Heating Curve Lag Factor	1.4
Cooling Curve Lag Factor	0
Total Processing Time(sec)	5000
Time When Cooling Starts(sec)	3600
Number of Time Increments	25
F-Value of Microorganism(min)	2.45
z-Value of Microorganism(deg C)	10
F-value of Nutrient(min)	1980
z-Value of Nutrient(deg C)	27

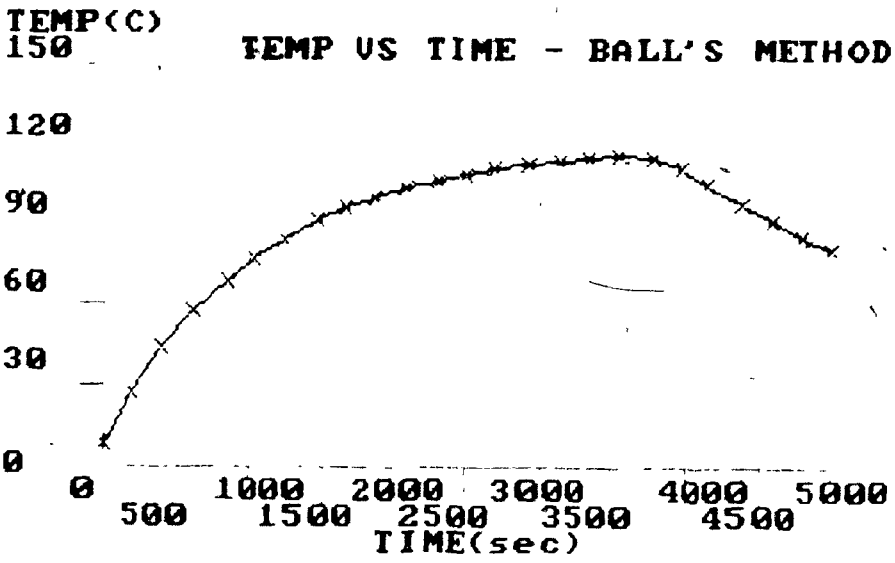
Model 1

Press "X" to exit or ↑-PrtSc to print


TIME sec
TEMPERATURE (deg C)

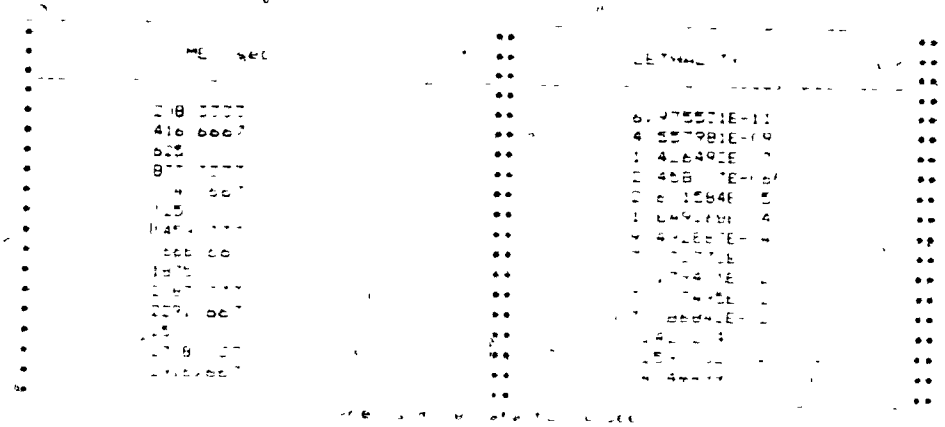

TIME sec
TEMPERATURE deg C
Press Print to print this page
Press any key to get back to results display menu

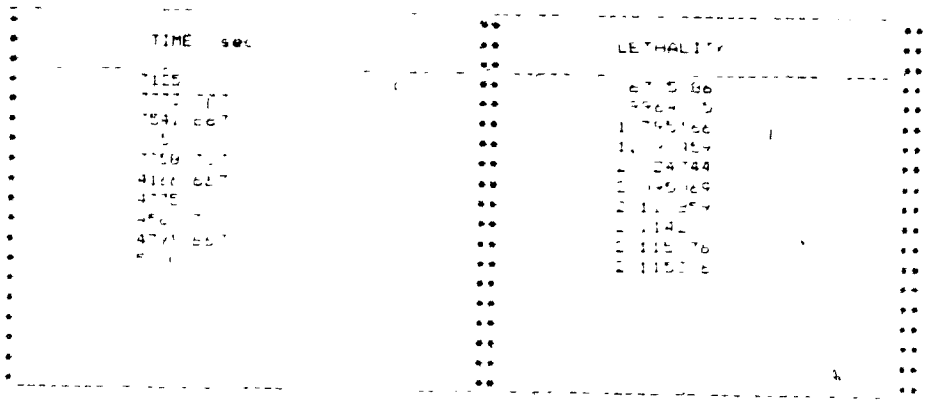
Model 1



Press "X" to exit or ↑-PrtSc to print

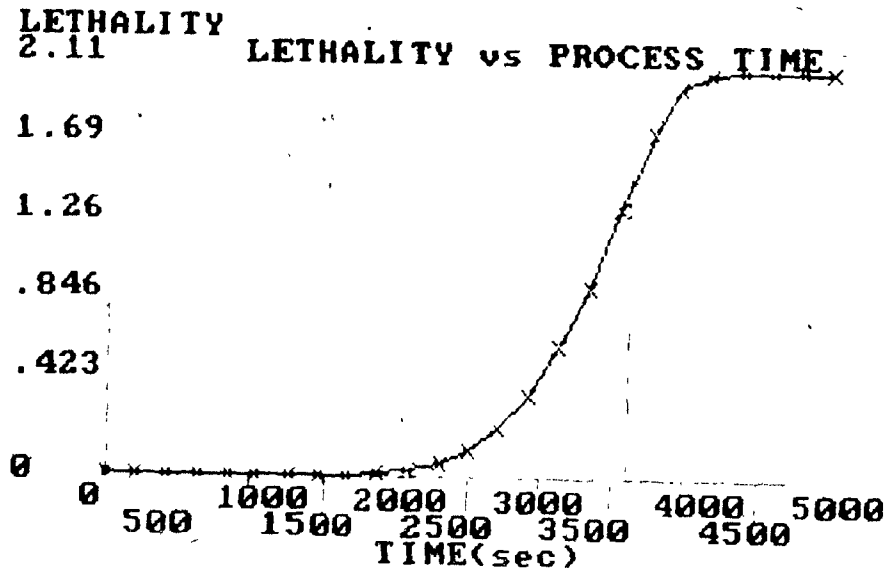
Model 1





Press print to print this page
Press any key to return to results display menu

Model 1



Press "X" to exit or ↑-PrtSc to print

Model 1

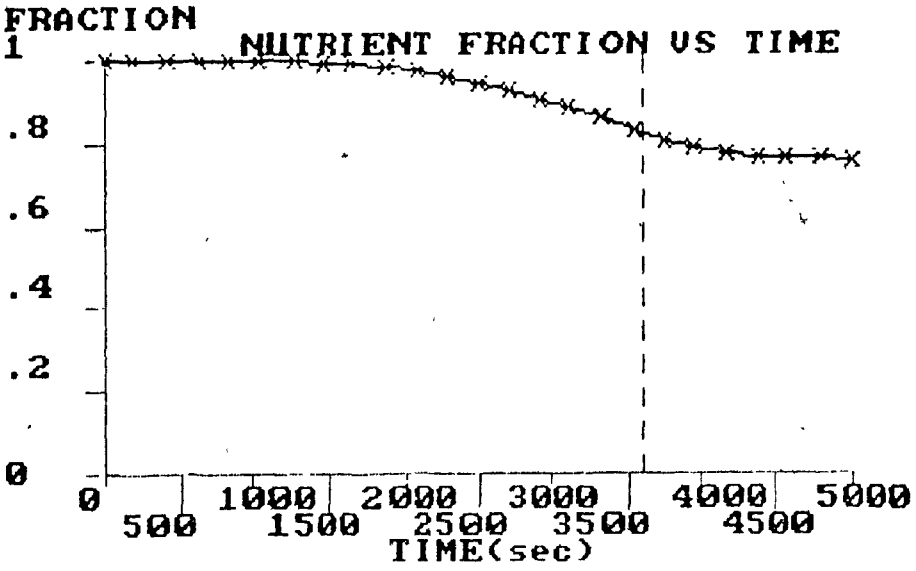
TIME (sec)	NUTRIENT FRACTION
208.3333	.9999927
416.6667	.9999582
625	.9998356
833.3333	.9994877
1041.667	.9986655
1250	.9970008
1458.333	.9941297
1666.667	.9892522
1875	.9811117
2083.333	.9715585
2291.667	.9600598
2500	.9448[illegible]
2708.333	.9266[illegible]
2916.667	.9067472

There is more data to be seen
To print this page press Print Screen
Press ↓ to see the other page of NUTRIENT FRACTION

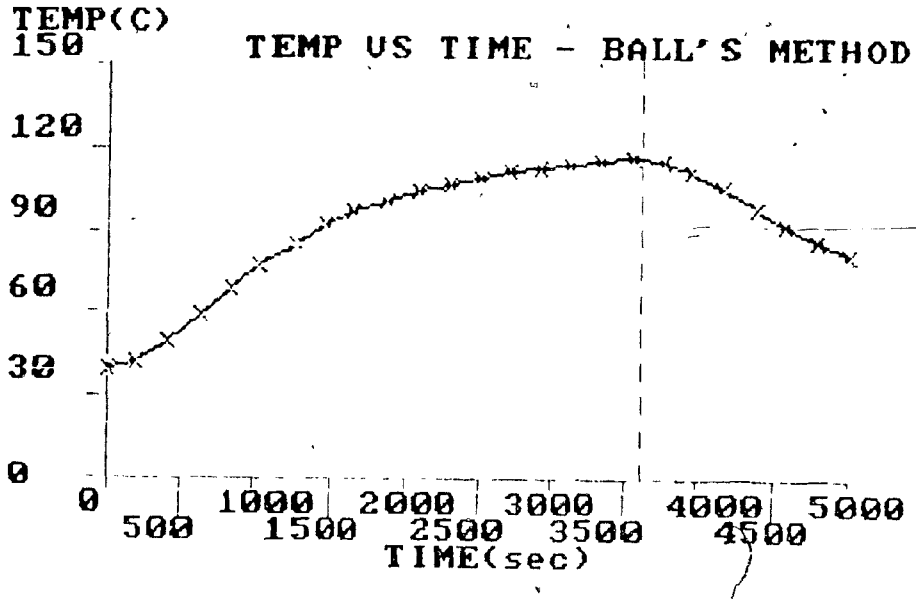
TIME (sec)	NUTRIENT FRACTION
3125	.8877755
3333.333	.8594642
3541.667	.8276224
3750	.7930868
3958.333	.7592452
4166.667	.7267117
4375	.6966818
4583.333	.6696769
4791.667	.6461109
5000	.6261821

Press ↑ PrtSc to print this page
Press any key to get back to results display menu

Model 1

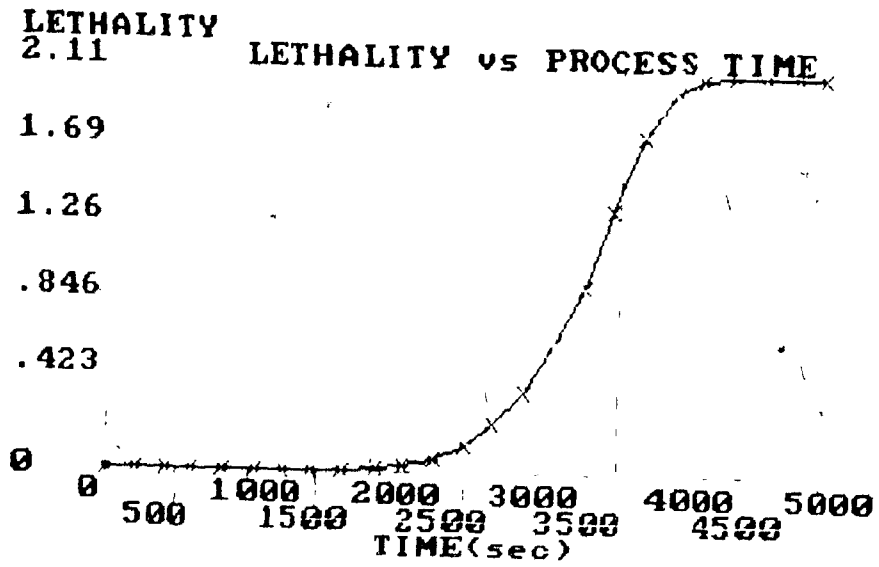


Press "X" to exit or ↑-PrtSc to print



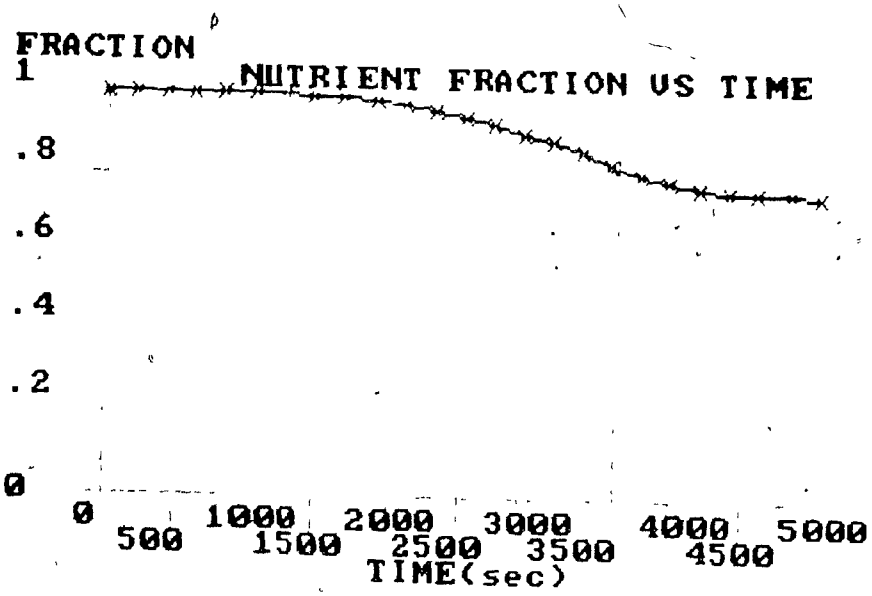
Press "X" to exit or ↑-PrtSc to print

Model 2

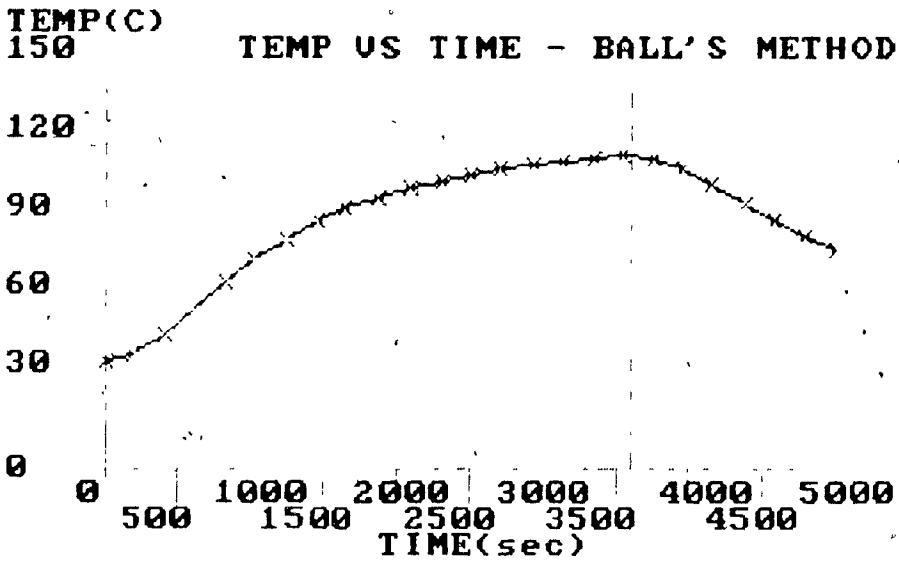


Press "X" to exit or ↑-PrtSc to print

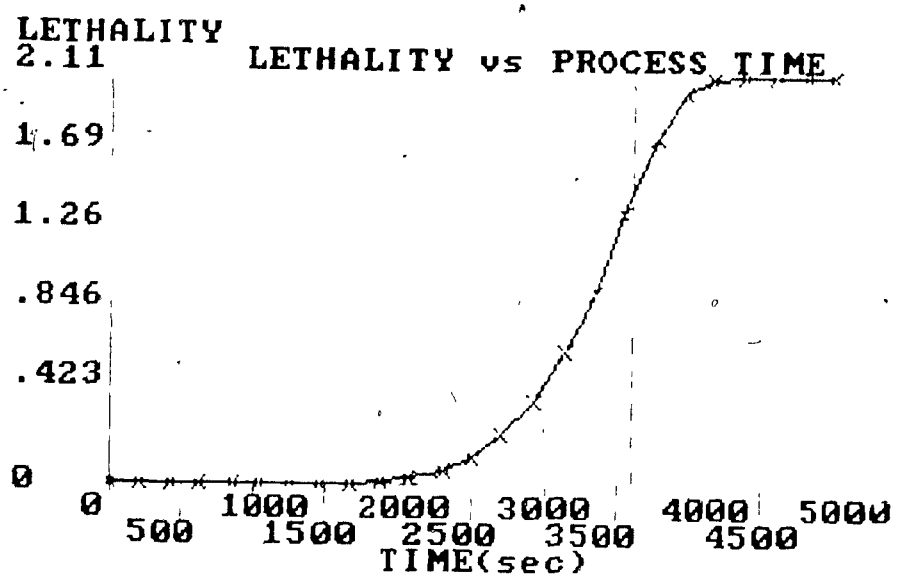
Model 2



Press "X" to exit or ↑-PrtSc to print

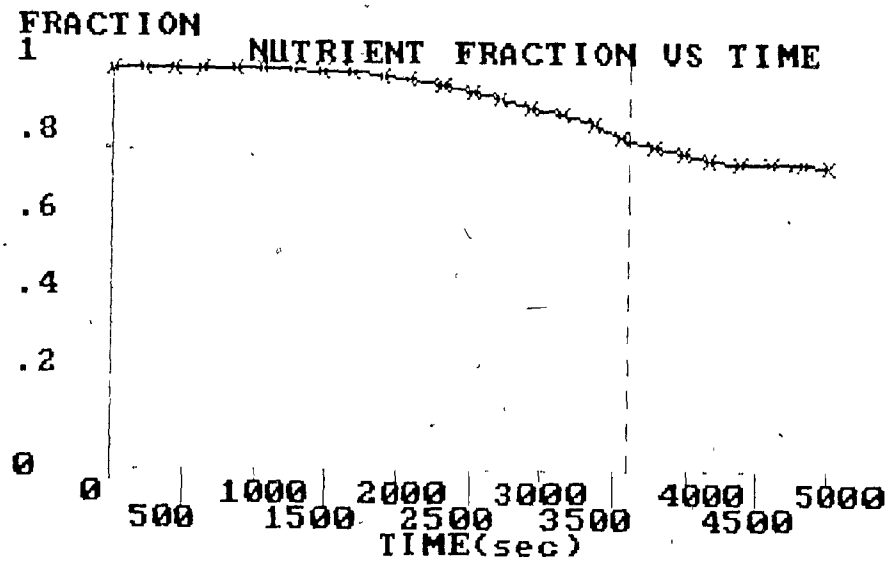


Press "X" to exit or ↑-PrtSc to print



Press "X" to exit or ↑-PrtSc to print

Model 3



Press "X" to exit or ↑-PrtSc to print

APPENDIX 2.2

Sample Runs from the Execution of MAINC

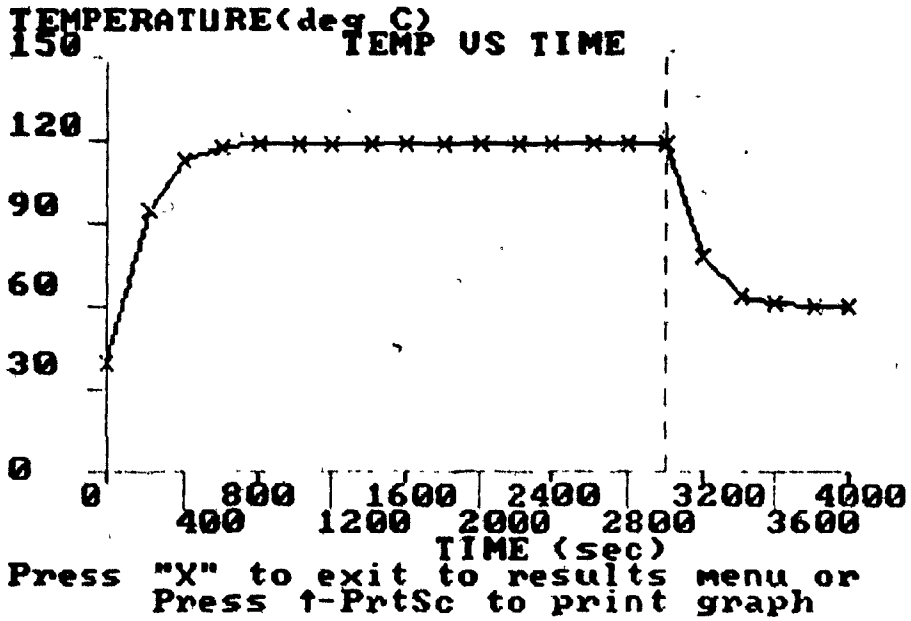
Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	4000
Time When Cooling Starts(sec)	3000
Number of Time Increments	21
Half X-Length(m)	.065
Half Y-Length(m)	.0475
Half Z-Length(m)	.0075
Number of X-values	5
Number of Y-values	5
Number of Z-values	5
Thermal Diffusivity(m**2/sec)	1.6E-07

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

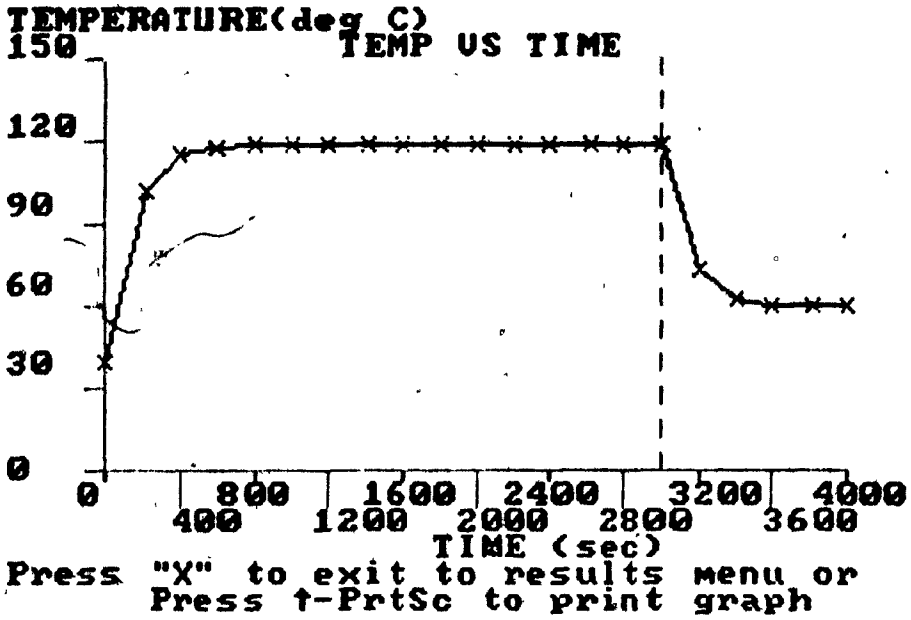
RESULTS

TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C
0.00000	0.00000	0.00000	0.00000	40.00000
200.00000	0.00000	0.00000	0.00000	94.97410
400.00000	0.00000	0.00000	0.00000	113.85100
600.00000	0.00000	0.00000	0.00000	118.49100
800.00000	0.00000	0.00000	0.00000	119.63100
1000.00000	0.00000	0.00000	0.00000	119.91000
1200.00000	0.00000	0.00000	0.00000	119.97800
1400.00000	0.00000	0.00000	0.00000	119.99400
1600.00000	0.00000	0.00000	0.00000	119.99800
1800.00000	0.00000	0.00000	0.00000	119.99900
2000.00000	0.00000	0.00000	0.00000	119.99900
2200.00000	0.00000	0.00000	0.00000	120.00000
2400.00000	0.00000	0.00000	0.00000	120.00000
2600.00000	0.00000	0.00000	0.00000	120.00000
2800.00000	0.00000	0.00000	0.00000	120.00000
3000.00000	0.00000	0.00000	0.00000	120.00000
3200.00000	0.00000	0.00000	0.00000	78.76930
3400.00000	0.00000	0.00000	0.00000	64.61120
3600.00000	0.00000	0.00000	0.00000	61.13160
3800.00000	0.00000	0.00000	0.00000	60.27660
4000.00000	0.00000	0.00000	0.00000	60.06720


TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

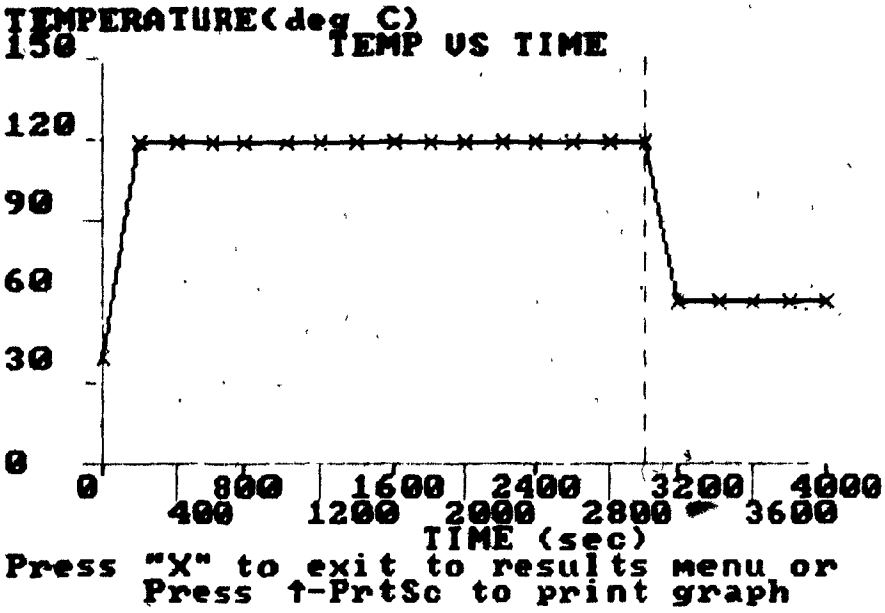
RESULTS

TIME (sec)	X (m)	Y (m)	Z (m)	TEMP (deg C)
0.00000	0.03250	0.02375	0.00375	40.00000
200.00000	0.03250	0.02375	0.00375	102.35700
400.00000	0.03250	0.02375	0.00375	115.82400
600.00000	0.03250	0.02375	0.00375	119.04200
800.00000	0.03250	0.02375	0.00375	119.78500
1000.00000	0.03250	0.02375	0.00375	119.95100
1200.00000	0.03250	0.02375	0.00375	119.98800
1400.00000	0.03250	0.02375	0.00375	119.99700
1600.00000	0.03250	0.02375	0.00375	119.99900
1800.00000	0.03250	0.02375	0.00375	119.99900
2000.00000	0.03250	0.02375	0.00375	120.00000
2200.00000	0.03250	0.02375	0.00375	120.00000
2400.00000	0.03250	0.02375	0.00375	120.00000
2600.00000	0.03250	0.02375	0.00375	120.00000
2800.00000	0.03250	0.02375	0.00375	120.00000
3000.00000	0.03250	0.02375	0.00375	120.00000
3200.00000	0.03250	0.02375	0.00375	73.23175
3400.00000	0.03250	0.02375	0.00375	63.13130
3600.00000	0.03250	0.02375	0.00375	60.71793
3800.00000	0.03250	0.02375	0.00375	60.16257
4000.00000	0.03250	0.02375	0.00375	60.03671


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

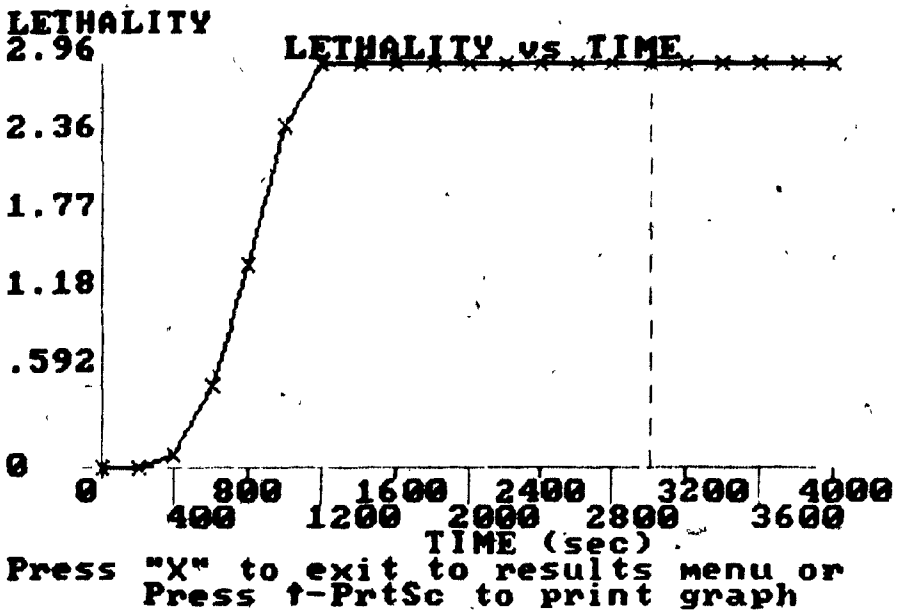
RESULTS

TIME (sec)	X (m)	Y (m)	Z (m)	TEMP (deg C)
0.00000	0.06500	0.04750	0.00750	40.00000
200.00000	0.06500	0.04750	0.00750	120.00000
400.00000	0.06500	0.04750	0.00750	120.00000
600.00000	0.06500	0.04750	0.00750	120.00000
800.00000	0.06500	0.04750	0.00750	120.00000
1000.00000	0.06500	0.04750	0.00750	120.00000
1200.00000	0.06500	0.04750	0.00750	120.00000
1400.00000	0.06500	0.04750	0.00750	120.00000
1600.00000	0.06500	0.04750	0.00750	120.00000
1800.00000	0.06500	0.04750	0.00750	120.00000
2000.00000	0.06500	0.04750	0.00750	120.00000
2200.00000	0.06500	0.04750	0.00750	120.00000
2400.00000	0.06500	0.04750	0.00750	120.00000
2600.00000	0.06500	0.04750	0.00750	120.00000
2800.00000	0.06500	0.04750	0.00750	120.00000
3000.00000	0.06500	0.04750	0.00750	120.00000
3200.00000	0.06500	0.04750	0.00750	60.00000
3400.00000	0.06500	0.04750	0.00750	60.00000
3600.00000	0.06500	0.04750	0.00750	60.00000
3800.00000	0.06500	0.04750	0.00750	60.00000
4000.00000	0.06500	0.04750	0.00750	60.00000


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

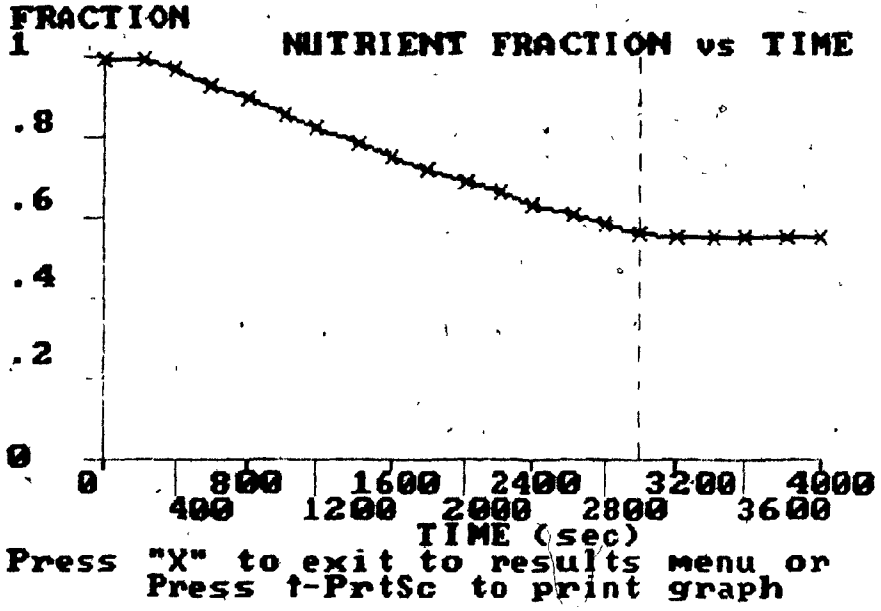
RESULTS

TIME (sec)	LETHALITY
0.00	0.00E+00
200.00	0.39E-04
400.00	0.95E-01
600.00	0.59E+00
800.00	0.15E+01
1000.00	0.25E+01
1200.00	0.30E+01
1400.00	0.30E+01
1600.00	0.30E+01
1800.00	0.30E+01
2000.00	0.30E+01
2200.00	0.30E+01
2400.00	0.30E+01
2600.00	0.30E+01
2800.00	0.30E+01
3000.00	0.30E+01
3200.00	0.30E+01
3400.00	0.30E+01
3600.00	0.30E+01
3800.00	0.30E+01
4000.00	0.30E+01


LETHALITY
2.96
LETHALITY vs TIME
2.36
1.77
1.18
.592
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	0.10E+01
200.00	0.10E+01
400.00	0.97E+00
600.00	0.94E+00
800.00	0.90E+00
1000.00	0.86E+00
1200.00	0.83E+00
1400.00	0.79E+00
1600.00	0.76E+00
1800.00	0.73E+00
2000.00	0.70E+00
2200.00	0.67E+00
2400.00	0.64E+00
2600.00	0.61E+00
2800.00	0.59E+00
3000.00	0.56E+00
3200.00	0.56E+00
3400.00	0.56E+00
3600.00	0.56E+00
3800.00	0.56E+00
4000.00	0.56E+00


FRACTION
NUTRIENT FRACTION vs TIME
1
.8
.6
.4
.2
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

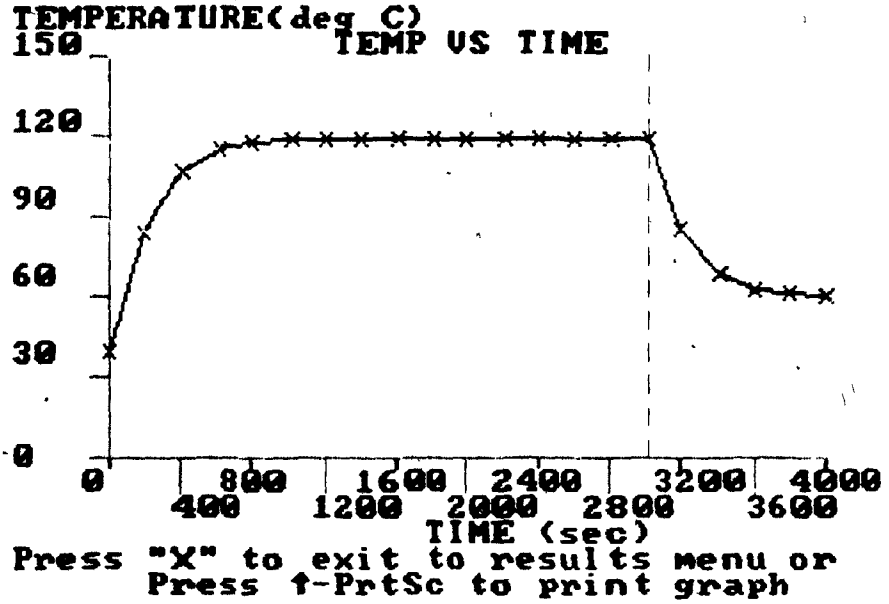
Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	4000
Time When Cooling Starts(sec)	3000
Number of Time Increments	21
Half X-Length(m)	.065
Half Y-Length(m)	.0475
Half Z-Length(m)	.0075
Number of X-values	5
Number of Y-values	5
Number of Z-values	5
Thermal Diffusivity(m**2/sec)	1.6E-07
Food Density (kg/m**3)	1500
Food Specific Heat (kJ/kg-degC	4000
H(W/m**2-degC)	800

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

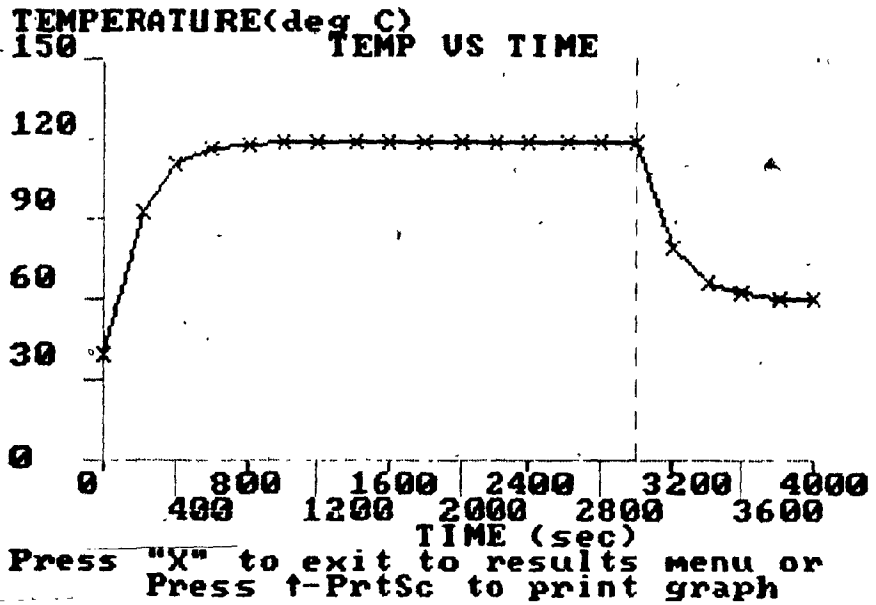
RESULTS

TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C)
0.00000	0.00000	0.00000	0.00000	40.00000
200.00000	0.00000	0.00000	0.00000	84.94020
400.00000	0.00000	0.00000	0.00000	107.70100
600.00000	0.00000	0.00000	0.00000	115.68900
800.00000	0.00000	0.00000	0.00000	118.49300
1000.00000	0.00000	0.00000	0.00000	119.47600
1200.00000	0.00000	0.00000	0.00000	119.81800
1400.00000	0.00000	0.00000	0.00000	119.93700
1600.00000	0.00000	0.00000	0.00000	119.97800
1800.00000	0.00000	0.00000	0.00000	119.99200
2000.00000	0.00000	0.00000	0.00000	119.99700
2200.00000	0.00000	0.00000	0.00000	119.99900
2400.00000	0.00000	0.00000	0.00000	119.99900
2600.00000	0.00000	0.00000	0.00000	119.99900
2800.00000	0.00000	0.00000	0.00000	120.00000
3000.00000	0.00000	0.00000	0.00000	120.00000
3200.00000	0.00000	0.00000	0.00000	86.29480
3400.00000	0.00000	0.00000	0.00000	69.22406
3600.00000	0.00000	0.00000	0.00000	63.23282
3800.00000	0.00000	0.00000	0.00000	61.12956
4000.00000	0.00000	0.00000	0.00000	60.39275


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

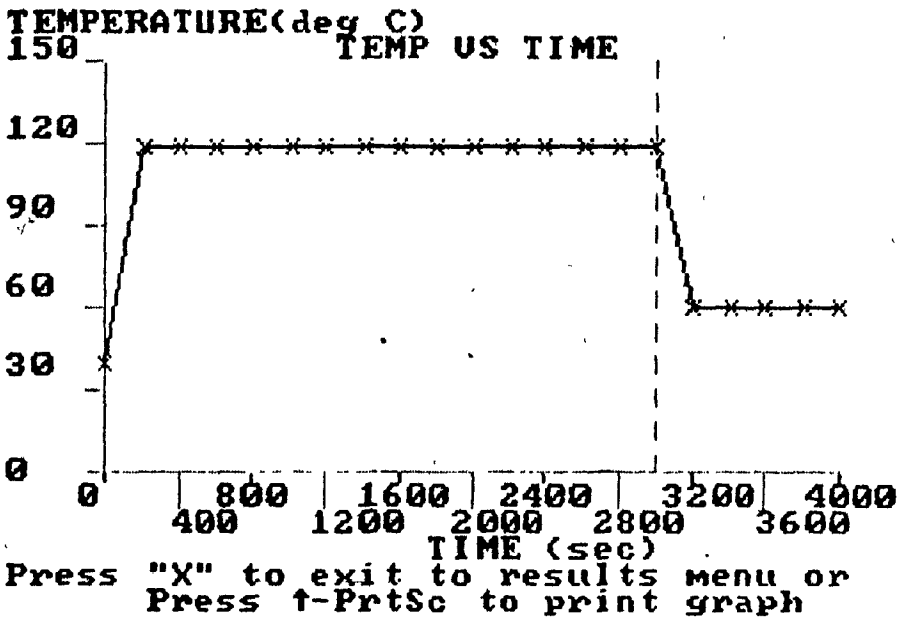
RESULTS

TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C)
0.00000	0.03250	0.02375	0.00375	40.00000
200.00000	0.03250	0.02375	0.00375	92.75830
400.00000	0.03250	0.02375	0.00375	110.72300
600.00000	0.03250	0.02375	0.00375	116.93300
800.00000	0.03250	0.02375	0.00375	119.00000
1000.00000	0.03250	0.02375	0.00375	119.67500
1200.00000	0.03250	0.02375	0.00375	119.89400
1400.00000	0.03250	0.02375	0.00375	119.96500
1600.00000	0.03250	0.02375	0.00375	119.98800
1800.00000	0.03250	0.02375	0.00375	119.99600
2000.00000	0.03250	0.02375	0.00375	119.99800
2200.00000	0.03250	0.02375	0.00375	119.99900
2400.00000	0.03250	0.02375	0.00375	119.99900
2600.00000	0.03250	0.02375	0.00375	120.00000
2800.00000	0.03250	0.02375	0.00375	120.00000
3000.00000	0.03250	0.02375	0.00375	120.00000
3200.00000	0.03250	0.02375	0.00375	80.43127
3400.00000	0.03250	0.02375	0.00375	66.95761
3600.00000	0.03250	0.02375	0.00375	62.29999
3800.00000	0.03250	0.02375	0.00375	60.74999
4000.00000	0.03250	0.02375	0.00375	60.24355


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

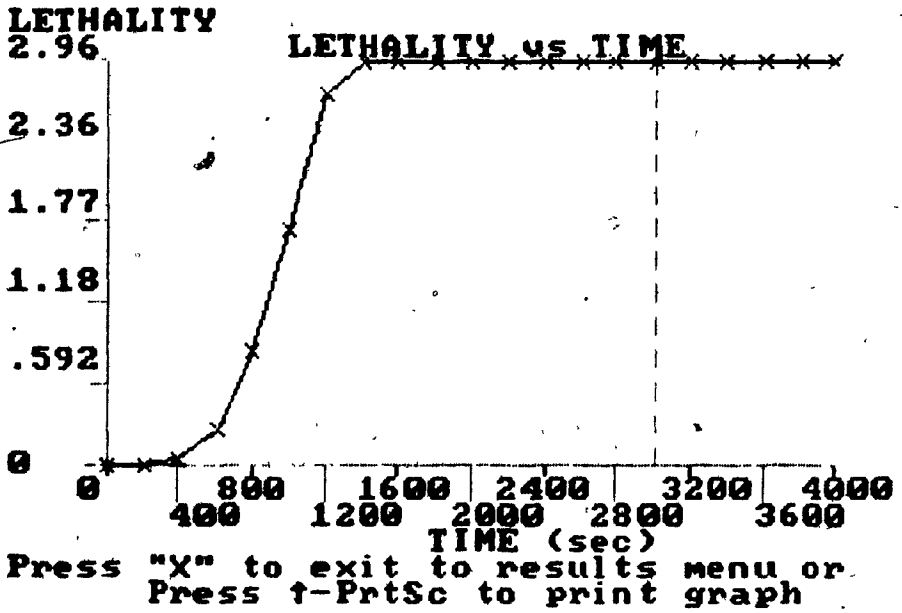
RESULTS

TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C
0.00000	0.06500	0.04750	0.00750	40.00000
200.00000	0.06500	0.04750	0.00750	119.89700
400.00000	0.06500	0.04750	0.00750	119.98100
600.00000	0.06500	0.04750	0.00750	119.99500
800.00000	0.06500	0.04750	0.00750	119.99800
1000.00000	0.06500	0.04750	0.00750	119.99900
1200.00000	0.06500	0.04750	0.00750	119.99900
1400.00000	0.06500	0.04750	0.00750	120.00000
1600.00000	0.06500	0.04750	0.00750	120.00000
1800.00000	0.06500	0.04750	0.00750	120.00000
2000.00000	0.06500	0.04750	0.00750	120.00000
2200.00000	0.06500	0.04750	0.00750	120.00000
2400.00000	0.06500	0.04750	0.00750	120.00000
2600.00000	0.06500	0.04750	0.00750	120.00000
2800.00000	0.06500	0.04750	0.00750	120.00000
3000.00000	0.06500	0.04750	0.00750	120.00000
3200.00000	0.06500	0.04750	0.00750	60.07660
3400.00000	0.06500	0.04750	0.00750	60.01370
3600.00000	0.06500	0.04750	0.00750	60.00324
3800.00000	0.06500	0.04750	0.00750	60.00085
4000.00000	0.06500	0.04750	0.00750	60.00024


TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
800
1600
2400
3200
4000
400
1200
2000
2800
3600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

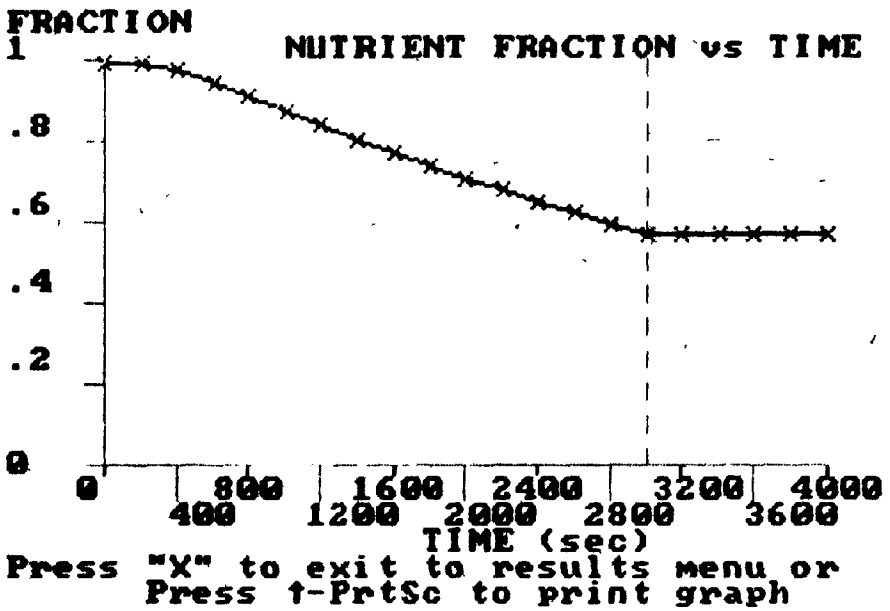
RESULTS

TIME (sec)	LETHALITY
0.00	0.00E+00
200.00	0.19E-04
400.00	0.45E-01
600.00	0.26E+00
800.00	0.85E+00
1000.00	0.17E+01
1200.00	0.27E+01
1400.00	0.30E+01
1600.00	0.30E+01
1800.00	0.30E+01
2000.00	0.30E+01
2200.00	0.30E+01
2400.00	0.30E+01
2600.00	0.30E+01
2800.00	0.30E+01
3000.00	0.30E+01
3200.00	0.30E+01
3400.00	0.30E+01
3600.00	0.30E+01
3800.00	0.30E+01
4000.00	0.30E+01


LETHALITY
2.96
LETHALITY vs TIME
2.36
1.77
1.18
.592
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	0.10E+01
200.00	0.10E+01
400.00	0.98E+00
600.00	0.95E+00
800.00	0.92E+00
1000.00	0.88E+00
1200.00	0.84E+00
1400.00	0.81E+00
1600.00	0.78E+00
1800.00	0.74E+00
2000.00	0.71E+00
2200.00	0.68E+00
2400.00	0.65E+00
2600.00	0.63E+00
2800.00	0.60E+00
3000.00	0.57E+00
3200.00	0.57E+00
3400.00	0.57E+00
3600.00	0.57E+00
3800.00	0.57E+00
4000.00	0.57E+00


FRACTION
NUTRIENT FRACTION vs TIME
1
.8
.6
.4
.2
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

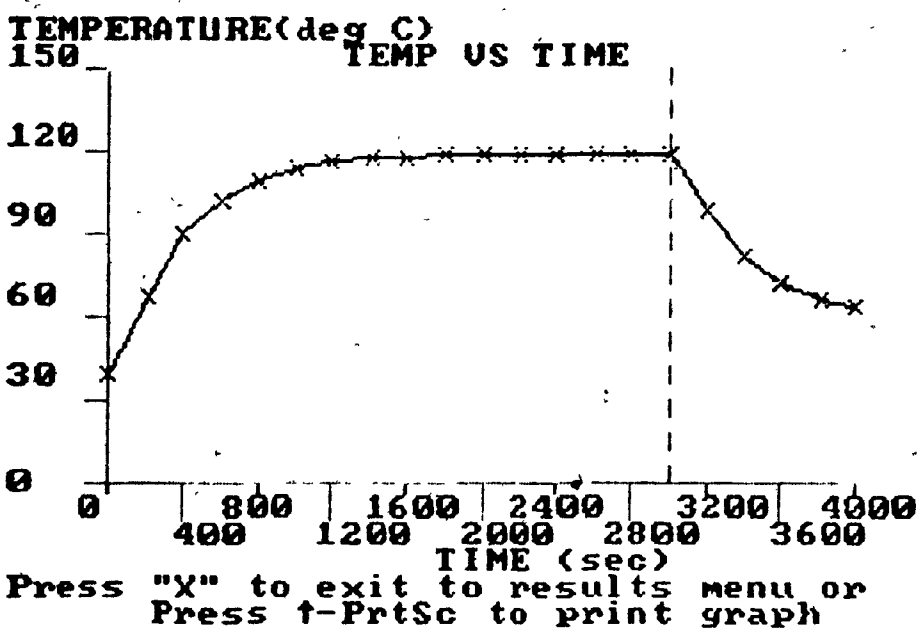
Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	4000
Time When Cooling Starts(sec)	3000
Number of Time Increments	21
Half X-Length(m)	.065
Half Y-Length(m)	.0475
Half Z-Length(m)	.0075
Number of X-values	5
Number of Y-values	5
Number of Z-values	5
Thermal Diffusivity(m**2/sec)	1.6E-07
Food Density (kg/m**3)	1500
Food Specific Heat (kJ/kg-degC	4000
H(W/m**2-degC)	200

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

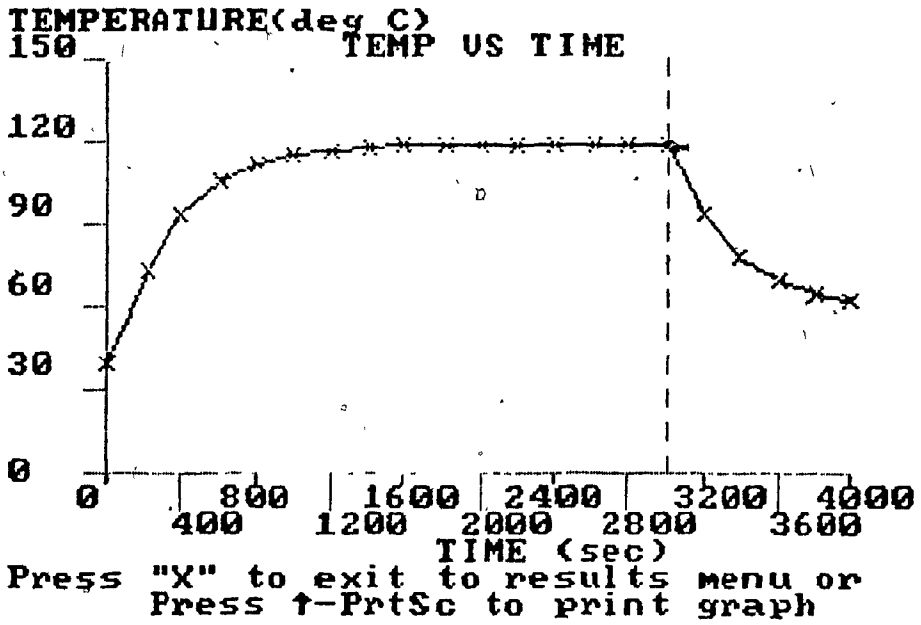
RESULTS

TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C)
0.00000	0.00000	0.00000	0.00000	40.00000
200.00000	0.00000	0.00000	0.00000	67.65970
400.00000	0.00000	0.00000	0.00000	90.39530
600.00000	0.00000	0.00000	0.00000	103.26700
800.00000	0.00000	0.00000	0.00000	110.56100
1000.00000	0.00000	0.00000	0.00000	114.69400
1200.00000	0.00000	0.00000	0.00000	117.02900
1400.00000	0.00000	0.00000	0.00000	118.34400
1600.00000	0.00000	0.00000	0.00000	119.08100
1800.00000	0.00000	0.00000	0.00000	119.49100
2000.00000	0.00000	0.00000	0.00000	119.71900
2200.00000	0.00000	0.00000	0.00000	119.84600
2400.00000	0.00000	0.00000	0.00000	119.91500
2600.00000	0.00000	0.00000	0.00000	119.95300
2800.00000	0.00000	0.00000	0.00000	119.97480
3000.00000	0.00000	0.00000	0.00000	119.98621
3200.00000	0.00000	0.00000	0.00000	99.24765
3400.00000	0.00000	0.00000	0.00000	82.19940
3600.00000	0.00000	0.00000	0.00000	72.54746
3800.00000	0.00000	0.00000	0.00000	67.07797
4000.00000	0.00000	0.00000	0.00000	63.97879


TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

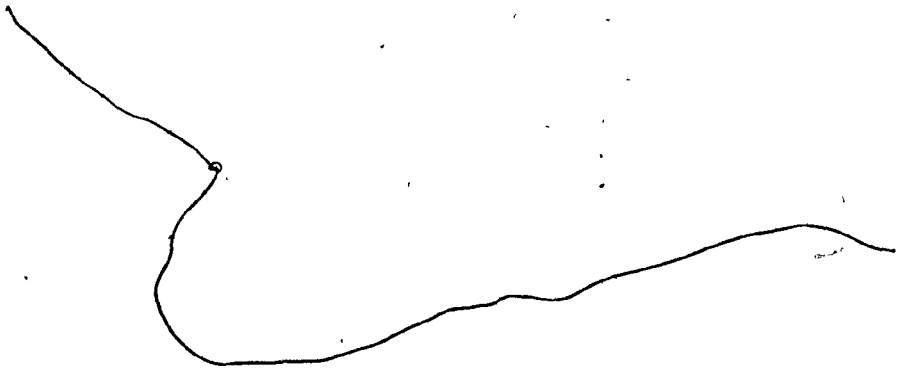
RESULTS

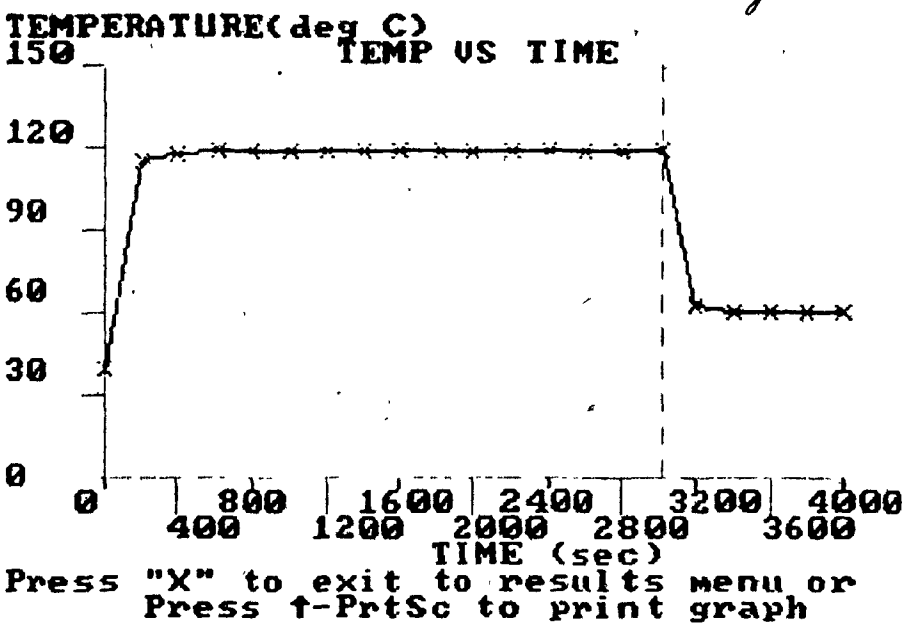
TIME(sec)	X(m)	Y(m)	Z(m)	TEMP(deg C
0.00000	0.03250	0.02375	0.00375	40.00000
200.00000	0.03250	0.02375	0.00375	74.11200
400.00000	0.03250	0.02375	0.00375	94.50820
600.00000	0.03250	0.02375	0.00375	106.15400
800.00000	0.03250	0.02375	0.00375	112.58000
1000.00000	0.03250	0.02375	0.00375	116.04700
1200.00000	0.03250	0.02375	0.00375	117.89800
1400.00000	0.03250	0.02375	0.00375	118.88200
1600.00000	0.03250	0.02375	0.00375	119.40300
1800.00000	0.03250	0.02375	0.00375	119.68300
2000.00000	0.03250	0.02375	0.00375	119.83700
2200.00000	0.03250	0.02375	0.00375	119.90900
2400.00000	0.03250	0.02375	0.00375	119.95100
2600.00000	0.03250	0.02375	0.00375	119.97400
2800.00000	0.03250	0.02375	0.00375	119.98610
3000.00000	0.03250	0.02375	0.00375	119.99249
3200.00000	0.03250	0.02375	0.00375	94.41193
3400.00000	0.03250	0.02375	0.00375	79.11661
3600.00000	0.03250	0.02375	0.00375	70.38260
3800.00000	0.03250	0.02375	0.00375	65.56430
4000.00000	0.03250	0.02375	0.00375	62.96402


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

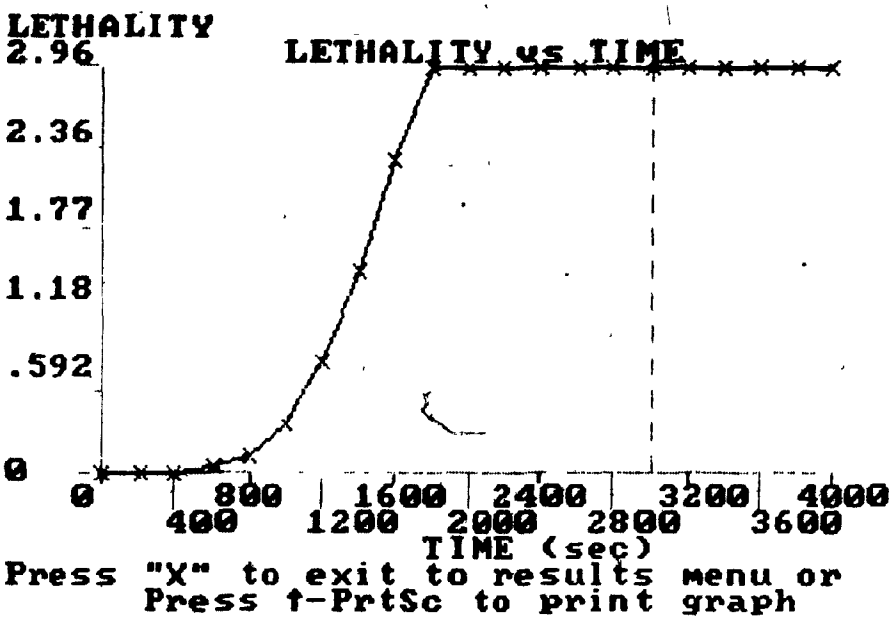
TIME (sec)	X (m)	Y (m)	Z (m)	TEMP (deg C)
0.00000	0.06500	0.04750	0.00750	40.00000
200.00000	0.06500	0.04750	0.00750	115.84800
400.00000	0.06500	0.04750	0.00750	118.59800
600.00000	0.06500	0.04750	0.00750	119.43000
800.00000	0.06500	0.04750	0.00750	119.74800
1000.00000	0.06500	0.04750	0.00750	119.88300
1200.00000	0.06500	0.04750	0.00750	119.94300
1400.00000	0.06500	0.04750	0.00750	119.97200
1600.00000	0.06500	0.04750	0.00750	119.98600
1800.00000	0.06500	0.04750	0.00750	119.99300
2000.00000	0.06500	0.04750	0.00750	119.99600
2200.00000	0.06500	0.04750	0.00750	119.99800
2400.00000	0.06500	0.04750	0.00750	119.99900
2600.00000	0.06500	0.04750	0.00750	119.99900
2800.00000	0.06500	0.04750	0.00750	119.99970
3000.00000	0.06500	0.04750	0.00750	119.99991
3200.00000	0.06500	0.04750	0.00750	63.11328
3400.00000	0.06500	0.04750	0.00750	61.05148
3600.00000	0.06500	0.04750	0.00750	60.42694
3800.00000	0.06500	0.04750	0.00750	60.18872
4000.00000	0.06500	0.04750	0.00750	60.08768




TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

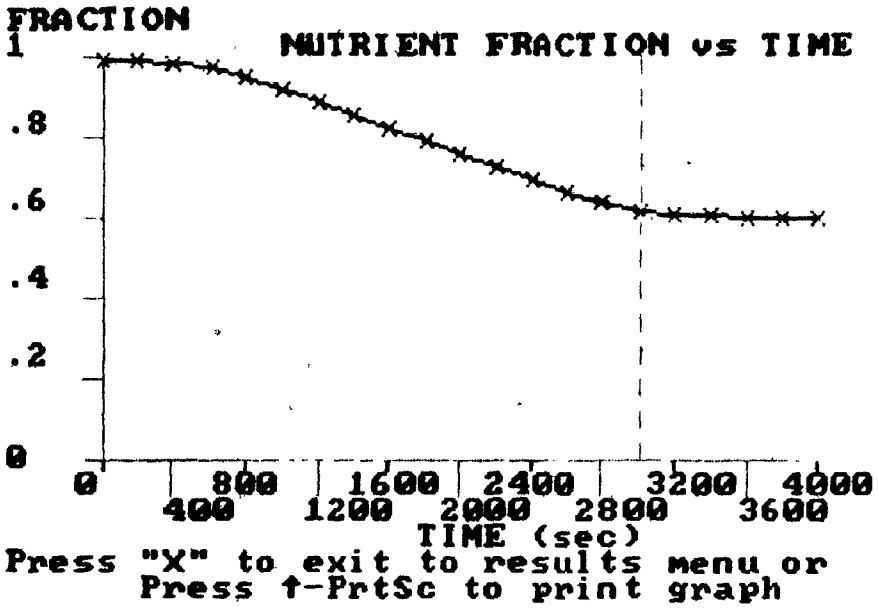
RESULTS

TIME (sec)	LETHALITY
0.00	0.00E+00
200.00	0.29E-05
400.00	0.56E-02
600.00	0.36E-01
800.00	0.13E+00
1000.00	0.37E+00
1200.00	0.81E+00
1400.00	0.15E+01
1600.00	0.23E+01
1800.00	0.30E+01
2000.00	0.30E+01
2200.00	0.30E+01
2400.00	0.30E+01
2600.00	0.30E+01
2800.00	0.30E+01
3000.00	0.30E+01
3200.00	0.30E+01
3400.00	0.30E+01
3600.00	0.30E+01
3800.00	0.30E+01
4000.00	0.30E+01


LETHALITY
2.96
LETHALITY vs TIME
2.36
1.77
1.18
.592
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	0.10E+01
200.00	0.10E+01
400.00	0.99E+00
600.00	0.98E+00
800.00	0.96E+00
1000.00	0.93E+00
1200.00	0.90E+00
1400.00	0.86E+00
1600.00	0.83E+00
1800.00	0.80E+00
2000.00	0.76E+00
2200.00	0.73E+00
2400.00	0.70E+00
2600.00	0.67E+00
2800.00	0.64E+00
3000.00	0.62E+00
3200.00	0.61E+00
3400.00	0.61E+00
3600.00	0.61E+00
3800.00	0.61E+00
4000.00	0.61E+00


FRACTION
NUTRIENT FRACTION vs TIME
1
.8
.6
.4
.2
0
0
400
800
1200
1600
2000
2400
2800
3200
3600
4000
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

APPENDIX 2.3

Sample Runs from the Execution of MAIND

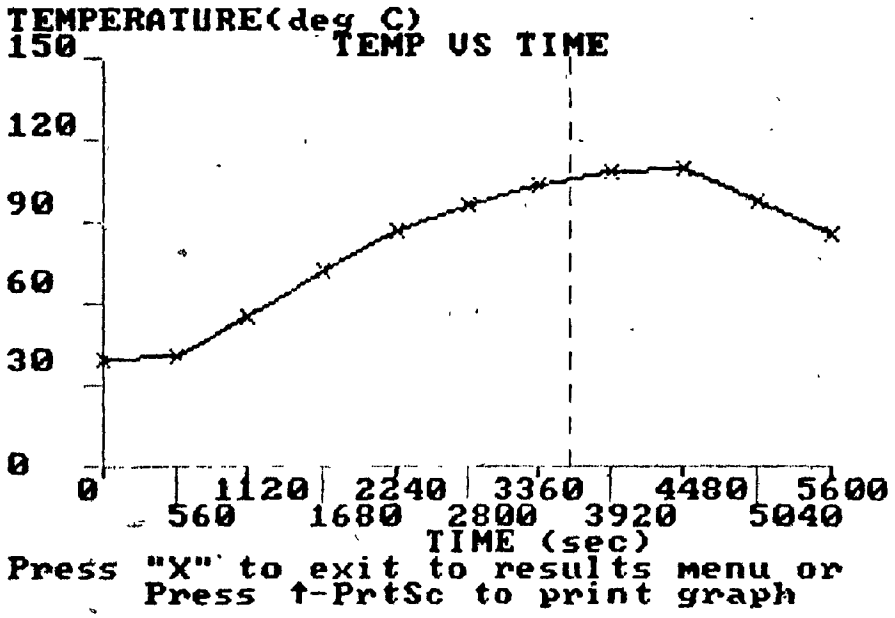
Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	5600
Time When Cooling Starts(sec)	3600
Number of Time Increments	11
Can Radius (m)	.0405
Half Z-Length(m)	.0556
Number of R-values	5
Number of Z-values	3
Thermal Diffusivity(m**2/sec)	1.6E-07

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

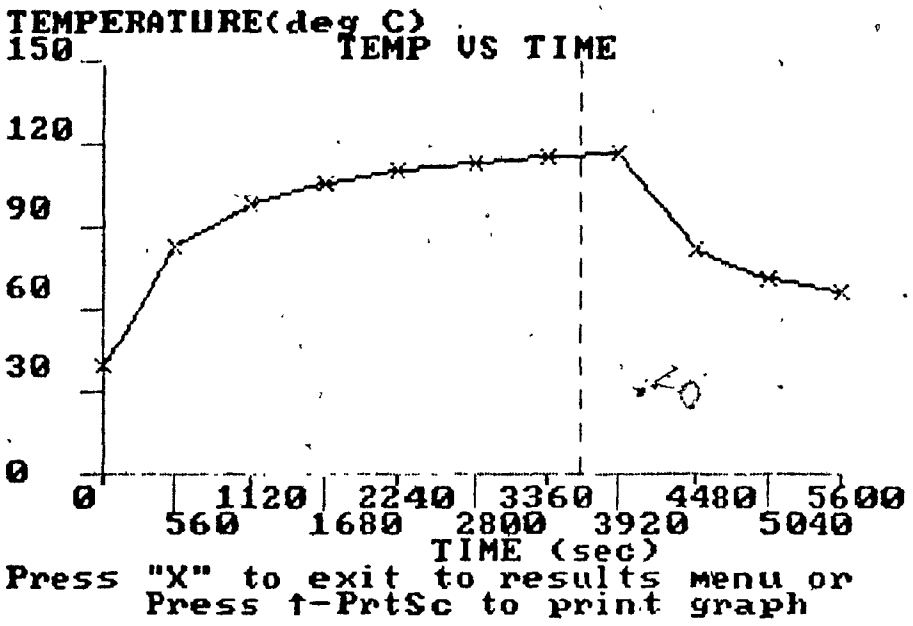
RESULTS

TIME (s)	RADIUS (m)	Z-VALUES (m)	TEMPERATURE (deg C)
0.00000	0.00000	0.00000	40.00000
560.00000	0.00000	0.00000	41.57560
1120.00000	0.00000	0.00000	55.33290
1680.00000	0.00000	0.00000	72.51980
2240.00000	0.00000	0.00000	86.62110
2800.00000	0.00000	0.00000	96.94940
3360.00000	0.00000	0.00000	104.21200
3920.00000	0.00000	0.00000	109.23000
4480.00000	0.00000	0.00000	110.22300
5040.00000	0.00000	0.00000	98.48370
5600.00000	0.00000	0.00000	85.82070


TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

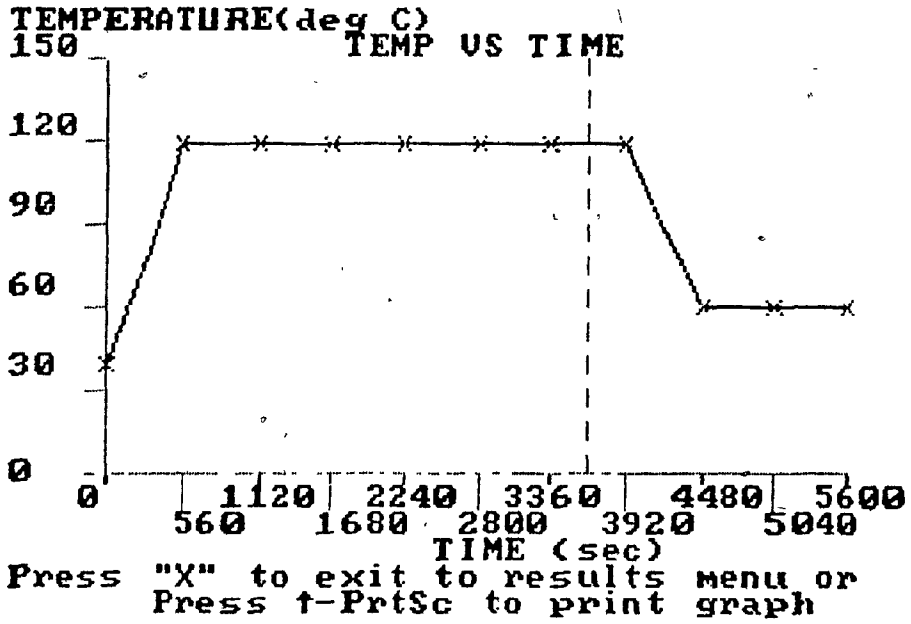
RESULTS

TIME(s)	RADIUS(m)	Z-VALUES(m)	TEMPERATURE(deg C)
0.00000	0.03038	0.02780	40.00000
560.00000	0.03038	0.02780	83.37570
1120.00000	0.03038	0.02780	99.23780
1680.00000	0.03038	0.02780	106.91900
2240.00000	0.03038	0.02780	111.41500
2800.00000	0.03038	0.02780	114.26900
3360.00000	0.03038	0.02780	116.14400
3920.00000	0.03038	0.02780	117.39500
4480.00000	0.03038	0.02780	82.54940
5040.00000	0.03038	0.02780	71.76540
5600.00000	0.03038	0.02780	66.87860


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

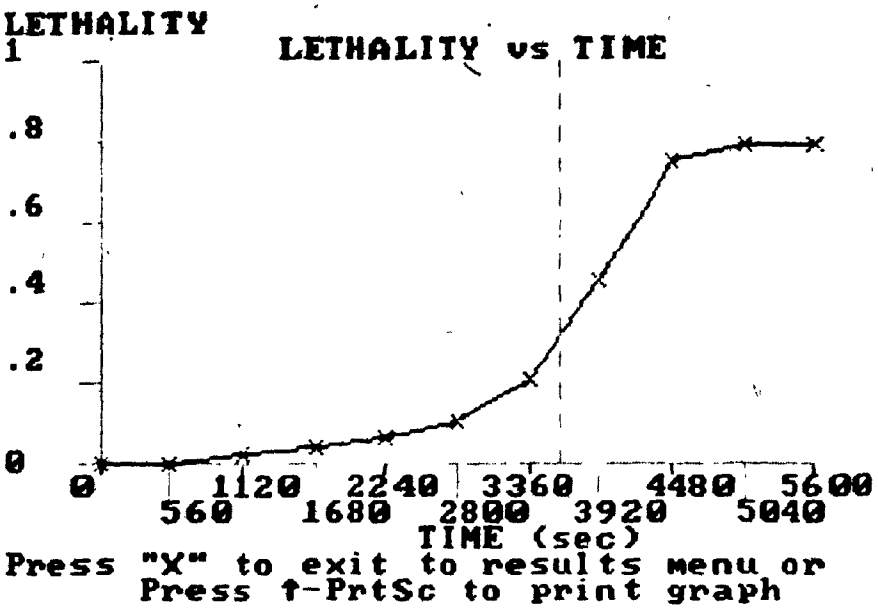
RESULTS

TIME(s)	RADIUS(m)	Z-VALUES(m)	TEMPERATURE(deg C)
0.00000	0.04050	0.05560	40.00000
560.00000	0.04050	0.05560	120.00000
1120.00000	0.04050	0.05560	120.00000
1680.00000	0.04050	0.05560	120.00000
2240.00000	0.04050	0.05560	120.00000
2800.00000	0.04050	0.05560	120.00000
3360.00000	0.04050	0.05560	120.00000
3920.00000	0.04050	0.05560	120.00000
4480.00000	0.04050	0.05560	60.00000
5040.00000	0.04050	0.05560	60.00000
5600.00000	0.04050	0.05560	60.00000


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

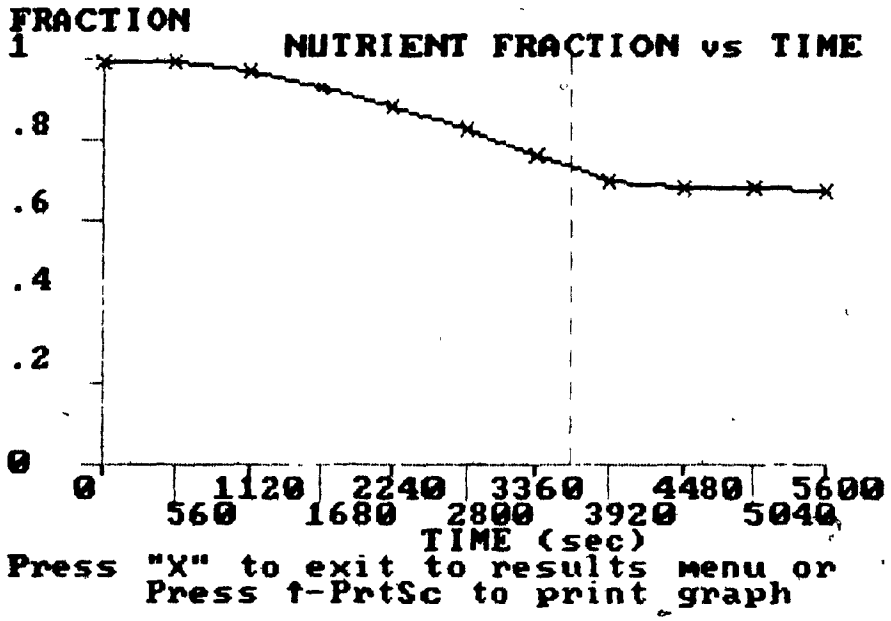
RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
560.00	34.353E-06
1120.00	26.967E-03
1680.00	43.016E-03
2240.00	67.685E-03
2800.00	10.563E-02
3360.00	20.699E-02
3920.00	45.988E-02
4480.00	75.653E-02
5040.00	79.563E-02
5600.00	79.[illegible]E-02


LETHALITY
1
LETHALITY vs TIME
.8
.6
.4
.2
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	10.000E-01
560.00	99.860E-02
1120.00	97.248E-02
1680.00	93.387E-02
2240.00	88.504E-02
2800.00	82.827E-02
3360.00	76.620E-02
3920.00	70.166E-02
4480.00	68.603E-02
5040.00	68.253E-02
5600.00	68.102E-02


FRACTION
1
NUTRIENT FRACTION vs TIME
.8
.6
.4
.2
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press f-PrtSc to print graph

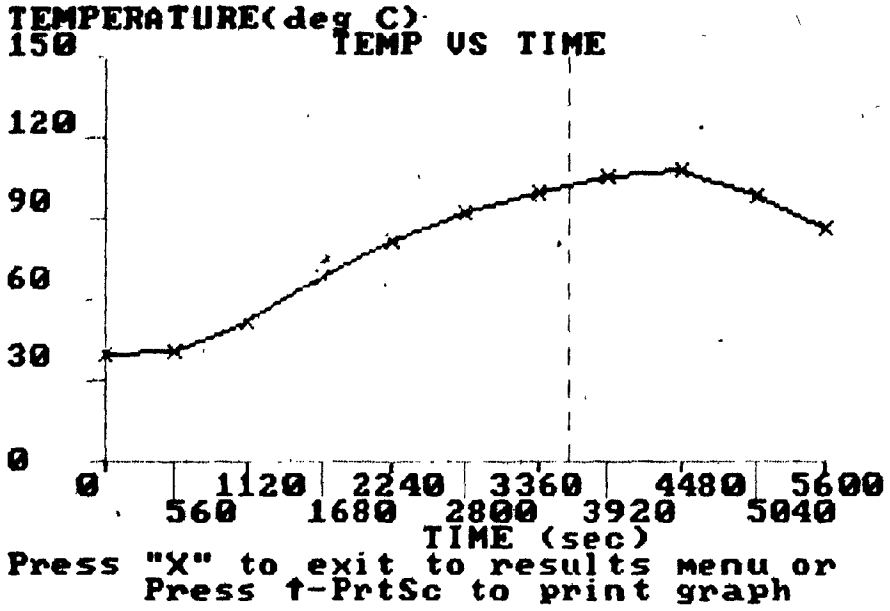
Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	5600
Time When Cooling Starts(sec)	3600
Number of Time Increments	21
Can Radius (m)	.0405
Half Z-Length(m)	.0556
Number of R-values	5
Number of Z-values	3
Thermal Diffusivity(m**2/sec)	1.6E-07
Food Density(kg/m**3)	1500
Food Specific Heat(kJ/kg-degC)	4000
H(W/m**2-degC)	500

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

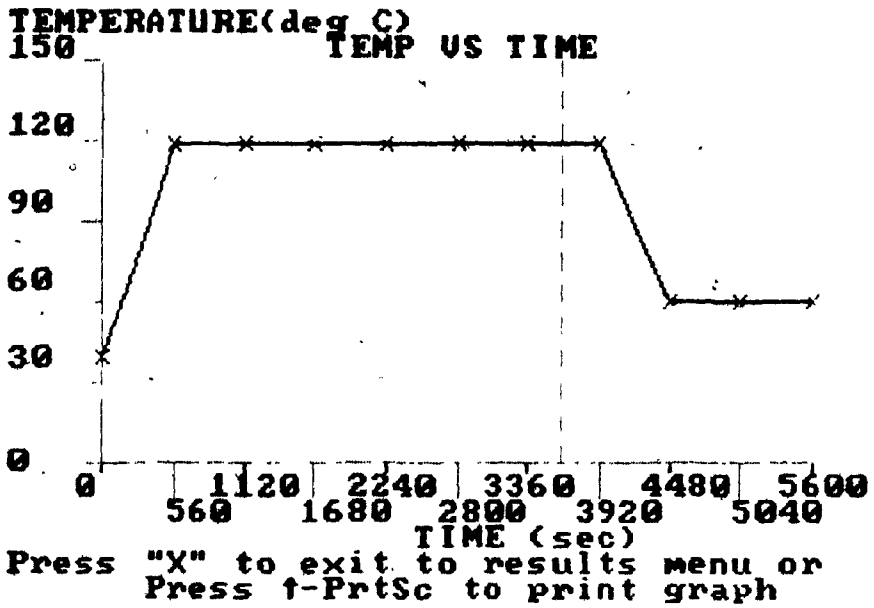
RESULTS

TIME (s)	RADIUS (m)	Z-VALUES (m)	TEMPERATURE (deg C)
0.00000	0.00000	0.00000	40.00000
560.00000	0.00000	0.00000	41.09100
1120.00000	0.00000	0.00000	52.56830
1680.00000	0.00000	0.00000	68.49170
2240.00000	0.00000	0.00000	82.41080
2800.00000	0.00000	0.00000	93.09920
3360.00000	0.00000	0.00000	100.92400
3920.00000	0.00000	0.00000	106.53600
4480.00000	0.00000	0.00000	108.72800
5040.00000	0.00000	0.00000	99.35890
5600.00000	0.00000	0.00000	87.69460


TEMPERATURE(deg C)
150
TEMP VS TIME
120
90
60
30
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

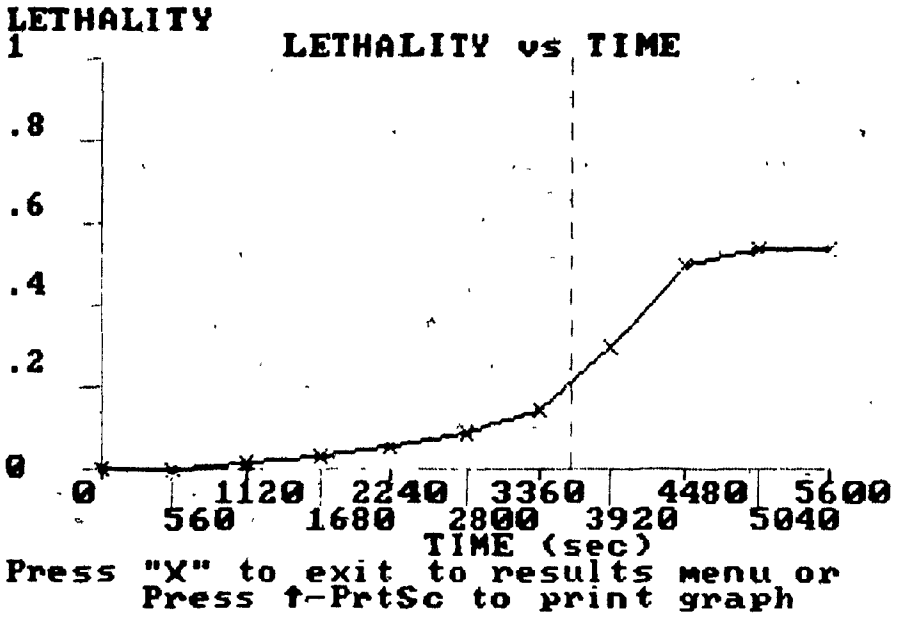
RESULTS

TIME(s)	RADIUS(m)	Z-VALUES(m)	TEMPERATURE(deg C)
0.00000	0.04050	0.05560	40.00000
560.00000	0.04050	0.05560	119.17500
1120.00000	0.04050	0.05560	119.63400
1680.00000	0.04050	0.05560	119.78700
2240.00000	0.04050	0.05560	119.86300
2800.00000	0.04050	0.05560	119.90900
3360.00000	0.04050	0.05560	119.93800
3920.00000	0.04050	0.05560	119.95700
4480.00000	0.04050	0.05560	60.47900
5040.00000	0.04050	0.05560	60.19850
5600.00000	0.04050	0.05560	60.10820


TEMPERATURE(deg C)
TEMP VS TIME
150
120
90
60
30
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press f-PrtSc to print graph

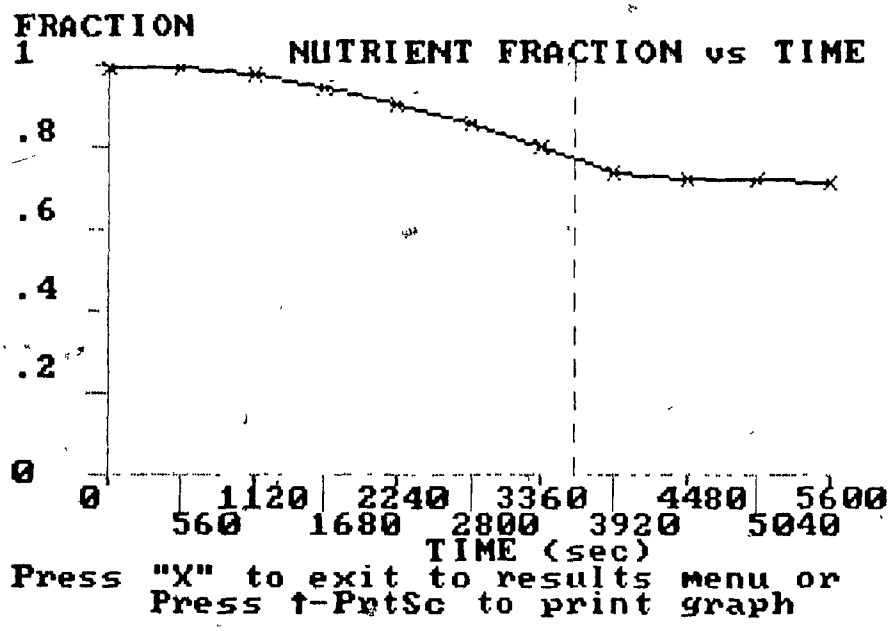
RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
560.00	18.493E-06
1120.00	16.796E-03
1680.00	35.609E-03
2240.00	54.259E-03
2800.00	85.271E-03
3360.00	14.470E-02
3920.00	29.623E-02
4480.00	50.306E-02
5040.00	54.203E-02
5600.00	54.429E-02


LETHALITY
1
LETHALITY vs TIME
.8
.6
.4
.2
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

RESULTS

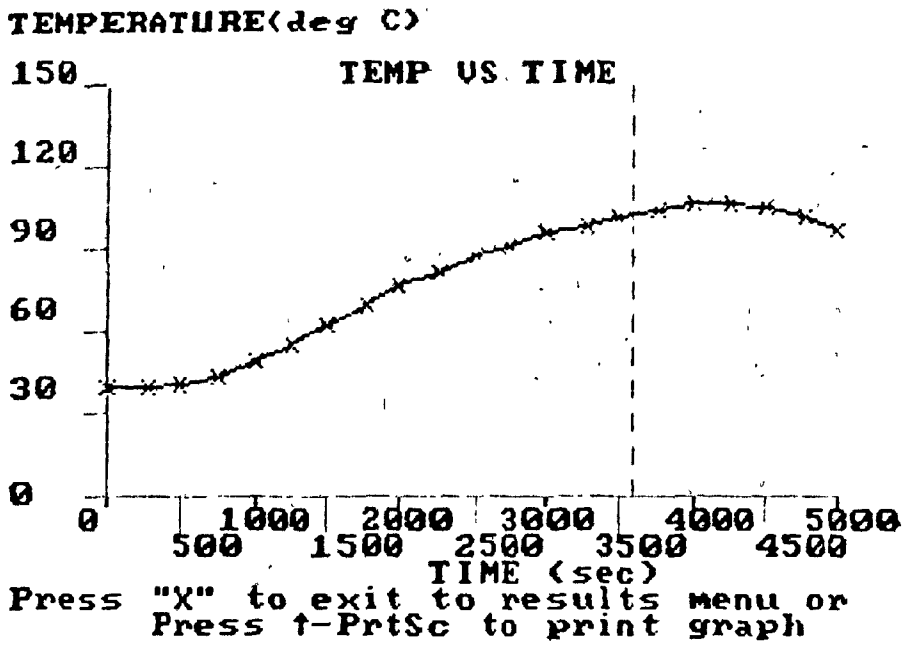
TIME (sec)	NUTRIENT FRACTION
0.00	10.000E-01
560.00	99.890E-02
1120.00	98.113E-02
1680.00	95.138E-02
2240.00	91.087E-02
2800.00	86.118E-02
3360.00	80.447E-02
3920.00	74.334E-02
4480.00	72.639E-02
5040.00	72.214E-02
5600.00	72.028E-02


FRACTION
1
NUTRIENT FRACTION vs TIME
.8
.6
.4
.2
0
0
560
1120
1680
2240
2800
3360
3920
4480
5040
5600
TIME (sec)
Press "X" to exit to results menu or
Press ↑-PrtSc to print graph

Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	5000
Time When Cooling Starts(sec)	3600
Number of Time Increments	21
Can Radius (m)	.0405
Half Z-Length(m)	.0556
Number of R-values	5
Number of Z-values	3
Thermal Diffusivity(m**2/sec)	1.6E-07
Food Density(kg/m**3)	1500
Food Specific Heat(kJ/kg-degC)	4000
H(W/m**2-degC)	500

Press "X" to exit to results menu or
Press ↑-PrtSc to print graph



Centre Temperature

RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
250.00	31.729E-07
500.00	37.735E-04
750.00	98.839E-04
1000.00	15.617E-03
1250.00	23.555E-03
1500.00	31.958E-03
1750.00	39.031E-03
2000.00	46.726E-03
2250.00	56.904E-03
2500.00	69.785E-03
2750.00	85.016E-03
3000.00	10.435E-02
3250.00	13.396E-02
3500.00	18.024E-02
3750.00	24.838E-02
4000.00	34.450E-02
4250.00	44.789E-02
4500.00	50.547E-02
4750.00	52.382E-02
5000.00	52.890E-02

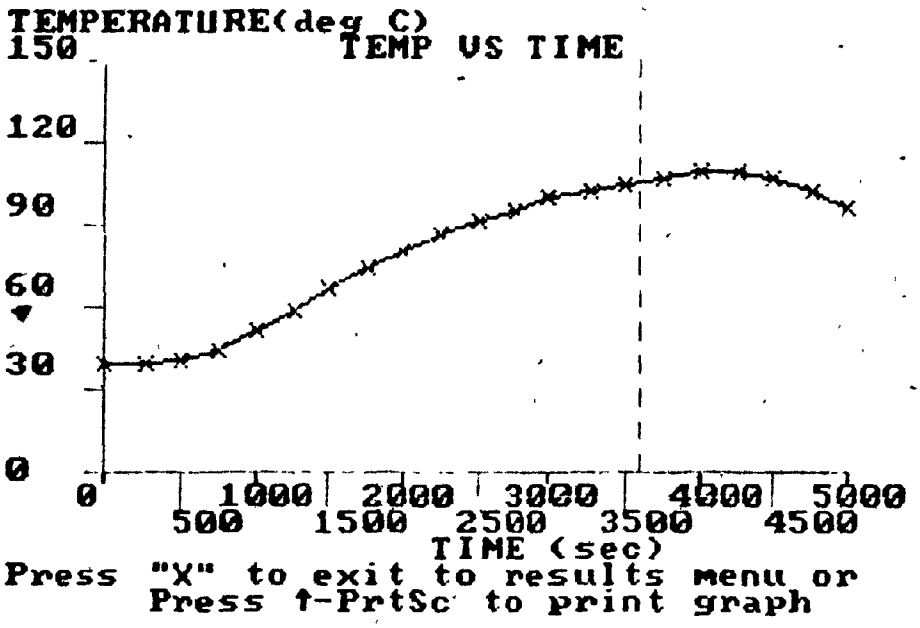
RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	10.000E-01
250.00	99.967E-02
500.00	99.591E-02
750.00	98.970E-02
1000.00	98.109E-02
1250.00	97.013E-02
1500.00	95.691E-02
1750.00	94.150E-02
2000.00	92.401E-02
2250.00	90.457E-02
2500.00	88.332E-02
2750.00	86.045E-02
3000.00	83.616E-02
3250.00	81.066E-02
3500.00	78.420E-02
3750.00	75.701E-02
4000.00	72.936E-02
4250.00	74.206E-02
4500.00	77.916E-02
4750.00	77.728E-02
5000.00	77.601E-02

Process Parameter Table

Initial Temperature(deg C)	40
Retort Temperature(deg C)	120
Water Temperature(deg C)	60
Total Processing Time(sec)	5000
Time When Cooling Starts(sec)	3600
Number of Time Increments	11
Can Radius (m)	.0405
Half Z-Length(m)	.0556
Number of R-values	5
Number of Z-values	3
Thermal Diffusivity(m**2/sec)	1.6E-07

Press "X" to exit to results menu or
Press ↑-Prt$c to print graph



Centre Temperatures

RESULTS

TIME (sec)	LETHALITY
0.00	0.000E+00
250.00	77.304E-07
500.00	97.449E-04
750.00	16.714E-03
1000.00	25.493E-03
1250.00	33.061E-03
1500.00	39.490E-03
1750.00	47.293E-03
2000.00	57.884E-03
2250.00	71.083E-03
2500.00	85.973E-03
2750.00	10.555E-02
3000.00	13.781E-02
3250.00	18.916E-02
3500.00	26.571E-02
3750.00	37.906E-02
4000.00	54.051E-02
4250.00	69.569E-02
4500.00	76.518E-02
4750.00	78.329E-02
5000.00	78.761E-02

RESULTS

TIME (sec)	NUTRIENT FRACTION
0.00	10.000E-01
250.00	99.954E-02
500.00	99.287E-02
750.00	98.324E-02
1000.00	97.092E-02
1250.00	95.612E-02
1500.00	93.903E-02
1750.00	91.983E-02
2000.00	89.868E-02
2250.00	87.579E-02
2500.00	85.134E-02
2750.00	82.556E-02
3000.00	79.871E-02
3250.00	77.102E-02
3500.00	74.274E-02
3750.00	71.414E-02
4000.00	70.447E-02
4250.00	70.033E-02
4500.00	69.784E-02
4750.00	69.629E-02
5000.00	69.527E-02

APPENDIX 2.4

Sample Runs from the Execution of MAINE

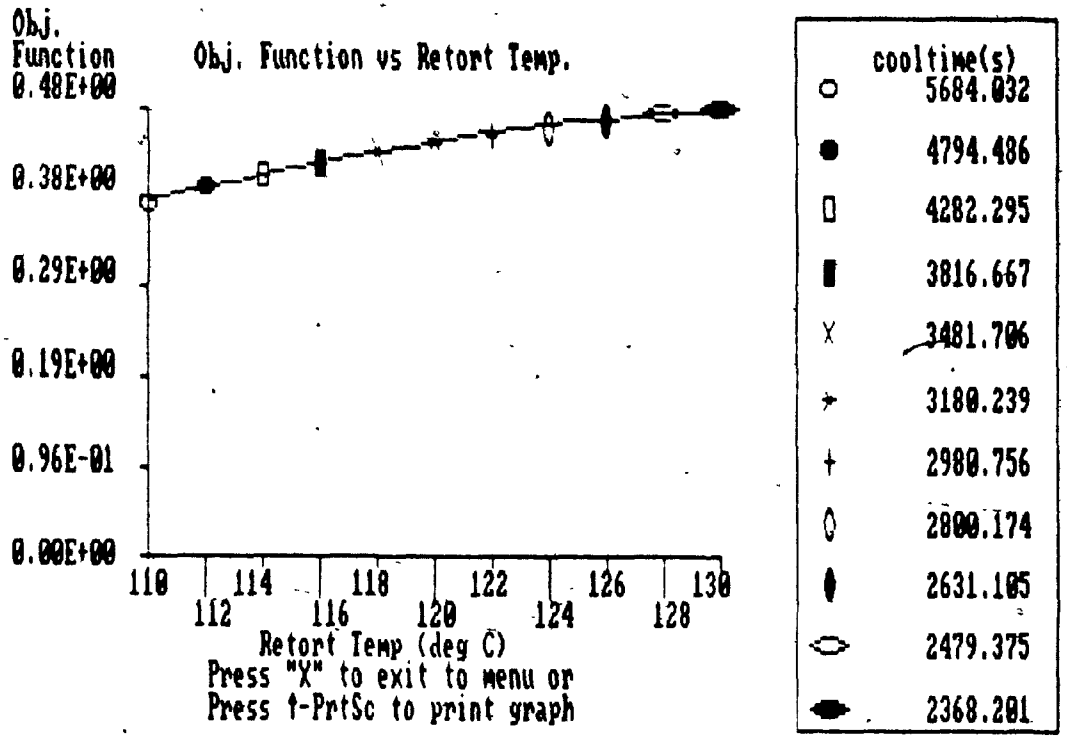
Process Parameter Table

Initial Temperature(deg C)	40
Water Temperature(deg C)	25
Slope of Heating Curve(sec)	2460
Slope of Cooling Curve(sec)	2460
Heating Curve Lag Factor	1.4
Cooling Curve Lag Factor	0
Total Processing Time(sec)	6666.667
F-Value of Microorganism(min)	2.45
z-Value of Microorganism(min)	10
F-Value of Nutrient #1(min)	1980
F-Value of Nutrient #2(min)	213.6
F-Value of Nutrient #3(min)	66
F-Value of Nutrient #4(min)	1200
z-Value of Nutrient #1(deg C)	27
z-Value of Nutrient #2(deg C)	45.4
z-Value of Nutrient #3(deg C)	31.1
z-Value of Nutrient #4(deg C)	28
Weighting Factor #1	.75
Weighting Factor #2	.8
Weighting Factor #3	.29
Weighting Factor #4	.55

Press "X" to exit to menu or
Press ↑-PrtSc to print graph

Results

Retort Temp(C)	Cooling starts at (s)	Objective Function value
110.000	5684.032	0.3783E+00
112.000	4794.486	0.3937E+00
114.000	4282.295	0.4079E+00
116.000	3816.667	0.4210E+00
118.000	3481.706	0.4329E+00
120.000	3180.239	0.4436E+00
122.000	2980.756	0.4531E+00
124.000	2800.174	0.4612E+00
126.000	2631.105	0.4681E+00
128.000	2479.375	0.4737E+00
130.000	2368.201	0.4778E+00
130.001	2368.201	0.4778E+00


Obj.
Function
Obj. Function vs Retort Temp.
0.48E+00
0.38E+00
0.29E+00
0.19E+00
0.96E-01
0.00E+00
110
112
114
116
118
120
122
124
126
128
130
Retort Temp (deg C)
Press "X" to exit to menu or
Press ↑-PrtSc to print graph
cooltime(s)
5684.032
4794.486
4282.295
3816.667
3481.706
3180.239
2980.756
2800.174
2631.105
2479.375
2368.281

Results

Retort Temp (C)	Cooling starts at (s)	Nutrient #1	Nutrient #2	Nutrient #3	Nutrie #4
130.00	2388.20	0.86E+00	0.91E-01	0.52E-02	0.77E+0

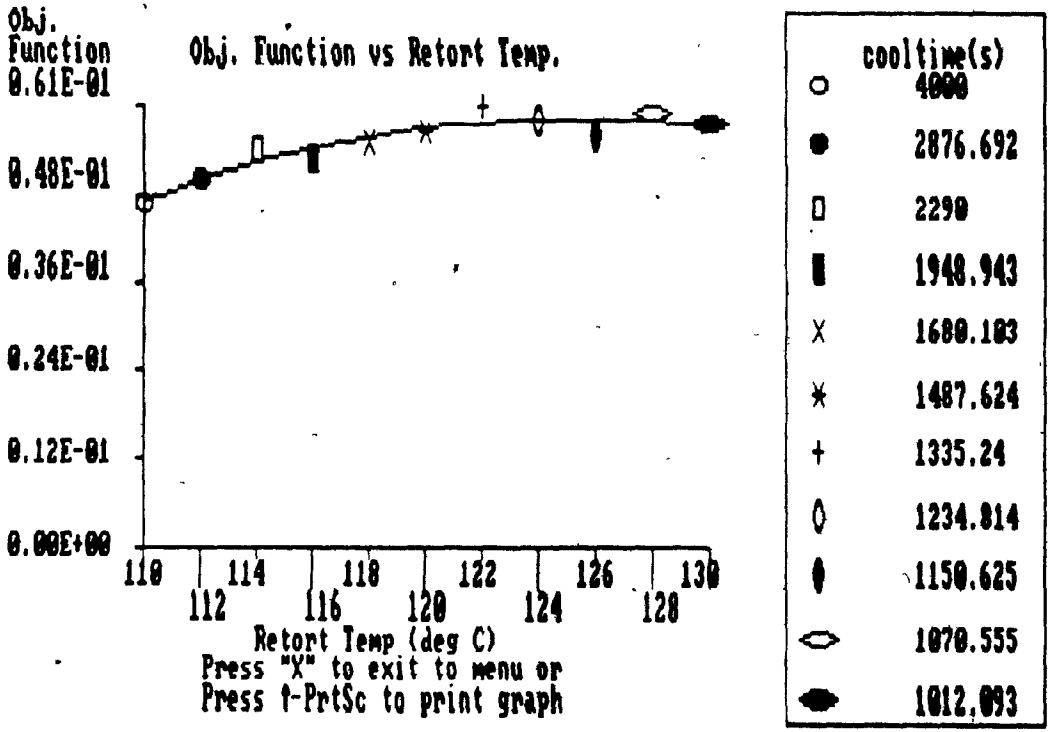
Process Parameter Table

Initial Temperature(deg C)	55
Water Temperature(deg C)	38
Slope of Heating Curve(sec)	1000
Slope of Cooling Curve(sec)	2000
Heating Curve Lag Factor	1.5
Cooling Curve Lag Factor	0
Total Processing Time(sec)	4000
F-Value of Microorganism(min)	2.5
z-Value of Microorganism(min)	10
F-Value of Nutrient #1(min)	45
F-Value of Nutrient #2(min)	35
F-Value of Nutrient #3(min)	26
F-Value of Nutrient #4(min)	66
z-Value of Nutrient #1(deg C)	45
z-Value of Nutrient #2(deg C)	29.8
z-Value of Nutrient #3(deg C)	49
z-Value of Nutrient #4(deg C)	11
Weighting Factor #1	.9
Weighting Factor #2	.76
Weighting Factor #3	.5
Weighting Factor #4	.81

Press "X" to exit to menu or
Press ↑-PrtSc to print graph.

Results

Retort Temp(C)	Cooling starts at (s)	Objective Function value
110.000	4000.000	0.4782E-01
112.000	2876.692	0.5036E-01
114.000	2290.000	0.5257E-01
116.000	1948.943	0.5445E-01
118.000	1680.103	0.5599E-01
120.000	1487.624	0.5718E-01
122.000	1335.240	0.5801E-01
124.000	1234.814	0.5849E-01
126.000	1150.625	0.5859E-01
128.000	1070.555	0.5832E-01
130.000	1012.093	0.5766E-01
125.551	1160.748	0.5860E-01


Obj.
Function
Obj. Function vs Retort Temp.
0.61E-01
0.48E-01
0.36E-01
0.24E-01
0.12E-01
0.00E+00
110
112
114
116
118
120
122
124
126
128
130
Retort Temp (deg C)
Press "X" to exit to menu or
Press f-PrtSc to print graph
cooltime(s)
4000
2876.692
2290
1948.943
1680.103
1487.624
1335.24
1234.814
1150.625
1070.555
1012.093

Results

Retort Temp	Cooling starts at	Nutrient #1	Nutrient #2	Nutrient #3	Nutrient #4
125.55	1160.	0.12E-1	0.12E-1	0.47E-07	0.22E+00

APPENDIX 3 -

Program Listings

APPENDIX 3.1

Program Listing of AUTOEXEC

```
***** AUTOEXEC *****
echo off
echo PACKAGE STARTUP MESSAGES  (DISREGARD)
superspl lpt1:/m=24/b
superdrv c:/m=450/2
for %%a in (alpha alphab alphac alphae alpha1) do copy %%a c:
for %%b in (b:alphad) do copy %%b c:
graphics
echo on
start
```

APPENDIX 3.2

Program Listing of START

```
10 '
20 '      START program
30 '
40 '
50 ' Writing system code into STORE1 for MAIN
60 DEFINT I-N
70 OPEN "b:store1" FOR OUTPUT AS 1
80 NCODE=0
90 PRINT#1,NCODE
100 CLOSE 1
110 '
120 ' Loading the Macdonald Logo screen (MACLOGO.SCR)
130 KEY OFF
140 SCREEN 1
150 COLOR 0,0
160 DEF SEG = &HB800
170 BLOAD "b:maclogo.scr",0
180 '
190 'Display time for screen
200 FOR ISTALL = 1 TO 13000
210 IF INKEY$<>"" THEN BEEP ELSE GOTO 230
220 ON ERROR GOTO 230     'to trap errors due to premature keyboard usage
230 NEXT
240 '
250 ' Loading the Title Page screen (TITLE.SCR)
260 CLS
270 COLOR 9,1
280 BLOAD "b:title.scr",0
290 FOR ISTALL = 1 TO 12000
300 ON ERROR GOTO 310
310 IF INKEY$<>"" THEN BEEP ELSE GOTO 320
320 NEXT
330 '
340 CHAIN "main"
```

APPENDIX 3.3

Program Listing of MAIN

```
10 '
20 '
30 '
40 '          ********* MAIN *************
50 '
60 '
70 '*****************************************************************************
80  'Variable Listing:
90  'COMMENT$=the record contents of ALPHA file
100 'JRECNO=the record number of ALPHA file
110 'NCODE=the start-up code in STORE1
120 'NOPCODE=the operation code in STORE2
130 'NRECCOUNT=the current record number in file COMMC
140 'NSUBMAIN=the number corresponding to the desired subMAIN
150 '*****************************************************************************
160 '
170 'Initialization
180 DEFINT I-N
190 '
200 '
210 'Setting up the screen
220 KEY OFF
230 SCREEN 0,1
240 WIDTH 80
250 COLOR 7,1,1:CLS
260 '
270 '
280 ' OPENING AND READYING ASSOCIATED FILES
290 OPEN "c:alpha" AS 1
300 FIELD 1,80 AS COMMENT$
310 '
320 OPEN "b:commc"  AS 2
330 FIELD 2,80 AS COMMC$
340 '
350 OPEN "b:store1" FOR INPUT AS 3
360 '
370 OPEN "b:comma" FOR OUTPUT AS 4
380 '
390 '
400 ' SETTING THE POINTER TO CURRENT RECORD NUMBER OF COMMC
410 GET 2,1: NRECCOUNT=VAL(COMMC$)
420 BLANK$=" ": LSET COMMC$=BLANK$: PUT 2, NRECCOUNT+1
430 '
440 '
450 ' CHECKING IF IT'S A SESSION RESTART, A CONTINUING SESSION OR NEW SESSION
460 INPUT#3,NCODE
470 '
480 ' CONTINUING SESSION:
490 IF NCODE<>0 THEN GOTO 640
500 '
510 ' CHECKING IF IT'S A SESSION RESTART OR NEW SESSION:
520 JRECNO=1
530 CLS
540 GOSUB 1650
550 IF JOK<>-1 THEN GOTO 520
560 IF (DY$="n") OR (DY$="N") THEN GOTO 790
570 '
```

```
580 'SESSION RESTART
590 GET 1,3: LSET COMMC$=COMMENT$: PUT 2,LOC(2)+1
600 CLOSE 1,2,3,4
610 CHAIN "sessrest"
620 '
630 '
640 ' CONTINUING SESSION
650 GET 1,2: LSET COMMC$=COMMENT$: PUT 2,LOC(2)+1
660 '
670 '     input answers from store1    ***
680 '                                 ***
690 '
700 OPEN "b:commb" FOR INPUT AS 5
710 '                                 ***
720 '          read info from commb    ***
730 '                                 ***
740 CLOSE 5
750 '
760 GOTO 860
770 '
780 '
790 ' NEW SESSION:
800 GET 1,4:  LSET COMMC$=COMMENT$:  PUT 2,LOC(2)+1
810 '
820 '
830 '       ASKING QUESTIONS- ANSWERS GO INTO COMMA      **
840 '                                                    **
850 '
860 ' PRINTING MENU
870 CLS
880 GET 1,5: PRINT COMMENT$
890 GET 1,6: PRINT COMMENT$
900 GET 1,7: PRINT COMMENT$
910 GET 1,8: PRINT COMMENT$
920 GET 1,12: PRINT COMMENT$
930 GET 1,13: PRINT COMMENT$
940 GET 1,14: PRINT COMMENT$:PRINT
950 '
960 '
970 'DETERMINING USER'S SELECTION FROM THE MENU
980 JRECNO=9
990 GOSUB 2220
1000 IF JOK<>-1 THEN GOTO 860
1010 IF (VAL(ANSI$)>=1) AND (VAL(ANSI$)<=5) THEN NSUBMAIN=VAL(ANSI$): GOTO 1180

1020 IF VAL(ANSI$)=6 GOTO 1090
1030 GET 1,10: PRINT COMMENT$
1040 GET 1,11: PRINT COMMENT$
1050 DUMMM$=INKEY$: IF DUMMM$="" THEN GOTO 1050
1060 GOTO 860
1070 '
1080 '
1090 'Loading the exit screen
1100 close: CLS:SCREEN 1: COLOR 0,1
1110 DEF SEG=&HB800
1120 BLOAD "b:end.scn",0
1130 GOTO 1130
```

```
1140 'Now have exited the package
1150 '
1160 '
1170 'Closing STORE1 as input file
1180 CLOSE 3
1182 if nsubmain>1 then goto 1210
1184 cls: locate 12,1: print "          Sorry - this option is not yet availabl
e": print
1186 get 1,11: print comment$
1188 dummy$=inkey$: if dummy$="" then goto 1188 else goto 860
1190 '
1200 '
1210 'STORING INFO IN STORE1 THAT WILL TELL MAIN WHERE IT HAS JUST BEEN
1220 OPEN "b:store1" FOR OUTPUT AS 3
1230 PRINT#3,NSUBMAIN
1240 '
1250 '
1260 ' STORING INFO IN STORE2 THAT WILL TELL subMAIN IT JUST CAME FROM MAIN
1270 OPEN "b:store2" FOR OUTPUT AS 6
1280 NOPCODE=0
1290 PRINT#6,NOPCODE
1300 '
1310 '
1320 ' STORING THE CURRENT RECORD NUMBER OF COMMC INTO THE FIRST REC NO OF COMM
C
1330 NRECCOUNT=LOC(2)
1340 LSET COMMC$=STR$(NRECCOUNT)
1350 PUT 2,1
1360 '
1370 '
1380 ' STORING INFO IN COMMA THAT IS NEEDED BY subMAIN
1390 '                                    ***
1400 '     write info in COMMA            ***
1410 '                                    ***
1420 '
1430 CLOSE 1,2,3,4,6
1440 '
1450 '
1460 'Loading the "Please Wait" screen
1470 CLS:SCREEN 1
1480 COLOR 0,1
1490 DEF SEG = &HB800
1500 BLOAD "b:retort.scr",0
1510 ON ERROR GOTO 1520
1520 IF INKEY$<>"" THEN BEEP ELSE GOTO 1530
1530 '
1540 '
1550 ' CHAINING TO subMAIN PROGRAM CHOSEN FROM MENU
1560 IF NSUBMAIN=1 THEN CHAIN "maina"
1570 IF NSUBMAIN=2 THEN CHAIN "mainb"
1580 IF NSUBMAIN=3 THEN CHAIN "mainc"
1590 IF NSUBMAIN=4 THEN CHAIN "maind"
1600 IF NSUBMAIN=5 THEN CHAIN "maine"
1610 PRINT "NSUBMAIN is not a number from 1 to 5"
1620 PRINT "MAIN could not chain to a subMAIN"
1630 PRINT "MAIN was terminated"
1640 STOP
```

```
1650 '
1660 '
1670 '
1680 '
1690 '  Y/N SUBROUTINE
1700 JOK=0
1710 GET 1,JRECNO
1720 COMMENT1Y$="You did not enter Y or N -     You entered: "
1730 COMMENT2Y$="You did not enter a value"
1740 PRINT COMMENT$
1750 LINE INPUT "",ANSY$
1760 NY=0
1770 NY = NY+1
1780 DY$=MID$(ANSY$,NY,1)
1790 IF DY$="" THEN GOTO 1860
1800 IF DY$=" " THEN GOTO 1770
1810 IF (DY$="Y") OR (DY$="y") OR (DY$="N") OR (DY$="n") THEN JOK=-1: RETURN
1820 PRINT COMMENT1Y$,ANSY$
1830 GET 1,11: PRINT COMMENT$
1840 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 1840
1850 RETURN
1860 PRINT COMMENT2Y$
1870 GET 1,11: PRINT COMMENT$
1880 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 1880
1890 RETURN
1900 '
1910 '
1920 '
1930 ' CHECKING FOR  REAL  NUMBERS
1940 JOK=0
1950 GET 1,JRECNO
1960 COMMENT1R$="You did not enter a REAL number  -  You entered: "
1970 COMMENT2R$="you did not enter a value"
1980 PRINT COMMENT$
1990 LINE INPUT"",ANSR$
2000 JFLAGR=0                          'flag to check that number exists
2010 IFLAGR=0                          'flag to check for decimal points
2020 JR=1
2030 NR=LEN(ANSR$)
2040 IF NR=0 THEN GOTO 2180
2050 WHILE JR<=NR
2060 DR$=MID$(ANSR$,JR,1)
2070 IF DR$=" " THEN GOTO 2150
2080 IF (ASC(DR$)>47) AND (ASC(DR$)<58) THEN JFLAGR=-1: GOTO 2150
2090 IF DR$="." THEN IFLAGR=IFLAGR+1: GOTO 2140
2100 PRINT COMMENT1R$,ANSR$
2110 GET 1,11:PRINT COMMENT$
2120 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 2120
2130 RETURN
2140 IF IFLAGR>1 GOTO 2100
2150 JR=JR+1
2160 WEND
2170 IF (DR$<>" ") OR (JFLAGR=-1) THEN JOK=-1: RETURN
2180 PRINT COMMENT2R$
2190 GET 1,11: PRINT COMMENT$
2200 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 2200
2210 RETURN
```

```
2220 '
2230 '
2240 '
2250 ' CHECKING FOR  INTEGER  NUMBERS
2260 JOK=0
2270 GET 1,JRECNO
2280 COMMENT1I$="You did not enter an INTEGER value -  You entered: "
2290 COMMENT2I$="you did not enter a value"
2300 PRINT COMMENT$
2310 LINE INPUT "",ANSI$
2320 NI=LEN(ANSI$)
2330 IF NI=0 THEN GOTO 2460
2340 JI=1
2350 WHILE JI<=NI
2360 DI$=MID$(ANSI$,JI,1)
2370 IF DI$=" " THEN GOTO 2430
2380 IF (ASC(DI$)>47) AND (ASC(DI$)<58) THEN IFLAGI=-1: GOTO 2430
2390 PRINT COMMENT1I$,ANSI$
2400 GET 1,11: PRINT COMMENT$
2410 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 2410
2420 RETURN
2430 JI=JI+1
2440 WEND
2450 IF (DI$<>" ") OR (IFLAGI=-1) THEN JOK=-1: RETURN
2460 PRINT COMMENT2I$
2470 GET 1,11: PRINT COMMENT$
2480 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 2480
2490 RETURN
```

ALPHA

1 Is this a session restart? (enter Y or N and RETURN)

2 This is a continuing session

3 This is a session restart

4 This is a new session

5 ***** M E N U *****

6 1 Spatial Temp Distribution at one Time - Can or Pouch

7 2 Can Center Temps, Lethality + Nutrient Destruction - Ball's Method

8 3 Spatial Temps, Lethality + Nutrient Destruction - Pouch

9 Select an item from the menu (enter 1 to 6 and RETURN)

10 You did not enter a number between 1 and 6

11 PRESS ANY KEY TO CONTINUE

12 4 Spatial Temps, Lethality + Nutrient Destruction - Can

13 5 Optimization Module - Nutrient Retention in Can

14 6 EXIT THE PACKAGE

APPENDIX 3.4

Program Listing of MAINB

```
15 '
20 '
25 '          ********** M A I N B **********
30 '
35 '
40 'Initialization and Dimensioning
50 DIM TEMP(501),TIME(501),TAVG(500),RATEL(500),XLETH(501)
55 DIM RATEN(500), XNUTOT(500), FRANUT(501)
57 DIM XX(50),YY(50)
60 '
65 '
70 'Setting up the text screen
75 KEY OFF
80 SCREEN 0,0,0,0
85 WIDTH 80
90 COLOR 7,1,1
92 CLS
95 '
100 '
105 ' opening and readying the associated files
110 OPEN "c:alphab" AS 1
115 FIELD 1,80 AS COMMENT$
120 '
125 OPEN "b:commc" AS 2
130 FIELD 2,80 AS COMMC$
135 '
140 '
145 ' setting the pointer to current record number of commc
150 GET 2,1: NRECCOUNT=VAL(COMMC$)
155 BLANK$=" ": LSET COMMC$=BLANK$: PUT 2,NRECCOUNT+1
160 '
165 '
170 ' Checking if control has come from MAIN or from SUBs
175 OPEN "b:store2" FOR INPUT AS 3
180 INPUT#3,NOPCODE
185 CLOSE 3
190 '
195 ' Not coming from MAIN:
200 IF NOPCODE<>0 THEN GOTO 1100
205 '
210 ' Coming from MAIN:
215 OPEN "b:comma" FOR INPUT AS 4
220 '
225 '     ** read info from comma**
230 '
235 CLOSE 4
240 '
245 '
250 ' Printing the menu
255 CLS
260 GET 1,1:  PRINT COMMENT$
265 GET 1,2:  PRINT COMMENT$
270 GET 1,3:  PRINT COMMENT$
275 GET 1,4:  PRINT COMMENT$
280 GET 1,5:  PRINT COMMENT$
285 '
290 '
```

```
295 'determining user's selection from the menu
300 JRECNO=10
305 GOSUB 2550
310 IF JOK<>-1 THEN GOTO 250
315 IF (VAL(ANSI$)=4)  GOTO 360
320 IF (VAL(ANSI$))=1) AND (VAL(ANSI$)<=3) THEN MODEL=VAL(ANSI$):NSUB=1:GOTO 46
5
325 GET 1,11
330 PRINT COMMENT$
335 GET 1,12: PRINT COMMENT$
340 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 340
345 GOTO 250
350 '
355 '
360 'if want to go straight back to MAIN
365 '
370 OPEN "b:commb" FOR OUTPUT AS 7
375 '
380 ' ** write info into b:commb **
385 '
390 CLOSE 7
395 '
400 CLOSE 1,2
405 '
410 CLS: SCREEN 1
415 COLOR 4,1
420 DEF SEG = &HB800
425 BLOAD "b:sandcl.scr",0
430 ON ERROR GOTO 435
435 IF INKEY$<>"" THEN BEEP ELSE GOTO 440
440 '
445 CHAIN "MAIN"
450 '
455 '
460 '
465 'if want to go to BALLST
470 '
475 'BALLST arguments
480 '
485 'determining and checking food temperature inputs
490 CLS
495 JRECNO =13: GOSUB 2390
500 IF JOK<>-1 THEN GOTO 490 ELSE TINIT=VAL(ANSR$)
505 CLS
510 JRECNO=20: GOSUB 2390
515 IF JOK<>-1 THEN GOTO 505 ELSE TRETRT=VAL(ANSR$)
520 CLS
525 JRECNO=21: GOSUB 2390
530 IF JOK<>-1 THEN GOTO 520 ELSE TWATER=VAL(ANSR$)'
535 IF (TRETRT>=TINIT) AND (TRETRT>=TWATER) THEN GOTO 595
540 CLS: GET 1,22: PRINT COMMENT$;
545 GET 1,23:  PRINT COMMENT$
550 GET 1,24:  PRINT COMMENT$;
555 GET 1,25:  PRINT COMMENT$,TINIT
560 GET 1,26:  PRINT COMMENT$,TWATER
565 GET 1,27:  PRINT COMMENT$,TRETRT
570 GET 1,28:  PRINT: PRINT COMMENT$
```

```
575 GET 1,12:  PRINT COMMENT$
580 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 580
585 GOTO 490
590 '
595 'determining and checking the time inputs
600 CLS:JRECNO=31: GOSUB 2550
605 IF JOK<>-1 THEN GOTO 595 ELSE NBA=VAL(ANSI$)
610 IF (2<=NBA) AND (NBA<=50) THEN GOTO 630
615 CLS:GET 1,36:PRINT COMMENT$
620 GET 1,12: PRINT COMMENT$
625 DUMMMA$=INKEY$:IF DUMMMA$="" THEN GOTO 625 ELSE GOTO 600
630 CLS: JRECNO=29: GOSUB 2390
635 IF JOK<>-1 THEN GOTO 630 ELSE TOTIME=VAL(ANSR$)
640 CLS: JRECNO=30: GOSUB 2390
645 IF JOK<>-1 THEN GOTO 640 ELSE COOLTI=VAL(ANSR$)
650 IF (TOTIME>0!) AND (COOLTI>0!) AND (TOTIME>=COOLTI) THEN GOTO 700
655 CLS: GET 1,32: PRINT COMMENT$
660 GET 1,33:  PRINT COMMENT$
665 GET 1,24:  PRINT COMMENT$;
670 GET 1,34:  PRINT COMMENT$;COOLTI
675 GET 1,35:  PRINT COMMENT$;TOTIME
680 GET 1,28:  PRINT: PRINT COMMENT$
685 GET 1,12:  PRINT COMMENT$
690 DUMMMA$=INKEY$:IF DUMMMA$="" THEN GOTO 690 ELSE GOTO 630
695 '
700 'determining and checking the (1/slopes)of the heating and cooling curves
705 CLS: JRECNO=37: GOSUB 2390
710 IF JOK<>-1 THEN GOTO 705 ELSE FH=VAL(ANSR$)
715 CLS: JRECNO=42: GOSUB 2390
720 IF JOK<>-1 THEN GOTO 715 ELSE FC=VAL(ANSR$)
725 IF (FH>700) AND (FC>700) THEN GOTO 775
730 CLS: GET 1,38: PRINT COMMENT$
735 GET 1,39:  PRINT COMMENT$
740 GET 1,24:  PRINT COMMENT$;
745 GET 1,40:  PRINT COMMENT$,FH
750 GET 1,41:  PRINT COMMENT$,FC
755 GET 1,28:  PRINT: PRINT COMMENT$
760 GET 1,12:  PRINT COMMENT$
765 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 765 ELSE GOTO 700
770 '
775 'determining and checking the heating curve lag factor
780 IF MODEL>1 THEN GOTO 870
785 CLS: JRECNO=43:GOSUB 2390
790 IF JOK<>-1 THEN GOTO 775 ELSE JH!=VAL(ANSR$)
795 IF JH!>=1! THEN GOTO 830
800 CLS: GET 1,44: PRINT COMMENT$
805 JRECNO=45: GOSUB 2265
810 IF JOK<>-1 THEN GOTO 800
815 IF (DY$="n") OR (DY$="N") THEN GOTO 830
820 GET 1,12: PRINT: PRINT COMMENT$
825 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 825 ELSE GOTO 775
830 IF JH!<=8! THEN GOTO 935
835 CLS: GET 1,46: PRINT COMMENT$
840 JRECNO=45: GOSUB 2265
845 IF JOK<>-1 THEN GOTO 835
850 IF (DY$="n") OR (DY$="N") THEN GOTO 935
855 GET 1,12: PRINT: PRINT COMMENT$
```

```
860 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 860 ELSE GOTO 775
865 '
870 ' for models 2 and 3
875 CLS: JRECNO=48: GOSUB 2390
880 IF JOK<>-1 THEN GOTO 875 ELSE JH!=VAL(ANSR$)
885 IF ABS(JH!-1.866)>.01 THEN GOTO 905
890 CLS:GET 1,81:PRINT COMMENT$;
895 GET 1,80:PRINT COMMENT$: GET 1,12:PRINT COMMENT$
900 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 900 ELSE GOTO 870
905 IF JH!<>0 THEN GOTO 795
910 CLS: GET 1,49: PRINT COMMENT$
915 GET 1,28: PRINT COMMENT$
920 GET 1,12 : PRINT COMMENT$
925 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 925 ELSE GOTO 870
930 '
935 ' determining and checking the cooling curve lag factor
940 IF MODEL<>3 THEN GOTO 1085
945 CLS: JRECNO=50: GOSUB 2390
950 IF JOK<>-1 THEN GOTO 945 ELSE JC!=VAL(ANSR$)
955 IF JC!<>0! THEN GOTO 980
960 CLS: GET 1,51: PRINT COMMENT$
965 GET 1,28: PRINT COMMENT$
970 GET 1,12: PRINT COMMENT$
975 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 975 ELSE GOTO 945
980 IF JC!>=1 THEN GOTO 1015
985 CLS: GET 1,52: PRINT COMMENT$
990 JRECNO=47: GOSUB 2265
995 IF JOK<>-1 THEN GOTO 985
1000 IF (DY$="n") OR (DY$="N") THEN GOTO 1085
1005 GET 1,12: PRINT COMMENT$
1010 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1010 ELSE GOTO 945
1015 IF JC!<=8! THEN GOTO 1050
1020 CLS: GET 1,53:  PRINT COMMENT$
1025 JRECNO=47: GOSUB 2265
1030 IF JOK<>-1 THEN GOTO 1020
1035 IF (DY$="N") OR (DY$="n") THEN GOTO 1085
1040 GET 1,12: PRINT COMMENT$
1045 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1045 ELSE GOTO 945
1050 IF ABS(JC!-1.866)>.01 THEN GOTO 1085
1055 CLS: GET 1,79: PRINT COMMENT$;
1060 GET 1,80: PRINT COMMENT$
1065 GET 1,12: PRINT COMMENT$
1070 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1070 ELSE GOTO 945
1075 '
1080 '
1085 GOTO 2115
1090 '
1095 '
1100 OPEN "b:store2" FOR INPUT AS 3  'to align pointer at 1st rec of STORE2
1110 INPUT#3,NSUB,MODEL,TINIT,TRETRT,TWATER,NBA,TOTIME,COOLTI,FH,FC,JH!,JC!,NSU
BMAIN
1115 N=NBA
1120 CLOSE 3
1125 '
1130 '
1135 ' reading the results from the calculating SUB programs from COMM2
1140 OPEN "c:comm2" FOR INPUT AS 7
```

```
1145 FOR I=1 TO N
1150 INPUT#7,TIME(I),TEMP(I)
1155 NEXT I
1160 CLOSE 7
1165 '
1170 '
1175 'Getting the FLETH value
1180 CLS: JRECNO=82: GOSUB 2390
1185 IF JOK<>-1 THEN GOTO 1180 ELSE FLETH=VAL(ANSR$)
1190 IF (FLETH>0!) AND (FLETH<=20!) GOTO 1250
1195 IF FLETH>20! THEN GOTO 1215
1200 GET 1,83: PRINT COMMENT$
1205 GET 1,12: PRINT: PRINT COMMENT$
1210 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1210 ELSE GOTO 1180
1215 CLS: GET 1,84: PRINT COMMENT$
1220 GET 1,24: PRINT COMMENT$;FLETH
1225 JRECNO=85: GOSUB 2265
1230 IF JOK<>-1 GOTO 1215
1235 IF (DY$="n") OR (DY$="N") GOTO 1250 ELSE GOTO 1180
1240 '
1245 '
1250 'Getting the ZLETH value
1255 CLS: JRECNO=86: GOSUB 2390
1260 IF JOK<>-1 THEN GOTO 1255 ELSE ZLETH=VAL(ANSR$)
1265 IF (ZLETH>0!) AND (ZLETH<=30!) GOTO 1325
1270 IF ZLETH>30! THEN GOTO 1290
1275 GET 1,87: PRINT COMMENT$
1280 GET 1,12: PRINT: PRINT COMMENT$
1285 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1285 ELSE GOTO 1255
1290 CLS: GET 1,88: PRINT COMMENT$
1295 GET 1,24: PRINT COMMENT$;ZLETH
1300 JRECNO=85: GOSUB 2265
1305 IF JOK<>-1 GOTO 1290
1310 IF (DY$="n") OR (DY$="N") GOTO 1325 ELSE GOTO 1255
1315 '
1320 '
1325 'calculating lethality
1330 DTIME = TOTIME/(N-1)/ 60!  'to convert seconds into minutes (like FLETH)
1332 xleth(1)=0.0
1335 FOR K=1 TO N-1
1340 TAVG(K) = (TEMP(K)+TEMP(K+1))/ 2!
1345 IF (-121!+TAVG(K))/ZLETH<-35 THEN RATEL(K)=0!: GOTO 1355
1350 RATEL(K)=1 / ELETH * (10^ ( (-121!+TAVG(K))/ZLETH) )
1355 IF K=1 THEN  XLETH(K+1)=RATEL(K)*DTIME  ELSE   XLETH(K+1)=XLETH(K) + RATEL
(K)*DTIME
1365 NEXT K
1375 '
1380 '
1385 'Getting the FXNUT value
1390 CLS: JRECNO=89: GOSUB 2390
1395 IF JOK<>-1 THEN GOTO 1390 ELSE FXNUT=VAL(ANSR$)
1400 IF (FXNUT>0!) AND (FXNUT<=100!) GOTO 1460
1405 IF FXNUT>100! THEN GOTO 1425
1410 GET 1,83: PRINT COMMENT$
1415 GET 1,12: PRINT: PRINT COMMENT$
1420 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1420 ELSE GOTO 1390
1425 CLS: GET 1,90: PRINT COMMENT$
```

```
1430 GET 1,24: PRINT COMMENT$;FXNUT
1435 JRECNO=85: GOSUB 2265
1440 IF JOK<>-1 GOTO 1425
1445 IF (DY$="n") OR (DY$="N") GOTO 1460 ELSE GOTO 1390
1450 '
1455 '
1460 'Getting the Znut value
1465 CLS: JRECNO=91: GOSUB 2390
1470 IF JOK<>-1 THEN GOTO 1465 ELSE ZNUT=VAL(ANSR$)
1475 IF (ZNUT>0!) AND (ZNUT<=50!) GOTO 1535
1480 IF ZNUT>50! THEN GOTO 1500
1485 GET 1,87: PRINT COMMENT$
1490 GET 1,12: PRINT: PRINT COMMENT$
1495 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1495 ELSE GOTO 1465
1500 CLS: GET 1,92: PRINT COMMENT$
1505 GET 1,24: PRINT COMMENT$;ZNUT
1510 JRECNO=85: GOSUB 2265
1515 IF JOK<>-1 GOTO 1500
1520 IF (DY$="n") OR (DY$="N") GOTO 1535 ELSE GOTO 1465
1525 '
1530 '
1535 'Calculating the nutrient fraction retained
1537 franut(1)=1.0
1540 FOR J=1 TO N-1
1545 IF (-121!+TAVG(J))/ZNUT<-35 THEN RATEN(J)=0!: GOTO 1552
1550 RATEN(J)=1 / FXNUT * (10^ ( (-121!+TAVG(J))/ZNUT) )
1552 IF (12!*RATEN(J)*DTIME) > 39 THEN XNUTOT(J)=0!: GOTO 1560
1555 IF J=1 THEN XNUTOT(J)=1! * 10!^(-12!*RATEN(J)*DTIME)  ELSE  XNUTOT(J)=XNUT
OT(J-1) * 10!^(-12!*RATEN(J)*DTIME)
1560 FRANUT(J+1) = XNUTOT(J) / 1!
1570 NEXT J
1580 '
1581 '
1582 'Going to Result Display Section
1583 GOTO 4000
1584 '
1585 '
1590 ' Ask if want to go to main or back to sub
1595 CLS: JRECNO=54: GOSUB 2265
1600 IF JOK<>-1 THEN GOTO 1590
1605 IF (DY$="y") OR (DY$="Y") THEN GOTO 1715
1610 '
1615 'when want to go to MAIN:
1620 OPEN "b:commb" FOR OUTPUT AS 7
1625 '
1630 '### write info to commb
1635 CLOSE 7
1640 '
1645 'storing the current record number of COMMC into the first rec no of COMMC

1650 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
1655 CLOSE 1,2
1660 '
1665 CLS: SCREEN 1
1670 COLOR 1,1
1675 DEF SEG = &HB800
1680 BLOAD "b:sandcl.scr",0
```

```
1685 ON ERROR GOTO 1690
1690 IF INKEY$<>"" THEN BEEP ELSE GOTO 1695
1695 '
1700 CHAIN "main"
1705 '
1710 '
1715 'when want to go back to SUB used:
1720 'changing the arguments used for the SUB:
1725 N1STTHRU=0
1730 'determining MODEL wanted
1735 NOLD = MODEL
1740 JRECNO=58: NMARKSUB=0: GOSUB 2695
1745 NEW1=VAL(NEWVALUE$)
1750 IF (NEW1>=1) AND (NEW1<=3) THEN GOTO 1765
1755 JRECNO=19: JRECNO2=71: NMARKSET=1: GOSUB 2850
1760 GOTO 1740
1765 MODEL=NEW1
1770 '
1775 'determining temperatures
1780 OLD=TINIT: JRECNO=60:NMARKSUB=1: GOSUB 2695
1785 XNEW1=VAL(NEWVALUE$)
1790 OLD=TRETRT: JRECNO=61:NMARKSUB=1: GOSUB 2695
1795 XNEW2=VAL(NEWVALUE$)
1800 OLD=TWATER: JRECNO=62:NMARKSUB=1: GOSUB 2695
1805 XNEW3=VAL(NEWVALUE$)
1810 IF (XNEW2>=XNEW1) AND (XNEW2>=XNEW3)THEN GOTO 1825
1815 JRECNO=23: JRECNO2=72: NMARKSET=3: GOSUB 2850
1820 GOTO 1780
1825 TINIT=XNEW1: TRETRT=XNEW2: TWATER=XNEW3
1830 '
1835 'determining time values
1840 OLD=TOTIME: JRECNO=63: NMARKSUB=1: GOSUB 2695
1845 XNEW1=VAL(NEWVALUE$)
1850 OLD=COOLTI: JRECNO=64: NMARKSUB=1: GOSUB 2695
1855 XNEW2=VAL(NEWVALUE$)
1860 IF (XNEW1>0) AND (XNEW2>0) AND (XNEW1>=XNEW2) THEN GOTO 1875
1865 JRECNO=33: JRECNO2=73: NMARKSET=2: GOSUB 2850
1870 GOTO 1840
1875 TOTIME=XNEW1: COOLTI=XNEW2
1880 '
1885 'determining number of time values
1890 NOLD=NBA: JRECNO=65: NMARKSUB=0: GOSUB 2695
1895 NEW1=VAL(NEWVALUE$)
1900 IF (2<=NEW1) AND (NEW1<=50) THEN GOTO 1915
1905 JRECNO=36: JRECNO2=74: NMARKSET=1: GOSUB 2850
1910 GOTO 1890
1915 NBA=NEW1
1920 '
1925 'determining (1/slopes) of heating and cooling curves
1930 OLD=FH: JRECNO=66: NMARKSUB=1: GOSUB 2695
1935 XNEW1=VAL(NEWVALUE$)
1940 OLD=FC: JRECNO=67: NMARKSUB=1: GOSUB 2695
1945 XNEW2=VAL(NEWVALUE$)
1950 IF (XNEW1>700) AND (XNEW2>700) THEN GOTO 1965
1955 JRECNO=39: JRECNO2=75: NMARKSET=2: GOSUB 2850
1960 GOTO 1930
1965 FH=XNEW1: FC=XNEW2
```

```
1970 '
1975  'determining lag factor of heating curve
1980 OLD=JH!: IF MODEL=1 THEN JRECNO=68 ELSE JRECNO=69
1985 NMARKSUB=1: GOSUB 2695
1990 XNEW1=VAL(NEWVALUE$)
1995 IF (XNEW1>0) THEN GOTO 2015
2000 IF (MODEL<>1) AND (XNEW1=0) THEN GOTO 2005 ELSE GOTO 2015
2005 JRECNO=49: JRECNO2=76: NMARKSET=1: GOSUB 2850
2010 GOTO 1980
2015 IF MODEL=1 THEN JH!=XNEW1: GOTO 2040
2020 IF ABS(XNEW1-1.866)>.01 THEN JH!=XNEW1: GOTO 2040
2025 JRECNO=81: JRECNO2=76: NMARKSET=1: GOSUB 2850
2030 GOTO 1980
2035 '
2040 'determining lag factor of cooling curve
2045 IF MODEL<>3 THEN GOTO 2095
2050 OLD=JC!: JRECNO=70: NMARKSUB=1: GOSUB 2695
2055 XNEW1=VAL(NEWVALUE$)
2060 IF XNEW1>0 THEN GOTO 2075
2065 JRECNO=51: JRECNO2=77: GOSUB 2850
2070 GOTO 2050
2075 IF ABS(XNEW1-1.866)>.01 THEN JC!=XNEW1: GOTO 2095
2080 JRECNO=79: JRECNO2=77: NMARKSET=1: GOSUB 2850
2085 GOTO 2045
2090 '
2095 'setting nsub equal to first time through
2100 NSUB=1
2105 '
2110 '
2115 ' copying arguments into COMM1 to run the SUB chosen
2120 OPEN "c:comm1" FOR OUTPUT AS 6
2125 NSUBMAIN=2
2130 PRINT#6,MODEL;TINIT;TRETRT;TWATER;NBA;TOTIME;COOLTI;FH;FC;JH!;JC!;NSUBMAIN
2135 CLOSE 6
2140 '
2145 '
2150 'putting info into STORE2 to tell MAINB what SUB it chained to
2155 OPEN "b:store2" FOR OUTPUT AS 3
2160 PRINT#3,NSUB;MODEL;TINIT;TRETRT;TWATER;NBA;TOTIME;COOLTI;FH;FC;JH!;JC!;NSU
BMAIN
2165 CLOSE 3
2170 '
2175 '
2180 'storing the current record number of COMMC into the first rec no of COMMC
2185 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
2190 '
2195 CLOSE 1,2
2200 '
2205 '
2210 'Displaying "Please Wait" symbol
2215 CLS:SCREEN 1,0,0,0
2220 COLOR 5,1
2225 DEF SEG = &HB800
2230 BLOAD "b:clock.scr",0
2235 ON ERROR GOTO 2240
```

```
2240 IF INKEY$<>"" THEN BEEP ELSE GOTO 2245
2245 '
2250 ' chaining to BALLST
2255 CHAIN "BALLST"
2260 STOP
2265 '
2267 '
2269 '
2271 '
2273 '
2275 'getting and checking  Y/N SUBROUTINE
2277 JOK=0
2279 GET 1,JRECNO
2281 COMMENT1Y$="You did not enter Y or N -    You entered: "
2283 COMMENT2Y$="You did not enter a value"
2285 PRINT COMMENT$;
2287 LINE INPUT "",ANSY$
2289 JY=0
2291 JY = JY+1
2293 DY$=MID$(ANSY$,JY,1)
2295 IF DY$="" THEN GOTO 2309
2297 IF DY$=" " THEN GOTO 2291
2299 IF (DY$="Y") OR (DY$="y") OR (DY$="N") OR (DY$="n") THEN JOK=-1: RETURN
2301 PRINT COMMENT1Y$,ANSY$
2303 GET 1,12: PRINT: PRINT COMMENT$
2305 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2305
2307 RETURN
2309 PRINT COMMENT2Y$
2311 GET 1,12: PRINT: PRINT COMMENT$
2313 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2313
2315 RETURN
2390 '
2392 '
2394 '
2396 ' getting and CHECKING FOR  REAL  NUMBERS
2398 JOK=0
2400 GET 1,JRECNO
2402 COMMENT1R$="You did not enter a REAL number -  You entered: "
2404 COMMENT2R$="you did not enter a value"
2406 PRINT COMMENT$;
2408 LINE INPUT"",ANSR$
2410 JFLAGR=0                          'flag to check that number exists
2412 IFLAGR=0                          'flag to check for decimal points
2414 JR=1
2416 KR=LEN(ANSR$)
2418 IF KR=0 THEN GOTO 2446
2420 WHILE JR<=KR
2422 DR$=MID$(ANSR$,JR,1)
2424 IF DR$=" " THEN GOTO 2440
2426 IF (ASC(DR$)>47) AND (ASC(DR$)<58) THEN JFLAGR=-1: GOTO 2440
2428 IF DR$="." THEN IFLAGR=IFLAGR+1: GOTO 2438
2430 PRINT COMMENT1R$,ANSR$
2432 GET 1,12: PRINT: PRINT COMMENT$
2434 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 2434
2436 RETURN
2438 IF IFLAGR>1 GOTO 2430
2440 JR=JR+1
```

```
2442 WEND
2444 IF (DR$<>" ") OR (JFLAGR=-1) THEN JOK=-1: RETURN
2446 PRINT COMMENT2R$
2448 GET 1,12: PRINT: PRINT COMMENT$
2450 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 2450
2452 RETURN
2550 '
2552 '
2554 '
2556 'getting and CHECKING FOR  INTEGER  NUMBERS
2558 JOK=0
2560 IFLAGI=0
2562 GET 1,JRECNO
2564 COMMENT1I$="You did not enter an INTEGER value  -  You entered: "
2566 COMMENT2I$="you did not enter a value"
2568 PRINT COMMENT$;
2570 LINE INPUT "",ANSI$
2572 NI=LEN(ANSI$)
2574 IF NI=0 THEN GOTO 2600
2576 JI=1
2578 WHILE JI<=NI
2580 DI$=MID$(ANSI$,JI,1)
2582 IF DI$=" " THEN GOTO 2594
2584 IF (ASC(DI$)>47) AND (ASC(DI$)<58) THEN IFLAGI=-1: GOTO 2594
2586 PRINT COMMENT1I$,ANSI$
2588 GET 1,12: PRINT: PRINT COMMENT$
2590 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 2590
2592 RETURN
2594 JI=JI+1
2596 WEND
2598 IF (DI$<>" ") OR (IFLAGI=-1) THEN JOK=-1: RETURN
2600 PRINT COMMENT2I$
2602 GET 1,12: PRINT: PRINT COMMENT$
2604 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 2604
2606 RETURN
2695 '
2697 '
2699 '
2701 '
2703 ' Table Creation subroutine
2705 IF N1STTHRU=1 GOTO 2725
2707 NROW=7:NCOLUMN=1
2709 SCREEN 0,0,1,1
2711 CLS
2713 'printing table heading
2715 GET 1,55:PRINT COMMENT$
2717 GET 1,56:PRINT COMMENT$;
2719 GET 1,57:PRINT COMMENT$
2721 GET 1,59:PRINT COMMENT$
2723 N1STTHRU=1
2725 LOCATE NROW,NCOLUMN
2727 GET 1,JRECNO
2729 BLANK$="   "
2731 NBLANK=INSTR(COMMENT$,BLANK$)
2733 FINAL$=LEFT$(COMMENT$,NBLANK-1)
2735 IF IDIFF=1 THEN DIFF$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(67);:
GOTO 2739
```

```
2737 IF NMARKSUB=1 THEN PRINT FINAL$;TAB(53);OLD;TAB(67); ELSE PRINT FINAL$;TAB
(53);NOLD;TAB(67);
2739 LINE INPUT NEWVALUE$
2741 NRTEMP=CSRLIN: NCTEMP=POS(0)
2743 IF LEN(NEWVALUE$)=0 THEN GOTO 2753
2745 IF NMARKSUB=1 THEN GOSUB 2985 ELSE GOSUB 3095
2747 IF (NOK=-1) THEN NROW=CSRLIN: NCOLUMN=POS(0): RETURN
2749 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
2751 GOTO 2735
2753 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
2755 IF IDIFF=1 THEN NEWVALUE$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(6
7);VAL(NEWVALUE$):GOTO 2759
2757 IF NMARKSUB=1 THEN NEWVALUE$=STR$(OLD):PRINT FINAL$;TAB(53);OLD;TAB(67);VA
L(NEWVALUE$) ELSE NEWVALUE$=STR$(NOLD):PRINT FINAL$;TAB(53);NOLD;TAB(67);VAL(NE
WVALUE$)
2759 NROW=CSRLIN: NCOLUMN=POS(0)
2761 RETURN
2850 '
2852 '
2854 '
2856 '
2858 ' Erase Lines in table and show error page subroutine
2860 SCREEN ,,2,2
2862 CLS
2864 GET 1,JRECNO: PRINT COMMENT$;
2866 GET 1,24: PRINT COMMENT$;
2868 GET 1,JRECNO2: PRINT COMMENT$;
2870 IF NMARKSET<>3 THEN GOTO 2874
2872 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2,XNEW3:GOTO 2880 ELSE PRINT NEW1,NEW2,
NEW3:GOTO 2880
2874 IF NMARKSET<>2 THEN GOTO 2878
2876 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2:GOTO 2880 ELSE PRINT NEW1,NEW2: GOTO
2880
2878 IF NMARKSUB=1 THEN PRINT XNEW1 ELSE PRINT NEW1
2880 GET 1,12
2882 PRINT: PRINT COMMENT$
2884 DUMMEL$=INKEY$: IF DUMMEL$="" THEN GOTO 2884
2886 SCREEN ,,1,1
2888 LOCATE NROW,NCOLUMN
2890 LOCATE NROW-1,NCOLUMN:PRINT "
                                  ": LOCATE NROW-1,NCOLUMN
2892 IF NMARKSET=1 THEN GOTO 2900
2894 LOCATE NROW-2,NCOLUMN:PRINT "
                                  ":LOCATE NROW-2,NCOLUMN
2896 IF NMARKSET=2 THEN GOTO 2900
2898 LOCATE NROW-3,NCOLUMN:PRINT "
                                  ": LOCATE NROW-3,NCOLUMN
2900 NROW=CSRLIN: NCOLUMN=POS(0)
2902 RETURN
2985 '
2987 '
2989 '
2991 '
2993 'checking Real2 numbers (in table)
2995 NOK=0
```

```
2997 JFLAGR2=0
2999 IFLAGR2=0
3001 JR2=1
3003 NR2=LEN(NEWVALUE$)
3005 WHILE JR2=<NR2
3007 DR2$=MID$(NEWVALUE$,JR2,1)
3009 IF DR2$=" " THEN GOTO 3019
3011 IF DR2$="." THEN IFLAGR2=IFLAGR2+1: GOTO 3019
3013 IF (ASC(DR2$)>47) AND (ASC(DR2$)<58) THEN JFLAGR2=-1: GOTO 3019
3015 NOK=0
3017 RETURN
3019 IF IFLAGR2>1 THEN NOK=0: RETURN
3021 JR2=JR2+1
3023 WEND
3025 IF JFLAGR2=0 THEN NOK=0 ELSE NOK=-1
3027 RETURN
3095 '
3097 '
3099 '
3101 '
3103 ' checking Integer2 numbers (in table)
3105 NOK=0
3107 IFLAGI2=0
3109 NI2=LEN(NEWVALUE$)
3111 JI2=1
3113 WHILE JI2<=NI2
3115 DI2$=MID$(NEWVALUE$,JI2,1)
3117 IF DI2$=" " THEN GOTO 3123
3119 IF (ASC(DI2$)>47) AND (ASC(DI2$)<58) THEN IFLAGI2=-1: GOTO 3123
3121 NOK=0: RETURN
3123 JI2=JI2+1
3125 WEND
3127 IF IFLAGI2=0 THEN NOK=0 ELSE NOK=-1
3129 RETURN
3185 '
3190 '
3195 '
4000 '*****************************************************************
4005 '
4010 '                        RESULTS DISPLAY SECTION
4015 '
4020 SCREEN 1,0,0,0:COLOR 9,1
4021 '
4022 '
4023 ' common variable initialization
4024 DUMMY1=0
4025 XXMAX=TOTIME
4026 XXMIN=0
4028 NYY=6
4030 '
4031 'Writing the results display menu
4035 '
4040 CLS:LOCATE 1,1
4045 GET 1,100:GOSUB 5165:PRINT COMMT40$;
4050 LOCATE 2,1
4055 GET 1,101:GOSUB 5165:PRINT COMMT40$;
4060 LOCATE 4,1
```

```
4065 GET 1,102:GOSUB 5165:PRINT COMMT40$;
4070 LOCATE 6,1
4075 GET 1,104:GOSUB 5165:PRINT COMMT40$;
4080 LOCATE 8,1
4085 GET 1,148:GOSUB 5165:PRINT COMMT40$;
4086 LOCATE 10,1
4087 GET 1,149:GOSUB 5165:PRINT COMMT40$;
4088 LOCATE 12,1
4090 GET 1,150:GOSUB 5165:PRINT COMMT40$;
4091 LOCATE 14,1
4092 GET 1,151:GOSUB 5165:PRINT COMMT40$;
4093 LOCATE 16,1
4094 GET 1,103:GOSUB 5165:PRINT COMMT40$;
4097 LOCATE 18,1:GET 1,105:GOSUB 5165:PRINT COMMT40$;
4098 LOCATE 23,1
4100 GET 1,106:GOSUB 5165:PRINT COMMT40$;
4105 '
4110 '
4115 'Getting the user's choice from menu
4120 GOSUB 5145
4125 IANSR=VAL(IANSR$):IF (IANSR>=1) AND (IANSR<=8) THEN GOTO 4130 ELSE LOCATE
24,1:BEEP:GET 1,160:GOSUB 5165:PRINT COMMT40$;:GOTO 4120
4130 IF IANSR=1 THEN GOTO 4151 'time-temp graph
4135 IF IANSR=2 THEN GOTO 4151 'time-temp table
4140 IF IANSR=3 THEN GOTO 4180 'lethality-time graph
4141 IF IANSR=4 THEN GOTO 4180 'lethality-time table
4142 IF IANSR=5 THEN GOTO 4206 'nutrient-fraction-time graph
4143 IF IANSR=6 THEN GOTO 4206 'nutrient-fraction-time table
4144 IF IANSR=7 THEN GOTO 4580 'process parameter, table
4145 GOTO 4930
4146 '
4150 '
4151 ' time-temp initializing part
4155 CLS: NXX=NBA: IF IANSR=1 GOTO 4160 ELSE GOTO 4172
4160 YYHVAR=150!: YYLVAR=0!
4163 FOR I=1 TO NXX
4166 YY(I)=TEMP(I)
4169 NEXT I
4170 IREC1=107:IREC2=108:IREC3=109:GOTO 4233
4172 IREC4=136:FOR I=1 TO NXX:XX(I)=TIME(I):YY(I)=TEMP(I):NEXT:GOTO 4668
4174 '
4175 '
4180 ' lethality-time initializing part
4185 CLS: NXX=NBA: IF IANSR=3 GOTO 4187 ELSE GOTO 4196
4187 YYHVAR=2!: YYLVAR=0!
4190 FOR I=1 TO NXX
4192 YY(I)=XLETH(I)
4193 NEXT I
4195 IREC1=141:IREC2=142:IREC3=109:GOTO 4233
4196 IREC4=143:FOR I=1 TO Nxx:XX(I)=TIME(I):NEXT
4198 FOR I=1 TO NXX: YY(I)=XLETH(I): NEXT: GOTO 4668
4200 '
4202 '
4206 ' nutrient-time initializing part
4208 CLS: NXX=NBA: IF IANSR=5 GOTO 4210 ELSE GOTO 4220
4210 YYHVAR=1!: YYLVAR=0!
4212 FOR I=1 TO NXX
```

```
4214 YY(I)=FRANUT(I)
4216 NEXT I
4218 IREC1=152:IREC2=153:IREC3=109:GOTO 4233
4220 IREC4=154:FOR I=1 TO nxx:XX(I)=TIME(I): NEXT
4222 FOR I=1 TO NXX: YY(I)=FRANUT(I): NEXT: GOTO 4668
4231 '
4232 '
4233 ' calculating x-array
4234 DXX=XXMAX!/CSNG(NXX-1)
4235 FOR I=1 TO NXX
4237 XX(I)=CSNG((I-1)*DXX)
4240 NEXT I
4242 GOSUB 6000
4245 '
4250 '
4545 ' Going to the parameter table
4555 LOCATE 24,1:GET 1,111:GOSUB 5165:PRINT COMMT40$;:LOCATE 24,22:PRINT CHR$(2
4);:GOSUB 5010
4565 IF IFLAG=1 THEN GOTO 4021 ELSE 4930
4570 '
4575 '
4580 ' Process parameter table
4585 CLS:NROW=7
4590 LOCATE 2,1:GET 1,113:GOSUB 5165:PRINT COMMT40$;:LOCATE 3,1:GET 1,114:GOSUB
 5165:PRINT COMMT40$;
4595 GET 1,115:GOSUB 5165:VAR$=COMMT40$:VAR!=TINIT!:GOSUB 4980
4600 GET 1,116:GOSUB 5165:VAR$=COMMT40$:VAR!=TRETRT!:GOSUB 4980
4605 GET 1,117:GOSUB 5165:VAR$=COMMT40$:VAR!=TWATER!:GOSUB 4980
4610 GET 1,118:GOSUB 5165:VAR$=COMMT40$:VAR!=FH!:GOSUB 4980
4615 GET 1,119:GOSUB 5165:VAR$=COMMT40$:VAR!=FC!:GOSUB 4980
4620 GET 1,120:GOSUB 5165:VAR$=COMMT40$:VAR!=JH!:GOSUB 4980
4625 GET 1,121:GOSUB 5165:VAR$=COMMT40$:VAR!=JC!:GOSUB 4980
4630 GET 1,122:GOSUB 5165:VAR$=COMMT40$:VAR!=TOTIME!:GOSUB 4980
4635 GET 1,123:GOSUB 5165:VAR$=COMMT40$:VAR!=COOLTI!:GOSUB 4980
4640 GET 1,124:GOSUB 5165:VAR$=COMMT40$:VAR!=NBA:GOSUB 4980
4641 GET 1,144:GOSUB 5165:VAR$=COMMT40$:VAR!=FLETH!:GOSUB 4980
4642 GET 1,145:GOSUB 5165:VAR$=COMMT40$:VAR!=ZLETH!:GOSUB 4980
4643 GET 1,146:GOSUB 5165:VAR$=COMMT40$:VAR!=FXNUT!:GOSUB 4980
4644 GET 1,147:GOSUB 5165:VAR$=COMMT40$:VAR!=ZNUT!:GOSUB 4980
4650 '
4655 '
4660 ' deciding on the next step: graph or menu
4665 LOCATE 24,1:GET 1,111:GOSUB 5165:PRINT COMMT40$;:LOCATE 24,22:PRINT CHR$(2
4);:GOSUB 5010
4666 IF IFLAG=1 THEN GOTO 4021 ELSE 4930
4667 '
4668 ' this is the table section
4670 SCREEN 2:SCREEN 0,0,0,0:COLOR 7,1,1
4675 '
4680 '
4685 ' listing of the results in tabular form
4690 CLS
4695 LOCATE 1,1
4700 GET 1,125:GOSUB 5165:PRINT COMMENT$;
4705 FOR I=1 TO 19
4710 GET 1,126:PRINT COMMENT$;
4715 NEXT I
```

```
4720 LOCATE 4,1
4725 GET 1,127:PRINT COMMENT$;
4730 LOCATE 20,1
4735 GET 1,127:PRINT COMMENT$;
4740 LOCATE 3,1
4745 GET 1,IREC4:PRINT COMMENT$;
4750 NROW=4:NPAGE=1: ISET = 1
4755 FOR I=ISET TO NXX+1
4760 NROW=NROW+1
4765 IF (I+NPAGE)/NPAGE >16 THEN NROW=4:NPAGE=NPAGE+1: ISET=I: GOTO 4790
4770 IF I=NXX+1 GOTO 4890
4775 LOCATE NROW,12:PRINT XX(I):LOCATE NROW,53:PRINT YY(I)
4777 NEXT I
4790 LOCATE 21,1:GET 1,131:PRINT COMMENT$;
4795 LOCATE 23,1: GET 1,135: PRINT COMMENT$;: LOCATE 22,1: GET 1,134: PRINT COM
MENT$;: LOCATE 22,48: PRINT CHR$(24);
4800 LOCATE 23,1:GET 1,128:PRINT COMMENT$;
4805 '
4810 '
4815 ' deciding if the user wants to see next page, print table or exit to menu
4820 ANSR$=INPUT$(1)
4825 IF ANSR$="Y" OR ANSR$="y" GOTO 4855
4830 IF ANSR$="N" OR ANSR$="n" THEN COLOR 7,1,1:CLS:GOTO 4020
4835 COLOR 20,1,1:BEEP:LOCATE 23,1:GET 1,129:PRINT COMMENT$;:COLOR 7,1,1:GOTO 4
820
4840 '
4845 '
4850 ' clearing the page for next page of table
4855 LOCATE 21,1: GET 1,135: PRINT COMMENT$;
4860 LOCATE 22,1: GET 1,135: PRINT COMMENT$;
4865 LOCATE 23,1: GET 1,135: PRINT COMMENT$;
4870 LOCATE 5,1:FOR IK=1 TO 15
4875 GET 1,126:PRINT COMMENT$;
4880 NEXT IK
4885 GOTO 4755
4890 NROW = 21
4895 FOR I=1 TO 5
4900 GOSUB 4925
4905 NROW=NROW+1
4910 NEXT
4915 LOCATE 22,1:GET 1,130:PRINT COMMENT$;:LOCATE 22,28:PRINT CHR$(24);:LOCATE
23,1:GET 1,133:PRINT COMMENT$;:ANSR$=INKEY$:IF ANSR$="" THEN 4915
4920 COLOR 7,1,1:CLS:GOTO 4020
4925 LOCATE NROW,1: PRINT "                                                    "
;: RETURN
4930 '
4935 '
4940 ' exit to the MAINB program
4945 CLS:SCREEN 2:SCREEN 0:COLOR 7,1,1: GOTO 1584
4950 '
4955 '
4960 ' returning to graph or results display menu
4975 STOP
4980 '
4985 '
4990 'subrdutine to position process parameters in table
```

```
4995 LOCATE NROW,1:PRINT VAR$:LOCATE NROW,32
4997 PARAM$=STR$(VAR): PARAM$=LEFT$(PARAM$,8): PRINT PARAM$;
5000 NROW=NROW+1
5005 RETURN
5010 '
5015 '
5020 ' error checking subroutine
5025 ANSR$=INPUT$(1)
5035 IF ASC(ANSR$)=88 OR ASC(ANSR$)=120 THEN IFLAG=1:RETURN
5040 BEEP:LOCATE 25,1:GET 1,164:GOSUB 5165:PRINT COMMT40$;:GOTO 5025
5045 RETURN
5130 '
5135 '
5140 ' error-checking subroutine for integers (results display section)
5145 IANSR$=INPUT$(1)
5150 IF ASC(IANSR$)>=49 AND ASC(IANSR$)<=58 THEN IFLAG=-1 ELSE IFLAG=0
5155 IF IFLAG=0 THEN LOCATE 24,1:BEEP:GET 1,162:GOSUB 5165:PRINT COMMT40$;:GOTO
 5145
5160 RETURN
5162 '
5163 '
5164 ' subroutine to reduce the size of the record
5165 COMMT40$=LEFT$(COMMENT$,39):RETURN
5170 '
6000 CLS
6020 ' drawing the graph
6040 '
6055 ' calculating yymax & yymin
6060 YYMIN=YY(1):YYMAX=YY(1)
6065 FOR I=2 TO NXX
6070 IF YY(I)< YYMIN THEN YYMIN=YY(I)
6075 IF YY(I)>YYMAX THEN YYMAX=YY(I)
6080 NEXT
6085 '
6090 '
6095 ' drawing x and y axis
6100 LINE(33,140)-(283,140),1
6105 LINE(33,140)-(33,16),1
6110 NXXPIX=-(250/(11-1))
6115 NYYPIX=-(124/(NYY-1))
6120 '
6125 '
6130 ' loop for drawing xbars and xnumbers on graph
6135 FOR I=1 TO 11
6140 NXXPIX=NXXPIX+(250/(11-1))
6142 XXAXIS = INT( (I-1) * (XXMAX/(11-1))  )
6146 DUMMY1=DUMMY1+1
6147 IF DUMMY1=3 THEN DUMMY1=1
6148 IF DUMMY1=1 THEN NROW=19:LINE(33+NXXPIX,140)-(33+NXXPIX,145),1:NCOLUM=INT(
3+(NXXPIX/7.96)) ELSE NROW=20:LINE(33+NXXPIX,140)-(33+NXXPIX,155),1:NCOLUM=INT(
2+(NXXPIX/7.96))
6149 IF I=1 THEN LOCATE 19,3:PRINT XX!(1)
6150 LOCATE NROW,NCOLUM
6151 PRINT XXAXIS
6152 NEXT
6155 '
6160 '
```

```
6165 ' drawing the axis titles
6170 LOCATE 1,1
6175 GET 1,IREC1:GOSUB 5165:PRINT COMMT40$;
6180 LOCATE 2,1
6185 GET 1,IREC2:GOSUB 5165:PRINT COMMT40$;
6190 LOCATE 21,1
6195 GET 1,IREC3:GOSUB 5165:PRINT COMMT40$;
6200 '
6205 '
6210 'drawing the y numbers on graph
6215 IF YYMAX>YYHVAR OR YYMIN<YYLVAR THEN 6220 ELSE YYMAX=YYHVAR:YYMIN=YYLVAR:G
OTO 6220
6220 YYSPACE=((YYMAX-YYMIN)/(NYY-1))
6225 YYMIN2=YYMIN
6230 FOR I=1 TO NYY
6235 NYYPIX=NYYPIX+(124/(NYY-1))
6240 NROW=INT(18-(NYYPIX/7.96))
6245 IYYMIN$=STR$(YYMIN2)
6250 NCHAR=LEN(IYYMIN$)
6255 IF ASC(MID$(IYYMIN$,1,1))=32 THEN IYYMIN$=MID$(IYYMIN$,2,NCHAR-1):IYYMIN=V
AL(IYYMIN$)
6260 IF NCHAR>4 THEN IYYMIN$=MID$(IYYMIN$,1,4):IYYMIN=VAL(IYYMIN$)
6265 LOCATE NROW,1
6270 PRINT IYYMIN$
6275 YYMIN2=YYMIN2+YYSPACE
6280 LINE(26,140-NYYPIX)-(32,140-NYYPIX),1
6285 NEXT
6290 FOR I=1 TO NXX
6295 XX(I)=33+(((XX(I)-XXMIN)/(XXMAX-XXMIN))*250)
6300 YY(I)=140-(((YY(I)-YYMIN)/(YYMAX-YYMIN))*124)
6305 NEXT
6310 FOR I=2 TO NXX
6315 LINE(XX(I-1),YY(I-1))-(XX(I),YY(I)),3
6320 NEXT
6325 FOR I=1 TO NXX
6330 LINE(XX(I)-2,YY(I)-2)-(XX(I)+2,YY(I)+2),2
6335 LINE(XX(I)+2,YY(I)-2)-(XX(I)-2,YY(I)+2),2
6337 NEXT
6338 COOLXX=33+((COOLTI-XXMIN)/(XXMAX-XXMIN))*250
6339 YYLINE=140!
6340 FOR I=1 TO 13
6342 LINE(COOLXX,YYLINE)-(COOLXX,YYLINE-5),1:YYLINE=YYLINE-10:NEXT
6345 RETURN
6350 '*************************************************************************
```

ALPHAB

```
1                    ******* M E N U *******
2 1  Ball's Basic Method - log heating, hyp & log cooling
3 2  Modified Ball's Method - hyp & log heating, hyp & log cooling
4 3  More General Method - hyp & log heating, hyp & log cooling
5 4  Exit to MAIN Selection Menu (containing can/pouch choice)
6
7
8
9
10 Select an item from the menu (enter 1 to 4 and RETURN)
11 You did not enter a number between 1 and 4
12 PRESS ANY KEY TO CONTINUE
13 What is the Initial Temperature of the food (deg C) ?
14
15
16
17
18
19 The model number must be between 1 and 3
20 What is the Retort Temperature (deg C) ?
21 What is the Cooling Water Temperature (deg C) ?
22 There is a Problem with Temperature Inputs
23 The Initial and Cooling Water Temp. must be < = Retort Temp.
24 You entered the following values:
25 Initial Temperature=
26 Cooling Water Temperature=
27 Retort Temperature=
28 Please try again
29 What is the Total Modelling Time (heating and cooling time) in seconds ?
```

```
 30 At what time will Cooling start (in seconds) ?

 31 How many TIME values do you want (must be >=2,<=50 including TIME=0) ?

 32 There is a problem with the Time Inputs

 33 The Total Modelling Time and the time Cooling starts must be>0 & Cooling<=T
otal
 34 Time Cooling starts at =

 35 Total Modelling Time =

 36 You did not enter a number between 2 and 50

 37 What is the fh (1/slope) of the heating curve in seconds (>700)?

 38 There is a problem with the f (1/slope) of the Heating and/or Cooling curve
s
 39 The f (1/slope) must be >700

 40 The fh (1/slope) of heating curve=

 41 The fc (1/slope) of cooling curve=

 42 What is the fc (1/slope) of the cooling curve in seconds (>700)?

 43 What is the Heating curve Lag Factor (Jh>=0) ?

 44 Entering a Jh value between 0 and 1 will give a Heating curve Jump, not Lag

 45 Would you like to reenter the Jh value (Y or N and RETURN) ?

 46 A Jh value greater than 8.0 is very unusual

 47 Would you like to reenter the Jc value (Y or N and RETURN) ?

 48 What is the Heating curve Lag Factor (Jh>0) ?

 49 You entered Jh=0 ; for the model you've chosen Jh can't = 0

 50 What is the Cooling curve Lag Factor (Jc>0) ?

 51 You entered Jc=0 ; for the model you've chosen Jc can't = 0

 52 Entering a Jc value between 0 and 1 will give a Cooling curve Jump, not Lag

 53 A Jc value greater than 8.0 is very ususual

 54 Do you want to redo the calculations (Y or N and RETURN) ?

 55                  ARGUMENT LISTING FOR SUBROUTINE

 56 Enter the new value and press Return.

 57 Hit Return alone if you want to keep the old value.
```

```
58 Model  (1-basic,2-basic modified,3-general)
59 ARGUMENT                                          OLD          NEW
60 Initial Food Temperature(deg C)
61 Retort Temperature(deg C)
62 Cooling Water Temperature(deg C)
63 Total Processing Time(heating + cooling) sec
64 Cooling Time(sec)
65 Number of Time Values (>=2,<=50)
66 Heating Curve fh (1/slope) (>700 sec)
67 Cooling Curve fc (1/slope) (>700 sec)
68 Heating Curve Lag Factor (must=>0) (>1=lag,<1=jump)
69 Heating Curve Lag Factor (must>0) (>1=lag,<1=jump)
70 Cooling Curve Lag Factor (must>0) (>1=lag,<1=jump)
71 Model:
72 Initial Temp:  Retort Temp:  Water Temp:
73 Total Time:  Cooling Time:
74 Number of Time Values:
75 fh:          fc:
76 Lag Factor (heating curve):
77 Lag Factor (cooling curve):
78
79 The Cooling curve Lag Factor can't be between 1.856 to 1.876 inclusively
80 This is because of discontinuities in the formula in this range
81 The Heating curve Lag Factor can't be between 1.856 to 1.876 inclusively
82 What is the F-value for the contaminant (at Tref=121 deg C) - in minutes?
83 The F-value cannot equal 0
84 A F-value greater than 20 minutes is unusual
85 Do you want to reenter the value (Y or N and RETURN)
86 What is the z-value for the contaminant (at Tref=121 deg C) - in deg C?
```

```
87 The z-value cannot equal 0
88 A z-value greater than 30 deg C is unusual
89 What is the F-value for the nutrient (at Tref=121 deg C) - in minutes?
90 A F-value greater than 100 minutes is unusual
91 What is the z-value for the nutrient (at Tref=121 deg C) - in deg C?
92 A z-value greater than 50 deg C is unusual
93
94
95
96
97
98
99
100            RESULTS DISPLAY MENU
101            ____________________
102        1) Time-Temp Graph
103        7) Process Parameters
104        2) Time-Temp Table
105        8) Exit
106              ENTER 1, 2, ... 8
107 TEMP(C)
108            TEMP VS TIME - BALL'S METHOD
109                  TIME(sec)
110 Press "Y" for process parameters
111 Press "X" to exit or  -PrtSc to print
112 Press "Y" for Time-Temp Graph
113          PROCESS PARAMETER TABLE
114          ----------------------
```

```
115 Initial Temperature(deg C)
116 Retort Temperature(deg C)
117 Water Temperature(deg C)
118 Slope of Heating Curve(sec)
119 Slope of Cooling Curve(sec)
120 Heating Curve Lag Factor
121 Cooling Curve Lag Factor
122 Total Processing Time(sec)
123 Time When Cooling Starts(sec)
124 Number of Time Increments
125 ____________________________________________________________________
126 *                                                                  **
  **
127 *__________________________________________________________________**_____
  **
128                    Do you want to see the other page(Y/N)?
129              You did not enter Y or N -please try again
130                    Press -PrtSc to print this page
131                         There is more data to be seen
132            Press any other key to get back to results display menu
133          Press any key to get back to results display menu
134                    To print this page press -PrtSc
135
136 *            TIME (sec)                **       TEMPERATURE (deg C)
  **
137           LETHAL RATE MENU
138           ---------------
139         1) Lethality-Time Graph
140         3) Lethality-Time Table
141 LETHALITY
142          LETHALITY vs PROCESS TIME
143 *            TIME (sec)                **          LETHALITY
```

```
 **
144 F-Value of Microorganism(min)
145 z-Value of Microorganism(deg C)
146 F-value of Nutrient(min)
147 z-Value of Nutrient(deg C)
148       3) Lethality-Time Graph
149       4) Lethality-Time Table
150       5) Nutrient Fraction-Time Graph
151       6) Nutrient Fraction-Time Table
152 FRACTION
153           NUTRIENT FRACTION VS TIME
154  *             TIME  (sec)              **       NUTRIENT  FRACTION
  **
155
156
157                  You did not enter Y or N -please try again
158                       To print this page press  -PrtSc
159
160         You did not enter an integer
161            Please try again
162          You did not enter an integer
163
164    You did not enter X, try again
```

APPENDIX 3.5

Program Listing of MAINC

```
10 '
15 '
20 '
25 '          ********** M A I N C **********
30 '
35 '
40 'Initialization and Dimensioning
45 DEFINT I-N
50 DIM X(5), Y(5), Z(5), TIME(21)
55 DIM XX(21),YY(21),ZZ(21)
60 DIM TEMP(5,5,5,21)
65 DIM TCNT(64,21), TAVG(64,20), VELE(64)
70 DIM ORGEO(64), RATEL(64,20), ORGELE(64,20), XLETH(21)
75 DIM XNUTEO(64), XNUTE(64,20), FRANUT(21)
80 '
85 '
90 'Setting up the text screen .
95 KEY OFF
100 SCREEN 0,0,0,0
105 WIDTH 80
110 COLOR 7,1,1: CLS
115 '
120 '
125 ' opening and readying the associated files
130 OPEN "c:alphac" AS 1
135 FIELD 1,80 AS COMMENT$
140 '
145 OPEN "b:commc" AS 2
150 FIELD 2,80 AS COMMC$
155 '
160 '
165 ' setting the pointer to current record number of commc
170 GET 2,1: NRECCOUNT=VAL(COMMC$)
175 BLANK$=" ": LSET COMMC$=BLANK$: PUT 2,NRECCOUNT+1
180 '
185 '
190 ' Checking if control has come from MAIN or from SUBs
195 OPEN "b:store2" FOR INPUT AS 3
200 INPUT#3,NOPCODE
205 CLOSE 3
210 '
215 ' Not coming from MAIN:
220 IF NOPCODE<>0 THEN GOTO 1285
225 '
230 ' Coming from MAIN:
235 OPEN "b:comma" FOR INPUT AS 4
240 '
245 '    ** read info from comma**
250 '
255 CLOSE 4
260 '
265 '
270 ' Printing the menu
275 CLS: PRINT
280 GET 1,1:  PRINT COMMENT$
285 GET 1,2:  PRINT COMMENT$
290 GET 1,3:  PRINT COMMENT$ : PRINT
```

```
295 '
300 '
305 'determining user's selection from the menu
310 JRECNO=4
315 GOSUB 3045
320 IF JOK<>-1 THEN GOTO 270
325 IF (VAL(ANSI$)=2)  GOTO 370
330 IF (VAL(ANSI$)=1) GOTO 475
335 GET 1,5
340 PRINT COMMENT$
345 GET 1,12: PRINT COMMENT$
350 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 350
355 GOTO 270
360 '
365 '
370 'if want to go straight back to MAIN'
375 '
380 OPEN "b:commb" FOR OUTPUT AS 7
385 '
390 ' ** write info into b:commb **
395 '
400 CLOSE 7
405 '
410 CLOSE 1,2
415 '
420 CLS: SCREEN 1
425 COLOR 4,1
430 DEF SEG = &HB800
435 BLOAD "b:sandcl.scr",0
440 ON ERROR GOTO 445
445 IF INKEY$<>"" THEN BEEP ELSE GOTO 450
450 '
455 CHAIN "MAIN"
460 '
465 '
470 '
475 'if want to go to a POUCH sub
480 N1STTHRU=0
485 INDIC=0
490 'determining NSUB wanted
495 NOLD = NSUB-1 : IF NSUB=22 THEN NOLD=1
500 IF NSUB=33 THEN NOLD=2
505 IF NSUB=0 THEN INDIC=1: NOLD=1
510 JRECNO=7: NMARKSUB=0: GOSUB 3190
515 NEW1=VAL(NEWVALUE$)
520 IF (NEW1>=1) AND (NEW1<=2) THEN GOTO 535
525 JRECNO=8: JRECNO2=9: NMARKSET=1: GOSUB 3360
530 GOTO 495
535 NSUB=NEW1 + 1
540 '
545 'determining temperatures
550 IF INDIC=1 THEN OLD=40! ELSE OLD=TINIT
555 JRECNO=10:NMARKSUB=1: GOSUB 3190
560 XNEW1=VAL(NEWVALUE$)
565 IF INDIC=1 THEN OLD=120! ELSE OLD=TRETRT
570 JRECNO=11:NMARKSUB=1: GOSUB 3190
575 XNEW2=VAL(NEWVALUE$)
```

```
580 IF INDIC=1 THEN OLD=60! ELSE OLD=TWATER
585 JRECNO=6:NMARKSUB=1: GOSUB 3190
590 XNEW3=VAL(NEWVALUE$)
595 IF (XNEW2>=XNEW1) AND (XNEW2>=XNEW3)THEN GOTO 610
600 JRECNO=13: JRECNO2=14: NMARKSET=3: GOSUB 3360
605 GOTO 550
610 TINIT=XNEW1: TRETRT=XNEW2: TWATER=XNEW3
615 '
620 'determining time values
625 IF INDIC=1 THEN OLD=5600! ELSE OLD=TOTIME
630 JRECNO=15: NMARKSUB=1: GOSUB 3190
635 XNEW1=VAL(NEWVALUE$)
640 IF INDIC=1 THEN OLD=3600! ELSE OLD=COOLTI
645 JRECNO=16: NMARKSUB=1: GOSUB 3190
650 XNEW2=VAL(NEWVALUE$)
655 IF (XNEW1>0) AND (XNEW1<=10000) AND (XNEW2>0) AND (XNEW1>=XNEW2) THEN GOTO
670
660 JRECNO=17: JRECNO2=18: NMARKSET=2: GOSUB 3360
665 GOTO 625
670 TOTIME=XNEW1: COOLTI=XNEW2
675 '
680 'determining number of time values
685 IF INDIC=1 THEN NOLD=11 ELSE NOLD=NTIME
690 JRECNO=19: NMARKSUB=0: GOSUB 3190
695 NEW1=VAL(NEWVALUE$)
700 IF (NEW1>=2) AND (NEW1<=21) THEN GOTO 715
705 JRECNO=20: JRECNO2=21: NMARKSET=1: GOSUB 3360
710 GOTO 685
715 NTIME=NEW1
720 '
725 'determining thermal diffusivity
730 IDIFF=1
735 IF INDIC=1 THEN OLD=1.6E-07 ELSE OLD=ALPHA
740 JRECNO=22: NMARKSUB=1: GOSUB 3190
745 XNEW1=VAL(NEWVALUE$)
750 IF (XNEW1=>.0000001) and (xnew1=<9.9e-07) THEN GOTO 765
755 JRECNO=23: JRECNO2=44: NMARKSET=1: GOSUB 3360
760 GOTO 735
765 ALPHA=XNEW1 : IDIFF=0
770 '
775 'determining food density
780 IF NSUB=2 THEN GOTO 870
785 IF (RHO=0!) THEN OLD=1500! ELSE OLD=RHO
790 JRECNO=25: NMARKSUB=1: GOSUB 3190
795 XNEW1=VAL(NEWVALUE$)
800 IF (XNEW1>0) THEN GOTO 815
805 JRECNO=26: JRECNO2=27: NMARKSET=1: GOSUB 3360
810 GOTO 785
815 RHO=XNEW1
820 '
825 ' determining specific heat
830 IF (CP=0!) THEN OLD=4000! ELSE OLD=CP
835 JRECNO=28: NMARKSUB=1: GOSUB 3190
840 XNEW1=VAL(NEWVALUE$)
845 IF (XNEW1>0) THEN GOTO 860
850 JRECNO=29: JRECNO2=30: NMARKSET=1: GOSUB 3360
855 GOTO 830
```

```
860 CP=XNEW1
865 '
870 'determining half-lengths of the pouch
875 IF INDIC=1 THEN OLD=.04 ELSE OLD=HALFX
880 JRECNO=31:NMARKSUB=1: GOSUB 3190
885 XNEW1=VAL(NEWVALUE$)
890 IF INDIC=1 THEN OLD=.05 ELSE OLD=HALFY
895 JRECNO=32: NMARKSUB=1: GOSUB 3190
900 XNEW2=VAL(NEWVALUE$)
905 IF INDIC=1 THEN OLD=.02 ELSE OLD=HALFZ
910 JRECNO=33: NMARKSUB=1: GOSUB 3190
915 XNEW3=VAL(NEWVALUE$)
920 IF (XNEW1>0) AND (XNEW2>0) AND (XNEW3>0) GOTO 935
925 JRECNO=34: JRECNO2=35: NMARKSET=3: GOSUB 3360
930 GOTO 875
935 HALFX=XNEW1: HALFY=XNEW2: HALFZ=XNEW3
940 '
945 'determining the nx,ny,nz values
950 IF INDIC=1 THEN NOLD=3 ELSE NOLD=NX
955 JRECNO=36: NMARKSUB=0: GOSUB 3190
960 NEW1=VAL(NEWVALUE$)
965 IF INDIC=1 THEN NOLD=3 ELSE NOLD=NY
970 JRECNO=37: NMARKSUB=0: GOSUB 3190
975 NEW2=VAL(NEWVALUE$)
980 IF INDIC=1 THEN NOLD=3 ELSE NOLD=NZ
985 JRECNO=38: NMARKSUB=0: GOSUB 3190
990 NEW3=VAL(NEWVALUE$)
995 IF (NEW1>=2) AND (NEW1<=5) AND (NEW2>=2) AND (NEW2<=5) AND (NEW3>=2) AND (N
EW3<=5) GOTO 1010
1000 JRECNO=39:JRECNO2=40: NMARKSET=3: GOSUB 3360
1005 GOTO 950
1010 NX=NEW1: NY=NEW2: NZ=NEW3
1015 '
1020 'determining the convective surface heat transfer coefficient
1025 IF NSUB=2 GOTO 1075
1030 IF H=0! THEN OLD=500! ELSE OLD=H
1035 JRECNO=41: NMARKSUB=1: GOSUB 3190
1040 XNEW1=VAL(NEWVALUE$)
1045 IF (XNEW1>0) GOTO 1060
1050 JRECNO=42: JRECNO2=43: NMARKSET=1: GOSUB 3360
1055 GOTO 1030
1060 H=XNEW1
1065 '
1070 '
1075 'Putting info into STORE2 to tell MAINB what SUB it chained to
1080 NSUBMAIN=3
1085 OPEN "b:store2" FOR OUTPUT AS 3
1090 IF NSUB=3 GOTO 1110
1095 PRINT#3,NSUB
1100 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,
COOLTI,NSUBMAIN
1105 GOTO 1120
1110 PRINT#3,NSUB
1115 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,
COOLTI,RHO,CP,H,NSUBMAIN
1120 CLOSE 3
1125 '
```

```
1130 '
1135 'Putting arguments into COMM1 to run the SUB to get the heating results
1140 OPEN "c:comm1" FOR OUTPUT AS 6
1145 IF NSUB=3 GOTO 1160
1150 PRINT#6,TRETRT,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,NSUBMAI
N
1155 GOTO 1165
1160 PRINT#6,TRETRT,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,RHO,CP,
H,NSUBMAIN
1165 CLOSE 6
1170 '
1175 '
1180 'Preparing to chain to a POUCH sub
1185 'storing the current record number of COMMC into the first rec no of COMMC

1190 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
1195 '
1200 '
1205 CLOSE 1,2
1210 '
1215 'Displaying "Please Wait" symbol
1220 CLS:SCREEN 1,0,0,0
1225 COLOR 5,1
1230 DEF SEG = &HB800
1235 BLOAD "b:clock.scr",0
1240 ON ERROR GOTO 1245
1245 IF INKEY$<>"" THEN BEEP ELSE GOTO 1250
1250 '
1255 'Chaining to the SUB
1260 IF NSUB=2 THEN CHAIN "pouch1"
1265 IF NSUB=3 THEN CHAIN "pouch2"
1270 PRINT"Problems with nsub in MAINC":STOP
1275 '
1280 '
1285 'Read the information in STORE2 to determine if this is heating or cooling

1287 cls:locate 12,1: get 1,199: print comment$
1290 OPEN "b:store2" FOR INPUT AS 3
1295 INPUT#3, NSUB
1300 IF NSUB=22 OR NSUB=33 GOTO 1655
1305 IF NSUB=3 GOTO 1320
1310 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,
COOLTI,NSUBMAIN
1315 GOTO 1325
1320 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,TOTIME,
COOLTI,RHO,CP,H,NSUBMAIN
1325 CLOSE 3
1330 '
1335 '
1340 'Reading the heating results from COMM2
1345 OPEN "c:comm2" FOR INPUT AS 7
1350 FOR LMC=1 TO NTIME
1355 FOR IMC=1 TO NX
1360 FOR JMC=1 TO NY
1365 FOR KMC=1 TO NZ
1370 INPUT#7,X(IMC),Y(JMC),Z(KMC),TIME(LMC),TEMP(IMC,JMC,KMC,LMC)
1375 NEXT KMC,JMC,IMC,LMC
```

```
1380 CLOSE 7
1385 '
1390 '
1395 IF TOTIME = COOLTI THEN GOTO 1800 'don't need to do cooling part
1400 'Storing arguments in COMM1 to get the cooling results
1405 '(first calculate the "ntime" for the cooling portion alone)
1410 NCTIME = NTIME - FIX(  (TOTIME-COOLTI)/ (TOTIME/(NTIME-1)) )
1415 NCOOLT = NTIME - NCTIME + 1       '(the "ntime" for the cooling part)
1417 if ncoolt=1 then ncoolt=2: nctime=nctime-1
1420 OPEN "c:comm1" FOR OUTPUT AS 6
1425 IF NSUB=3 GOTO 1440
1430 PRINT#6,TWATER,TRETRT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NCOOLT,TOTIME-COOLT
I,NSUBMAIN
1435 GOTO 1445
1440 PRINT#6,TWATER,TRETRT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NCOOLT,TOTIME-COOLT
I,RHO,CP,H,NSUBMAIN
1445 CLOSE 6
1450 '
1455 '
1460 'Storing codes, parameters in STORE2
1465 OPEN "b:store2" FOR OUTPUT AS 3
1470 IF NSUB=2 THEN NOPCODE=22: PRINT#3,NOPCODE: GOTO 1480
1475 IF NSUB=3 THEN NOPCODE=33: PRINT#3,NOPCODE: GOTO 1490
1480 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,NCTIME,
TOTIME,COOLTI,NSUBMAIN
1485 GOTO 1505
1490 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,NCTIME,
TOTIME,COOLTI,RHO,CP,H,NSUBMAIN
1495 '
1500 '
1505 'Storing the heating results in STORE2 so can have them when chain back to
 MAINC in cooling section
1510 FOR LMC=1 TO NTIME
1515 FOR IMC=1 TO NX
1520 FOR JMC=1 TO NY
1525 FOR KMC=1 TO NZ
1530 PRINT#3,TEMP(IMC,JMC,KMC,LMC)
1535 NEXT KMC,JMC,IMC,LMC
1540 CLOSE 3
1545 '
1550 '
1555 'Preparing to chain to a POUCH sub
1560 'storing the current record number of COMMC into the first rec no of COMMC

1565 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
1570 '
1575 '
1580 CLOSE 1,2
1585 '
1590 'Displaying "Please Wait" symbol
1595 CLS:SCREEN 1,0,0,0
1600 COLOR 5,1
1605 DEF SEG = &HB800
1610 BLOAD "b:clock.scr",0
1615 ON ERROR GOTO 1245
1620 IF INKEY$<>"" THEN BEEP ELSE GOTO 1250
1625 '
```

```
1630 'Chaining to the SUB
1635 IF NSUB=2 OR NSUB=22 THEN CHAIN "pouch1"
1640 IF NSUB=3 OR NSUB=33 THEN CHAIN "pouch2"
1645 '
1650 '
1655 'Reading the rest of the parameters from STORE2
1660 IF NOPCODE=22 THEN INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,
NY,NZ,NTIME,NCTIME,TOTIME,COOLTI,NSUBMAIN: GOTO 1670
1665 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFX,HALFY,HALFZ,NX,NY,NZ,NTIME,NCTIME,
TOTIME,COOLTI,RHO,CP,H,NSUBMAIN
1670 '
1675 '
1680 'Reading the heating results from STORE2
1685 FOR LMC=1 TO NTIME
1690 FOR IMC=1 TO NX
1695 FOR JMC=1 TO NY
1700 FOR KMC=1 TO NZ
1705 INPUT#3,TEMP(IMC,JMC,KMC,LMC)
1710 NEXT KMC,JMC,IMC,LMC
1715 CLOSE 3
1720 '
1725 '
1730 'Reading the TCOOL value from COMM2 one at a time = the cool temp
1735 ' and calculating the true cooling curve temperatures
1740 OPEN "c:comm2" FOR INPUT AS 7
1745 FOR LMC=NCTIME TO NTIME
1750 FOR IMC=1 TO NX
1755 FOR JMC=1 TO NY
1760 FOR KMC=1 TO NZ
1765 INPUT#7,X(IMC),Y(JMC),Z(KMC),TIME(LMC),TCOOL
1770 TIME(LMC) = TIME(LMC) + COOLTI
1775 TEMP(IMC,JMC,KMC,LMC)=TEMP(IMC,JMC,KMC,LMC) - (TRETRT-TCOOL)
1780 NEXT KMC,JMC,IMC,LMC
1785 CLOSE 7
1790 '
1795 '
1800 'Getting the FLETH value
1805 CLS: JRECNO=45: GOSUB 2885
1810 IF JOK<>-1 THEN GOTO 1805 ELSE FLETH=VAL(ANSR$)
1815 IF (FLETH>0!) AND (FLETH<=20!) GOTO 1875
1820 IF FLETH>20! THEN GOTO 1840
1825 GET 1,46: PRINT COMMENT$
1830 GET 1,12: PRINT: PRINT COMMENT$
1835 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1835 ELSE GOTO 1805
1840 CLS: GET 1,47: PRINT COMMENT$
1845 GET 1,24: PRINT COMMENT$;FLETH
1850 JRECNO=48: GOSUB 2760
1855 IF JOK<>-1 GOTO 1840
1860 IF (DY$="n") OR (DY$="N") GOTO 1875 ELSE GOTO 1805
1865 '
1870 '
1875 'Getting the ZLETH value
1880 CLS: JRECNO=49: GOSUB 2885
1885 IF JOK<>-1 THEN GOTO 1880 ELSE ZLETH=VAL(ANSR$)
1890 IF (ZLETH>0!) AND (ZLETH<=30!) GOTO 1950
1895 IF ZLETH>30! THEN GOTO 1915
1900 GET 1,50: PRINT COMMENT$
```

```
1905 GET 1,12: PRINT: PRINT COMMENT$
1910 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1910 ELSE GOTO 1880
1915 CLS: GET 1,51: PRINT COMMENT$
1920 GET 1,24: PRINT COMMENT$;ZLETH
1925 JRECNO=48: GOSUB 2760
1930 IF JOK<>-1 GOTO 1915
1935 IF (DY$="n") OR (DY$="N") GOTO 1950 ELSE GOTO 1880
1940 '
1945 '
1950 'Getting the FXNUT value
1955 CLS: JRECNO=52: GOSUB 2885
1960 IF JOK<>-1 THEN GOTO 1955 ELSE FXNUT=VAL(ANSR$)
1965 IF (FXNUT>0!) AND (FXNUT<=100!) GOTO 2025
1970 IF FXNUT>100! THEN GOTO 1990
1975 GET 1,46: PRINT COMMENT$
1980 GET 1,12: PRINT: PRINT COMMENT$
1985 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1985 ELSE GOTO 1955
1990 CLS: GET 1,53: PRINT COMMENT$
1995 GET 1,24: PRINT COMMENT$;FXNUT
2000 JRECNO=48: GOSUB 2760
2005 IF JOK<>-1 GOTO 1990
2010 IF (DY$="n") OR (DY$="N") GOTO 2025 ELSE GOTO 1955
2015 '
2020 '
2025 'Getting the Znut value
2030 CLS: JRECNO=54: GOSUB 2885
2035 IF JOK<>-1 THEN GOTO 2030 ELSE ZNUT=VAL(ANSR$)
2040 IF (ZNUT>0!) AND (ZNUT<=50!) GOTO 2100
2045 IF ZNUT>50! THEN GOTO 2065
2050 GET 1,50: PRINT COMMENT$
2055 GET 1,12: PRINT: PRINT COMMENT$
2060 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 2060 ELSE GOTO 2030
2065 CLS: GET 1,60: PRINT COMMENT$
2070 GET 1,24: PRINT COMMENT$;ZNUT
2075 JRECNO=48: GOSUB 2760
2080 IF JOK<>-1 GOTO 2065
2085 IF (DY$="n") OR (DY$="N") GOTO 2100 ELSE GOTO 2030
2090 '
2095 '
2100 '*****************************************************************************
2105 'calculating lethality and nutrient fraction retained
2110 CLS: LOCATE 12,1 : GET 1,199: PRINT COMMENT$
2115 '
2120 LET CORG=10!^(37!): LET CNUT=10!^(37!)     'arbitrarily set
2125 IF RHO=0! THEN RHO=1500!                   'when nsub=2
2130 '
2135 'number of elements:
2140 NELE = (NX-1) * (NY-1) * (NZ-1)
2145 '
2150 'Total and elemental volumes
2155 VTOTP = (2!*HALFX) * (2!*HALFY) * (2!*HALFZ)
2160 FOR MMC=1 TO NELE
2165 VELE(MMC) = (HALFX/(NX-1)) * (HALFY/(NY-1)) * (HALFZ/(NZ-1))
2170 NEXT MMC
2175 '
2180 'calculate original number of organisms in pouch, and in each element
2185 ORGO = (1!/8!*VTOTP) * CORG * RHO
```

```
2190 FOR MMC=1 TO NELE
2195 ORGE0(MMC) = VELE(MMC) * CORG * RHO
2200 NEXT MMC
2205 '
2210 'calculate original amount of nutrient in pouch, and in each element
2215 XNUT0 = (1!/8!*VTOTP) * CNUT * RHO
2220 FOR MMC=1 TO NELE
2225 XNUTE0(MMC) = VELE(MMC) * CNUT * RHO
2230 NEXT MMC
2235 '
2240 '
2245 'calculate TCNT (average of eight nodal temps, at each time value)
2250 FOR LMC = 1 TO NTIME
2255 MCOUNT=0
2260  FOR IMC = 1 TO (NX-1)
2265   FOR JMC=1 TO (NY-1)
2270      FOR KMC=1 TO (NZ-1)
2275      MCOUNT=MCOUNT+1
2280      ATCNT=TEMP(IMC,JMC,KMC,LMC)+TEMP(IMC,JMC,KMC+1,LMC)+TEMP(IMC,JMC+1,K
MC,LMC)+TEMP(IMC,JMC+1,KMC+1,LMC)
2285      BTCNT=TEMP(IMC+1,JMC,KMC,LMC)+TEMP(IMC+1,JMC,KMC+1,LMC)+TEMP(IMC+1,J
MC+1,KMC,LMC)+TEMP(IMC+1,JMC+1,KMC+1,LMC)
2290      TCNT(MCOUNT,LMC) = (ATCNT + BTCNT ) / 8!
2295      NEXT KMC
2300   NEXT JMC
2305  NEXT IMC
2310 NEXT LMC
2315 '
2320 'calculate TAVG (the average TCNT at start and end of time increment)
2325 FOR LL=1 TO (NTIME-1)
2330   FOR MMC = 1 TO NELE
2335   TAVG(MMC,LL)=(TCNT(MMC,LL) + TCNT(MMC,LL+1))/2!
2340   NEXT MMC
2345 NEXT LL
2350 '
2355 'calculate the lethal rate, the total number of organisms left
2360 'at end of time increment, and lethality at end of time increment
2365 DTIME = TOTIME /(NTIME-1)/60!     'to convert into minutes like F,z value
s
2370 XLETH(1)=0!
2375 FOR LL=1 TO (NTIME-1)
2380 ORGTOT = 0!
2385    FOR MMC=1 TO NELE
2390    IF (TAVG(MMC,LL)-121!)/ZLETH <-37! THEN RATEL(MMC,LL)=0!: GOTO 2400
2395    RATEL(MMC,LL)=(1!/FLETH) * 10^((TAVG(MMC,LL)-121!)/ZLETH)
2400    IF LL>1 GOTO 2420
2405    IF (-12!*RATEL(MMC,LL)*DTIME)<-37! THEN ORGELE(MMC,1)=10!^-37!: GOTO 2
430
2410    ORGELE(MMC,1)= ORGE0(MMC) * 10!^(-12!*RATEL(MMC,LL)*DTIME)
2415    GOTO 2430
2420    IF (-12!*RATEL(MMC,LL)*DTIME)<-37! THEN ORGELE(MMC,LL)=10!^-37!: GOTO
2430
2425    ORGELE(MMC,LL)=ORGELE(MMC,LL-1) * 10!^(-12!*RATEL(MMC,LL)*DTIME)
2430    ORGTOT= ORGTOT + ORGELE(MMC,LL)
2435    NEXT MMC
2440 IF ORGTOT<1! THEN ORGTOT=1!
2445 XLETH(LL+1)= .434294482# * LOG(ORG0/ORGTOT) / 12!
```

```
2450 NEXT LL
2455 '
2460 '
2465 'calculate the nutrient destruction rate, the total amount of nutrient
2470 'left at the end of time increment, and nutrient fraction retained at
2475 'end of each time increment
2480 FOR LL=1 TO (NTIME-1)
2485 FRANUT(1)=1!
2490 XNUTOT = 0!
2495    FOR MMC=1 TO NELE
2500    IF (TAVG(MMC,LL)-121)/ZNUT < -37! THEN RATEL(MMC,LL)=0!:GOTO 2510
2505    RATEL(MMC,LL)=(1!/FXNUT) * 10^((TAVG(MMC,LL)-121!)/ZNUT)
2510    IF LL>1 GOTO 2530
2515    IF (12!*RATEL(MMC,LL)*DTIME)>37! THEN XNUTE(MMC,1)=10!^-37!:GOTO 2540
2520    XNUTE(MMC,1)= XNUTEO(MMC) * 10!^(-12!*RATEL(MMC,LL)*DTIME)
2525    GOTO 2540
2530    IF (12!*RATEL(MMC,LL)*DTIME)>37! THEN XNUTE(MMC,LL)=10!^-37!:GOTO 2540

2535    XNUTE(MMC,LL)=XNUTE(MMC,LL-1) * 10!^(-12!*RATEL(MMC,LL)*DTIME)
2540    XNUTOT = XNUTOT + XNUTE(MMC,LL)
2545    NEXT MMC
2550 IF XNUTOT<1! THEN XNUTOT=1!
2555 FRANUT(LL+1)= (XNUTOT/ XNUTO)
2560 NEXT LL
2565 '
2570 '*************************************************************************
****
2575 '
2580 '
2585 'Going to the results display section
2590 GOTO 3695
2595 '
2600 '
2605 'Back from results display, ask if want to go to MAIN or back to SUB
2610 CLS: JRECNO=61: GOSUB 2760
2615 IF JOK<>-1 THEN GOTO 2610
2620 IF (DY$="y") OR (DY$="Y") THEN GOTO 475
2625 '
2630 '
2635 'when want to go to MAIN:
2640 OPEN "b:commb" FOR OUTPUT AS 7
2645 '
2650 '
2655 '*** write info to commb
2660 CLOSE 7
2665 '
2670 '
2675 'storing the current record number of COMMC into the first rec no of COMMC

2680 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
2685 CLOSE 1,2
2690 '
2695 '
2700 CLS: SCREEN 1
2705 COLOR 1,1
2710 DEF SEG = &HB800
2715 BLOAD "b:sandcl.scr",0
```

```
2720 ON ERROR GOTO 2725
2725 IF INKEY$<>"" THEN BEEP ELSE GOTO 2730
2730 '
2735 '
2740 CHAIN "main"
2745 '
2750 '
2755 '
2760 '
2765 '
2770 '
2775 '
2780 'getting and checking  Y/N SUBROUTINE
2785 JOK=0
2790 GET 1,JRECNO
2795 COMMENT1Y$="You did not enter Y or N -    You entered: "
2800 COMMENT2Y$="You did not enter a value"
2805 PRINT COMMENT$;
2810 LINE INPUT "",ANSY$
2815 JY=0
2820 JY = JY+1
2825 DY$=MID$(ANSY$,JY,1)
2830 IF DY$="" THEN GOTO 2865
2835 IF DY$=" " THEN GOTO 2820
2840 IF (DY$="Y") OR (DY$="y") OR (DY$="N") OR (DY$="n") THEN JOK=-1: RETURN
2845 PRINT COMMENT1Y$,ANSY$
2850 GET 1,12: PRINT: PRINT COMMENT$
2855 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2855
2860 RETURN
2865 PRINT COMMENT2Y$
2870 GET 1,12: PRINT: PRINT COMMENT$
2875 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2875
2880 RETURN
2885 '
2890 '
2895 '
2900 ' getting and CHECKING FOR  REAL  NUMBERS
2905 JOK=0
2910 GET 1,JRECNO
2915 COMMENT1R$="You did not enter a REAL number  -  You entered: "
2920 COMMENT2R$="you did not enter a value"
2925 PRINT COMMENT$;
2930 LINE INPUT"",ANSR$
2935 JFLAGR=0                          'flag to check that number exists
2940 IFLAGR=0                          'flag to check for decimal points
2945 JR=1
2950 KR=LEN(ANSR$)
2955 IF KR=0 THEN GOTO 3025
2960 WHILE JR<=KR
2965 DR$=MID$(ANSR$,JR,1)
2970 IF DR$=" " THEN GOTO 3010
2975 IF (ASC(DR$)>47) AND (ASC(DR$)<58) THEN JFLAGR=-1: GOTO 3010
2980 IF DR$="." THEN IFLAGR=IFLAGR+1: GOTO 3005
2985 PRINT COMMENT1R$,ANSR$
2990 GET 1,12: PRINT: PRINT COMMENT$
2995 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 2995
3000 RETURN
```

```
3005 IF IFLAGR>1 GOTO 2985
3010 JR=JR+1
3015 WEND
3020 IF (DR$<>" ") OR (JFLAGR=-1) THEN JOK=-1: RETURN
3025 PRINT COMMENT2R$
3030 GET 1,12: PRINT: PRINT COMMENT$
3035 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 3035
3040 RETURN
3045 '
3050 '
3055 '
3060 'getting and CHECKING FOR  INTEGER  NUMBERS
3065 JOK=0
3070 IFLAGI=0
3075 GET 1,JRECNO
3080 COMMENT1I$="You did not enter an INTEGER value  -  You entered: "
3085 COMMENT2I$="you did not enter a value"
3090 PRINT COMMENT$;
3095 LINE INPUT "",ANSI$
3100 NI=LEN(ANSI$)
3105 IF NI=0 THEN GOTO 3170
3110 JI=1
3115 WHILE JI<=NI
3120 DI$=MID$(ANSI$,JI,1)
3125 IF DI$=" " THEN GOTO 3155
3130 IF (ASC(DI$)>47) AND (ASC(DI$)<58) THEN IFLAGI=-1: GOTO 3155
3135 PRINT COMMENT1I$,ANSI$
3140 GET 1,12: PRINT: PRINT COMMENT$
3145 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 3145
3150 RETURN
3155 JI=JI+1
3160 WEND
3165 IF (DI$<>" ") OR (IFLAGI=-1) THEN JOK=-1: RETURN
3170 PRINT COMMENT2I$
3175 GET 1,12: PRINT: PRINT COMMENT$
3180 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 3180
3185 RETURN
3190 '
3195 '
3200 '
3205 '
3210 ' Table Creation subroutine
3215 IF N1STTHRU=1 GOTO 3265
3220 NROW=7:NCOLUMN=1
3225 SCREEN 0,0,1,1
3230 CLS
3235 'printing table heading
3240 GET 1,55:PRINT COMMENT$
3245 GET 1,56:PRINT COMMENT$;
3250 GET 1,57:PRINT COMMENT$
3255 GET 1,59:PRINT COMMENT$
3260 N1STTHRU=1
3265 LOCATE NROW,NCOLUMN
3270 GET 1,JRECNO
3275 BLANK$="    "
3280 NBLANK=INSTR(COMMENT$,BLANK$)
3285 FINAL$=LEFT$(COMMENT$,NBLANK-1)
```

```
3290 IF IDIFF=1 THEN DIFF$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(67);:
GOTO 3300
3295 IF NMARKSUB=1 THEN PRINT FINAL$;TAB(53);OLD;TAB(67); ELSE PRINT FINAL$;TAB
(53);NOLD;TAB(67);
3300 LINE INPUT NEWVALUE$
3305 NRTEMP=CSRLIN: NCTEMP=POS(0)
3310 IF LEN(NEWVALUE$)=0 THEN GOTO 3335
3315 IF NMARKSUB=1 THEN GOSUB 3495 ELSE GOSUB 3605
3320 IF (NOK=-1) THEN NROW=CSRLIN: NCOLUMN=POS(0): RETURN
3325 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
3330 GOTO 3290
3335 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
3340 IF IDIFF=1 THEN NEWVALUE$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(6
7);VAL(NEWVALUE$):GOTO 3350
3345 IF NMARKSUB=1 THEN NEWVALUE$=STR$(OLD):PRINT FINAL$;TAB(53);OLD;TAB(67);VA
L(NEWVALUE$) ELSE NEWVALUE$=STR$(NOLD):PRINT FINAL$;TAB(53);NOLD;TAB(67);VAL(NE
WVALUE$)
3350 NROW=CSRLIN: NCOLUMN=POS(0)
3355 RETURN
3360 '
3365 '
3370 '
3375 '
3380 ' Erase Lines in table and show error page subroutine
3385 SCREEN ,,2,2
3390 CLS
3395 GET 1,JRECNO: PRINT COMMENT$;
3400 GET 1,24: PRINT COMMENT$;
3405 GET 1,JRECNO2: PRINT COMMENT$;
3410 IF NMARKSET<>3 THEN GOTO 3420
3415 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2,XNEW3:GOTO 3435 ELSE PRINT NEW1,NEW2,
NEW3:GOTO 3435
3420 IF NMARKSET<>2 THEN GOTO 3430
3425 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2:GOTO 3435 ELSE PRINT NEW1,NEW2: GOTO
3435
3430 IF NMARKSUB=1 THEN PRINT XNEW1 ELSE PRINT NEW1
3435 GET 1,12
3440 PRINT: PRINT COMMENT$
3445 DUMMEL$=INKEY$: IF DUMMEL$="" THEN GOTO 3445
3450 SCREEN ,,1,1
3455 LOCATE NROW,NCOLUMN
3460 LOCATE NROW-1,NCOLUMN:PRINT "
                                  ": LOCATE NROW-1,NCOLUMN
3465 IF NMARKSET=1 THEN GOTO 3485
3470 LOCATE NROW-2,NCOLUMN:PRINT "
                                  ":LOCATE NROW-2,NCOLUMN
3475 IF NMARKSET=2 THEN GOTO 3485
3480 LOCATE NROW-3,NCOLUMN:PRINT "
                                  ": LOCATE NROW-3,NCOLUMN
3485 NROW=CSRLIN: NCOLUMN=POS(0)
3490 RETURN
3495 '
3500 '
3505 '
3510 '
```

```
3515 'checking Real2 numbers (in table)
3520 NOK=0
3525 JFLAGR2=0
3530 IFLAGR2=0
3535 JR2=1
3540 NR2=LEN(NEWVALUE$)
3545 WHILE JR2=<NR2
3550 DR2$=MID$(NEWVALUE$,JR2,1)
3555 IF DR2$=" " THEN GOTO 3580
3560 IF DR2$="." THEN IFLAGR2=IFLAGR2+1: GOTO 3580
3565 IF (ASC(DR2$)>47) AND (ASC(DR2$)<58) THEN JFLAGR2=-1: GOTO 3580
3570 NOK=0
3575 RETURN
3580 IF IFLAGR2>1 THEN NOK=0: RETURN
3585 JR2=JR2+1
3590 WEND
3595 IF JFLAGR2=0 THEN NOK=0 ELSE NOK=-1
3600 RETURN
3605 '
3610 '
3615 '
3620 '
3625 ' checking Integer2 numbers (in table)
3630 NOK=0
3635 IFLAGI2=0
3640 NI2=LEN(NEWVALUE$)
3645 JI2=1
3650 WHILE JI2<=NI2
3655 DI2$=MID$(NEWVALUE$,JI2,1)
3660 IF DI2$=" " THEN GOTO 3675
3665 IF (ASC(DI2$)>47) AND (ASC(DI2$)<58) THEN IFLAGI2=-1: GOTO 3675
3670 NOK=0: RETURN
3675 JI2=JI2+1
3680 WEND
3685 IF IFLAGI2=0 THEN NOK=0 ELSE NOK=-1
3690 RETURN
3695 '*****************************************************************************

3700 '
3705 '
3710 '                                  RESULTS DISPLAY SECTION
3715 '
3720 NXX=NX
3725 NYY=NY:NZZ=NZ
3730 '
3735 '
3740 ' initializing the values
3745 FOR I=1 TO NXX:XX(I)=X(I):NEXT
3750 FOR I=1 TO NYY:YY(I)=Y(I):NEXT
3755 FOR I=1 TO NZZ:ZZ(I)=Z(I):NEXT
3760 '
3765 '
3770 ' this is the pouch results displays
3775 CLS:SCREEN 1: COLOR 9,1: KEY OFF
3780 LOCATE 2,1:GET 1,150:GOSUB 5215:PRINT COMMT40$;
3785 LOCATE 6,1:GET 1,151:GOSUB 5215:PRINT COMMT40$;
3790 LOCATE 8,1:GET 1,152:GOSUB 5215:PRINT COMMT40$;
```

```
3795 LOCATE 10,1:GET 1,153:GOSUB 5215:PRINT COMMT40$;
3800 LOCATE 12,1:GET 1,218:GOSUB 5215:PRINT COMMT40$;
3805 LOCATE 14,1:GET 1,219:GOSUB 5215:PRINT COMMT40$;
3810 LOCATE 16,1:GET 1,220:GOSUB 5215:PRINT COMMT40$;
3815 LOCATE 18,1:GET 1,245:GOSUB 5215:PRINT COMMT40$;
3820 LOCATE 20,1:GET 1,221:GOSUB 5215:PRINT COMMT40$;
3825 LOCATE 23,1:GET 1,154:GOSUB 5215:PRINT COMMT40$;
3830 GOSUB 5105
3835 IF(VAL(ANSWERI$)>=1) AND (VAL(ANSWERI$)<=8) THEN ICHOICE=VAL(ANSWERI$):GOT
O 3845
3840 LOCATE 24,1:BEEP:GET 1,251:GOSUB 5215:PRINT COMMT40$;:GOTO 3830
3845 IF ICHOICE=1 GOTO 3885
3850 IF ICHOICE=2 GOTO 3995
3855 IF ICHOICE=3 GOTO 4055
3860 IF ICHOICE=4 GOTO 4235
3865 IF ICHOICE=5 GOTO 4115
3870 IF ICHOICE=6 GOTO 4175
3875 IF ICHOICE=7 GOTO 4710
3880 IF ICHOICE=8 GOTO 4845
3885 '
3890 '
3895 ' graphic display section
3900 '
3905 ' displaying the x, y, z length increments
3910 JRECNO=155
3915 GOSUB 5065
3920 ' inputting graphic parameter
3925 GOSUB 5265
3930 '
3935 ' going to the temp-time graph
3940 IRECNO2=197 :IRECNO3=196:IRECNO1=215
3945 XXMIN=0:XXMAX=TOTIME:N=NTIME
3950 YYHVAR=150!:YYLVAR=0!
3955 DXX=(XXMAX/(N-1))
3960 FOR I=1 TO N
3965 XX(I)=(I-1)*DXX:NEXT
3970 FOR I=1 TO NTIME
3975 YY(I)=TEMP(XPOSITION,YPOSITION,ZPOSITION,I):NEXT
3980 GOSUB  5330
3985 '
3990 '
3995 ' lethality setting value section
4000 XXMIN=0:XXMAX=TOTIME:N=NTIME
4005 YYHVAR=2!:YYLVAR=0
4010 FOR I=1 TO N
4015 YY(I)=XLETH(I):NEXT
4020 IRECNO1=215:IRECNO2=212 :IRECNO3=211
4025 DXX=(XXMAX/(N-1))
4030 FOR I=1 TO N
4035 XX(I)=(I-1)*DXX:NEXT
4040 GOSUB 5330
4045 '
4050 '
4055 ' nutrient setting value section
4060 XXMIN=0:XXMAX=TOTIME:N=NTIME
4065 YYHVAR=1!:YYLVAR=0
4070 FOR I=1 TO N
```

```
4075 YY(I)=FRANUT(I): NEXT
4080 IRECNO1=215:IRECNO2=216:IRECNO3=217
4085 DXX=(XXMAX/(N-1))
4090 FOR I=1 TO N
4095 XX(I)=(I-1)*DXX:NEXT
4100 GOSUB 5330
4105 '
4110 '
4115 'setting the lethality table
4120 N=NTIME
4125 DXX=(TOTIME/(NTIME-1))
4130 FOR I=1 TO NTIME
4135 XX(I)=(I-1)*DXX
4140 NEXT
4145 FOR I=1 TO N
4150 YY(I)=XLETH(I):NEXT
4155 IRECNO4=222 :IRECNO5=225
4160 GOTO 4860
4165 '
4170 '
4175 'setting the nutrient table
4180 N=NTIME
4185 DXX=(TOTIME/(NTIME-1))
4190 FOR I=1 TO NTIME
4195 XX(I)=(I-1)*DXX
4200 NEXT
4205 FOR I=1 TO N
4210 YY(I)=FRANUT(I):NEXT
4215 IRECNO4=223 :IRECNO5=224
4220 GOTO 4860
4225 '
4230 '
4235 ' printing of the table menu (all data or specific data)
4240 CLS:SCREEN 1
4245 LOCATE 2,1:GET 1,163:GOSUB 5215:PRINT COMMT40$;
4250 LOCATE 4,1:GET 1,164:GOSUB 5215:PRINT COMMT40$;
4255 LOCATE 7,1:GET 1,165:GOSUB 5215:PRINT COMMT40$;
4260 LOCATE 9,1:GET 1,166:GOSUB 5215:PRINT COMMT40$;
4265 LOCATE 23,1:GET 1,214:GOSUB 5215:PRINT COMMT40$;
4270 GOSUB 5105
4275 IF (VAL(ANSWERI$)>=1) AND (VAL(ANSWERI$)<=2) THEN ICHOICE2=VAL(ANSWERI$):G
OTO 4285
4280 LOCATE 24,1:BEEP:GET 1,260:GOSUB 5215:PRINT COMMT40$;:GOTO 4270
4285 IF ICHOICE2=1 THEN IITIME=1:NSTART1=1:NSTART2=1:NSTART3=1:NEND1=NXX:NEND2=
NYY:NEND3=NZZ:GOTO 4430
4290 IF ICHOICE2=2 THEN JRECNO=229:GOSUB 5065 :GOTO 4305
4295 '
4300 '
4305 ' asking value to start and where to end printing
4310 N1=NXX:N2=NYY:N3=NZZ:GET 1,175:GOSUB 5215:VAR1$=COMMT40$:GET 1,178:GOSUB 5
215:VAR2$=COMMT40$:GET 1,176:GOSUB 5215:VAR3$=COMMT40$:GET 1,179:GOSUB 5215:VAR
4$=COMMT40$:GET 1,177:GOSUB 5215:VAR5$=COMMT40$:GET 1,180:GOSUB 5215:VAR6$=COMM
T40$
4315 '
4320 ' subroutine asking what are the dimensions for the printing of results
4325 LOCATE 23,1:PRINT VAR1$;:GOSUB 5105
4330 NSTART1=VAL(ANSWERI$):IF NSTART1>N1 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB
```

```
5215:PRINT COMMT40$;:GOTO 4325
4335 NROW=24:GOSUB 5195
4340 LOCATE 23,1:PRINT VAR2$;:GOSUB 5105
4345 NEND1=VAL(ANSWERI$):IF NSTART1>NEND1 THEN LOCATE 24,1:BEEP:GET 1,256:GOSUB
 5215:PRINT COMMT40$;:GOTO 4340
4350 IF NEND1>N1 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB 5215:PRINT COMMT40$;:GOT
O 4340
4355 NROW=24:GOSUB 5195
4360 LOCATE 23,1:PRINT VAR3$;:GOSUB 5105
4365 NSTART2=VAL(ANSWERI$):IF NSTART2>N2 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB
5215:PRINT COMMT40$;:GOTO 4360
4370 NROW=24:GOSUB 5195
4375 LOCATE 23,1:PRINT VAR4$;:GOSUB 5105
4380 NEND2=VAL(ANSWERI$):IF NEND2>N2 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB 5215
:PRINT COMMT40$;:GOTO 4375
4385 IF (NEND2<NSTART2) THEN LOCATE 24,1:BEEP:GET 1,256:GOSUB 5215:PRINT COMMT4
0$;:GOTO 4375
4390 NROW=24:GOSUB 5195
4395 LOCATE 23,1:PRINT VAR5$;:GOSUB 5105
4400 NSTART3=VAL(ANSWERI$):IF NSTART3>N3 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB
5215:PRINT COMMT40$;:GOTO 4395
4405 NROW=24:GOSUB 5195
4410 LOCATE 23,1:PRINT VAR6$;:GOSUB 5105
4415 NEND3=VAL(ANSWERI$):IF NEND3>N3 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB 5215
:PRINT COMMT40$;:GOTO 4410
4420 IF (NEND3<NSTART3) THEN LOCATE 24,1:BEEP:GET 1,256:GOSUB 5215:PRINT COMMT4
0$;:GOTO 4410
4425 '
4430 ' printing the results
4435 CLS:ISTART1=NSTART1:ISTART2=NSTART2:ISTART3=NSTART3
4440 LOCATE 2,1:GET 1,182:GOSUB 5215:PRINT COMMT40$;
4445 LOCATE 4,1:GET 1,183:GOSUB 5215:PRINT COMMT40$;
4450 INC=5
4455 DTIME=TOTIME/(NTIME-1)
4460 FOR ITIME=1 TO NTIME
4465 TIME(ITIME)=(ITIME-1)*DTIME:NEXT ITIME
4470 '
4475 FOR ITIME=1 TO NTIME
4480 FOR I=NSTART1 TO NEND1
4485 FOR J=NSTART2 TO NEND2
4490 FOR K=NSTART3 TO NEND3
4495 INC=INC+1
4500 LOCATE INC,1:print using "#####.##"; time(itime)  'TTIME$=STR$(TIME(ITIME)
):TTIME$=LEFT$(TTIME$,8):TIME(ITIME)=VAL(TTIME$):PRINT TIME(ITIME);
4505 LOCATE INC,11: print using "#.####"; xx(i)  'XX$=STR$(XX(I)):XX$=LEFT$(XX$
,5):XX(I)=VAL(XX$):PRINT XX(I)
4510 LOCATE INC,18:print using "#.####"; yy(j)  'YY$=STR$(YY(J)):YY$=LEFT$(YY$,
5):YY(J)=VAL(YY$):PRINT YY(J)
4515 LOCATE INC,25:print using "#.####"; zz(k)  'ZZ$=STR$(ZZ(K)):ZZ$=LEFT$(ZZ$,
5):ZZ(K)=VAL(ZZ$):PRINT ZZ(K)
4520 LOCATE INC,31:TEMP$=STR$(TEMP(I,J,K,ITIME)):TEMP$=LEFT$(TEMP$,8):TEMP(I,J,
K,ITIME)=VAL(TEMP$):PRINT TEMP(I,J,K,ITIME);
4525 IF INC>=19 THEN 4530  ELSE 4625
4530 INC=5
4535 LOCATE 21,1:GET 1,186:GOSUB 5215:PRINT COMMT40$;
4540 IF (I>=NEND1) AND (J>=NEND2) AND (K>=NEND3) AND (ITIME>=NTIME) THEN 4550 E
LSE 4545
```

```
4545 LOCATE 22,1:GET 1,187:GOSUB 5215:PRINT COMMT40$;
4550 LOCATE 23,1:GET 1,188:GOSUB 5215:PRINT COMMT40$;
4555 '
4560 ' error checking for x, y, p
4565 ANSWERY$=INPUT$(1)
4570 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 4595
4575 IF (I>=NEND1) AND (J>=NEND2) AND (K>=NEND3) AND (ITIME>=NTIME) THEN 4580 E
LSE 4585
4580 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=-1:GOTO 4595
4585 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=1:GOTO 4595
4590 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 4595
4595 IF IFLAG=-1 THEN BEEP:IF (I>=NEND1) AND (J>=NEND2) AND (K>=NEND3) AND (ITI
ME>=NTIME) THEN GOTO 4620 ELSE LOCATE 24,1:GET 1,257:GOSUB 5215:PRINT COMMT40$;
:GOTO 4565
4600 IF IFLAG=2 THEN CLS:GOTO 4645
4605 IF IFLAG=0 THEN 3730
4610 IF IFLAG=1 THEN FOR II=5 TO 24:NROW=II:GOSUB 5190:NEXT
4615 IF(I>=NEND1)AND(J>=NEND2)AND(K>=NEND3) AND (ITIME>=NTIME) THEN 4620 ELSE 4
625
4620 NROW=22:GOSUB 5190:LOCATE 24,1:GET 1,258:GOSUB 5215:BEEP:PRINT COMMT40$;:G
OTO 4565
4625 NEXT K:NEXT J:NEXT I:NEXT ITIME
4630 GOTO 4535
4635 '
4640 '
4645 OPEN "lpt1:" AS #10
4650 WIDTH #10,100
4655 CLS: LOCATE 12,1
4660 GET 1,198: GOSUB 5215: PRINT COMMT40$
4665 GET 1,227:PRINT #10,COMMENT$:PRINT #10,:PRINT #10,
4670 GET 1,189:PRINT #10,COMMENT$;:PRINT #10,:PRINT #10,
4675 FOR ITIME=1 TO NTIME
4680 FOR I=ISTART1 TO NEND1
4685 FOR J=ISTART2 TO NEND2
4690 FOR K=ISTART3 TO NEND3
4695 PRINT #10,USING" ####.#####    ";TIME(ITIME);XX(I);YY(J);ZZ(K);TEMP(I,J,K,I
TIME)
4700 NEXT K,J,I,ITIME
4705 CLOSE #10:GOTO 3730
4710 '
4715 '
4720 ' Process parameter table
4725 CLS:NROW=3
4730 LOCATE 1,1:GET 1,230:GOSUB 5215:PRINT COMMT40$;:LOCATE 2,1:GET 1,231:GOSUB
 5215:PRINT COMMT40$;
4735 GET 1,232:GOSUB 5215:VAR$=COMMT40$:VAR!=TINIT!:GOSUB 5675
4740 GET 1,233:GOSUB 5215:VAR$=COMMT40$:VAR!=TRETRT!:GOSUB 5675
4745 GET 1,234:GOSUB 5215:VAR$=COMMT40$:VAR!=TWATER!:GOSUB 5675
4750 GET 1,239:GOSUB 5215:VAR$=COMMT40$:VAR!=TOTIME!:GOSUB 5675
4755 GET 1,240:GOSUB 5215:VAR$=COMMT40$:VAR!=COOLTI!:GOSUB 5675
4760 GET 1,241:GOSUB 5215:VAR$=COMMT40$:VAR!=NTIME:GOSUB 5675
4765 GET 1,242:GOSUB 5215:VAR$=COMMT40$:VAR!=HALFX!:GOSUB 5675
4770 GET 1,243:GOSUB 5215:VAR$=COMMT40$:VAR!=HALFY!:GOSUB 5675
4775 GET 1,244:GOSUB 5215:VAR$=COMMT40$:VAR!=HALFZ!:GOSUB 5675
4780 GET 1,246:GOSUB 5215:VAR$=COMMT40$:VAR!=NX:GOSUB 5675
4785 GET 1,247:GOSUB 5215:VAR$=COMMT40$:VAR!=NY:GOSUB 5675
4790 GET 1,264:GOSUB 5215:VAR$=COMMT40$:VAR!=NZ:GOSUB 5675
```

```
4795 GET 1,266:GOSUB 5215:VAR$=COMMT40$:VAR!=ALPHA!:GOSUB 5675
4800 IF (NSUB=2) OR (NSUB=22) THEN GOTO 4820
4805 GET 1,248:GOSUB 5215:VAR$=COMMT40$:VAR!=RHO!:GOSUB 5675
4810 GET 1,249:GOSUB 5215:VAR$=COMMT40$:VAR!=CP!:GOSUB 5675
4815 GET 1,265:GOSUB 5215:VAR$=COMMT40$:VAR!=H!:GOSUB 5675
4820 GET 1,161:LOCATE 23,1:GOSUB 5215:PRINT COMMT40$;
4825 GET 1,162:LOCATE 24,1:GOSUB 5215:PRINT COMMT40$;:LOCATE 24,13:PRINT CHR$(2
4);
4830 GOSUB 5235
4835 '
4840 'exit the program
4845 CLS:SCREEN 2:SCREEN 0: COLOR 7,1,1: GOTO 2605
4850 '
4855 '
4860 ' printing a table for lethality or nutrient fraction
4865 CLS
4870 LOCATE 2,1:GET 1,182:GOSUB 5215:PRINT COMMT40$;
4875 LOCATE 4,1:GET 1,IRECNO4:GOSUB 5215:PRINT COMMENT$;
4880 INC =5:NVAR=1
4885 FOR I=NVAR TO N
4890 INC=INC+1
4895 LOCATE INC,4:PRINT XX(I) :LOCATE INC,24:PRINT USING "#.###^^^^";YY(I);
4900 IF INC>= 17 THEN 4910
4905 NEXT
4910 LOCATE 21,1:GET 1,186:GOSUB 5215:PRINT COMMT40$;
4915 IF I>=N THEN 4925 ELSE 4920
4920 LOCATE 22,1:GET 1,187:GOSUB 5215:PRINT COMMT40$;
4925 LOCATE 23,1:GET 1,188:GOSUB 5215:PRINT COMMT40$;
4930 ANSWERY$=INPUT$(1)
4935 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 4960
4940 IF (I>=N) THEN 4945 ELSE 4950
4945 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=-1:GOTO 4960
4950 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=1:GOTO 4960
4955 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 4960
4960 IF IFLAG=-1 THEN BEEP:LOCATE 24,1:IF I>=N THEN GOTO 4980 ELSE GET 1,257:GO
SUB 5215:PRINT COMMT40$;:GOTO 4930
4965 IF IFLAG=2 THEN CLS:GOTO 4990
4970 IF IFLAG=0 THEN 3730
4975 IF IFLAG=1 THEN FOR II=5 TO 24:NROW=II:GOSUB 5195:NEXT:IF(I>N) THEN 4980 E
LSE INC=5:NVAR=I:GOTO 4885
4980 NROW=22:GOSUB 5195:LOCATE 24,1:GET 1,258:GOSUB 5215:BEEP:PRINT COMMT40$;:G
OTO 4930
4985 RETURN
4990 OPEN "lpt1:" AS #10
4995 WIDTH #10,100
5000 CLS: LOCATE 12,1
5005 GET 1,198: GOSUB 5215: PRINT COMMT40$
5010 GET 1,227:GOSUB 5215:PRINT #10,COMMT40$:PRINT #10,:PRINT #10,
5015 GET 1,IRECNO5:PRINT #10,COMMENT$;:PRINT #10,:PRINT #10,
5020 FOR I=1 TO N
5025 PRINT #10,USING"################.##";XX(I);
5035 PRINT #10,"                                ";
5040 PRINT #10,USING "#.##^^^^";YY(I)
5045 NEXT I
5050 CLOSE#10:GOTO 3730
5055 '
5060 '
```

```
5065 ' printing the x, y z values
5070 CLS:LOCATE 2,1:GET 1,JRECNO:GOSUB 5215:PRINT COMMT40$;
5075 LOCATE 9,1:GET 1,156:GOSUB 5215:PRINT COMMT40$;
5080 LOCATE 6,1:GET 1,168:GOSUB 5215:PRINT COMMT40$;
5085 NROW=10:FOR I=1 TO NXX:NROW=NROW+1:LOCATE NROW,1:PRINT I;")";:print using
"#.###^^^^";XX(I);:NEXT
5090 NROW=10:FOR I=1 TO NYY:NROW=NROW+1:LOCATE NROW,14:PRINT I;")";:print using
 "#.###^^^^";YY(I);:NEXT
5095 NROW=10:FOR I=1 TO NZZ:NROW=NROW+1:LOCATE NROW,28:PRINT I;")";:print using
 "#.###^^^^";ZZ(I);:NEXT
5100 RETURN
5105 '
5110 '
5115 ' error checking for integers
5120 ANSWERI$=INPUT$(1)
5125 IF ASC(ANSWERI$)>=49 AND ASC(ANSWERI$)<=58 THEN RETURN
5130 LOCATE 24,1:BEEP:GET 1,250:GOSUB 5215:PRINT COMMT40$;:GOTO 5120
5135 RETURN
5140 '
5145 '
5150 '
5155 ' error checking for x, y or z
5160 ANSWERY$=INPUT$(1)
5165 IF (ANSWERY$="x") OR (ANSWERY$="X") OR (ANSWERY$="y") OR (ANSWERY$="Y") OR
 (ANSWERY$="z") OR (ANSWERY$="Z") THEN RETURN
5170 BEEP:LOCATE 24,1:GET 1,252:GOSUB 5215:PRINT COMMT40$;:GOTO 5160
5175 RETURN
5180 '
5185 '
5190 ' blank line subroutine
5195 LOCATE NROW,1:PRINT"                                                    ";:RETURN
5200 '
5205 '
5210 ' subroutine that reduce the length of the records
5215 COMMT40$=LEFT$(COMMENT$,39):RETURN
5220 '
5225 '
5230 ' error checking subroutine for X
5235 ANSWERY$=INPUT$(1)
5240 IF (ANSWERY$="X") OR (ANSWERY$="x") THEN GOTO 3730
5245 LOCATE 24,1:GET 1,254:GOSUB 5215:BEEP:PRINT COMMT40$;:GOTO 5235
5250 RETURN
5255 '
5260 '
5265 ' inputting graphics parameters
5270 LOCATE 23,1:GET 1,157:GOSUB 5215:PRINT COMMT40$;
5275 N=NXX:GOSUB 5115:XPOSITION=VAL(ANSWERI$)
5280 IF XPOSITION>N THEN BEEP:LOCATE 24,1:GET 1,253:GOSUB 5215:PRINT COMMT40$;:
GOTO 5270
5285 LOCATE 23,1:GET 1,158:GOSUB 5215:PRINT COMMT40$;
5290 N=NYY:GOSUB 5115:YPOSITION=VAL(ANSWERI$)
5295 IF YPOSITION>N THEN BEEP:LOCATE 24,1:GET 1,253:GOSUB 5215:PRINT COMMT40$;:
GOTO 5285
5300 LOCATE 23,1:GET 1,159:GOSUB 5215:PRINT COMMT40$;
5305 N=NZZ:GOSUB 5115:ZPOSITION=VAL(ANSWERI$)
5310 IF ZPOSITION>N THEN BEEP:LOCATE 24,1:GET 1,253:GOSUB 5215:PRINT COMMT40$;:
GOTO 5300
```

```
5315 RETURN
5320 '
5325 '
5330 ' graph subroutine
5335 ' draw titles
5340 CLS:LOCATE 2,1
5345 GET 1,IRECNO2:GOSUB 5215:PRINT COMMT40$;
5350 LOCATE 1,1:GET 1,IRECNO3:GOSUB 5215:PRINT COMMT40$;:LOCATE 21,1:GET 1,IREC
NO1:GOSUB 5215:PRINT COMMT40$;
5355 '
5360 ' print x-axis number
5365 LINESET=0
5370 NXXPIX=-(250/(11-1))
5375 FOR I=1 TO 11
5380 NXXPIX=NXXPIX+(250/(11-1))
5385 XX=INT((I-1)*(XXMAX/(11-1)))
5390 LINESET=LINESET+1
5395 IF LINESET=3 THEN LINESET=1
5400 IF LINESET=1 THEN NROW=19:LINE(33+NXXPIX,140)-(33+NXXPIX,145),2:NCOLUM=INT
(3+(NXXPIX/7.96)) ELSE NROW=20:LINE(33+NXXPIX,140)-(33+NXXPIX,155),1:NCOLUM=INT
(4+(NXXPIX/7.96))
5405 IF I=1 THEN LOCATE 19,3:PRINT XX(1)
5410 LOCATE NROW,NCOLUM:PRINT XX
5415 NEXT
5420 '
5425 ' printing y-axis numbers
5430 TEMPMAX=YY(1)
5435 TEMPMIN=YY(1)
5440 FOR I=2 TO N
5445 IF YY(I)<TEMPMIN THEN TEMPMIN=YY(I)
5450 IF YY(I)>TEMPMAX THEN TEMPMAX=YY(I)
5455 NEXT
5460 YYMAX=TEMPMAX
5465 YYMIN=TEMPMIN
5470 IF YYMAX>YYHVAR THEN 5475 ELSE YYMAX=YYHVAR
5475 IF YYMIN<YYLVAR THEN 5480 ELSE YYMIN=YYLVAR
5480 NYYSET=6
5485 YYSPACE=(YYMAX-YYMIN)/(NYYSET-1)
5490 YYMIN2=YYMIN
5495 NYYPIX=-(124/(NYYSET-1))
5500 FOR I=1 TO NYYSET
5505 NYYPIX=NYYPIX+(124/(NYYSET-1))
5510 NROW=INT(18-(NYYPIX/7.96))
5515 IYYMIN$=STR$(YYMIN2)
5520 NCHAR=LEN(IYYMIN$)
5525 IF ASC(MID$(IYYMIN$,1,1))=32 THEN IYYMIN$=MID$(IYYMIN$,2,NCHAR-1):IYYMIN=V
AL(IYYMIN$)
5530 IF NCHAR>4 THEN IYYMIN$=MID$(IYYMIN$,1,4):IYYMIN=VAL(IYYMIN$)
5535 LOCATE NROW,1
5540 PRINT IYYMIN$
5545 YYMIN2=YYMIN2+YYSPACE
5550 LINE(26,140-NYYPIX)-(32,140-NYYPIX),1
5555 NEXT
5560 '
5565 ' draws axis lines
5570 LINE(33,140)-(283,140),1
5575 LINE(33,140)-(33,16),1
```

```
5580 '
5585 ' drawing the curve
5590 FOR I=1 TO N
5595 XX(I)=33+((XX(I)/XXMAX)*250)
5600 YY(I)=140-(((YY(I)-YYMIN)/(YYMAX-YYMIN))*124)
5605 NEXT
5610 FOR I=2 TO N
5615 LINE (XX(I-1),YY(I-1))-(XX(I),YY(I)),3:NEXT
5620 FOR I=1 TO N
5625 LINE(XX(I)-2,YY(I)-2)-(XX(I)+2,YY(I)+2),2
5630 LINE(XX(I)+2,YY(I)-2)-(XX(I)-2,YY(I)+2),2:NEXT
5635 COOLXX=33+((COOLTI-XXMIN)/(XXMAX-XXMIN))*250
5640 YYLINE=140!
5645 FOR I=1 TO 13
5650 LINE(COOLXX,YYLINE)-(COOLXX,YYLINE-5),1:YYLINE=YYLINE-10:NEXT
5655 GET 1,161:LOCATE 22,1:GOSUB 5215:PRINT COMMT40$;
5660 GET 1,162:LOCATE 23,1:GOSUB 5215:PRINT COMMT40$;:LOCATE 23,13:PRINT CHR$(2
4);
5665 GOSUB 5235
5670 STOP
5675 LOCATE NROW,1:PRINT VAR$:LOCATE NROW,31:PRINT VAR!
5680 NROW=NROW+1
5685 RETURN
5690 '
5695 '
5700 '*************************************************************************
```

```
                    ALPHAC
1                 * * * * * M E N U * * * * * *
2 1  Do process calculations for a pouch
3 2  Exit to MAIN selection menu (with can/pouch choices)
4 Select an item from the menu (enter 1 or 2 and RETURN)
5 You did not enter either 1 or 2
6 Cooling Water Temperature (deg C)
7 Enter 1 to let h=infinite, 2 to specify h
8 The number must be 1 or 2
9 Number:
10 Initial Food Temperature (deg C)
11 Retort Temperature (deg C)
12 PRESS ANY KEY TO CONTINUE
13 The Initial and Cooling Water Temp must be <= Retort Temp
14 Initial Temp:   Retort Temp:   Water Temp:
15 Total Processing Time (heat + cool) (<10000 sec)
16 Time at Which Cooling Starts (sec)
17 Total Time and Cooling Time must be>0, Total>=Cooling, & Total<=10000
18 Total Time:   Cooling Time:
19 Number of Time Increments (>=2, <=21)
20 You did not enter a number between 2 and 21
21 Number of Time Values:
22 Thermal Diffusivity (m**2/sec)
23 The Thermal Diffusivity must be between .0000001 and .00000099
24 You entered the following value(s):
25 Food Density (kg/m**3)
26 The Food Density must be > 0
27 Food Density:
28 Specific Heat of Food (kJ/kg-deg C)
29 The Specific Heat must be > 0
```

```
30 Specific Heat:
31 Half-X Pouch Length (m)
32 Half-Y Pouch Length (m)
33 Half-Z Pouch Length (m)
34 The Half-X, Half-Y and Half-Z lengths must be > 0
35 Half-X:    Half-Y:    Half-Z:
36 The Number of X values (2<=nx<=5)
37 The Number of Y values (2<=ny<=5)
38 The Number of Z values (2<=nz<=5)
39 The Number of X, Y and Z values must all be between 2 and 5
40 nx:            ny:          nz:
41 Convective Heat Transfer Coeff (W/m**2-deg C)
42 The surface convective heat transfer coefficient must be > 0
43 H:
44 Thermal Diffusivity:
45 What is the F-value for the contaminant at Tref=121 deg C) - in minutes?
46 The F-value cannot equal 0
47 A F-value greater than 20 minutes is unusual
48 Do you want to reenter the value (Y or N and RETURN)?
49 What is the z-value for the contaminant (at Tref=121 deg C) - in deg C?
50 The z-value cannot equal 0
51 A z-value greater than 30 deg C is unusual
52 What is the F-value for the nutrient (at Tref=121 deg C) - in minutes?
53 A F-value greater than 100 minutes is unusual
54 What is the z-value for the nutrient (at Tref=121 deg C) - in deg C?
55                    ARGUMENT LISTING FOR SUBROUTINE
56 Enter the new value and press RETURN
57 Press RETURN alone if you want to keep the old value
```

```
58
59 ARGUMENT                                          OLD          NEW
60 A z-value greater than 50 deg C is unusual
61 Do you want to redo the POUCH calculations (Y or N and RETURN) ?
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
```

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

```
144
145
146
147
148
149
150         RESULTS DISPLAY MENU
151         1) Temp-Time Graph
152         2) Lethality-Time Graph
153         3) Nutrient Fraction-Time Graph
154               Enter 1, 2, ... 8
155              GRAPHIC DISPLAY MENU
156   X(m)            Y(m)              Z(m)
157 At which X do you want Temp-Time Graph
158 At which Y do you want Temp-Time Graph
159 At which Z do you want Temp-Time Graph
160     Select the Z position (1, 2 or ...)
161 Press "X" to exit to results menu or
162       Press -PrtSc to print graph
163                PRINTING MENU
164  Choose the printing options dimension
165         1) Print all the data
166         2) Print specific data
167
168     Here is a list of various values
169      X(m)             Y(m)          Z(m)
170      Y(m)             X(m)          Z(m)
171      Z(m)             Y(m)          x(m)
```

```
172 X
173 Y
174 Z
175 Which X are you starting at(1,2 or ...)
176 Which Y are you starting at(1,2 or ...)
177 Which Z are you starting at(1,2 or ...)
178 Which X do you want to stop(1,2 or ...)
179 Which Y do you want to stop(1,2 or ...)
180 Which Z do you want to stop(1,2 or ...)
181          ENTER "X", "Y" OR "Z"
182                  RESULTS
183 TIME(s)   X(m)   Y(m)   Z(m) TEMP(degC)
184 185
185 Z(m)      Y(m)       X(m)   Temp(deg C)
186 Press "X" to exit to results menu
187 Press "Y" to continue
188 Press "P" to print specified data
189   TIME(sec)        X(m)           Y(m)           Z(m)        TEMP(deg C)
190
191
192 Radius          Z-values
193 Select the axis to be plotted: R or Z
194   Select the R position (1, 2, or ...)
195   Select the Z position (1, 2, or ...)
196 TEMPERATURE(deg C)
197            TEMP VS TIME
198     * * * P L E A S E  W A I T * * *
199                    * * * P L E A S E  W A I T * * *
200
```

```
201
202                 1)  R, Z
203                 2)  Z, R
204         Z-value           Radius
205 Which R are you starting at(1,2, or...)
206 Which Z are you starting at(1,2, or...)
207 Which R do you want to stop(1,2, or...)
208 Which Z do you want to stop(1,2, or...)
209     Radius          Z          TEMPERATURE
210      Z            RADIUS       TEMPERATURE
211 LETHALITY
212              LETHALITY vs TIME
213             ENTER 1, 2 OR 3
214            ENTER  1  OR  2
215                   TIME (sec)
216              NUTRIENT FRACTION vs TIME
217 FRACTION
218        4) Temp-Time Table
219        5) Lethality-Time Table
220        6) Nutrient Fraction-Time Table
221        8) Exit
222     TIME (sec)         LETHALITY
223      TIME (sec)        NUTRIENT FRACTION
224             TIME (sec)                          NUTRIENT FRACTION
225             TIME (sec)                          LETHALITY
226        7) Process Parameter Table
227                                  RESULTS
228                                                                  RESULTS
```

```
229
230         Process Parameter Table
231         ----------------------
232 Initial Temperature(deg C)
233 Retort Temperature(deg C)
234 Water Temperature(deg C)
235
236
237
238
239 Total Processing Time(sec)
240 Time When Cooling Starts(sec)
241 Number of Time Increments
242 Half X-Length(m)
243 Half Y-Length(m)
244 Half Z-Length(m)
245       7) Process Parameter Table
246 Number of X-values
247 Number of Y-values
248 Food Density (kg/m**3)
249 Food Specific Heat (kJ/kg-degC)
250 You did not enter an integer, try again
251 You did not enter 1,2,...,8 -try again
252 You did not enter X, Y or Z -try again
253    The value was too big - try again
254          You did not enter "X"
255        Value is too big, try again
256 End value can not be < then start value
257 There is no more data, enter P or X
```

```
258 There is no more data,enter P or X
259 You did not enter R or Z - try again
260 You did not enter 1 or 2 - try again
261 You did not enter 1, 2 or 3 - try again
262
263
264 Number of Z-values
265 H(W/m**2-degC)
266 Thermal Diffusivity(m**2/sec)
267 You did not enter X or P
```

APPENDIX 3.6

Program Listing of MAIND

```
10 '
15 '
20 '
25 '          ********** M A I N D **********
30 '
35 '
40 'Initialization and Dimensioning
45 DEFINT I-N
50 DIM R(10), Z(10), TIME(21)
55 DIM RR(21),ZZ(21),XX(21),YY(21)
60 DIM TEMP(10,10,21)
65 DIM TCNT(81,21), TAVG(81,20), VELE(81)
70 DIM ORGE0(81), RATEL(81,20), ORGELE(81,20), XLETH(21)
75 DIM XNUTE0(81), XNUTE(81,20), FRANUT(21)
80 '
85 '
90 'Setting up the text screen
95 KEY OFF
100 SCREEN 0,0,0,0
105 WIDTH 80
110 COLOR 7,1,1: CLS
115 '
120 '
125 ' opening and readying the associated files
130 OPEN "c:alphad" AS 1
135 FIELD 1,80 AS COMMENT$
140 '
145 OPEN "b:commc" AS 2
150 FIELD 2,80 AS COMMC$
155 '
160 '
165 ' setting the pointer to current record number of commc
170 GET 2,1: NRECCOUNT=VAL(COMMC$)
175 BLANK$=" ": LSET COMMC$=BLANK$: PUT 2,NRECCOUNT+1
180 '
185 '
190 ' Checking if control has come from MAIN or from SUBs
195 OPEN "b:store2" FOR INPUT AS 3
200 INPUT#3,NOPCODE
205 CLOSE 3
210 '
215 ' Not coming from MAIN:
220 IF NOPCODE<>0 THEN GOTO 1300
225 '
230 ' Coming from MAIN:
235 OPEN "b:comma" FOR INPUT AS 4
240 '
245 '    ** read info from comma**
250 '
255 CLOSE 4
260 '
265 '
270 ' Printing the menu
275 CLS: PRINT
280 GET 1,1:  PRINT COMMENT$
285 GET 1,2:  PRINT COMMENT$
290 GET 1,3:  PRINT COMMENT$ : PRINT
```

```
295 '
300 '
305 'determining user's selection from the menu
310 JRECNO=4
315 GOSUB 3070
320 IF JOK<>-1 THEN GOTO 270
325 IF (VAL(ANSI$)=2)  GOTO 370
330 IF (VAL(ANSI$)=1) GOTO 475
335 GET 1,5
340 PRINT COMMENT$
345 GET 1,12: PRINT COMMENT$
350 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 350
355 GOTO 270
360 '
365 '
370 'if want to go straight back to MAIN
375 '
380 OPEN "b:commb" FOR OUTPUT AS 7
385 '
390 ' ** write info into b:commb **
395 '
400 CLOSE 7
405 '
410 CLOSE 1,2
415 '
420 CLS: SCREEN 1
425 COLOR 4,1
430 DEF SEG = &HB800
435 BLOAD "b:sandcl.scr",0
440 ON ERROR GOTO 445
445 IF INKEY$<>"" THEN BEEP ELSE GOTO 450
450 '
455 CHAIN "MAIN"
460 '
465 '
470 '
475 'if want to go to a CAN sub
480 N1STTHRU=0
485 INDIC=0
490 'determining NSUB wanted
495 NOLD = NSUB-3 : IF NSUB=44 THEN NOLD=1
500 IF NSUB=55 THEN NOLD=2
505 IF NSUB=0 THEN INDIC=1: NOLD=1
510 JRECNO=7: NMARKSUB=0: GOSUB 3215
515 NEW1=VAL(NEWVALUE$)
520 IF (NEW1>=1) AND (NEW1<=2) THEN GOTO 535
525 JRECNO=8: JRECNO2=9: NMARKSET=1: GOSUB 3385
530 GOTO 495
535 NSUB=NEW1 + 3
540 '
545 'determining temperatures
550 IF INDIC=1 THEN OLD=40! ELSE OLD=TINIT
555 JRECNO=10:NMARKSUB=1: GOSUB 3215
560 XNEW1=VAL(NEWVALUE$)
565 IF INDIC=1 THEN OLD=120! ELSE OLD=TRETRT
570 JRECNO=11:NMARKSUB=1: GOSUB 3215
575 XNEW2=VAL(NEWVALUE$)
```

```
580 IF INDIC=1 THEN OLD=60! ELSE OLD=TWATER
585 JRECNO=6:NMARKSUB=1: GOSUB 3215
590 XNEW3=VAL(NEWVALUE$)
595 IF (XNEW2>=XNEW1) AND (XNEW2>=XNEW3)THEN GOTO 610
600 JRECNO=13: JRECNO2=14: NMARKSET=3: GOSUB 3385
605 GOTO 550
610 TINIT=XNEW1: TRETRT=XNEW2: TWATER=XNEW3
615 '
620 'determining time values
625 IF INDIC=1 THEN OLD=5600! ELSE OLD=TOTIME
630 JRECNO=15: NMARKSUB=1: GOSUB 3215
635 XNEW1=VAL(NEWVALUE$)
640 IF INDIC=1 THEN OLD=3600! ELSE OLD=COOLTI
645 JRECNO=16: NMARKSUB=1: GOSUB 3215
650 XNEW2=VAL(NEWVALUE$)
655 IF (XNEW1>0) AND (XNEW1<=10000) AND (XNEW2>0) AND (XNEW1>=XNEW2) THEN GOTO
670
660 JRECNO=17: JRECNO2=18: NMARKSET=2: GOSUB 3385
665 GOTO 625
670 TOTIME=XNEW1: COOLTI=XNEW2
675 '
680 'determining number of time values
685 IF INDIC=1 THEN NOLD=11 ELSE NOLD=NTIME
690 JRECNO=19: NMARKSUB=0: GOSUB 3215
695 NEW1=VAL(NEWVALUE$)
700 IF (NEW1>=2) AND (NEW1<=21) THEN GOTO 795
705 JRECNO=20: JRECNO2=21: NMARKSET=1: GOSUB 3385
710 GOTO 685
715 NTIME=NEW1
720 '
725 'determining thermal diffusivity
730 IDIFF=1
735 IF INDIC=1 THEN OLD=1.6E-07 ELSE OLD=ALPHA
740 JRECNO=22: NMARKSUB=1: GOSUB 3215
745 XNEW1=VAL(NEWVALUE$)
750 IF (XNEW1>0) THEN GOTO 765
755 JRECNO=23: JRECNO2=44: NMARKSET=1: GOSUB 3385
760 GOTO 735
765 IF (XNEW1=>.0000001) AND (XNEW1=<9.99E-07) GOTO 780
770 JRECNO=33: JRECNO2=44: NMARKSET=1: GOSUB 3385
775 GOTO 735
780 ALPHA=XNEW1: IDIFF=0
785 '
790 'determining food density
795 IF NSUB=4 THEN GOTO 885
800 IF RHO=0! THEN OLD=1500! ELSE OLD=RHO
805 JRECNO=25: NMARKSUB=1: GOSUB 3215
810 XNEW1=VAL(NEWVALUE$)
815 IF (XNEW1>0) THEN GOTO 830
820 JRECNO=26: JRECNO2=27: NMARKSET=1: GOSUB 3385
825 GOTO 800
830 RHO=XNEW1
835 '
840 ' determining specific heat
845 IF CP=0! THEN OLD=4000! ELSE OLD=CP
850 JRECNO=28: NMARKSUB=1: GOSUB 3215
855 XNEW1=VAL(NEWVALUE$)
```

```
860 IF (XNEW1>0) THEN GOTO 875
865 JRECNO=29: JRECNO2=30: NMARKSET=1: GOSUB 3385
870 GOTO 845
875 CP=XNEW1
880 '
885 'determining dimensions of the can
890 IF INDIC=1 THEN OLD=.03 ELSE OLD=RADIUS
895 JRECNO=31:NMARKSUB=1: GOSUB 3215
900 XNEW1=VAL(NEWVALUE$)
905 IF INDIC=1 THEN OLD=.04 ELSE OLD=HALFL
910 JRECNO=32: NMARKSUB=1: GOSUB 3215
915 XNEW2=VAL(NEWVALUE$)
920 '
925 '
930 '
935 IF (XNEW1>0) AND (XNEW2>0) GOTO 950
940 JRECNO=34: JRECNO2=35: NMARKSET=2: GOSUB 3385
945 GOTO 890
950 RADIUS=XNEW1: HALFL=XNEW2
955 '
960 'determining the nr,nz values
965 IF INDIC=1 THEN NOLD=5 ELSE NOLD=NR
970 JRECNO=36: NMARKSUB=0: GOSUB 3215
975 NEW1=VAL(NEWVALUE$)
980 IF INDIC=1 THEN NOLD=5 ELSE NOLD=NZ
985 JRECNO=37: NMARKSUB=0: GOSUB 3215
990 NEW2=VAL(NEWVALUE$)
995 '
1000 '
1005 '
1010 IF (NEW1>=2) AND (NEW1<=10) AND (NEW2>=2) AND (NEW2<=10) GOTO 1025
1015 JRECNO=39:JRECNO2=40: NMARKSET=2: GOSUB 3385
1020 GOTO 965
1025 NR=NEW1: NZ=NEW2
1030 '
1035 'determining the convective surface heat transfer coefficient
1040 IF NSUB=4 GOTO 1090
1045 IF H=0! THEN OLD=500! ELSE OLD=H
1050 JRECNO=41: NMARKSUB=1: GOSUB 3215
1055 XNEW1=VAL(NEWVALUE$)
1060 IF (XNEW1>0) GOTO 1075
1065 JRECNO=42: JRECNO2=43: NMARKSET=1: GOSUB 3385
1070 GOTO 1045
1075 H=XNEW1
1080 '
1085 '
1090 'Putting info into STORE2 to tell MAINB what SUB it chained to
1095 NSUBMAIN=4
1100 OPEN "b:store2" FOR OUTPUT AS 3
1105 IF NSUB=5 GOTO 1125
1110 PRINT#3,NSUB
1115 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,COOLTI,N
SUBMAIN
1120 GOTO 1135
1125 PRINT#3,NSUB
1130 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,COOLTI,R
HO,CP,H,NSUBMAIN
```

```
1135 CLOSE 3
1140 '
1145 '
1150 'Putting arguments into COMM1 to run the SUB to get the heating results
1155 OPEN "c:comm1" FOR OUTPUT AS 6
1160 IF NSUB=5 GOTO 1175
1165 PRINT#6,TRETRT,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,NSUBMAIN
1170 GOTO 1180
1175 PRINT#6,TRETRT,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,RHO,CP,H,NSUBMA
IN
1180 CLOSE 6
1185 '
1190 '
1195 'Preparing to chain to a CAN sub
1200 'storing the current record number of COMMC into the first rec no of COMMC

1205 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
1210 '
1215 '
1220 CLOSE 1,2
1225 '
1230 'Displaying "Please Wait" symbol
1235 CLS:SCREEN 1,0,0,0
1240 COLOR 0,0
1245 DEF SEG = &HB800
1250 BLOAD "b:fire.scn",0
1255 ON ERROR GOTO 1260
1260 IF INKEY$<>"" THEN BEEP ELSE GOTO 1265
1265 '
1270 'Chaining to the SUB
1275 IF NSUB=4 THEN CHAIN "can1"
1280 IF NSUB=5 THEN CHAIN "can2"
1285 PRINT"Problems with nsub in MAIND":STOP
1290 '
1295 '
1300 'Read the information in STORE2 to determine if this is heating or cooling

1302 cls:locate 12,10: get 1,198:print comment$
1305 OPEN "b:store2" FOR INPUT AS 3
1310 INPUT#3, NSUB
1315 IF NSUB=44 OR NSUB=55 GOTO 1670
1320 IF NSUB=5 GOTO 1335
1325 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,COOLTI,N
SUBMAIN
1330 GOTO 1340
1335 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,TOTIME,COOLTI,R
HO,CP,H,NSUBMAIN
1340 CLOSE 3
1345 '
1350 '
1355 'Reading the heating results from COMM2
1360 OPEN "c:comm2" FOR INPUT AS 7
1365 FOR LMD=1 TO NTIME
1370 FOR IMD=1 TO NZ
1375 FOR JMD=1 TO NR
1380 '
1385 INPUT#7,Z(IMD),R(JMD),TIME(LMD),TEMP(IMD,JMD,LMD)
```

```
1390 NEXT JMD,IMD,LMD
1395 CLOSE 7
1400 '
1405 '
1410 IF TOTIME = COOLTI THEN GOTO 1815 'don't need to do cooling part
1415 'Storing arguments in COMM1 to get the cooling results
1420 '(first calculate the "ntime" for the cooling portion alone)
1425 NCTIME = NTIME - FIX(  (TOTIME-COOLTI)/ (TOTIME/(NTIME-1)) )
1430 NCOOLT = NTIME - NCTIME + 1       '(the "ntime" for the cooling part)
1432 if ncoolt=1 then ncoolt=2: nctime=nctime-1
1435 OPEN "c:comm1" FOR OUTPUT AS 6
1440 IF NSUB=5 GOTO 1455
1445 PRINT#6,TWATER,TRETRT,ALPHA,HALFL,RADIUS,NZ,NR,NCOOLT,TOTIME-COOLTI,NSUBMA
IN
1450 GOTO 1460
1455 PRINT#6,TWATER,TRETRT,ALPHA,HALFL,RADIUS,NZ,NR,NCOOLT,TOTIME-COOLTI,RHO,CP
,H,NSUBMAIN
1460 CLOSE 6
1465 '
1470 '
1475 'Storing codes, parameters in STORE2
1480 OPEN "b:store2" FOR OUTPUT AS 3
1485 IF NSUB=4 THEN NOPCODE=44: PRINT#3,NOPCODE: GOTO 1495
1490 IF NSUB=5 THEN NOPCODE=55: PRINT#3,NOPCODE: GOTO 1505
1495 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,NCTIME,TOTIME,C
OOLTI,NSUBMAIN
1500 GOTO 1520
1505 PRINT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,NCTIME,TOTIME,C
OOLTI,RHO,CP,H,NSUBMAIN
1510 '
1515 '
1520 'Storing the heating results in STORE2 so can have them when chain back to
 MAIND in cooling section
1525 FOR LMD=1 TO NTIME
1530 FOR IMD=1 TO NZ
1535 FOR JMD=1 TO NR
1540 '
1545 PRINT#3,TEMP(IMD,JMD,LMD)
1550 NEXT JMD,IMD,LMD
1555 CLOSE 3
1560 '
1565 '
1570 'Preparing to chain to a CAN sub
1575 'storing the current record number of COMMC into the first rec no of COMMC

1580 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
1585 '
1590 '
1595 CLOSE 1,2
1600 '
1605 'Displaying "Please Wait" symbol
1610 CLS:SCREEN 1,0,0,0
1615 COLOR 0,0
1620 DEF SEG = &HB800
1625 BLOAD "b:fire.scn",0
1630 ON ERROR GOTO 1260
1635 IF INKEY$<>"" THEN BEEP ELSE GOTO 1265
```

```
1640 '
1645 'Chaining to the SUB
1650 IF NSUB=4 OR NSUB=44 THEN CHAIN "can1"
1655 IF NSUB=5 OR NSUB=55 THEN CHAIN "can2"
1660 '
1665 '
1670 'Reading the rest of the parameters from STORE2
1675 IF NOPCODE=44 THEN INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NT
IME,NCTIME,TOTIME,COOLTI,NSUBMAIN: GOTO 1685
1680 INPUT#3,TRETRT,TWATER,TINIT,ALPHA,HALFL,RADIUS,NZ,NR,NTIME,NCTIME,TOTIME,C
OOLTI,RHO,CP,H,NSUBMAIN
1685 '
1690 '
1695 'Reading the heating results from STORE2
1700 FOR LMD=1 TO NTIME
1705 FOR IMD=1 TO NZ
1710 FOR JMD=1 TO NR
1715 '
1720 INPUT#3,TEMP(IMD,JMD,LMD)
1725 NEXT JMD,IMD,LMD
1730 CLOSE 3
1735 '
1740 '
1745 'Reading the TCOOL value from COMM2 one at a time = the cool temp
1750 ' and calculating the true cooling curve temperatures
1755 OPEN "c:comm2" FOR INPUT AS 7
1760 FOR LMD=NCTIME TO NTIME
1765 FOR IMD=1 TO NZ
1770 FOR JMD=1 TO NR
1775 '
1780 INPUT#7,Z(IMD),R(JMD),TIME(LMD),TCOOL
1785 TIME(LMD) = TIME(LMD) + COOLTI
1790 TEMP(IMD,JMD,LMD)=TEMP(IMD,JMD,LMD) - (TRETRT-TCOOL)
1795 NEXT JMD,IMD,LMD
1800 CLOSE 7
1805 '
1810 '
1815 'Getting the FLETH value
1820 CLS: JRECNO=45: GOSUB 2910
1825 IF JOK<>-1 THEN GOTO 1820 ELSE FLETH=VAL(ANSR$)
1830 IF (FLETH>0!) AND (FLETH<=20!) GOTO 1890
1835 IF FLETH>20! THEN GOTO 1855
1840 GET 1,46: PRINT COMMENT$
1845 GET 1,12: PRINT: PRINT COMMENT$
1850 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1850 ELSE GOTO 1820
1855 CLS: GET 1,47: PRINT COMMENT$
1860 GET 1,24: PRINT COMMENT$;FLETH
1865 JRECNO=48: GOSUB 2785
1870 IF JOK<>-1 GOTO 1855
1875 IF (DY$="n") OR (DY$="N") GOTO 1890 ELSE GOTO 1820
1880 '
1885 '
1890 'Getting the ZLETH value
1895 CLS: JRECNO=49: GOSUB 2910
1900 IF JOK<>-1 THEN GOTO 1895 ELSE ZLETH=VAL(ANSR$)
1905 IF (ZLETH>0!) AND (ZLETH<=30!) GOTO 1965
1910 IF ZLETH>30! THEN GOTO 1930
```

```
1915 GET 1,50: PRINT COMMENT$
1920 GET 1,12: PRINT: PRINT COMMENT$
1925 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1925 ELSE GOTO 1895
1930 CLS: GET 1,51: PRINT COMMENT$
1935 GET 1,24: PRINT COMMENT$;ZLETH
1940 JRECNO=48: GOSUB 2785
1945 IF JOK<>-1 GOTO 1930
1950 IF (DY$="n") OR (DY$="N") GOTO 1965 ELSE GOTO 1895
1955 '
1960 '
1965 'Getting the FXNUT value
1970 CLS: JRECNO=52: GOSUB 2910
1975 IF JOK<>-1 THEN GOTO 1970 ELSE FXNUT=VAL(ANSR$)
1980 IF (FXNUT>0!) AND (FXNUT<=100!) GOTO 2040
1985 IF FXNUT>100! THEN GOTO 2005
1990 GET 1,46: PRINT COMMENT$
1995 GET 1,12: PRINT: PRINT COMMENT$
2000 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 2000 ELSE GOTO 1970
2005 CLS: GET 1,53: PRINT COMMENT$
2010 GET 1,24: PRINT COMMENT$;FXNUT
2015 JRECNO=48: GOSUB 2785
2020 IF JOK<>-1 GOTO 2005
2025 IF (DY$="n") OR (DY$="N") GOTO 2040 ELSE GOTO 1970
2030 '
2035 '
2040 'Getting the Znut value
2045 CLS: JRECNO=54: GOSUB 2910
2050 IF JOK<>-1 THEN GOTO 2045 ELSE ZNUT=VAL(ANSR$)
2055 IF (ZNUT>0!) AND (ZNUT<=50!) GOTO 2115
2060 IF ZNUT>50! THEN GOTO 2080
2065 GET 1,50: PRINT COMMENT$
2070 GET 1,12: PRINT: PRINT COMMENT$
2075 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 2075 ELSE GOTO 2045
2080 CLS: GET 1,60: PRINT COMMENT$
2085 GET 1,24: PRINT COMMENT$;ZNUT
2090 JRECNO=48: GOSUB 2785
2095 IF JOK<>-1 GOTO 2080
2100 IF (DY$="n") OR (DY$="N") GOTO 2115 ELSE GOTO 2045
2105 '
2110 '
2115 '*****************************************************************************
2120 'calculating lethality and nutrient fraction retained
2125 CLS: LOCATE 12,1: GET 1,198: PRINT COMMENT$
2130 '
2135 LET CORG=10!^(37!): LET CNUT=10!^(37!)     'arbitrarily set
2140 IF RHO=0! THEN RHO=1500!                  'when nsub=4
2145 '
2150 'number of elements:
2155 NELE = (NZ-1) * (NR-1)
2160 '
2165 'Total and elemental volumes
2170 PI = 3.14159265#
2175 VTOTC = PI * (RADIUS^2) * (2!*HALFL)
2180 ICOUNTR=1
2185   FOR NLOOP=1 TO NELE STEP (NZ-1)
2190   ICOUNTR=ICOUNTR + 1
2195     FOR MMD=NLOOP TO (NLOOP+(NZ-2))
```

```
2200      VELE(MMD) = PI * (HALFL/(NZ-1)) * ((R(ICOUNTR)^2) - (R(ICOUNTR-1)^2))

2205      NEXT MMD
2210    NEXT NLOOP
2215 '
2220 'calculate original number of organisms in half-can, and in each element
2225 ORG0 = (1!/2!*VTOTC) * CORG * RHO
2230 FOR MMD=1 TO NELE
2235 ORGE0(MMD) = VELE(MMD) * CORG * RHO
2240 NEXT MMD
2245 '
2250 'calculate original amount of nutrient in half-can, and in each element
2255 XNUT0 = (1!/2!*VTOTC) * CNUT * RHO
2260 FOR MMD=1 TO NELE
2265 XNUTE0(MMD) = VELE(MMD) * CNUT * RHO
2270 NEXT MMD
2275 '
2280 '
2285 'calculate TCNT (average of four radial temps, at each time value)
2290 FOR LMD = 1 TO NTIME
2295 MCOUNT=0
2300  FOR IRMD = 1 TO (NR-1)
2305   FOR IZMD=1 TO (NZ-1)
2310 '
2315    MCOUNT=MCOUNT+1
2320    TCNT(MCOUNT,LMD) = (TEMP(IZMD,IRMD,LMD)+TEMP(IZMD+1,IRMD,LMD)+TEMP(IZMD
,IRMD+1,LMD)+TEMP(IZMD+1,IRMD+1,LMD) )/4!
2325    NEXT IZMD
2330  NEXT IRMD
2335 NEXT LMD
2340 '
2345 'calculate TAVG (the average TCNT at start and end of time increment)
2350 FOR LL=1 TO (NTIME-1)
2355   FOR MMD = 1 TO NELE
2360   TAVG(MMD,LL)=(TCNT(MMD,LL) + TCNT(MMD,LL+1))/2!
2365   NEXT MMD
2370 NEXT LL
2375 '
2380 'calculate the lethal rate, the total number of organisms left
2385 'at end of time increment, and lethality at end of time increment
2390 DTIME = TOTIME /(NTIME-1)/60!
2395 XLETH(1)=0!
2400 FOR LL=1 TO (NTIME-1)
2405 ORGTOT = 0!
2410    FOR MMD=1 TO NELE
2415    IF (TAVG(MMD,LL)-121!)/ZLETH <-37! THEN RATEL(MMD,LL)=0!: GOTO 2425
2420    RATEL(MMD,LL)=(1!/FLETH) * 10^((TAVG(MMD,LL)-121!)/ZLETH)
2425    IF LL>1 GOTO 2445
2430    IF (-12!*RATEL(MMD,LL)*DTIME)<-37! THEN ORGELE(MMD,1)=10!^-37!: GOTO 2
455
2435    ORGELE(MMD,1)= ORGE0(MMD) * 10!^(-12!*RATEL(MMD,LL)*DTIME)
2440    GOTO 2455
2445    IF (-12!*RATEL(MMD,LL)*DTIME)<-37! THEN ORGELE(MMD,LL)=10!^-37!: GOTO
2455
2450    ORGELE(MMD,LL)=ORGELE(MMD,LL-1) * 10!^(-12!*RATEL(MMD,LL)*DTIME)
2455    ORGTOT = ORGTOT + ORGELE(MMD,LL)
2460    NEXT MMD
```

```
2465 IF ORGTOT<1! THEN ORGTOT=1!
2470 XLETH(LL+1)= .434294482# * LOG(ORGO/ORGTOT) /12!
2475 NEXT LL
2480 '
2485 '
2490 'calculate the nutrient destruction rate, the total amount of nutrient
2495 'left at the end of time increment, and nutrient fraction retained at
2500 'end of each time increment
2505 FRANUT(1)=1!
2510 FOR LL=1 TO (NTIME-1)
2515 XNUTOT = 0!
2520    FOR MMD=1 TO NELE
2525    IF (TAVG(MMD,LL)-121)/ZNUT < -37! THEN RATEL(MMD,LL)=0!:GOTO 2535
2530    RATEL(MMD,LL)=(1!/FXNUT) * 10^((TAVG(MMD,LL)-121!)/ZNUT)
2535    IF LL>1 GOTO 2555
2540    IF (12!*RATEL(MMD,LL)*DTIME)>37! THEN XNUTE(MMD,1)=10!^-37!:GOTO 2565
2545    XNUTE(MMD,1)= XNUTE0(MMD) * 10!^(-12!*RATEL(MMD,LL)*DTIME)
2550    GOTO 2565
2555    IF (12!*RATEL(MMD,LL)*DTIME)>37! THEN XNUTE(MMD,LL)=10!^-37!:GOTO 2565

2560    XNUTE(MMD,LL)=XNUTE(MMD,LL-1) * 10!^(-12!*RATEL(MMD,LL)*DTIME)
2565    XNUTOT = XNUTOT + XNUTE(MMD,LL)
2570    NEXT MMD
2575 IF XNUTOT<1! THEN XNUTOT=1!
2580 FRANUT(LL+1)= (XNUTOT/ XNUTO)
2585 NEXT LL
2590 '
2595 '*************************************************************************
****
2600 '
2605 '
2610 'Going to the results display section
2615 GOTO 3750
2620 '
2625 '
2630 'Back from results display, ask if want to go to MAIN or back to SUB
2635 CLS: JRECNO=61: GOSUB 2785
2640 IF JOK<>-1 THEN GOTO 2635
2645 IF (DY$="y") OR (DY$="Y") THEN GOTO 475
2650 '
2655 '
2660 'when want to go to MAIN:
2665 OPEN "b:commb" FOR OUTPUT AS 7
2670 '
2675 '
2680 '*** write info to commb
2685 CLOSE 7
2690 '
2695 '
2700 'storing the current record number of COMMC into the first rec no of COMMC

2705 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
2710 CLOSE 1,2
2715 '
2720 '
2725 CLS: SCREEN 1
2730 COLOR 1,1
```

```
2735 DEF SEG = &HB800
2740 BLOAD "b:sandcl.scr",0
2745 ON ERROR GOTO 2750
2750 IF INKEY$<>"" THEN BEEP ELSE GOTO 2755
2755 '
2760 '
2765 CHAIN "main"
2770 '
2775 '
2780 '
2785 '
2790 '
2795 '
2800 '
2805 'getting and checking  Y/N SUBROUTINE
2810 JOK=0
2815 GET 1,JRECNO
2820 COMMENT1Y$="You did not enter Y or N -    You entered: "
2825 COMMENT2Y$="You did not enter a value"
2830 PRINT COMMENT$;
2835 LINE INPUT "",ANSY$
2840 JY=0
2845 JY = JY+1
2850 DY$=MID$(ANSY$,JY,1)
2855 IF DY$="" THEN GOTO 2890
2860 IF DY$=" " THEN GOTO 2845
2865 IF (DY$="Y") OR (DY$="y") OR (DY$="N") OR (DY$="n") THEN JOK=-1: RETURN
2870 PRINT COMMENT1Y$,ANSY$
2875 GET 1,12: PRINT: PRINT COMMENT$
2880 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2880
2885 RETURN
2890 PRINT COMMENT2Y$
2895 GET 1,12: PRINT: PRINT COMMENT$
2900 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 2900
2905 RETURN
2910 '
2915 '
2920 '
2925 ' getting and CHECKING FOR  REAL  NUMBERS
2930 JOK=0
2935 GET 1,JRECNO
2940 COMMENT1R$="You did not enter a REAL number  -  You entered: "
2945 COMMENT2R$="you did not enter a value"
2950 PRINT COMMENT$;
2955 LINE INPUT"",ANSR$
2960 JFLAGR=0                          'flag to check that number exists
2965 IFLAGR=0                          'flag to check for decimal points
2970 JR=1
2975 KR=LEN(ANSR$)
2980 IF KR=0 THEN GOTO 3050
2985 WHILE JR<=KR
2990 DR$=MID$(ANSR$,JR,1)
2995 IF DR$=" " THEN GOTO 3035
3000 IF (ASC(DR$)>47) AND (ASC(DR$)<58) THEN JFLAGR=-1: GOTO 3035
3005 IF DR$="." THEN IFLAGR=IFLAGR+1: GOTO 3030
3010 PRINT COMMENT1R$,ANSR$
3015 GET 1,12: PRINT: PRINT COMMENT$
```

```
3020 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 3020
3025 RETURN
3030 IF IFLAGR>1 GOTO 3010
3035 JR=JR+1
3040 WEND
3045 IF (DR$<>" ") OR (JFLAGR=-1) THEN JOK=-1: RETURN
3050 PRINT COMMENT2R$
3055 GET 1,12: PRINT: PRINT COMMENT$
3060 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 3060
3065 RETURN
3070 '
3075 '
3080 '
3085 'getting and CHECKING FOR  INTEGER  NUMBERS
3090 JOK=0
3095 IFLAGI=0
3100 GET 1,JRECNO
3105 COMMENT1I$="You did not enter an INTEGER value  -  You entered: "
3110 COMMENT2I$="you did not enter a value"
3115 PRINT COMMENT$;
3120 LINE INPUT "",ANSI$
3125 NI=LEN(ANSI$)
3130 IF NI=0 THEN GOTO 3195
3135 JI=1
3140 WHILE JI<=NI
3145 DI$=MID$(ANSI$,JI,1)
3150 IF DI$=" " THEN GOTO 3180
3155 IF (ASC(DI$)>47) AND (ASC(DI$)<58) THEN IFLAGI=-1: GOTO 3180
3160 PRINT COMMENT1I$,ANSI$
3165 GET 1,12: PRINT: PRINT COMMENT$
3170 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 3170
3175 RETURN
3180 JI=JI+1
3185 WEND
3190 IF (DI$<>" ") OR (IFLAGI=-1) THEN JOK=-1: RETURN
3195 PRINT COMMENT2I$
3200 GET 1,12: PRINT: PRINT COMMENT$
3205 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 3205
3210 RETURN
3215 '
3220 '
3225 '
3230 '
3235 ' Table Creation subroutine
3240 IF N1STTHRU=1 GOTO 3290
3245 NROW=7:NCOLUMN=1
3250 SCREEN 0,0,1,1
3255 CLS
3260 'printing table heading
3265 GET 1,55:PRINT COMMENT$
3270 GET 1,56:PRINT COMMENT$;
3275 GET 1,57:PRINT COMMENT$
3280 GET 1,59:PRINT COMMENT$
3285 N1STTHRU=1
3290 LOCATE NROW,NCOLUMN
3295 GET 1,JRECNO
3300 BLANK$="    "
```

```
3305 NBLANK=INSTR(COMMENT$,BLANK$)
3310 FINAL$=LEFT$(COMMENT$,NBLANK-1)
3315 IF IDIFF=1 THEN DIFF$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(67);:
GOTO 3325
3320 IF NMARKSUB=1 THEN PRINT FINAL$;TAB(53);OLD;TAB(67); ELSE PRINT FINAL$;TAB
(53);NOLD;TAB(67);
3325 LINE INPUT NEWVALUE$
3330 NRTEMP=CSRLIN: NCTEMP=POS(0)
3335 IF LEN(NEWVALUE$)=0 THEN GOTO 3360
3340 IF NMARKSUB=1 THEN GOSUB 3520 ELSE GOSUB 3630
3345 IF (NOK=-1) THEN NROW=CSRLIN: NCOLUMN=POS(0): RETURN
3350 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
3355 GOTO 3315
3360 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
3365 IF IDIFF=1 THEN NEWVALUE$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(6
7);VAL(NEWVALUE$):GOTO 3375
3370 IF NMARKSUB=1 THEN NEWVALUE$=STR$(OLD):PRINT FINAL$;TAB(53);OLD;TAB(67);VA
L(NEWVALUE$) ELSE NEWVALUE$=STR$(NOLD):PRINT FINAL$;TAB(53);NOLD;TAB(67);VAL(NE
WVALUE$)
3375 NROW=CSRLIN: NCOLUMN=POS(0)
3380 RETURN
3385 '
3390 '
3395 '
3400 '
3405 ' Erase Lines in table and show error page subroutine
3410 SCREEN ,,2,2
3415 CLS
3420 GET 1,JRECNO: PRINT COMMENT$;
3425 GET 1,24: PRINT COMMENT$;
3430 GET 1,JRECNO2: PRINT COMMENT$;
3435 IF NMARKSET<>3 THEN GOTO 3445
3440 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2,XNEW3:GOTO 3460 ELSE PRINT NEW1,NEW2,
NEW3:GOTO 3460
3445 IF NMARKSET<>2 THEN GOTO 3455
3450 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2:GOTO 3460 ELSE PRINT NEW1,NEW2: GOTO
3460
3455 IF NMARKSUB=1 THEN PRINT XNEW1 ELSE PRINT NEW1
3460 GET 1,12
3465 PRINT: PRINT COMMENT$
3470 DUMMEL$=INKEY$: IF DUMMEL$="" THEN GOTO 3470
3475 SCREEN ,,1,1
3480 LOCATE NROW,NCOLUMN
3485 LOCATE NROW-1,NCOLUMN:PRINT "
                                  ": LOCATE NROW-1,NCOLUMN
3490 IF NMARKSET=1 THEN GOTO 3510
3495 LOCATE NROW-2,NCOLUMN:PRINT "
                                  ":LOCATE NROW-2,NCOLUMN
3500 IF NMARKSET=2 THEN GOTO 3510
3505 LOCATE NROW-3,NCOLUMN:PRINT "
                                  ": LOCATE NROW-3,NCOLUMN
3510 NROW=CSRLIN: NCOLUMN=POS(0)
3515 RETURN
3520 '
3525 '
```

```
3530 '
3535 '
3540 'checking Real2 numbers (in table)
3545 NOK=0
3550 JFLAGR2=0
3555 IFLAGR2=0
3560 JR2=1
3565 NR2=LEN(NEWVALUE$)
3570 WHILE JR2=<NR2
3575 DR2$=MID$(NEWVALUE$,JR2,1)
3580 IF DR2$=" " THEN GOTO 3605
3585 IF DR2$="." THEN IFLAGR2=IFLAGR2+1: GOTO 3605
3590 IF (ASC(DR2$)>47) AND (ASC(DR2$)<58) THEN JFLAGR2=-1: GOTO 3605
3595 NOK=0
3600 RETURN
3605 IF IFLAGR2>1 THEN NOK=0: RETURN
3610 JR2=JR2+1
3615 WEND
3620 IF JFLAGR2=0 THEN NOK=0 ELSE NOK=-1
3625 RETURN
3630 '
3635 '
3640 '
3645 '
3650 ' checking Integer2 numbers (in table)
3655 NOK=0
3660 IFLAGI2=0
3665 NI2=LEN(NEWVALUE$)
3670 JI2=1
3675 WHILE JI2<=NI2
3680 DI2$=MID$(NEWVALUE$,JI2,1)
3685 IF DI2$=" " THEN GOTO 3700
3690 IF (ASC(DI2$)>47) AND (ASC(DI2$)<58) THEN IFLAGI2=-1: GOTO 3700
3695 NOK=0: RETURN
3700 JI2=JI2+1
3705 WEND
3710 IF IFLAGI2=0 THEN NOK=0 ELSE NOK=-1
3715 RETURN
3720 '
3725 '
3730 '
3735 '*************************************************************************
3740 '
3745 '     RESULTS DISPLAY SECTION
3750 ' initializing the values
3755 NRADIUS=NR
3760 NZLEN=NZ
3765 FOR I=1 TO NRADIUS
3770 RR(I)=R(I):NEXT
3775 FOR I=1 TO NZLEN
3780 ZZ(I)=Z(I):NEXT
3785 '
3790 '
3795 ' this is the pouch results displays
3800 CLS:SCREEN 1:COLOR 9,1: KEY OFF
3805 LOCATE 2,1:GET 1,150:GOSUB 5240:PRINT COMMT40$;
3810 LOCATE 5,1:GET 1,151:GOSUB 5240:PRINT COMMT40$;
```

```
3815 LOCATE 7,1:GET 1,152:GOSUB 5240:PRINT COMMT40$;
3820 LOCATE 9,1:GET 1,153:GOSUB 5240:PRINT COMMT40$;
3825 LOCATE 11,1:GET 1,218:GOSUB 5240:PRINT COMMT40$;
3830 LOCATE 13,1:GET 1,219:GOSUB 5240:PRINT COMMT40$;
3835 LOCATE 15,1:GET 1,220:GOSUB 5240:PRINT COMMT40$;
3840 LOCATE 17,1:GET 1,245:GOSUB 5240:PRINT COMMT40$;
3845 LOCATE 19,1:GET 1,221:GOSUB 5240:PRINT COMMT40$;
3850 LOCATE 23,1:GET 1,154:GOSUB 5240:PRINT COMMT40$;
3855 GOSUB 5135
3860 IF(VAL(ANSWERI$)>=1) AND (VAL(ANSWERI$)<=8) THEN ICHOICE=VAL(ANSWERI$):GOT
O 3870
3865 LOCATE 24,1:BEEP:GET 1,251:GOSUB 5240:PRINT COMMT40$;:GOTO 3855
3870 IF ICHOICE=1 GOTO 4035
3875 IF ICHOICE=2 GOTO 4150
3880 IF ICHOICE=3 GOTO 4210
3885 IF ICHOICE=4 GOTO 4390
3890 IF ICHOICE=5 GOTO 4275
3895 IF ICHOICE=6 GOTO 4330
3900 IF ICHOICE=7 GOTO 3910
3905 IF ICHOICE=8 GOTO 5025
3910 '
3915 ' Process parameter table
3920 CLS:NROW=5
3925 LOCATE 1,1:GET 1,230:GOSUB 5235:PRINT COMMT40$;:LOCATE 2,1:GET 1,231:GOSUB
 5235:PRINT COMMT40$;
3930 GET 1,232:GOSUB 5235:VAR$=COMMT40$:VAR!=TINIT!:GOSUB 5045
3935 GET 1,233:GOSUB 5235:VAR$=COMMT40$:VAR!=TRETRI!:GOSUB 5045
3940 GET 1,234:GOSUB 5235:VAR$=COMMT40$:VAR!=TWATER!:GOSUB 5045
3945 GET 1,239:GOSUB 5235:VAR$=COMMT40$:VAR!=TOTIME!:GOSUB 5045
3950 GET 1,240:GOSUB 5235:VAR$=COMMT40$:VAR!=COOLTI!:GOSUB 5045
3955 GET 1,241:GOSUB 5235:VAR$=COMMT40$:VAR!=NTIME:GOSUB 5045
3960 GET 1,262:GOSUB 5235:VAR$=COMMT40$:VAR!=RADIUS!:GOSUB 5045
3965 GET 1,244:GOSUB 5235:VAR$=COMMT40$:VAR!=HALFL!:GOSUB 5045
3970 GET 1,246:GOSUB 5235:VAR$=COMMT40$:VAR!=NR:GOSUB 5045
3975 GET 1,264:GOSUB 5235:VAR$=COMMT40$:VAR!=NZ:GOSUB 5045
3980 GET 1,266:GOSUB 5235:VAR$=COMMT40$:VAR!=ALPHA!:GOSUB 5045
3985 IF (NSUB=4) OR (NSUB=44) THEN GOTO 4005
3990 GET 1,248:GOSUB 5235:VAR$=COMMT40$:VAR!=RHO!:GOSUB 5045
3995 GET 1,249:GOSUB 5235:VAR$=COMMT40$:VAR!=CP!:GOSUB 5045
4000 GET 1,265:GOSUB 5235:VAR$=COMMT40$:VAR!=H!:GOSUB 5045
4005 GET 1,161:LOCATE 23,1:GOSUB 5235:PRINT COMMT40$;
4010 GET 1,162:LOCATE 24,1:GOSUB 5235:PRINT COMMT40$;:LOCATE 24,13:PRINT CHR$(2
4);
4015 GOSUB 5255
4020 GOTO 3795
4025 '
4030 '
4035 ' graphic display section
4040 CLS:LOCATE 2,1:GET 1,155:GOSUB 5240:PRINT COMMT40$;
4045 JRECNO=155
4050 '
4055 ' displaying the r, z length increments
4060 GOSUB 5075
4065 ' inputing graphic parameter
4070 '
4075 GOSUB 5285
4080 '
```

```
4085 ' going to temp-time graph
4090 IRECNO2=197:IRECNO3=196:IRECNO1=215
4095 XXMIN=0:XXMAX=TOTIME
4100 YYHVAL=150:YYLVAL=0
4105 N=NTIME
4110 DXX=(XXMAX/(N-1))
4115 FOR I=1 TO N
4120 XX(I)=(I-1)*DXX:NEXT
4125 FOR I=1 TO N
4130 YY(I)=TEMP(ZPOSITION,RPOSITION,I):NEXT
4135 GOSUB 5335
4140 '
4145 '
4150 ' lethality vs time graph
4155 XXMIN=0:XXMAX=TOTIME:N=NTIME
4160 YYHVAL=1:YYLVAL=0
4165 FOR I=1 TO N
4170 YY(I)=XLETH(I):NEXT
4175 IRECNO1=215:IRECNO2=212:IRECNO3=211
4180 DXX=(XXMAX/(N-1))
4185 FOR I=1 TO N
4190 XX(I)=(I-1)*DXX:NEXT
4195 GOSUB 5335
4200 '
4205 '
4210 ' nutrient fraction vs time initialization section graph
4215 XXMIN=0:XXMAX=TOTIME:N=NTIME
4220 YYHVAL=1:YYLVAL=0
4225 FOR I=1 TO N
4230 YY(I)=FRANUT(I):NEXT
4235 IRECNO1=215:IRECNO2=216:IRECNO3=217
4240 DXX=(XXMAX/(N-1))
4245 FOR I=1 TO N
4250 XX(I)=(I-1)*DXX:NEXT
4255 GOSUB 5335
4260 '
4265 '
4270 ' setting the lethality vs time table
4275 CLS:N=NTIME
4280 DXX=(TOTIME/(NTIME-1))
4285 FOR I=1 TO NTIME
4290 XX(I)=(I-1)*DXX
4295 NEXT
4300 FOR I=1 TO N
4305 YY(I)=XLETH(I):NEXT
4310 IRECNO4=222:IRECNO5=225
4315 GOTO 4810
4320 '
4325 '
4330 ' setting the nutrient fraction vs time table
4335 CLS:N=NTIME
4340 DXX=(TOTIME/(NTIME-1))
4345 FOR I=1 TO NTIME
4350 XX(I)=(I-1)*DXX
4355 NEXT
4360 FOR I=1 TO N
4365 YY(I)=FRANUT(I):NEXT
```

```
4370 IRECNO4=223:IRECNO5=224
4375 GOTO 4810
4380 '
4385 '
4390 ' printing options table menu
4395 CLS:SCREEN 1
4400 LOCATE 2,1:GET 1,163:GOSUB 5240:PRINT COMMT40$;
4405 LOCATE 4,1:GET 1,164:GOSUB 5240:PRINT COMMT40$;
4410 LOCATE 7,1:GET 1,165:GOSUB 5240:PRINT COMMT40$;
4415 LOCATE 9,1:GET 1,166:GOSUB 5240:PRINT COMMT40$;
4420 LOCATE 23,1:GET 1,214:GOSUB 5240:PRINT COMMT40$;
4425 GOSUB 5135
4430 IF (VAL(ANSWERI$)>=1) AND (VAL(ANSWERI$)<=2) THEN ICHOICE2=VAL(ANSWERI$):G
OTO 4440
4435 LOCATE 24,1:BEEP:GET 1,260:GOSUB 5240:PRINT COMMT40$;:GOTO 4425
4440 IF ICHOICE2=1 THEN IITIME=1:NSTART1=1:NSTART2=1:NEND1=NRADIUS:NEND2=NZLEN:
GOTO 4555
4445 IF ICHOICE2=2 THEN JRECNO=229:CLS:GOSUB 5075:LOCATE 2,1:GET 1,163:GOSUB 52
40:PRINT COMMT40$;:GOTO 4460
4450 '
4455 '
4460 ' asking value to start and where to end printing
4465 N1=NRADIUS:N2=NZLEN:GET 1,205:GOSUB 5240:VAR1$=COMMT40$:GET 1,207:GOSUB 52
40:VAR2$=COMMT40$:GET 1,206:GOSUB 5240:VAR3$=COMMT40$:GET 1,208:GOSUB 5240:VAR4
$=COMMT40$
4470 ' getting the start and end dimensions for the print
4475 LOCATE 23,1:PRINT VAR1$;:GOSUB 5135
4480 NSTART1=VAL(ANSWERI$):IF NSTART1>N1 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB
5240:PRINT COMMT40$;:GOTO 4475
4485 NROW=24:GOSUB 5220
4490 LOCATE 23,1:PRINT VAR2$;:GOSUB 5135
4495 NEND1=VAL(ANSWERI$):IF NSTART1>NEND1 THEN LOCATE 24,1:BEEP:GET 1,256:GOSUB
 5240:PRINT COMMT40$;:GOTO 4490
4500 IF NEND1>N1 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB 5240:PRINT COMMT40$;:GOT
O 4490
4505 NROW=24:GOSUB 5220
4510 LOCATE 23,1:PRINT VAR3$;:GOSUB 5135
4515 NSTART2=VAL(ANSWERI$):IF NSTART2>N2 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB
5240:PRINT COMMT40$;:GOTO 4510
4520 NROW=24:GOSUB 5220
4525 LOCATE 23,1:PRINT VAR4$;:GOSUB 5135
4530 NEND2=VAL(ANSWERI$):IF NEND2>N2 THEN LOCATE 24,1:BEEP:GET 1,255:GOSUB 5240
:PRINT COMMT40$;:GOTO 4525
4535 IF (NEND2<NSTART2) THEN LOCATE 24,1:BEEP:GET 1,256:GOSUB 5240:PRINT COMMT4
0$;:GOTO 4525
4540 NROW=24:GOSUB 5220
4545 '
4550 ' printing the numbers in the table
4555 CLS:ISTART1=NSTART1:ISTART2=NSTART2
4560 LOCATE 2,1:GET 1,182:GOSUB 5240:PRINT COMMT40$;
4565 LOCATE 4,1:GET 1,209:GOSUB 5240:PRINT COMMT40$;
4570 INC=5
4575 DTIME=TOTIME/(NTIME-1)
4580 FOR ITIME=1 TO NTIME
4585 TIME(ITIME)=(ITIME-1)*DTIME:NEXT ITIME
4590 FOR ITIME=1 TO NTIME
4595 FOR I=NSTART1 TO NEND1
```

```
4600 FOR J=NSTART2 TO NEND2
4605 INC=INC+1
4610 LOCATE INC,1:TTIME$=STR$(TIME(ITIME)):TTIME$=LEFT$(TTIME$,6):TIME(ITIME)=V
AL(TTIME$):PRINT TIME(ITIME);
4615 LOCATE INC,9:print using "#.####";rr(i)  'RR$=STR$(RR(I)):RR$=LEFT$(RR$,6)
:RR(I)=VAL(RR$):PRINT RR(I);
4620 LOCATE INC,19:print using "#.####";zz(j)  'ZZ$=STR$(ZZ(J)):ZZ$=LEFT$(ZZ$,6
):ZZ(J)=VAL(ZZ$):PRINT ZZ(J);
4625 LOCATE INC,30:TEMP$=STR$(TEMP(J,I,ITIME)):TEMP$=LEFT$(TEMP$,8):TEMP(J,I,IT
IME)=VAL(TEMP$):PRINT TEMP(J,I,ITIME);
4630 IF INC>=19 THEN 4635 ELSE 4735
4635 INC=5
4640 LOCATE 21,1:GET 1,186:GOSUB 5240:PRINT COMMT40$;
4645 IF (I>=NEND1)AND(J>=NEND2)AND(ITIME>=NTIME) THEN 4655 ELSE 4650
4650 LOCATE 22,1:GET 1,187:GOSUB 5240:PRINT COMMT40$;
4655 LOCATE 23,1:GET 1,188:GOSUB 5240:PRINT COMMT40$;
4660 '
4665 ' error checking subroutine for X/Y/P  +  printing table to printer
4670 ANSWERY$=INPUT$(1)
4675 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 4700
4680 IF(I>=NEND1)AND(J>=NEND2)AND(ITIME>=NTIME) THEN 4685 ELSE 4690
4685 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=-1:GOTO 4700
4690 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=1:GOTO 4700
4695 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 4700
4700 '
4705 IF IFLAG=2 THEN CLS:GOTO 4750
4710 IF IFLAG=0 THEN 3795
4715 IF IFLAG=1 THEN FOR II=5 TO 24:NROW=II:GOSUB 5220:NEXT
4720 IF IFLAG=-1 THEN BEEP:IF(I>=NEND1)AND(J>=NEND2)AND(ITIME>=NTIME) THEN 4730
 ELSE LOCATE 24,1:GET 1,257:GOSUB 5240:PRINT COMMT40$;:GOTO 4670
4725 IF (ITIME>=NTIME)AND(I>=NEND1)AND(J>=NEND2) THEN 4730 ELSE GOTO 4735
4730 NROW=22:GOSUB 5220:LOCATE 24,1:GET 1,258:GOSUB 5240:BEEP:PRINT COMMT40$;:G
OTO 4670
4735 NEXT J:NEXT I:NEXT ITIME
4740 GOTO 4640
4745 '
4750 OPEN "lpt1:" AS #10
4755 WIDTH #10,100
4760 CLS: LOCATE 12,1: GET 1,198: GOSUB 5240: PRINT COMMT40$
4765 GET 1,227:GOSUB 5240:PRINT #10,COMMT40$:PRINT #10,:PRINT #10,
4770 GET 1,190:PRINT #10,COMMENT$;:PRINT #10,:PRINT #10,
4775 FOR IITIME=1 TO NTIME
4780 FOR I=ISTART1 TO NEND1
4785 FOR J=ISTART2 TO NEND2
4790 PRINT #10,USING"####.#####      ";TIME(IITIME);RR(I);ZZ(J);TEMP(J,I,IITIME)

4795 NEXT J:NEXT I:NEXT IITIME
4800 CLOSE #10:GOTO 3795
4805 '
4810 ' table for lethality and nutrient fraction
4815 CLS
4820 LOCATE 2,1:GET 1,182:GOSUB 5240:PRINT COMMT40$;
4825 LOCATE 4,1:GET 1,IRECNO4:GOSUB 5240:PRINT COMMT40$;
4830 INC=5:NVAR=1
4835 FOR I=NVAR TO N
4840 INC=INC+1
4845 LOCATE INC,5:PRINT XX(I) :LOCATE INC,24:PRINT USING "#.###^^^^";YY(I);
```

```
4850 IF INC>=19 THEN 4860
4855 NEXT
4860 LOCATE 21,1:GET 1,186:GOSUB 5240:PRINT COMMT40$;
4865 IF (I>=N) THEN 4875 ELSE 4870
4870 LOCATE 22,1:GET 1,187:GOSUB 5240:PRINT COMMT40$;
4875 LOCATE 23,1:GET 1,188:GOSUB 5240:PRINT COMMT40$;
4880 '
4885 ' error checking subroutine for X/Y/P  +  printing table to printer
4890 ANSWERY$=INPUT$(1)
4895 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 4920
4900 IF(I>=N) THEN 4905 ELSE 4910
4905 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=-1:GOTO 4920
4910 IF(ANSWERY$="Y") OR (ANSWERY$="y") THEN IFLAG=1:GOTO 4920
4915 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 4920
4920 IF IFLAG=-1 THEN BEEP:LOCATE 24,1:IF I>N THEN GOTO 4940:ELSE GET 1,257:GOS
UB 5240:PRINT COMMT40$;:GOTO 4890
4925 IF IFLAG=2 THEN CLS:GOTO 4955
4930 IF IFLAG=0 THEN 3795
4935 IF IFLAG=1 THEN FOR II=5 TO 24:NROW=II:GOSUB 5220:NEXT:IF(I>N) THEN 4940 E
LSE INC=5:NVAR=I:GOTO 4835
4940 NROW=22:GOSUB 5220:LOCATE 24,1:GET 1,258:GOSUB 5240:BEEP:PRINT COMMT40$;:G
OTO 4890
4945 '
4950 '
4955 ' printing a table for lethality and nutrient fraction results
4960 OPEN "lpt1:" AS #10
4965 WIDTH #10,100
4970 CLS: LOCATE 12,1: GET 1,198: GOSUB 5240: PRINT COMMT40$
4975 GET 1,227:GOSUB 5240:PRINT #10,COMMT40$:PRINT #10,:PRINT #10,
4980 GET 1,IRECNO5:PRINT #10,COMMENT$;:PRINT #10,:PRINT #10,
4985 FOR I=1 TO N
4990 PRINT #10,USING "################.##";XX(I);
4995 PRINT #10,"                              ";
5000 PRINT #10,USING "###.###^^^^";YY(I)
5005 NEXT I
5010 CLOSE 10:GOTO 3795
5015 '
5020 '
5025 ' exit of this subprogram
5030 CLS: SCREEN 2: SCREEN 0: COLOR 7,1,1 : GOTO 2635
5035 '
5040 '
5045 'subroutine that print parameter and comment in the process parameter tabl
e
5050 LOCATE NROW,1:PRINT VAR$:LOCATE NROW,32:PRINT VAR!
5055 NROW=NROW+1
5060 RETURN
5065 '
5070 '
5075 ' printing value subroutine
5080 LOCATE 9,1:GET 1,192:GOSUB 5240:PRINT COMMT40$;
5085 LOCATE 2,1:GET 1,JRECNO:GOSUB 5240:PRINT COMMT40$;
5090 LOCATE 6,1:GET 1,168:GOSUB 5240:PRINT COMMT40$;
5095 NROW=11:NCOLUM=5
5100 FOR I=1 TO NRADIUS
5105 LOCATE NROW,NCOLUM:PRINT I;")";:print using "#.###^^^^";RR(I);:NROW=NROW+1
:NEXT
```

```
5110 NROW=11:NCOLUM=23
5115 FOR I=1 TO NZLEN
5120 LOCATE NROW,NCOLUM:PRINT I;")";:print using "#.###^^^^";ZZ(I)
5125 NROW=NROW+1:NEXT
5130 RETURN
5135 '
5140 '
5145 ' error checking subroutine for integers
5150 ANSWERI$=INPUT$(1)
5155 IF ASC(ANSWERI$)>=49 AND ASC(ANSWERI$)<=58 THEN RETURN
5160 LOCATE 24,1:BEEP:GET 1,250:GOSUB 5240:PRINT COMMT40$;:GOTO 5150
5165 RETURN
5170 '
5175 '
5180 ' error checking subroutine for r/z
5185 ANSWERY$=INPUT$(1)
5190 IF (ANSWERY$="R") OR (ANSWERY$="r") OR (ANSWERY$="z") OR (ANSWERY$="Z") TH
EN RETURN
5195 LOCATE 24,1:GET 1,259:GOSUB 5240:BEEP:PRINT COMMT40$;:GOTO 5185
5200 RETURN
5205 '
5210 '
5215 ' put a blank line subroutine
5220 LOCATE NROW,1:PRINT"                                                    ";:RETURN
5225 '
5230 '
5235 ' reduce record length to 39 characters
5240 COMMT40$=LEFT$(COMMENT$,39):RETURN
5245 '
5250 '
5255 'error checking subroutine for X
5260 ANSWERY$=INPUT$(1)
5265 IF (ANSWERY$="X") OR (ANSWERY$="x") THEN RETURN
5270 LOCATE 24,1:GET 1,254:GOSUB 5240:BEEP:PRINT COMMT40$;:GOTO 5260
5275 '
5280 '
5285 ' inputing graphics parameters
5290 LOCATE 23,1:GET 1,194:GOSUB 5240:PRINT COMMT40$;
5295 N=NRADIUS:GOSUB 5145:RPOSITION=VAL(ANSWERI$)
5300 IF RPOSITION>N THEN BEEP:LOCATE 24,1:GET 1,253:GOSUB 5240:PRINT COMMT40$;:
GOTO 5290
5305 LOCATE 23,1:GET 1,195:GOSUB 5240:PRINT COMMT40$;
5310 NROW=24:GOSUB 5215
5315 N=NZLEN:GOSUB 5145:ZPOSITION=VAL(ANSWERI$)
5320 IF ZPOSITION>N THEN BEEP:LOCATE 24,1:GET 1,253:GOSUB 5240:PRINT COMMT40$;:
GOTO 5315
5325 RETURN
5330 '
5335 '
5340 ' plotting graph
5345 ' draws titles
5350 CLS:LOCATE 1,1
5355 GET 1,IRECNO3:GOSUB 5240:PRINT COMMT40$;
5360 LOCATE 2,1:GET 1,IRECNO2:GOSUB 5240:PRINT COMMT40$;:LOCATE 21,1
5365 GET 1,IRECNO1:GOSUB 5240:PRINT COMMT40$;
5370 '
5375 ' print x-axis number
```

```
5380 LINESET=0
5385 DXX=(XXMAX/(11-1))
5390 NXXPIX=-(250/(11-1))
5395 FOR I=1 TO 11
5400 NXXPIX=NXXPIX+(250/(11-1))
5405 XX=INT((I-1)*(XXMAX/(11-1)))
5410 LINESET=LINESET+1
5415 IF LINESET=3 THEN LINESET=1
5420 IF LINESET=1 THEN NROW=19:LINE(33+NXXPIX,140)-(33+NXXPIX,145),2:NCOLUM=INT
(3+(NXXPIX/7.96)) ELSE NROW=20:LINE(33+NXXPIX,140)-(33+NXXPIX,155),1:NCOLUM=INT
(4+(NXXPIX/7.96))
5425 IF I=1 THEN LOCATE 19,3:PRINT XX(1)
5430 LOCATE NROW,NCOLUM:PRINT XX
5435 NEXT
5440 '
5445 ' printing y-axis numbers
5450 TEMPMAX=YY(1)
5455 TEMPMIN=YY(1)
5460 FOR I=2 TO N
5465 IF YY(I)<TEMPMIN THEN TEMPMIN=YY(I)
5470 IF YY(I)>TEMPMAX THEN TEMPMAX=YY(I)
5475 NEXT
5480 YYMAX=TEMPMAX
5485 YYMIN=TEMPMIN
5490 IF (YYMAX>YYHVAL) THEN 5495 ELSE YYMAX=YYHVAL
5495 IF (YYMIN<YYLVAL) THEN 5500 ELSE YYMIN=YYLVAL
5500 NYYSET=6
5505 YYSPACE=(YYMAX-YYMIN)/(NYYSET-1)
5510 YYMIN2=YYMIN
5515 NYYPIX=-(124/(NYYSET-1))
5520 FOR I=1 TO NYYSET
5525 NYYPIX=NYYPIX+(124/(NYYSET-1))
5530 NROW=INT(18-(NYYPIX/7.96))
5535 IYYMIN$=STR$(YYMIN2)
5540 NCHAR=LEN(IYYMIN$)
5545 IF ASC(MID$(IYYMIN$,1,1))=32 THEN IYYMIN$=MID$(IYYMIN$,2,NCHAR-1):IYYMIN=V
AL(IYYMIN$)
5550 IF NCHAR>4 THEN IYYMIN$=MID$(IYYMIN$,1,4):IYYMIN=VAL(IYYMIN$)
5555 LOCATE NROW,1
5560 PRINT IYYMIN$
5565 YYMIN2=YYMIN2+YYSPACE
5570 LINE(26,140-NYYPIX)-(32,140-NYYPIX),1
5575 NEXT
5580 '
5585 ' draw axis lines
5590 LINE(33,140)-(283,140),1
5595 LINE(33,140)-(33,16),1
5600 '
5605 '
5610 ' drawing the curve
5615 FOR I=1 TO N
5620 XX(I)=33+((XX(I)/XXMAX)*250)
5625 YY(I)=140-(((YY(I)-YYMIN)/(YYMAX-YYMIN))*124)
5630 NEXT
5635 FOR I=2 TO N
5640 LINE (XX(I-1),YY(I-1))-(XX(I),YY(I)),3:NEXT
5645 FOR I=1 TO N
```

```
5650 LINE(XX(I)-2,YY(I)-2)-(XX(I)+2,YY(I)+2),2
5655 LINE(XX(I)+2,YY(I)-2)-(XX(I)-2,YY(I)+2),2
5660 NEXT
5665 COOLXX=33+((COOLTI-XXMIN)/(XXMAX-XXMIN))*250
5670 YYLINE=140!
5675 FOR I=1 TO 13
5680 LINE(COOLXX,YYLINE)-(COOLXX,YYLINE-5),1:YYLINE=YYLINE-10:NEXT
5685 GET 1,161:LOCATE 22,1:GOSUB 5240:PRINT COMMT40$;
5690 GET 1,162:LOCATE 23,1:GOSUB 5240:PRINT COMMT40$;:LOCATE 23,13:PRINT CHR$(2
4);
5695 GOSUB 5260
5700 GOTO 3795
5705 '
5710 '*************************************************************************
```

ALPHAD

```
1          * * * * * M E N U * * * * *
2 1  Do process calculations for a can
3 2  Exit to MAIN selection menu (with can/pouch choices)
4 Select an item from the menu (enter 1 or 2 and RETURN)
5 You did not enter either 1 or 2
6 Cooling Water Temperature (deg C)
7 Enter 1 to let h=infinite, 2 to specify h
8 The number must be 1 or 2
9 Number:
10 Initial Food Temperature (deg C)
11 Retort Temperature (deg C)
12 PRESS ANY KEY TO CONTINUE
13 The Initial and Cooling Water Temp must be <= Retort Temp
14 Initial Temp:   Retort Temp:   Water Temp:
15 Total Processing Time (heat + cool) (<10000 sec)
16 Time at Which Cooling Starts (sec)
17 Total Time and Cooling Time must be>0, Total>=Cooling, & Total<=10000
18 Total Time:   Cooling Time:
19 Number of Time Increments (>=2, <=21)
20 You did not enter a number between 2 and 21
21 Number of Time Values:
22 Thermal Diffusivity (m**2/sec)
23 The Thermal Diffusivity must be > 0
24 You entered the following value(s):
25 Food Density (kg/m**3)
26 The Food Density must be > 0
27 Food Density:
28 Specific Heat of Food (kJ/kg-deg C)
29 The Specific Heat must be > 0
```

```
30 Specific Heat:
31 Can Radius (m)
32 Can Half-Length (m)
33 Thermal Diffusivity must be between 1.0e-07 and 9.99e-07
34 The Can Radius and Half-length must be > 0
35 Radius:      Half-Length:
36 The Number of Radial Values (2<=nr<=10)
37 The Number of Axial Values (2<=nz<=10)
38
39 The Number of Radial and Axial values must both be between 2 and 10
40 NR:       NZ:
41 Convective Heat Transfer Coeff (W/m**2-deg C)
42 The surface convective heat transfer coefficient must be > 0
43 H:
44 Thermal Diffusivity:
45 What is the F-value for the contaminant at Tref=121 deg C) - in minutes?
46 The F-value cannot equal 0
47 A F-value greater than 20 minutes is unusual
48 Do you want to reenter the value (Y or N and RETURN)?
49 What is the z-value for the contaminant (at Tref=121 deg C) - in deg C?
50 The z-value cannot equal 0
51 A z-value greater than 30 deg C is unusual
52 What is the F-value for the nutrient (at Tref=121 deg C) - in minutes?
53 A F-value greater than 100 minutes is unusual
54 What is the z-value for the nutrient (at Tref=121 deg C) - in deg C?
55                    ARGUMENT LISTING FOR SUBROUTINE
56 Enter the new value and press RETURN
57 Press RETURN alone if you want to keep the old value
```

58

59 ARGUMENT OLD NEW

60 A z-value greater than 50 deg C is unusual

61 Do you want to redo the CAN calculations (Y or N and RETURN)?

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

```
144
145
146
147
148
149
150          RESULTS DISPLAY MENU
151         1) Temp-Time Graph
152         2) Lethality-Time Graph
153         3) Nutrient Fraction-Time Graph
154               Enter 1, 2, ... 8
155             GRAPHIC DISPLAY MENU
156   X(m)            Y(m)              Z(m)
157 At which X do you want Temp-Time Graph
158 At which Y do you want Temp-Time Graph
159 At which Z do you want Temp-Time Graph
160    Select the Z position (1, 2 or ...)
161 Press "X" to exit to results menu or
162         Press  -PrtSc to print graph
163                PRINTING MENU
164  Choose the printing options dimension
165          1) Print all the data
166          2) Print specific data
167
168    Here is a list of various values
169      X(m)            Y(m)            Z(m)
170      Y(m)            X(m)            Z(m)
171      Z(m)            Y(m)            x(m)
```

```
172 X
173 Y
174 Z
175 Which X are you starting at(1,2 or ...)
176 Which Y are you starting at(1,2 or ...)
177 Which Z are you starting at(1,2 or ...)
178 Which X do you want to stop(1,2 or ...)
179 Which Y do you want to stop(1,2 or ...)
180 Which Z do you want to stop(1,2 or ...)
181          ENTER "X", "Y" OR "Z"
182                RESULTS
183 TIME(s)   X(m)   Y(m)   Z(m) TEMP(degC)
184 185
185 Z(m)      Y(m)       X(m)   Temp(deg C)
186 Press "X" to exit to results menu
187 Press "Y" to continue
188 Press "P" to print specified data
189   TIME(sec)        X(m)          Y(m)          Z(m)        TEMP(deg C)
190   TIME(s)        RADIUS(m)      Z-VALUES(m)    TEMPERATURE(deg C)
191
192        Radii(m)          Z-values(m)
193 Select the axis to be plotted: R or Z
194   Select the R position (1, 2, or ...)
195   Select the Z position (1, 2, or ...)
196 TEMPERATURE(deg C)
197          TEMP VS TIME
198     * * * P L E A S E  W A I T * * *
199
200
```

```
201
202                    1)  R, Z
203                    2)  Z, R
204          Z-value              Radius
205 Which R are you starting at(1,2, or...)
206 Which Z are you starting at(1,2, or...)
207 Which R do you want to stop(1,2, or...)
208 Which Z do you want to stop(1,2, or...)
209 TIME(s)     R(m)       Z(m)    TEMP(deg C)
210       Z              RADIUS        TEMPERATURE
211 LETHALITY
212                 LETHALITY vs TIME
213               ENTER 1, 2 OR 3
214              ENTER  1  OR  2
215                       TIME (sec)
216                 NUTRIENT FRACTION vs TIME
217 FRACTION
218          4) Temp-Time Table
219          5) Lethality-Time Table
220          6) Nutrient Fraction-Time Table
221          8) Exit
222       TIME (sec)          LETHALITY
223        TIME (sec)         NUTRIENT FRACTION
224              TIME (sec)                          NUTRIENT FRACTION
225              TIME (sec)                          LETHALITY
226          7) Process Parameter Table
227                                  RESULTS
228                                                              RESULTS
```

```
229
230         Process Parameter Table
231         ----------------------
232 Initial Temperature(deg C)
233 Retort Temperature(deg C)
234 Water Temperature(deg C)
235 Fh-1/Slope of Heating Curve(sec)
236 Fc-1/Slope of Heating Curve(sec)
237 Heating Curve Lag Factor
238 Cooling Curve Lag Factor
239 Total Processing Time(sec)
240 Time When Cooling Starts(sec)
241 Number of Time Increments
242 Half X-Length(m)
243 Half Y-Length(m)
244 Half Z-Length(m)
245       7) Process Parameter Table
246 Number of R-values
247 Number of Y-values
248 Food Density(kg/m**3)
249 Food Specific Heat(kJ/kg-degC)
250 You did not enter an integer, try again
251 You did not enter 1,2,...,8 -try again
252 You did not enter X, Y or Z -try again
253    The value was too big - try again
254          You did not enter "X"
255        Value is too big, try again
256 End value can not be < then start value
257 You did not enter X, Y or P
```

```
258 There is no more data,enter P or X
259 You did not enter R or Z - try again
260 You did not enter 1 or 2 - try again
261 You did not enter 1, 2 or 3 - try again
262 Can Radius (m)
263
264 Number of Z-values
265 H(W/m**2-degC)
266 Thermal Diffusivity(m**2/sec)
267 You did not enter X or P
```

APPENDIX 3.7

Program Listing of MAINE

```
10 '
20 '
30 '          ********** M A I N E **********
40 '
50 '
60 'Initialization and Dimensioning
70 DIM TEMP(51),TIME(51),TAVG(50)
80 DIM TEMPOBJ(11),TIMCOOL(11)
90 DIM FXNUT(4),ZNUT(4),WTFACT(4)
100 DIM FRANUT(4,11), OBJFN(11), FRAOPT(4)
110 DIM XX(11),YY(11), XCURVE(21),YCURVE(21)
120 'curve-fitting
130 DIM ZCCF(4),ACCF(4,4),C1CCF(4),YCCF(35),UCCF(35,4)
140 DIM WCCF(4,1),BCCF(4,4), I2CCF%(4,3), XCCF(35), Y1CCF(35)
150 DIM Y2CCF(35), R3CCF(35), E2CCF(4)
160 DIM XXCURVE(401),YYCURVE(401)
170 '
180 '
190 '
200 'Setting up the text screen
210 KEY OFF
220 SCREEN 0,0,0,0
230 WIDTH 80
240 COLOR 7,1,1
250 CLS
260 '
270 '
280 'Initializing the temp range TEMPOBJ
290 TEMPOBJ(1)=110!: FOR IME=2 TO 11
300 TEMPOBJ(IME)=TEMPOBJ(IME-1) + 2!:  NEXT IME
310 '
320 ' opening and readying the associated files
330 OPEN "c:alphae" AS 1
340 FIELD 1,80 AS COMMENT$
350 '
360 OPEN "b:commc" AS 2
370 FIELD 2,80 AS COMMC$
380 '
390 '
400 ' setting the pointer to current record number of commc
410 GET 2,1: NRECCOUNT=VAL(COMMC$)
420 BLANK$=" ": LSET COMMC$=BLANK$: PUT 2,NRECCOUNT+1
430 '
440 '
450 ' Checking if control has come from MAIN or from SUBs
460 OPEN "b:store2" FOR INPUT AS 3
470 INPUT#3,NOPCODE
480 CLOSE 3
490 '
500 ' Not coming from MAIN:
510 IF NOPCODE=0 THEN GOTO 570
520 IF NOPCODE=1 THEN GOTO 2390
530 IF NOPCODE=2 THEN GOTO 3180
540 IF NOPCODE=3 OR NOPCODE=4 THEN GOTO 6540
550 '
560 '
570 ' Coming from MAIN:
```

```
580 OPEN "b:comma" FOR INPUT AS 4
590 '
600 '    ** read info from comma**
610 '
620 CLOSE 4
630 '
640 '
650 'if want to go to BALLST
660 'getting the arguments used for the SUB:
670 N1STTHRU=0
680 '
690 'determining MODEL wanted
700 IF MODEL=0 THEN NOLD=1 ELSE NOLD = MODEL
710 JRECNO=40: NMARKSUB=0: GOSUB 8280
720 NEW1=VAL(NEWVALUE$)
730 IF (NEW1>=1) AND (NEW1<=3) THEN GOTO 760
740 JRECNO=41: JRECNO2=42: NMARKSET=1: GOSUB 8620
750 GOTO 700
760 MODEL=NEW1
770 '
780 'determining temperatures
790 IF TINIT=0! THEN OLD=45! ELSE OLD=TINIT
800 JRECNO=8:NMARKSUB=1: GOSUB 8280
810 XNEW1=VAL(NEWVALUE$)
820 IF TWATER=0! THEN OLD=60! ELSE OLD=TWATER
830 JRECNO=9:NMARKSUB=1: GOSUB 8280
840 XNEW2=VAL(NEWVALUE$)
850 IF (XNEW1>0!) AND (XNEW2>0!) AND (XNEW2<=90!) THEN GOTO 880
860 JRECNO=10: JRECNO2=11: NMARKSET=2: GOSUB 8620
870 GOTO 790
880 TINIT=XNEW1: TWATER=XNEW2
890 '
900 'determining time values
910 IF TOTIME=0! THEN OLD=4500 ELSE OLD=TOTIME
920 JRECNO=13: NMARKSUB=1: GOSUB 8280
930 XNEW1=VAL(NEWVALUE$)
940 IF (XNEW1=>1800!) AND (XNEW1<=9000!) THEN GOTO 970
950 JRECNO=14: JRECNO2=15: NMARKSET=1: GOSUB 8620
960 GOTO 910
970 TOTIME=XNEW1
980 '
990 'determining number of time values
1000 NBA=51
1010 '
1020 'determining (1/slopes) of heating and cooling curves
1030 IF FH=0! THEN OLD=1000! ELSE OLD=FH
1040 JRECNO=16: NMARKSUB=1: GOSUB 8280
1050 XNEW1=VAL(NEWVALUE$)
1060 IF FC=0! THEN OLD=1000! ELSE OLD=FC
1070 JRECNO=17: NMARKSUB=1: GOSUB 8280
1080 XNEW2=VAL(NEWVALUE$)
1090 IF (XNEW1>700) AND (XNEW2>700) THEN GOTO 1120
1100 JRECNO=18: JRECNO2=19: NMARKSET=2: GOSUB 8620
1110 GOTO 1030
1120 FH=XNEW1: FC=XNEW2
1130 '
1140  'determining lag factor of heating curve
```

```
1150 IF JH!=0! THEN OLD=1.5 ELSE OLD=JH!
1160 IF MODEL=1 THEN JRECNO=20 ELSE JRECNO=21
1170 NMARKSUB=1: GOSUB 8280
1180 XNEW1=VAL(NEWVALUE$)
1190 IF (XNEW1>0) THEN GOTO 1230
1200 IF (MODEL<>1) AND (XNEW1=0) THEN GOTO 1210 ELSE GOTO 1230
1210 JRECNO=22: JRECNO2=23: NMARKSET=1: GOSUB 8620
1220 GOTO 1150
1230 IF MODEL=1 THEN JH!=XNEW1: GOTO 1280
1240 IF ABS(XNEW1-1.866)>.01 THEN JH!=XNEW1: GOTO 1280
1250 JRECNO=25: JRECNO2=23: NMARKSET=1: GOSUB 8620
1260 GOTO 1150
1270 '
1280 'determining lag factor of cooling curve
1290 IF MODEL<>3 THEN GOTO 1400
1300 IF JC!=0! THEN OLD=2! ELSE OLD=JC!
1310 JRECNO=26: NMARKSUB=1: GOSUB 8280
1320 XNEW1=VAL(NEWVALUE$)
1330 IF XNEW1>0 THEN GOTO 1360
1340 JRECNO=27: JRECNO2=28: GOSUB 8620
1350 GOTO 1300
1360 IF ABS(XNEW1-1.866)>.01 THEN JC!=XNEW1: GOTO 1400
1370 JRECNO=29: JRECNO2=28: NMARKSET=1: GOSUB 8620
1380 GOTO 1290
1390 '
1400 'getting the target process lethality (userleth) from the user
1410 IF USERLETH=0! THEN OLD=1.1 ELSE OLD=USERLETH
1420 JRECNO=30: NMARKSUB=1: GOSUB 8280
1430 XNEW1=VAL(NEWVALUE$)
1440 IF (XNEW1>=1!) AND (XNEW1<=2.5) THEN GOTO 1470
1450 JRECNO=31: JRECNO2=32: NMARKSET=1: GOSUB 8620
1460 GOTO 1410
1470 USERLETH=XNEW1
1480 '
1490 '
1500 'Getting the FLETH value
1510 CLS: JRECNO=33: GOSUB 7700
1520 IF JOK<>-1 THEN GOTO 1510 ELSE FLETH=VAL(ANSR$)
1530 IF (FLETH>0!) AND (FLETH<=20!) GOTO 1640
1540 IF FLETH>20! THEN GOTO 1580
1550 GET 1,34: PRINT COMMENT$
1560 GET 1,12: PRINT: PRINT COMMENT$
1570 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1570 ELSE GOTO 1510
1580 CLS: GET 1,35: PRINT COMMENT$
1590 GET 1,24: PRINT COMMENT$;FLETH
1600 JRECNO=36: GOSUB 7460
1610 IF JOK<>-1 GOTO 1580
1620 IF (DY$="n") OR (DY$="N") GOTO 1640 ELSE GOTO 1510
1630 '
1640 'Getting the ZLETH value
1650 CLS: JRECNO=37: GOSUB 7700
1660 IF JOK<>-1 THEN GOTO 1650 ELSE ZLETH=VAL(ANSR$)
1670 IF (ZLETH>0!) AND (ZLETH<=30!) GOTO 1780
1680 IF ZLETH>30! THEN GOTO 1720
1690 GET 1,38: PRINT COMMENT$
1700 GET 1,12: PRINT: PRINT COMMENT$
1710 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 1710 ELSE GOTO 1650
```

```
1720 CLS: GET 1,39: PRINT COMMENT$
1730 GET 1,24: PRINT COMMENT$;ZLETH
1740 JRECNO=36: GOSUB 7460
1750 IF JOK<>-1 GOTO 1720
1760 IF (DY$="n") OR (DY$="N") GOTO 1780 ELSE GOTO 1650
1770 '
1780 NOPCODE=1
1790 '
1800 'getting the FXNUT value for each nutrient
1810 KME=1: KREC=46: GOSUB 9550
1820 KME=2: KREC=47: GOSUB 9550
1830 KME=3: KREC=48: GOSUB 9550
1840 KME=4: KREC=49: GOSUB 9550
1850 '
1860 'getting the ZNUT value for each nutrient
1870 KME=1: MREC=51: GOSUB 9720
1880 KME=2: MREC=52: GOSUB 9720
1890 KME=3: MREC=53: GOSUB 9720
1900 KME=4: MREC=54: GOSUB 9720
1910 '
1920 'getting the weighting factor WTFACT for each nutrient
1930 KME=1: NREC=61: GOSUB 9890
1940 KME=2: NREC=62: GOSUB 9890
1950 KME=3: NREC=63: GOSUB 9890
1960 KME=4: NREC=64: GOSUB 9890
1970 '
1980 '
1990 'set TIMCOOL(1)=TOTIME at first TEMPOBJ to check that TOTIME is ok
2000 TIMCOOL(1)=TOTIME
2010 '
2020 '
2030 ' copying arguments into COMM1 to run the SUB chosen
2040 OPEN "c:comm1" FOR OUTPUT AS 6
2050 NSUBMAIN=5
2060 PRINT#6,MODEL;TINIT;TEMPOBJ(1);TWATER;NBA;TOTIME;TIMCOOL(1);FH;FC;JH!;JC!;
NSUBMAIN
2070 CLOSE 6
2080 '
2090 '
2100 'putting info into STORE2 to tell MAINE what SUB it chained to
2110 OPEN "b:store2" FOR OUTPUT AS 3
2120 PRINT#3,NOPCODE
2130 PRINT#3,MODEL;TINIT;TEMPOBJ(1);TWATER;NBA;TOTIME;TIMCOOL(1);FH;FC;JH!;JC!
2140 PRINT#3,USERLETH;FLETH;ZLETH;NSUBMAIN
2150 PRINT#3,FXNUT(1);FXNUT(2);FXNUT(3);FXNUT(4)
2160 PRINT#3,ZNUT(1);ZNUT(2);ZNUT(3);ZNUT(4)
2170 PRINT#3,WTFACT(1);WTFACT(2);WTFACT(3);WTFACT(4)
2180 CLOSE 3
2190 '
2200 '
2210 '
2220 ' chaining to BALLST
2230 'storing the current record number of COMMC into the first rec no of COMMC

2240 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
2250 '
2260 CLOSE 1,2
```

```
2270 '
2280 'Displaying "Please Wait" symbol
2290 CLS:SCREEN 1,0,0,0
2300 COLOR 5,1
2310 DEF SEG = &HB800
2320 BLOAD "b:clock.scr",0
2330 ON ERROR GOTO 2340
2340 IF INKEY$<>"" THEN BEEP
2350 CHAIN "BALLST"
2360 '
2370 '
2380 '****************** nopcode=1 ******************************************
2390 'Coming back from BALLST first time
2400 'Calculating lethality to see if TOTIME is big enough
2410 OPEN "b:store2" FOR INPUT AS 3
2420 INPUT#3,NOPCODE
2430 INPUT#3,MODEL,TINIT,TEMPOBJ(1),TWATER,NBA,TOTIME,TIMCOOL(1),FH,FC,JH!,JC!
2440 INPUT#3,USERLETH,FLETH,ZLETH,NSUBMAIN
2450 INPUT#3,FXNUT(1),FXNUT(2),FXNUT(3),FXNUT(4)
2460 INPUT#3,ZNUT(1),ZNUT(2),ZNUT(3),ZNUT(4)
2470 INPUT#3,WTFACT(1),WTFACT(2),WTFACT(3),WTFACT(4)
2480 CLOSE 3
2490 '
2500 'Getting TEMP results from COMM2 with user TOTIME
2510 OPEN "c:comm2" FOR INPUT AS 7
2520 FOR ITIME =1 TO NBA
2530 INPUT#7,TIME(ITIME),TEMP(ITIME)
2540 NEXT ITIME
2550 CLOSE 7
2560 '
2570 'Calculating lethality with user TOTIME
2580 DTIME=TOTIME/(NBA-1)/60!
2590 FOR KME=1 TO (NBA-1)
2600 TAVG(KME)= (TEMP(KME) + TEMP(KME+1)) / 2!
2610 IF (-121! + TAVG(KME))/ZLETH <-35! THEN RATEL=0! : GOTO 2630
2620 RATEL=1/FLETH * ( 10!^((-121!+TAVG(KME))/ZLETH) )
2630 IF KME=1 THEN XLETH=RATEL*DTIME  ELSE  XLETH=XLETH + RATEL*DTIME
2640 NEXT KME
2650 '
2660 'Checking process adequacy
2670 IF XLETH>=USERLETH THEN GOTO 2760 'when TOTIME is ok
2680 IF XLETH/USERLETH<.5 THEN TOTIME=TOTIME+TOTIME*.5 ELSE TOTIME=TOTIME+TOTIM
E*1!/3!
2690 CLS: GET 1,43: PRINT COMMENT$:PRINT
2700 GET 1,44: PRINT COMMENT$
2710 PRINT TOTIME:PRINT
2720 GET 1,12: PRINT COMMENT$
2730 DUMME$=INKEY$: IF DUMME$="" THEN GOTO 2730
2740 GOTO 1990
2750 '
2760 ITEMP=1
2770 '
2780 '
2790 'Once TOTIME is ok, now getting cooling times for each TEMPOBJ
2800 '
2810 'getting the temp-time array using timcool=totime
2820 TIMCOOL(ITEMP)=TOTIME
```

```
2830 '
2840 NOPCODE=2
2850 DCOOL=TOTIME/4!
2860 '
2870 OPEN "c:comm1" FOR OUTPUT AS 6
2880 NSUBMAIN=5
2890 PRINT#6,MODEL;TINIT;TEMPOBJ(ITEMP);TWATER;NBA;TOTIME;TIMCOOL(ITEMP);FH;FC;
JH!;JC!;NSUBMAIN
2900 CLOSE 6
2910 '
2920 '
2930 'putting info into STORE2 to tell MAINE what SUB it chained to
2940 OPEN "b:store2" FOR OUTPUT AS 3
2950 PRINT#3,NOPCODE
2960 PRINT#3,ITEMP,DCOOL
2970 PRINT#3,MODEL;TINIT;TEMPOBJ(ITEMP);TWATER;NBA;TOTIME;TIMCOOL(ITEMP);FH;FC;
JH!;JC!
2980 PRINT#3,USERLETH;FLETH;ZLETH;NSUBMAIN
2990 PRINT#3,FXNUT(1);FXNUT(2);FXNUT(3);FXNUT(4)
3000 PRINT#3,ZNUT(1);ZNUT(2);ZNUT(3);ZNUT(4)
3010 PRINT#3,WTFACT(1);WTFACT(2);WTFACT(3);WTFACT(4)
3020 PRINT#3,OLDLETH
3030 FOR JME=1 TO ITEMP
3040 PRINT#3,TIMCOOL(JME);TEMPOBJ(JME);FRANUT(1,JME);FRANUT(2,JME);FRANUT(3,JME
);FRANUT(4,JME);OBJFN(JME)
3050 NEXT JME
3060 CLOSE 3
3070 '
3080 '
3090 'chaining to BALLST
3100 'print screen message to wait
3110 CLS: LOCATE 12,30: GET 1,45: PRINT COMMENT$
3120 'storing the current record number of COMMC into the first rec no of COMMC

3130 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
3140 CLOSE 1,2
3150 CHAIN "BALLST"
3160 '
3170 '
3180 '***************** nopcode=2 ************************************
3190 'read info from STORE2
3200 OPEN "b:store2" FOR INPUT AS 3
3210 INPUT#3,NOPCODE
3220 INPUT#3,ITEMP,DCOOL
3230 INPUT#3,MODEL,TINIT,TEMPOBJ(ITEMP),TWATER,NBA,TOTIME,TIMCOOL(ITEMP),FH,FC,
JH!,JC!
3240 INPUT#3,USERLETH,FLETH,ZLETH,NSUBMAIN
3250 INPUT#3,FXNUT(1),FXNUT(2),FXNUT(3),FXNUT(4)
3260 INPUT#3,ZNUT(1),ZNUT(2),ZNUT(3),ZNUT(4)
3270 INPUT#3,WTFACT(1),WTFACT(2),WTFACT(3),WTFACT(4)
3280 INPUT#3,OLDLETH
3290 FOR JME=1 TO ITEMP
3300 INPUT#3,TIMCOOL(JME),TEMPOBJ(JME),FRANUT(1,JME),FRANUT(2,JME),FRANUT(3,JME
),FRANUT(4,JME),OBJFN(JME)
3310 NEXT JME
3320 CLOSE 3
3330 '
```

```
3340 '
3350 'Read temp-time info from COMM2
3360 OPEN "c:comm2" FOR INPUT AS 7
3370 FOR ITIME =1 TO NBA
3380 INPUT#7,TIME(ITIME),TEMP(ITIME)
3390 NEXT ITIME
3400 CLOSE 7
3410 '
3420 'Calculating lethality with TIMCOOL(itemp)
3430 DTIME=TOTIME/(NBA-1)/60!
3440 FOR KME=1 TO (NBA-1)
3450 TAVG(KME)= (TEMP(KME) + TEMP(KME+1)) / 2!
3460 IF (-121! + TAVG(KME))/ZLETH <-35! THEN RATEL=0! : GOTO 3480
3470 RATEL=1/FLETH * ( 10!^((-121!+TAVG(KME))/ZLETH) )
3480 IF KME=1 THEN XLETH=RATEL*DTIME  ELSE  XLETH=XLETH + RATEL*DTIME
3490 NEXT KME
3500 '
3510 '
3520 'Assigning lethalities to variables
3530 XNEWLETH=XLETH
3540 IF TIMCOOL(ITEMP)=TOTIME THEN OLDLETH=XLETH
3550 '
3560 '
3570 ' compare two lethalities
3580 IF ABS(XNEWLETH-USERLETH)<.05 THEN GOTO 3650
3590 IF (XNEWLETH>USERLETH) AND (OLDLETH>USERLETH) THEN DCOOL=.9*DCOOL: TIMCOOL
(ITEMP)=TIMCOOL(ITEMP)-DCOOL: OLDLETH=XNEWLETH: GOTO 2870
3600 IF (XNEWLETH<USERLETH) AND (OLDLETH<USERLETH) THEN DCOOL=1.1*DCOOL: TIMCOO
L(ITEMP)=TIMCOOL(ITEMP)+DCOOL: OLDLETH=XNEWLETH: GOTO 2870
3610 IF (XNEWLETH>USERLETH) AND (OLDLETH<USERLETH) THEN DCOOL=DCOOL/1.9: TIMCOO
L(ITEMP)=TIMCOOL(ITEMP)-DCOOL: OLDLETH=XNEWLETH: GOTO 2870
3620 IF (XNEWLETH<USERLETH) AND (OLDLETH>USERLETH) THEN DCOOL=DCOOL/2.1: TIMCOO
L(ITEMP)=TIMCOOL(ITEMP)+DCOOL: OLDLETH=XNEWLETH: GOTO 2870
3630 '
3640 '
3650 'now that TIMCOOL(itemp) is found, do nutrient retention calculations
3660 '
3670 CLS: LOCATE 12,30: GET 1,45: PRINT COMMENT$
3680 '
3690 'calculating nutrient fraction retained, for each nutrient, at TEMPOPJ(ite
mp)
3700 DTIME=TOTIME/50!/60!
3710 FOR KME=1 TO 4
3720    FOR NME=1 TO 50
3730       IF (-121!+TAVG(NME))/ZNUT(KME)<-35 THEN RATEN=0!: GOTO 3750
3740       RATEN= 1/FXNUT(KME) * (10!^( (-121!+TAVG(NME))/ZNUT(KME)) )
3750       IF NME=1 THEN XNUTOT=1!* 10!^(-12!*RATEN*DTIME) ELSE XNUTOT=XNUTOT*
 10!^(-12!*RATEN*DTIME)
3760    NEXT NME
3770    FRANUT(KME,ITEMP)=XNUTOT/1!
3780 NEXT KME
3790 '
3800 '
3810 'Calculating the value of the objective function for TEMPOBJ(itemp)
3820 '
3830 'Doing the calculation
3840 XNUMERATOR= WTFACT(1)*FRANUT(1,ITEMP) + WTFACT(2)*FRANUT(2,ITEMP) + WTFACT
```

```
(3)*FRANUT(3,ITEMP) + WTFACT(4)*FRANUT(4,ITEMP)
3850 DENOMINATOR=WTFACT(1) + WTFACT(2) + WTFACT(3) + WTFACT(4)
3860 OBJFN(ITEMP) = XNUMERATOR / DENOMINATOR
3870 'LPRINT"itemp,timcool,objfn=",ITEMP;TIMCOOL(ITEMP);OBJFN(ITEMP)
3880 '
3890 '
3900 'Deciding control pathway: do another TEMPOBJ, or do curve-fitting
3910 IF ITEMP=11 THEN GOTO 3950 ELSE ITEMP=ITEMP+1: GOTO 2790
3920 '
3930 '
3940 '
3950 '*************** Finding equation of objective function ****************
3960 '
3970 'Curve-fitting the objective function values
3980 '
3990 ACCF$=" ###  ##.#  ###.###^^^^  ###.###^^^^  ###.###^^^^"
4000 CCCF$="  ##.####^^^^    ##.###^^^^"
4010 M1CCF%=35
4020 '
4030 'lPRINT
4040 'lPRINT "Parabolic least-squares fit by";
4050 'lPRINT "Gauss-Jordan elimination"
4060 GOSUB 4150
4070 '
4080 GOSUB 4250 'set up the matrix
4090 GOSUB 4710 'square up the matrix
4100 GOSUB 4890 'Gauss-Jordan solution
4110 GOSUB 4350 'print results
4120 '
4130 GOTO 5890 'getting maximum of the objective function equation
4140 '
4150 'get the data
4160 N1CCF%=ITEMP
4170 N2CCF%=4
4180 FOR ICCF%=1 TO N1CCF%
4190   XCCF(ICCF%)=TEMPOBJ(ICCF%)
4200   Y1CCF(ICCF%)=OBJFN(ICCF%)
4210 'lPRINT XCCF(ICCF%),Y1CCF(ICCF%)
4220 NEXT ICCF%
4230 L3CCF%=(N1CCF%-1) * 2 +1
4240 RETURN
4250 '
4260 FOR ICCF%=1 TO N1CCF%
4270   UCCF(ICCF%,1) = 1
4280   FOR JCCF%=2 TO N2CCF%
4290    UCCF(ICCF%,JCCF%)=UCCF(ICCF%,JCCF%-1) * XCCF(ICCF%)
4300   NEXT JCCF%
4310   YCCF(ICCF%)=Y1CCF(ICCF%)
4320 NEXT ICCF%
4330 RETURN
4340 '
4350 'calculate the residuals and print results
4360 S7CCF=0
4370 S8CCF=0
4380 T6CCF=0
4390 FOR ICCF%=1 TO N1CCF%
4400   Y2CCF=0
```

```
4410   FOR JCCF%= 1 TO N2CCF%
4420     Y2CCF=Y2CCF + C1CCF(JCCF%) * UCCF(ICCF%,JCCF%)
4430   NEXT JCCF%
4440   R3CCF(ICCF%)= Y2CCF-YCCF(ICCF%)
4450   Y2CCF(ICCF%)= Y2CCF
4460   T6CCF =T6CCF + R3CCF(ICCF%)*R3CCF(ICCF%)
4470   S7CCF =S7CCF + YCCF(ICCF%)
4480   S8CCF =S8CCF + YCCF(ICCF%)*YCCF(ICCF%)
4490 NEXT ICCF%
4500 C3CCF=SQR(1-T6CCF/(S8CCF-S7CCF*S7CCF/N1CCF%))
4510 IF (N1CCF%=N2CCF%) THEN E5CCF=SQR(T6CCF)
4520 IF (N1CCF%<>N2CCF%) THEN E5CCF=SQR(T6CCF/(N1CCF%-N2CCF%))
4530 FOR JCCF%=1 TO N2CCF%
4540   E2CCF(JCCF%) =E5CCF*SQR(ABS(BCCF(JCCF%,JCCF%)))
4550 NEXT JCCF%
4560 'LPRINT "      x     y     ycalc     resid"
4570 FOR ICCF%=1 TO N1CCF%
4580  'LPRINT USING ACCF$;ICCF%;XCCF(ICCF%),YCCF(ICCF%),Y2CCF(ICCF%),R3CCF(ICCF
%)
4590 NEXT ICCF%
4600 'LPRINT
4610 'LPRINT "coefficients   errors"
4620 'LPRINT USING CCCF$;C1CCF(1),E2CCF(1);
4630 'LPRINT "   Constant term"
4640 FOR ICCF%=2 TO N2CCF%
4650   'LPRINT USING CCCF$;C1CCF(ICCF%),E2CCF(ICCF%)
4660 NEXT ICCF%
4670 'LPRINT
4680 'LPRINT "correlation coefficient is";C3CCF
4690 RETURN 'from printout
4700 '
4710 'Uccf and Yccf converted to accf and zccf
4720 ' n1ccf%   nrowccf%  number of rows
4730 ' n2ccf%   ncolccf%  number of columns
4740 FOR KCCF%=1 TO N2CCF%
4750   FOR LCCF%=1 TO KCCF%
4760    ACCF(KCCF%,LCCF%)=0
4770    FOR ICCF%=1 TO N1CCF%
4780      ACCF(KCCF%,LCCF%)=ACCF(KCCF%,LCCF%) + UCCF(ICCF%,LCCF%)*UCCF(ICCF%,K
CCF%)
4790      IF (KCCF%<>LCCF%) THEN ACCF(LCCF%,KCCF%) = ACCF(KCCF%,LCCF%)
4800    NEXT ICCF%
4810 NEXT LCCF%
4820 ZCCF(KCCF%)=0
4830 FOR ICCF%=1 TO N1CCF%
4840   ZCCF(KCCF%)=ZCCF(KCCF%) + YCCF(ICCF%)*UCCF(ICCF%,KCCF%)
4850 NEXT ICCF%
4860 NEXT KCCF%
4870 RETURN
4880 '
4890 ' Gauss-Jordan matrix inversion and solution
4900 E1CCF%=0 'becomes 1 for singular matrix
4910 I5CCF%=1 'print inverse matrix if zero
4920 N3CCF%=1 'number of constant vectors
4930 FOR ICCF%=1 TO N2CCF%
4940   FOR JCCF%=1 TO N2CCF%
4950    BCCF(ICCF%,JCCF%)=ACCF(ICCF%,JCCF%)
```

```
4960    NEXT JCCF%
4970    WCCF(ICCF%,1)=ZCCF(ICCF%)
4980    I2CCF%(ICCF%,3)=0
4990  NEXT ICCF%
5000  D3CCF=1
5010  FOR ICCF%=1 TO N2CCF%
5020  '
5030  ' search for largest (pivot) element
5040  B1CCF=0
5050  FOR JCCF%=1 TO N2CCF%
5060    IF (I2CCF%(JCCF%,3)=1) THEN 5150
5070    FOR KCCF%=1 TO N2CCF%
5080     IF (I2CCF%(KCCF%,3)>1) THEN 5820
5090     IF (I2CCF%(KCCF%,3)=1) THEN 5140
5100     IF (B1CCF>=ABS(BCCF(JCCF%,KCCF%))) THEN 5140
5110     I3CCF%=JCCF%
5120     I4CCF%=KCCF%
5130     B1CCF=ABS(BCCF(JCCF%,KCCF%))
5140    NEXT KCCF%
5150    NEXT JCCF%
5160    I2CCF%(I4CCF%,3)=I2CCF%(I4CCF%,3)+1
5170    I2CCF%(ICCF%,1)=I3CCF%
5180    I2CCF%(ICCF%,2)=I4CCF%
5190  '
5200    'interchange rows to put pivot on diagonal
5210    IF (I3CCF%=I4CCF%) THEN 5360
5220    D3CCF=-D3CCF
5230    FOR LCCF%=1 TO N2CCF%
5240      H1CCF=BCCF(I3CCF%,LCCF%)
5250      BCCF(I3CCF%,LCCF%)=BCCF(I4CCF%,LCCF%)
5260      BCCF(I4CCF%,LCCF%)=H1CCF
5270    NEXT LCCF%
5280    IF (N3CCF%<1) THEN 5360
5290    FOR LCCF%=1 TO N3CCF%
5300     H1CCF=WCCF(I3CCF%,LCCF%)
5310     WCCF(I3CCF%,LCCF%)=WCCF(I4CCF%,LCCF%)
5320     WCCF(I4CCF%,LCCF%)=H1CCF
5330    NEXT LCCF%
5340  '
5350    'divide pivot row by pivot element
5360    P1CCF=BCCF(I4CCF%,I4CCF%)
5370    D3CCF=D3CCF*P1CCF
5380    BCCF(I4CCF%,I4CCF%)=1
5390    FOR LCCF%=1 TO N2CCF%
5400     BCCF(I4CCF%,LCCF%)=BCCF(I4CCF%,LCCF%)/P1CCF
5410    NEXT LCCF%
5420    IF (N3CCF%<1) THEN 5480
5430    FOR LCCF%=1 TO N3CCF%
5440     WCCF(I4CCF%,LCCF%)=WCCF(I4CCF%,LCCF%)/P1CCF
5450    NEXT LCCF%
5460    '
5470    'reduce nonpivot rows
5480    FOR L1CCF%=1 TO N2CCF%
5490      IF (L1CCF%=I4CCF%) THEN 5590
5500      TCCF=BCCF(L1CCF%,I4CCF%)
5510      BCCF(L1CCF%,I4CCF%)=0
5520      FOR LCCF%=1 TO N2CCF%
```

```
5530        BCCF(L1CCF%,LCCF%)=BCCF(L1CCF%,LCCF%)-BCCF(I4CCF%,LCCF%)*TCCF
5540      NEXT LCCF%
5550      IF (N3CCF%<1) THEN 5590
5560      FOR LCCF%=1 TO N3CCF%
5570         WCCF(L1CCF%,LCCF%)=WCCF(L1CCF%,LCCF%)-WCCF(I4CCF%,LCCF%)*TCCF
5580      NEXT LCCF%
5590    NEXT L1CCF%
5600 NEXT ICCF%
5610 '
5620 'interchange columns
5630 FOR ICCF%=1 TO N2CCF%
5640    LCCF%=N2CCF%-ICCF%+1
5650    IF (I2CCF%(LCCF%,1)=I2CCF%(LCCF%,2)) THEN 5730
5660    I3CCF%=I2CCF%(LCCF%,1)
5670    I4CCF%=I2CCF%(LCCF%,2)
5680    FOR KCCF%=1 TO N2CCF%
5690           H1CCF=BCCF(KCCF%,I3CCF%)
5700           BCCF(KCCF%,I3CCF%)=BCCF(KCCF%,I4CCF%)
5710           BCCF(KCCF%,I4CCF%)=H1CCF
5720    NEXT KCCF%
5730 NEXT ICCF%
5740 FOR KCCF%=1 TO N2CCF%
5750    IF (I2CCF%(KCCF%,3)<>1) THEN 5820
5760 NEXT KCCF%
5770 E1CCF%=0
5780 FOR ICCF%=1 TO N2CCF%
5790    C1CCF(ICCF%)=WCCF(ICCF%,1)
5800 NEXT ICCF%
5810 IF (I5CCF%=1) THEN 5860
5820 E1CCF%=1
5830 'lPRINT "ERROR - matrix is singular or inverse matrix is zero "
5840 'lPRINT "Curve-fitting cannot be done here"
5850 IERRFLAG=1
5860 RETURN
5870 '
5880 '
5890 IF IERRFLAG=1 THEN GOTO 5900 ELSE GOTO 5940
5900 OBJOPT=OBJFN(1):
5910 FOR INOT=2 TO 11: IF OBJFN(INOT)>OBJOPT THEN OBJOPT=OBJFN(INOT):IOPT=INOT
5920 NEXT INOT
5930 TEMPOPT=TEMPOBJ(IOPT):COOLOPT=TIMCOOL(IOPT):FOR KME=1 TO 4: FRAOPT(KME)=FR
ANUT(4,IOPT):NEXT KME: GOTO 10020
5940 'getting the maximum of the objective function equation, in range 110-130
C
5950 '   first get x,y values of the curve
5960 FOR ICCF=1 TO 401
5970    IF ICCF=1 THEN XXCURVE(1)=110! ELSE XXCURVE(ICCF)=XXCURVE(ICCF-1)+.05
5980    YYCURVE(ICCF)= C1CCF(1) + C1CCF(2)*XXCURVE(ICCF) + C1CCF(3)*(XXCURVE(IC
CF)^2) + C1CCF(4)*(XXCURVE(ICCF)^3)
5990 NEXT ICCF
6000 '
6010 'find the minimum and maximum y values of the curve equation
6020 YCURMAX=YYCURVE(1)
6030 YCURMIN=YYCURVE(1)
6040 FOR JMAX=2 TO 401
6050    IF YYCURVE(JMAX)>YCURMAX THEN YCURMAX=YYCURVE(JMAX):IIMAX=JMAX
6060    IF YYCURVE(JMAX)<YCURMIN THEN YCURMIN=YYCURVE(JMAX)
```

```
6070 NEXT JMAX
6080 '
6090 '
6100 '
6110 'find COOLOPT corresponding to TEMPOPT
6120 'getting the temp-time array using COOLOPT=TOTIME
6130 COOLOPT=TOTIME
6140 TEMPOPT=XXCURVE(IIMAX)
6150 '
6160 NOPCODE=3 : OLDLETH=0!
6170 DCOOL=TOTIME/4!
6180 '
6190 OPEN "c:comm1" FOR OUTPUT AS 6
6200 NSUBMAIN=5
6210 PRINT#6,MODEL;TINIT;TEMPOPT;TWATER;NBA;TOTIME;COOLOPT;FH;FC;JH!;JC!;NSUBMA
IN
6220 CLOSE 6
6230 '
6240 '
6250 'putting info into STORE2 to tell MAINE what SUB it chained to
6260 OPEN "b:store2" FOR OUTPUT AS 3
6270 PRINT#3,NOPCODE
6280 PRINT#3,DCOOL
6290 PRINT#3,MODEL;TINIT;TEMPOPT;TWATER;NBA;TOTIME;COOLOPT;FH;FC;JH!;JC!
6300 PRINT#3,USERLETH;FLETH;ZLETH;NSUBMAIN
6310 PRINT#3,FXNUT(1);FXNUT(2);FXNUT(3);FXNUT(4)
6320 PRINT#3,ZNUT(1);ZNUT(2);ZNUT(3);ZNUT(4)
6330 PRINT#3,WTFACT(1);WTFACT(2);WTFACT(3);WTFACT(4)
6340 PRINT#3,OLDLETH
6350 FOR JME=1 TO 11
6360 PRINT#3,TIMCOOL(JME);TEMPOBJ(JME);FRANUT(1,JME);FRANUT(2,JME);FRANUT(3,JME
);FRANUT(4,JME);OBJFN(JME)
6370 NEXT JME
6380 PRINT#3,YCURMAX;YCURMIN
6390 FOR LME=1 TO 401 STEP 20
6400 PRINT#3,XXCURVE(LME);YYCURVE(LME)
6410 NEXT LME
6420 CLOSE 3
6430 '
6440 '
6450 'chaining to BALLST
6460 'print screen message to wait
6470 CLS: LOCATE 12,30: GET 1,45: PRINT COMMENT$
6480 'storing the current record number of COMMC into the first rec no of COMMC

6490 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
6500 CLOSE 1,2
6510 CHAIN "BALLST"
6520 '
6530 '
6540 '***************** nopcode=3 or 4 ************************************
6550 'read info from STORE2
6560 OPEN "b:store2" FOR INPUT AS 3
6570 INPUT#3,NOPCODE
6580 INPUT#3,DCOOL
6590 INPUT#3,MODEL,TINIT,TEMPOPT,TWATER,NBA,TOTIME,COOLOPT,FH,FC,JH!,JC!
6600 INPUT#3,USERLETH,FLETH,ZLETH,NSUBMAIN
```

```
6610 INPUT#3,FXNUT(1),FXNUT(2),FXNUT(3),FXNUT(4)
6620 INPUT#3,ZNUT(1),ZNUT(2),ZNUT(3),ZNUT(4)
6630 INPUT#3,WTFACT(1),WTFACT(2),WTFACT(3),WTFACT(4)
6640 INPUT#3,OLDLETH
6650 FOR JME=1 TO 11
6660 INPUT#3,TIMCOOL(JME),TEMPOBJ(JME),FRANUT(1,JME),FRANUT(2,JME),FRANUT(3,JME
),FRANUT(4,JME),OBJFN(JME)
6670 NEXT JME
6680 INPUT#3,YCURMAX,YCURMIN
6690 FOR LME=1 TO 401 STEP 20
6700 INPUT#3,XXCURVE(LME),YYCURVE(LME)
6710 NEXT LME
6720 CLOSE 3
6730 '
6740 '
6750 'Read temp-time info from COMM2
6760 OPEN "c:comm2" FOR INPUT AS 7
6770 FOR ITIME =1 TO NBA
6780 INPUT#7,TIME(ITIME),TEMP(ITIME)
6790 NEXT ITIME
6800 CLOSE 7
6810 '
6820 'Calculating lethality with COOLOPT temp values
6830 DTIME=TOTIME/(NBA-1)/60!
6840 FOR KME=1 TO (NBA-1)
6850 TAVG(KME)= (TEMP(KME) + TEMP(KME+1)) / 2!
6860 IF (-121! + TAVG(KME))/ZLETH <-35! THEN RATEL=0! : GOTO 6880
6870 RATEL=1/FLETH * ( 10!^((-121!+TAVG(KME))/ZLETH) )
6880 IF KME=1 THEN XLETH=RATEL*DTIME  ELSE  XLETH=XLETH + RATEL*DTIME
6890 NEXT KME
6900 '
6910 '
6920 'Assigning lethalities to variables
6930 XNEWLETH=XLETH
6940 IF NOPCODE=3 THEN OLDLETH=XLETH
6950 '
6960 '
6970 NOPCODE = 4
6980 ' compare two lethalities
6990 IF ABS(XNEWLETH-USERLETH)<.05 THEN GOTO 7060
7000 IF (XNEWLETH>USERLETH) AND (OLDLETH>USERLETH) THEN DCOOL=.9*DCOOL: COOLOPT
=COOLOPT-DCOOL: OLDLETH=XNEWLETH: GOTO 6190
7010 IF (XNEWLETH<USERLETH) AND (OLDLETH<USERLETH) THEN DCOOL=1.1*DCOOL: COOLOP
T=COOLOPT+DCOOL: OLDLETH=XNEWLETH: GOTO 6190
7020 IF (XNEWLETH>USERLETH) AND (OLDLETH<USERLETH) THEN DCOOL=DCOOL/1.9: COOLOP
T=COOLOPT-DCOOL: OLDLETH=XNEWLETH: GOTO 6190
7030 IF (XNEWLETH<USERLETH) AND (OLDLETH>USERLETH) THEN DCOOL=DCOOL/2.1: COOLOP
T=COOLOPT+DCOOL: OLDLETH=XNEWLETH: GOTO 6190
7040 '
7050 '
7060 'calculating nutrient fraction retained, for each nutrient, at TEMPOPT
7070 DTIME=TOTIME/50!/60!
7080 FOR KME=1 TO 4
7090    FOR NME=1 TO 50
7100       IF (-121!+TAVG(NME))/ZNUT(KME)<-35 THEN RATEN=0!: GOTO 7120
7110       RATEN= 1/FXNUT(KME) * (10!^( (-121!+TAVG(NME))/ZNUT(KME)) )
7120       IF NME=1 THEN XNUTOT=1!* 10!^(-12!*RATEN*DTIME) ELSE XNUTOT=XNUTOT*
```

```
 10!^(-12!*RATEN*DTIME)
7130    NEXT NME
7140    FRAOPT(KME)=XNUTOT/11
7150 NEXT KME
7160 '
7170 '
7180 GOTO 10020   'results display section
7190 '
7200 '
7210 'Ask if want to go to MAIN or back to BALLST
7220 CLS: JRECNO=66: GOSUB 7460
7230 IF JOK<>-1 THEN GOTO 7220
7240 IF (DY$="y") OR (DY$="Y") THEN GOTO  650
7250 '
7260 ' when want to go to MAIN
7270 OPEN "b:commb" FOR OUTPUT AS 7
7280 '
7290 '*** write info to commb ***
7300 CLOSE 7
7310 '
7320 'storing the current record number of COMMC into the first rec no of COMMC

7330 NRECCOUNT=LOC(2): LSET COMMC$=STR$(NRECCOUNT): PUT 2,1
7340 CLOSE 1,2
7350 '
7360 CLS: SCREEN 1: COLOR 1,1
7370 DEF SEG = &HB800
7380 BLOAD "b:sandcl.scr",0
7390 ON ERROR GOTO 7400
7400 IF INKEY$<>"" THEN BEEP
7410 '
7420 CHAIN "MAIN"
7430 '
7440 '
7450 '
7460 'getting and checking  Y/N SUBROUTINE
7470 JOK=0
7480 GET 1,JRECNO
7490 COMMENT1Y$="You did not enter Y or N -    You entered: "
7500 COMMENT2Y$="You did not enter a value"
7510 PRINT COMMENT$;
7520 LINE INPUT "",ANSY$
7530 JY=0
7540 JY = JY+1
7550 DY$=MID$(ANSY$,JY,1)
7560 IF DY$="" THEN GOTO 7630
7570 IF DY$=" " THEN GOTO 7540
7580 IF (DY$="Y") OR (DY$="y") OR (DY$="N") OR (DY$="n") THEN JOK=-1: RETURN
7590 PRINT COMMENT1Y$,ANSY$
7600 GET 1,12: PRINT: PRINT COMMENT$
7610 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 7610
7620 RETURN
7630 PRINT COMMENT2Y$
7640 GET 1,12: PRINT: PRINT COMMENT$
7650 DUMMY$=INKEY$: IF DUMMY$="" THEN GOTO 7650
7660 RETURN
7670 '
```

```
7680 '
7690 '
7700 ' getting and CHECKING FOR  REAL  NUMBERS
7710 JOK=0
7720 GET 1,JRECNO
7730 COMMENT1R$="You did not enter a REAL number  -  You entered: "
7740 COMMENT2R$="you did not enter a value"
7750 PRINT COMMENT$;
7760 LINE INPUT"",ANSR$
7770 JFLAGR=0                          'flag to check that number exists
7780 IFLAGR=0                          'flag to check for decimal points
7790 JR=1
7800 KR=LEN(ANSR$)
7810 IF KR=0 THEN GOTO 7950
7820 WHILE JR<=KR
7830 DR$=MID$(ANSR$,JR,1)
7840 IF DR$=" " THEN GOTO 7920
7850 IF (ASC(DR$)>47) AND (ASC(DR$)<58) THEN JFLAGR=-1: GOTO 7920
7860 IF DR$="." THEN IFLAGR=IFLAGR+1: GOTO 7910
7870 PRINT COMMENT1R$,ANSR$
7880 GET 1,12: PRINT: PRINT COMMENT$
7890 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 7890
7900 RETURN
7910 IF IFLAGR>1 GOTO 7870
7920 JR=JR+1
7930 WEND
7940 IF (DR$<>" ") OR (JFLAGR=-1) THEN JOK=-1: RETURN
7950 PRINT COMMENT2R$
7960 GET 1,12: PRINT: PRINT COMMENT$
7970 DUMMR$=INKEY$: IF DUMMR$="" THEN GOTO 7970
7980 RETURN
7990 '
8000 '
8010 '
8020 'getting and CHECKING FOR INTEGER  NUMBERS
8030 JOK=0
8040 IFLAGI=0
8050 GET 1,JRECNO
8060 COMMENT1I$="You did not enter an INTEGER value  -  You entered: "
8070 COMMENT2I$="you did not enter a value"
8080 PRINT COMMENT$;
8090 LINE INPUT "",ANSI$
8100 NI=LEN(ANSI$)
8110 IF NI=0 THEN GOTO 8240
8120 JI=1
8130 WHILE JI<=NI
8140 DI$=MID$(ANSI$,JI,1)
8150 IF DI$=" " THEN GOTO 8210
8160 IF (ASC(DI$)>47) AND (ASC(DI$)<58) THEN IFLAGI=-1: GOTO 8210
8170 PRINT COMMENT1I$,ANSI$
8180 GET 1,12: PRINT: PRINT COMMENT$
8190 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 8190
8200 RETURN
8210 JI=JI+1
8220 WEND
8230 IF (DI$<>" ") OR (IFLAGI=-1) THEN JOK=-1: RETURN
8240 PRINT COMMENT2I$
```

```
8250 GET 1,12: PRINT: PRINT COMMENT$
8260 DUMMI$=INKEY$: IF DUMMI$="" THEN GOTO 8260
8270 RETURN
8280 '
8290 '
8300 '
8310 '
8320 ' Table Creation subroutine
8330 IF N1STTHRU=1 GOTO 8430
8340 NROW=7:NCOLUMN=1
8350 SCREEN 0,0,1,1
8360 CLS
8370 'printing table heading
8380 GET 1,55:PRINT COMMENT$
8390 GET 1,56:PRINT COMMENT$;
8400 GET 1,57:PRINT COMMENT$
8410 GET 1,59:PRINT COMMENT$
8420 N1STTHRU=1
8430 LOCATE NROW,NCOLUMN
8440 GET 1,JRECNO
8450 BLANK$="    "
8460 NBLANK=INSTR(COMMENT$,BLANK$)
8470 FINAL$=LEFT$(COMMENT$,NBLANK-1)
8480 IF IDIFF=1 THEN DIFF$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(67);:
GOTO 8500
8490 IF NMARKSUB=1 THEN PRINT FINAL$;TAB(53);OLD;TAB(67); ELSE PRINT FINAL$;TAB
(53);NOLD;TAB(67);
8500 LINE INPUT NEWVALUE$
8510 NRTEMP=CSRLIN: NCTEMP=POS(0)
8520 IF LEN(NEWVALUE$)=0 THEN GOTO 8570
8530 IF NMARKSUB=1 THEN GOSUB 8890 ELSE GOSUB 9110
8540 IF (NOK=-1) THEN NROW=CSRLIN: NCOLUMN=POS(0): RETURN
8550 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
8560 GOTO 8480
8570 LOCATE NRTEMP-1,NCTEMP:PRINT "
                                  ":LOCATE NRTEMP-1,NCTEMP
8580 IF IDIFF=1 THEN NEWVALUE$=STR$(OLD): PRINT FINAL$;TAB(53);VAL(DIFF$);TAB(6
7);VAL(NEWVALUE$):GOTO 8600
8590 IF NMARKSUB=1 THEN NEWVALUE$=STR$(OLD):PRINT FINAL$;TAB(53);OLD;TAB(67);VA
L(NEWVALUE$) ELSE NEWVALUE$=STR$(NOLD):PRINT FINAL$;TAB(53);NOLD;TAB(67);VAL(NE
WVALUE$)
8600 NROW=CSRLIN: NCOLUMN=POS(0)
8610 RETURN
8620 '
8630 '
8640 '
8650 '
8660 ' Erase Lines in table and show error page subroutine
8670 SCREEN ,,2,2
8680 CLS
8690 GET 1,JRECNO: PRINT COMMENT$;
8700 GET 1,24: PRINT COMMENT$;
8710 GET 1,JRECNO2: PRINT COMMENT$;
8720 IF NMARKSET<>3 THEN GOTO 8740
8730 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2,XNEW3:GOTO 8770 ELSE PRINT NEW1,NEW2,
NEW3:GOTO 8770
```

```
8740 IF NMARKSET<>2 THEN GOTO 8760
8750 IF NMARKSUB=1 THEN PRINT XNEW1,XNEW2:GOTO 8770 ELSE PRINT NEW1,NEW2: GOTO
8770
8760 IF NMARKSUB=1 THEN PRINT XNEW1 ELSE PRINT NEW1
8770 GET 1,12
8780 PRINT: PRINT COMMENT$
8790 DUMMEL$=INKEY$: IF DUMMEL$="" THEN GOTO 8790
8800 SCREEN ,,1,1
8810 LOCATE NROW,NCOLUMN
8820 LOCATE NROW-1,NCOLUMN:PRINT "
                                  ": LOCATE NROW-1,NCOLUMN
8830 IF NMARKSET=1 THEN GOTO 8870
8840 LOCATE NROW-2,NCOLUMN:PRINT "
                                  ":LOCATE NROW-2,NCOLUMN
8850 IF NMARKSET=2 THEN GOTO 8870
8860 LOCATE NROW-3,NCOLUMN:PRINT "
                                  ": LOCATE NROW-3,NCOLUMN
8870 NROW=CSRLIN: NCOLUMN=POS(0)
8880 RETURN
8890 '
8900 '
8910 '
8920 '
8930 'checking Real2 numbers (in table)
8940 NOK=0
8950 JFLAGR2=0
8960 IFLAGR2=0
8970 JR2=1
8980 NR2=LEN(NEWVALUE$)
8990 WHILE JR2=<NR2
9000 DR2$=MID$(NEWVALUE$,JR2,1)
9010 IF DR2$=" " THEN GOTO 9060
9020 IF DR2$="." THEN IFLAGR2=IFLAGR2+1: GOTO 9060
9030 IF (ASC(DR2$)>47) AND (ASC(DR2$)<58) THEN JFLAGR2=-1: GOTO 9060
9040 NOK=0
9050 RETURN
9060 IF IFLAGR2>1 THEN NOK=0: RETURN
9070 JR2=JR2+1
9080 WEND
9090 IF JFLAGR2=0 THEN NOK=0 ELSE NOK=-1
9100 RETURN
9110 '
9120 '
9130 '
9140 '
9150 ' checking Integer2 numbers (in table)
9160 NOK=0
9170 IFLAGI2=0
9180 NI2=LEN(NEWVALUE$)
9190 JI2=1
9200 WHILE JI2<=NI2
9210 DI2$=MID$(NEWVALUE$,JI2,1)
9220 IF DI2$="" THEN GOTO 9250
9230 IF (ASC(DI2$)>47) AND (ASC(DI2$)<58) THEN IFLAGI2=-1: GOTO 9250
9240 NOK=0: RETURN.
9250 JI2=JI2+1
9260 WEND
```

```
9270 IF IFLAGI2=0 THEN NOK=0 ELSE NOK=-1
9280 RETURN
9290 '
9300 '
9310 '
9320 '
9330 ' the "goto MAIN preparation" subroutine
9340 OPEN "b:commb" FOR OUTPUT AS 7
9350 '
9360 ' ** write info into b:commb **
9370 '
9380 CLOSE 7
9390 '
9400 CLOSE 1,2
9410 '
9420 CLS: SCREEN 1
9430 COLOR 4,1
9440 DEF SEG = &HB800
9450 BLOAD "b:sandcl.scr",0
9460 ON ERROR GOTO 9470
9470 IF INKEY$<>"" THEN BEEP
9480 RETURN
9490 '
9500 '
9510 '
9520 '
9530 '
9540 '
9550 'Getting the FXNUT value subroutine
9560 CLS: JRECNO=KREC: GOSUB 7700
9570 IF JOK<>-1 THEN GOTO 9560 ELSE FXNUT(KME)=VAL(ANSR$)
9580 IF (FXNUT(KME)>0!) AND (FXNUT(KME)<=2000!) THEN RETURN
9590 IF FXNUT(KME)>2000! THEN GOTO 9630
9600 GET 1,34: PRINT COMMENT$
9610 GET 1,12: PRINT: PRINT COMMENT$
9620 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 9620 ELSE GOTO 9560
9630 CLS: GET 1,50: PRINT COMMENT$
9640 GET 1,24: PRINT COMMENT$;FXNUT(KME)
9650 JRECNO=36: GOSUB 7460
9660 IF JOK<>-1 GOTO 9630
9670 IF (DY$="n") OR (DY$="N") THEN RETURN ELSE GOTO 9560
9680 '
9690 '
9700 '
9710 '
9720 'Getting the ZNUT value subroutine
9730 CLS: JRECNO=MREC: GOSUB 7700
9740 IF JOK<>-1 THEN GOTO 9730 ELSE ZNUT(KME)=VAL(ANSR$)
9750 IF (ZNUT(KME)>0!) AND (ZNUT(KME)<=50!) THEN RETURN
9760 IF ZNUT(KME)>50! THEN GOTO 9800
9770 GET 1,38: PRINT COMMENT$
9780 GET 1,12: PRINT: PRINT COMMENT$
9790 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 9790 ELSE GOTO 9730
9800 CLS: GET 1,60: PRINT COMMENT$
9810 GET 1,24: PRINT COMMENT$;ZNUT(KME)
9820 JRECNO=36: GOSUB 7460
9830 IF JOK<>-1 GOTO 9800
```

```
9840 IF (DY$="n") OR (DY$="N") THEN RETURN ELSE GOTO 9730
9850 '
9860 '
9870 '
9880 '
9890 ' Getting the weighting factor subroutine
9900 CLS: JRECNO=NREC: GOSUB 7700
9910 IF JOK<>-1 THEN GOTO 9900 ELSE WTFACT(KME)=VAL(ANSR$)
9920 IF (WTFACT(KME)=>0!) AND (WTFACT(KME)<=1!) THEN RETURN
9930 CLS: GET 1,65: PRINT COMMENT$
9940 PRINT:GET 1,24: PRINT COMMENT$;WTFACT(KME)
9950 PRINT:GET 1,12: PRINT COMMENT$
9960 DUMMMA$=INKEY$: IF DUMMMA$="" THEN GOTO 9960 ELSE GOTO 9900
9970 '
9980 '
9990 '
10000 '
10010 '
10020 '*************************************************************************
10030 '
10040 '
10050 '                RESULTS DISPLAY SECTION
10060 '
10070 '
10080 NPOINTS = 11
10090 IF IERRFLAG=0 THEN OBJOPT=YCURMAX
10100 '
10110 ' initializing the values
10120 FOR I=1 TO NPOINTS:XX(I)=TEMPOBJ(I):NEXT
10130 FOR I=1 TO NPOINTS:YY(I)=OBJFN(I):NEXT
10140 '
10150 '
10160 ' this is the results display
10170 CLS:SCREEN 1:COLOR 9,1,1:KEY OFF
10180 LOCATE 2,1:GET 1,150:GOSUB 11840:PRINT COMMT40$;
10190 LOCATE 5,1:GET 1,151:GOSUB 11840:PRINT COMMT40$;
10200 LOCATE 7,1:GET 1,152:GOSUB 11840:PRINT COMMT40$;
10210 LOCATE 9,1:GET 1,190:GOSUB 11840:PRINT COMMT40$;
10220 LOCATE 11,1:GET 1,191:GOSUB 11840:PRINT COMMT40$;
10230 LOCATE 13,1:GET 1,153:GOSUB 11840:PRINT COMMT40$;
10240 LOCATE 23,1:GET 1,154:GOSUB 11840:PRINT COMMT40$;
10250 GOSUB 11710
10260 IF(VAL(ANSWERI$)>=1) AND (VAL(ANSWERI$)<=5) THEN ICHOICE=VAL(ANSWERI$):GO
TO 10280
10270 LOCATE 24,1:BEEP:GET 1,156:GOSUB 11840:PRINT COMMT40$;:GOTO 10250
10280 IF ICHOICE=1 GOTO 10630 'objective fnc graph
10290 IF ICHOICE=2 GOTO 10750 'objective fnc table
10300 IF ICHOICE=3 GOTO 11170 'nutrient retention table
10310 IF ICHOICE=4 GOTO 10330 'process parameter table
10320 IF ICHOICE=5 GOTO 11630 'exit
10330 '
10340 ' Process parameter table
10350 SCREEN 1:COLOR 9,1,1:CLS:NROW=3
10360 LOCATE 1,1:GET 1,157:GOSUB 11830:PRINT COMMT40$;
10370 GET 1,159:GOSUB 11830:VAR$=COMMT40$:VAR!=TINIT!:GOSUB 11660
10380 GET 1,161:GOSUB 11830:VAR$=COMMT40$:VAR!=TWATER!:GOSUB 11660
10390 GET 1,162:GOSUB 11830:VAR$=COMMT40$:VAR!=FH!:GOSUB 11660
```

```
10400 GET 1,163:GOSUB 11830:VAR$=COMMT40$:VAR!=FC!:GOSUB 11660
10410 GET 1,164:GOSUB 11830:VAR$=COMMT40$:VAR!=JH!:GOSUB 11660
10420 GET 1,165:GOSUB 11830:VAR$=COMMT40$:VAR!=JC!:GOSUB 11660
10430 GET 1,166:GOSUB 11830:VAR$=COMMT40$:VAR!=TOTIME!:GOSUB 11660
10440 GET 1,171:GOSUB 11830:VAR$=COMMT40$:VAR!=FLETH!:GOSUB 11660
10450 GET 1,172:GOSUB 11830:VAR$=COMMT40$:VAR!=ZLETH!:GOSUB 11660
10460 GET 1,169:GOSUB 11830:VAR$=COMMT40$:VAR!=FXNUT(1):GOSUB 11660
10470 GET 1,201:GOSUB 11830:VAR$=COMMT40$:VAR!=FXNUT(2):GOSUB 11660
10480 GET 1,202:GOSUB 11830:VAR$=COMMT40$:VAR!=FXNUT(3):GOSUB 11660
10490 GET 1,203:GOSUB 11830:VAR$=COMMT40$:VAR!=FXNUT(4):GOSUB 11660
10500 GET 1,170:GOSUB 11830:VAR$=COMMT40$:VAR!=ZNUT(1):GOSUB 11660
10510 GET 1,204:GOSUB 11830:VAR$=COMMT40$:VAR!=ZNUT(2):GOSUB 11660
10520 GET 1,205:GOSUB 11830:VAR$=COMMT40$:VAR!=ZNUT(3):GOSUB 11660
10530 GET 1,206:GOSUB 11830:VAR$=COMMT40$:VAR!=ZNUT(4):GOSUB 11660
10540 GET 1,185:GOSUB 11830:VAR$=COMMT40$:VAR!=WTFACT(1):GOSUB 11660
10550 GET 1,186:GOSUB 11830:VAR$=COMMT40$:VAR!=WTFACT(2):GOSUB 11660
10560 GET 1,187:GOSUB 11830:VAR$=COMMT40$:VAR!=WTFACT(3):GOSUB 11660
10570 GET 1,188:GOSUB 11830:VAR$=COMMT40$:VAR!=WTFACT(4):GOSUB 11660
10580 GET 1,173:LOCATE 24,1:GOSUB 11830:PRINT COMMT40$;
10590 GET 1,174:LOCATE 25,1:GOSUB 11830:PRINT COMMT40$;:LOCATE 25,13:PRINT CHR$
(24);
10600 GOSUB 11870
10610 IF IFLAG=-1 THEN GET 1,181:GOSUB 11830:BEEP:LOCATE 25,1:PRINT COMMT40$;:G
OTO 10600
10620 GOTO 10160
10630 '
10640 '
10650 ' preparing objective function temperature graph
10660 N=11
10670 FOR I=1 TO N
10680 XX(I)=TEMPOBJ(I):YY(I)=OBJFN(I):NEXT
10690 ' setting the x-axis, y-axis minimums and maximums
10700 '
10710 XXMAX=130!
10720 XXMIN=110!
10730 YYLVAL=0!
10740 GOSUB 11930
10750 '
10760 '
10770 ' table for objective function vs temp
10780 CLS:SCREEN 1:COLOR 9,1,1
10790 LOCATE 2,1:GET 1,175:GOSUB 11830:PRINT COMMT40$;
10800 LOCATE 4,1:GET 1,176:GOSUB 11830:PRINT COMMT40$;:LOCATE 5,1:GET 1,189:GOS
UB 11830:PRINT COMMT40$;
10810 N=NPOINTS:INC=6:NVAR=1: J=1
10820 FOR I=NVAR TO N
10830 INC=INC+1
10840 LOCATE INC,1:PRINT TEMPOBJ(I);:LOCATE INC,13:PRINT TIMCOOL(I);:LOCATE INC
,28:PRINT USING "#.####^^^^";YYCURVE(J):J=J+40
10850 NEXT
10860 LOCATE 20,1: PRINT TEMPOPT;: LOCATE 20,13: PRINT COOLOPT;: LOCATE 20,28:
PRINT USING "#.####^^^^";OBJOPT
10870 LOCATE 22,1:GET 1,173:GOSUB 11830:PRINT COMMT40$;
10880 LOCATE 23,1:GET 1,178:GOSUB 11830:PRINT COMMT40$;
10890 J=0
10900 '
10910 ' error checking subroutine for X/Y/P  +  printing table to printer
```

```
10920 ANSWERY$=INPUT$(1)
10930 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 10950
10940 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 10950
10950 IF IFLAG=-1 THEN BEEP:LOCATE 24,1: GET 1,179:GOSUB 11830:PRINT COMMT40$;:
GOTO 10920
10960 IF IFLAG=2 THEN CLS:GOTO 11000
10970 IF IFLAG=0 THEN 10160
10980 '
10990 '
11000 ' printing a table for objective function results
11010 OPEN "lpt1:" AS #10
11020 WIDTH #10,100:CLS:LOCATE 12,1:GET 1,200:GOSUB 11830:PRINT COMMT40$;
11030 GET 1,197:PRINT #10,COMMENT$:PRINT #10,:PRINT #10,
11040 GET 1,198:PRINT #10,COMMENT$:GET 1,199:PRINT #10,COMMENT$:PRINT #10,
11050 J=1
11060 FOR I=1 TO N
11070 PRINT #10,USING " ######.###                ";TEMPOBJ(I);TIMCOOL(I);
11080 PRINT #10,USING "   #.####^^^^";YYCURVE(J)
11090 J=J+40
11100 NEXT I
11110 J=0
11120 PRINT #10,USING " ######.###                ";TEMPOPT;COOLOPT;
11130 PRINT #10,USING "   #.####^^^^";OBJOPT
11140 CLOSE 10:GOTO 10160
11150 '
11160 '
11170 'screen for nutrient retention fraction
11180 SCREEN 2:SCREEN 0:COLOR 11,1,1:CLS
11190 LOCATE 3,1:GET 1,193:GOSUB 11830:PRINT COMMENT$;
11200 LOCATE 5,1:GET 1,194:GOSUB 11830:PRINT COMMENT$;
11210 LOCATE 7,1:GET 1,195:GOSUB 11830:PRINT COMMENT$;
11220 LOCATE 8,1:GET 1,196:GOSUB 11830:PRINT COMMENT$;
11230 N=NPOINTS:INC=8:NVAR=1
11240'FOR I=NVAR TO N
11250'INC=INC+1
11260'LOCATE INC,1:PRINT TEMPOBJ(I);:LOCATE INC,13:PRINT TIMCOOL(I);:LOCATE INC
,28
11270'PRINT USING "#.##^^^^";FRANUT(1,I); :LOCATE INC,43:PRINT USING "#.##^^^^"
; FRANUT(2,I);:LOCATE INC,58:PRINT USING "#.##^^^^"; FRANUT(3,I);:LOCATE INC,73
:PRINT USING "#.##^^^^"; FRANUT(4,I);
11280'NEXT
11290'COLOR 12,1,1
11300 LOCATE 14,1:PRINT TEMPOPT;:LOCATE 14,13:PRINT COOLOPT;:LOCATE 14,28
11310 PRINT USING "#.##^^^^";FRAOPT(1); :LOCATE 14,43:PRINT USING "#.##^^^^"; F
RAOPT(2);:LOCATE 14,58:PRINT USING "#.##^^^^"; FRAOPT(3);:LOCATE 14,73:PRINT US
ING "#.##^^^^"; FRAOPT(4);
11320 COLOR 11,1,1
11330 GET 1,173:LOCATE 21,20:GOSUB 11830:PRINT COMMT40$;
11340 GET 1,178:LOCATE 22,20:GOSUB 11830::PRINT COMMT40$;
11350 '
11360 ' error checking subroutine for X/Y/P  +  printing table to printer
11370 ANSWERY$=INPUT$(1)
11380 IF(ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:GOTO 11400
11390 IF(ANSWERY$="P") OR (ANSWERY$="p") THEN IFLAG=2:ELSE IFLAG=-1:GOTO 11400
11400 IF IFLAG=-1 THEN BEEP: GET 1,179:LOCATE 24,20:GOSUB 11830:PRINT COMMT40$;
:GOTO 11370
11410 IF IFLAG=2 THEN CLS:GOTO 11450
```

```
11420 IF IFLAG=0 THEN 10160
11430 '
11440 '
11450 ' printing a screen for nutrient fraction results
11460 OPEN "lpt1:" AS #10
11470 WIDTH #10,100:GET 1,45:LOCATE 12,1:PRINT COMMENT$
11480 GET 1,197:PRINT #10,COMMENT$:PRINT #10,:PRINT #10,
11490 GET 1,195:PRINT #10,COMMENT$:PRINT #10,
11500 GET 1,196:PRINT #10,COMMENT$;:PRINT #10,:PRINT #10,
11510'FOR I=1 TO N
11520'PRINT #10,USING "####.##  ";TEMPOBJ(I);
11530'PRINT #10,USING "  #######.##  ";TIMCOOL(I);
11540'PRINT #10,USING "    #.##^^^^  ";FRANUT(1,I);FRANUT(2,I);FRANUT(3,I);FRAN
UT(4,I)
11550'NEXT I
11560 PRINT #10,USING "####.##  ";TEMPOPT;
11570 PRINT #10,USING "  #######.##  ";COOLOPT;
11580 PRINT #10,USING "    #.##^^^^  ";FRAOPT(1);FRAOPT(2);FRAOPT(3);FRAOPT(4)
11590 CLOSE 10:GOTO 10160
11600 '
11610 '
11620 ' exit from this subprogram
11630 CLS: SCREEN 2: SCREEN 0: COLOR 7,1,1: GOTO 7210
11640 '
11650 '
11660 'subroutine that prints parameters and comments in the process parameter
table
11670 LOCATE NROW,1:PRINT VAR$:LOCATE NROW,31:PRINT VAR
11680 NROW=NROW+1
11690 RETURN
11700 '
11710 '
11720 ' error checking subroutine for integers
11730 ANSWERI$=INPUT$(1)
11740 IF ASC(ANSWERI$)>=49 AND ASC(ANSWERI$)<=58 THEN RETURN
11750 LOCATE 24,1:BEEP:GET 1,155:GOSUB 11840:PRINT COMMT40$;:GOTO 11730
11760 RETURN
11770 '
11780 '
11790 ' put a blank line subroutine
11800 LOCATE NROW,1:PRINT"                                                ";:RETURN
11810 '
11820 '
11830 ' reduce record length to 39 characters
11840 COMMT40$=LEFT$(COMMENT$,39):RETURN
11850 '
11860 '
11870 'error checking subroutine for X
11880 ANSWERY$=INPUT$(1)
11890 IF (ANSWERY$="X") OR (ANSWERY$="x") THEN IFLAG=0:RETURN
11900 IFLAG=-1:RETURN
11910 '
11920 '
11930 ' plotting graph
11940 ' draws titles
11950 SCREEN 2
11960 CLS:LOCATE 1,1
```

```
11970 GET 1,182:GOSUB 11830:PRINT COMMT40$;
11980 LOCATE 2,1:GET 1,183::PRINT COMMENT$;:LOCATE 21,12
11990 GET 1,184:GOSUB 11830:PRINT COMMT40$;
12000 '
12010 ' print x-axis number
12020 LINESET=0
12030 DXX=((XXMAX-XXMIN)/(11-1))
12040 NXXPIX=-(350/(11-1))
12050 FOR I=1 TO 11
12060 NXXPIX=NXXPIX+(350/(11-1))
12070 XX=XXMIN+(INT((I-1)*((XXMAX-XXMIN)/(11-1))))
12080 LINESET=LINESET+1
12090 IF LINESET=3 THEN LINESET=1
12100 IF LINESET=1 THEN NROW=19:LINE(83+NXXPIX,140)-(83+NXXPIX,145),1:NCOLUM=IN
T(9+(NXXPIX/7.96)) ELSE NROW=20:LINE(83+NXXPIX,140)-(83+NXXPIX,155),1:NCOLUM=IN
T(10+(NXXPIX/7.96))
12110 LOCATE NROW,NCOLUM:PRINT XX
12120 NEXT
12130 '
12140 ' printing y-axis numbers
12150 TEMPMAX=YY(1)
12160 TEMPMIN=YY(1)
12170 FOR I=2 TO N
12180 IF YY(I)<TEMPMIN THEN TEMPMIN=YY(I)
12190 IF YY(I)>TEMPMAX THEN TEMPMAX=YY(I)
12200 NEXT
12210 YYMAX=TEMPMAX
12220 YYMIN=TEMPMIN
12230 'IF (YYMAX>YYHVAL) THEN 26105 ELSE YYMAX=YYHVAL
12240 IF (YYMIN<YYLVAL) THEN 12250 ELSE YYMIN=YYLVAL
12250 IF (YYMAX<YCURMAX) THEN YYMAX=YCURMAX
12260 IF (YYMIN>YCURMIN) THEN YYMIN=YCURMIN
12270 NYYSET=6
12280 YYSPACE=(YYMAX-YYMIN)/(NYYSET-1)
12290 YYAXNUMBER=YYMIN
12300 NYYPIX=-(115/(NYYSET-1))
12310 FOR I=1 TO NYYSET
12320 NYYPIX=NYYPIX+(115/(NYYSET-1))
12330 NROW=INT(18-(NYYPIX/7.96))
12340 LOCATE NROW,1
12350 PRINT USING "#.##^^^^";YYAXNUMBER
12360 YYAXNUMBER=YYAXNUMBER+YYSPACE
12370 LINE(78,140-NYYPIX)-(83,140-NYYPIX),1
12380 NEXT
12390 '
12400 ' draw axis lines
12410 LINE(83,140)-(433,140),1
12420 LINE(83,140)-(83,25),1
12430 '
12440 '
12450 ' drawing the symbols
12460 FOR I=1 TO N
12470 XX(I)=83+(((XX(I)-XXMIN)/(XXMAX-XXMIN))*350)
12480 YY(I)=140-(((YY(I)-YYMIN)/(YYMAX-YYMIN))*115)
12490 XPOSITION=XX(I):YPOSITION=YY(I)
12500 ON I GOSUB 12820,12850,12890,12920,12950,12990,13040,13080,13110,13150,13
230
```

```
12510 NEXT
12520 '
12530 'drawing the index table
12540 XPOSITION=500!:YPOSITION=19
12550 FOR I=1 TO N
12560 ON I GOSUB 12820,12850,12890,12920,12950,12990,13040,13080,13110,13150,13
230
12570 YPOSITION=YPOSITION+16
12580 NEXT
12590 NROW=1:LINE(480,1)-(639,185),1,B
12600 FOR I=1 TO N
12610 NROW=NROW+2:LOCATE NROW,70:PRINT TIMCOOL(I);:NEXT
12620 '
12630 ' plotting the curve fit
12640 IF IERRFLAG=1 GOTO 12750
12650 IPOINTS=0
12660 FOR I=1 TO 401 STEP 20
12670 IPOINTS=IPOINTS + 1
12680 XCURVE(IPOINTS)=83+(((XXCURVE(I)-XXMIN)/(XXMAX-XXMIN))*350)
12690 YCURVE(IPOINTS)=140-(((YYCURVE(I)-YYMIN)/(YYMAX-YYMIN))*115)
12700 NEXT I
12710 FOR I=2 TO 21
12720 LINE (XCURVE(I-1),YCURVE(I-1))-(XCURVE(I),YCURVE(I)),1
12730 NEXT
12740 '
12750 ' getting the user's choice
12760 GET 1,173:LOCATE 22,10:GOSUB 11830:PRINT COMMT40$;
12770 GET 1,174:LOCATE 23,10:GOSUB 11830:PRINT COMMT40$;:LOCATE 23,22:PRINT CHR
$(24);
12780 GOSUB 11870
12790 IF IFLAG=-1 THEN GET 1,181:GOSUB 11830:BEEP:LOCATE 24,10:PRINT COMMT40$;:
GOTO 12780
12800 GOTO 10160
12810 '
12820 ' circle
12830 CIRCLE(XPOSITION,YPOSITION),5,1
12840 RETURN
12850 'point
12860 CIRCLE(XPOSITION,YPOSITION),5,1
12870 PAINT (XPOSITION,YPOSITION),1,1
12880 RETURN
12890 ' box
12900 LINE(XPOSITION-3,YPOSITION-3)-(XPOSITION+3,YPOSITION+3),1,B
12910 RETURN
12920 ' box full
12930 LINE(XPOSITION-3,YPOSITION-3)-(XPOSITION+3,YPOSITION+3),1,BF
12940 RETURN
12950 ' x
12960 LINE(XPOSITION-3,YPOSITION-3)-(XPOSITION+3,YPOSITION+3),1
12970 LINE(XPOSITION-3,YPOSITION+3)-(XPOSITION+3,YPOSITION-3),1
12980 RETURN
12990 'star
13000 LINE(XPOSITION-3,YPOSITION-3)-(XPOSITION+3,YPOSITION+3),1
13010 LINE(XPOSITION-3,YPOSITION+3)-(XPOSITION+3,YPOSITION-3),1
13020 LINE(XPOSITION+4,YPOSITION)-(XPOSITION-4,YPOSITION),1
13030 RETURN
13040 ' plus+
```

```
13050 LINE(XPOSITION-3,YPOSITION)-(XPOSITION+3,YPOSITION),1
13060 LINE(XPOSITION,YPOSITION-3)-(XPOSITION,YPOSITION+3),1
13070 RETURN
13080 'ellipse
13090 CIRCLE(XPOSITION,YPOSITION),4,1,,,3/2
13100 RETURN
13110 'full ellipse
13120 CIRCLE(XPOSITION,YPOSITION),4,1,,,3/2
13130 PAINT (XPOSITION,YPOSITION),1,1
13140 RETURN
13150 'hexagon
13160 LINE(XPOSITION-4,YPOSITION-2)-(XPOSITION+4,YPOSITION-2),1
13170 LINE(XPOSITION-4,YPOSITION+2)-(XPOSITION+4,YPOSITION+2),1
13180 LINE(XPOSITION-4,YPOSITION-2)-(XPOSITION-12,YPOSITION),1
13190 LINE(XPOSITION+4,YPOSITION-2)-(XPOSITION+12,YPOSITION),1
13200 LINE(XPOSITION-4,YPOSITION+2)-(XPOSITION-12,YPOSITION),1
13210 LINE(XPOSITION+4,YPOSITION+2)-(XPOSITION+12,YPOSITION),1
13220 RETURN
13230 'hexagon
13240 LINE(XPOSITION-4,YPOSITION-2)-(XPOSITION+4,YPOSITION-2),1
13250 LINE(XPOSITION-4,YPOSITION+2)-(XPOSITION+4,YPOSITION+2),1
13260 LINE(XPOSITION-4,YPOSITION-2)-(XPOSITION-12,YPOSITION),1
13270 LINE(XPOSITION+4,YPOSITION-2)-(XPOSITION+12,YPOSITION),1
13280 LINE(XPOSITION-4,YPOSITION+2)-(XPOSITION-12,YPOSITION),1
13290 LINE(XPOSITION+4,YPOSITION+2)-(XPOSITION+12,YPOSITION),1
13300 PAINT(XPOSITION,YPOSITION),1,1
13310 RETURN
13320 '
13330 '
13340 '*************************************************************************
```

ALPHAE

```
1       * * * * * M E N U * * * * *
2
3
4
5
6
7
8 Initial Food Temperature (deg C)
9 Cooling Water Temperature (deg C)
10 The Initial and Cooling Water Temp. must be >0 and Water Temp. <90 deg C
11 Initial Temp:   Water Temp:
12 PRESS ANY KEY TO CONTINUE
13 Total Processing Time (heating + cooling) (sec)
14 The total time must be between 1800 and 9000 sec
15 Total Processing Time:
16 Heating Curve fh (1/slope) (>700 sec)
17 Cooling Curve fc (1/slope) (>700 sec)
18 The fh and fc values (1/slope) must be > 700 sec
19 fh:          fc:
20 Heating Curve Lag Factor (must be>=0;>1=lag,<1=jump)
21 Heating Curve Lag Factor (must be>0;>1=lag,<1=jump)
22 You entered Jh (lag factor) =0; for the model you've chosen Jh can't = 0
23 Lag Factor (heating curve):
24 You entered the following value(s):
25 The Heating Curve Lag Factor can't be between 1.856 and 1.876 inclusively
26 Cooling Curve Lag Factor (must>0) (>1=lag,<1=jump)
27 You entered Jc=0; for the model you've chosen Jc can't = 0
28 Lag Factor (cooling curve):
29 The cooling curve lag factor can't be between 1.856 and 1.876 inclusively
```

```
30 Process Lethality (1.0<=lethality<=2.5)
31 The process lethality must be between 1.0 and 2.5
32 Lethality:
33 What is the F-value for the contaminant (at Tref=121 deg C) - in minutes?
34 The F-value cannot equal 0
35 A F-value greater than 20 minutes is unusual
36 Do you want to reenter the value (Y or N and RETURN)?
37 What is the z-value for the contaminant (at Tref=121 deg C) - in deg C?
38 The z-value cannot equal 0
39 A z-value greater than 30 deg C is unusual
40 Model (1-basic,2-basic modified,3-general)
41 The model number must be between 1 and 3
42 Model:
43 The total processing time you supplied was too short for the lethality desi
red
44 The processing time was automatically increased to: (sec)
45                        * * * * P L E A S E  W A I T * * * *
46 What is the F-value for Nutrient #1 ( at Tref=121 deg C) - in minutes?
47 What is the F-value for Nutrient #2 (at Tref=121 deg C) - in minutes?
48 What is the F-value for Nutrient #3 (at Tref=121 deg C) - in minutes?
49 What is the F-value for Nutrient #4 (at Tref=121 deg C) - in minutes?
50 A F-value greater than 2000 minutes is unusual
51 What is the z-value for Nutrient #1 (at Tref=121 deg C) - in deg C?
52 What is the z-value for Nutrient #2 (at Tref=121 deg C) - in deg C?
53 What is the z-value for Nutrient #3 (at Tref=121 deg C) - in deg C?
54 What is the z-value for Nutrient #4 (at Tref=121 deg C) - in deg C?
55       ARGUMENT LISTING FOR SUBROUTINE
56 Enter the new value and press RETURN
57 Hit RETURN alone if you want to keep the old value.
```

```
58
59    ARGUMENT                                        OLD          NEW
60 A z-value greater than 50 deg C is unusual
61 What is the weighting factor for Nutrient #1 (between 0 and 1) ?
62 What is the weighting factor for Nutrient #2 (between 0 and 1) ?
63 What is the weighting factor for Nutrient #3 (between 0 and 1) ?
64 What is the weighting factor for Nutrient #4 (between 0 and 1) ?
65 You did not enter a weighting factor value between 0 and 1
66 Do you want to redo the process calculations (Y or N and RETURN)?
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
```

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

```
144
145
146
147
148
149
150                    MENU
151      1) Objective Function vs Temp Graph
152      2) Objective Function vs Temp Table
153      5) EXIT The Result Display Section
154           Enter 1, 2 ... 5
155 you did not enter an integer, try again
156 You did not enter 1, 2 ...5 - try again
157           Process Parameter Table
158           ----------------------
159 Initial Temperature(deg C)
160 Retort Temperature(deg C)
161 Water Temperature(deg C)
162 Slope of Heating Curve(sec)
163 Slope of Cooling Curve(sec)
164 Heating Curve Lag Factor
165 Cooling Curve Lag Factor
166 Total Processing Time(sec)
167 Time When Cooling Starts(sec)
168 Number of Time Increments
169 F-Value of Nutrient #1(min)
170 z-Value of Nutrient #1(deg C)
171 F-Value of Microorganism(min)
```

```
172 z-Value of Microorganism(min)
173       Press "X" to exit to menu or
174       Press  -PrtSc to print graph
175                  RESULTS
176 Retort        Cooling        Objective
177       Press Y to continue
178       Press P to print specified data
179          You did not enter X or P
180   There is no more data, enter P or X
181         YOU DID NOT ENTER "X"
182 Obj.
183 Function      Obj. Function vs Retort Temp.                    cooltime
(s)
184         Retort Temp (deg C)
185 Weighting Factor #1
186 Weighting Factor #2
187 Weighting Factor #3
188 Weighting Factor #4
189 Temp(C)    starts at(s)    Func. Value
190     3) Optimal Nutrient Retention
191     4) Process Parameter Table
192 Weighting Factor #4
193                                OPTIMAL  NUTRIENT
194                                RETENTION FRACTION
195 Retort      Cooling       Nutrient       Nutrient       Nutrient       Nut
rient
196 Temp(C)   starts at(s)       #1             #2             #3
#4
197                           Results
198    Retort                 Cooling                    Objective
199   Temp(C)               starts at (s)                Function value
200     * * * P L E A S E  W A I T * * *
```

201 F-Value of Nutrient #2(min)

202 F-Value of Nutrient #3(min)

203 F-Value of Nutrient #4(min)

204 z-Value of Nutrient #2(deg C)

205 z-Value of Nutrient #3(deg C)

206 z-Value of Nutrient #4(deg C)

APPENDIX 3.8

Program Listing of SUB1 (BALLST)

```
1000 '
1005 '
1010 '
1015 ' ************************************************************************
1020 '                              BALLST
1025 ' SUBROUTINE BALLST PRODUCES VECTORS OF TIME AND TEMPERATURE VALUES
1030 ' ACCORDING TO BALL'S MODEL (BASIC, MODIFIED AND GENERAL - DEPENDING
1035 ' ON WHETHER THE VALUE OF THE PARAMETER 'MODEL' IS 1,2 OR 3).
1040 ' IT ALSO OUTPUTS VALUES OF G! AND M! .
1045 ' THE CONTAINER OF FOOD IS INITIALLY AT UNIFORM TEMPERATURE TINIT!,
1050 ' SUBJECTED TO TEMPERATURE TRETRT! IN A RETORT FOR THE FIRST COOLTI! SECS,

1055 ' THEN DUMPED IN WATER AT TEMPERATURE TWATER! AND LEFT THERE FOR THE REST
1060 ' OF THE TOTAL MODELLING TIME, TOTIME!.
1065 ' ************************************************************************
1070 ' PARAMETERS:
1075 ' tinit! ="INITIAL TEMPERATURE - DEG C - INPUT
1080 ' tretrt! ="RETORT TEMPERATURE - DEG C - INPUT
1085 ' twater! ="WATER TEMPERATURE - DEG C - INPUT
1090 ' fh! ="SLOPE OF HEATING CURVE - SEC - INPUT
1095 ' jh! ="HEATING CURVE LAG FACTOR - INPUT
1100 ' fc! ="SLOPE OF COOLING CURVE - SEC - INPUT
1105 ' jc! ="COOLING CURVE LAG FACTOR - INPUT
1110 ' totime! ="TOTAL MODELLING TIME - SEC - INPUT
1115 ' coolti! ="TIME AT WHICH COOLING STARTS - INPUT
1120 ' time! ="ARRAY OF TIME VALUES - SEC - OUTPUT
1125 ' temp! ="ARRAY OF TEMPERATURE VALUES - DEG C - OUTPUT
1130 ' g! ="tretrt!-tcmaxba - DEG C - OUTPUT
1135 ' m! ="tcmaxba-twater! - DEG C - OUTPUT
1140 ' Nba ="NUMBER OF time! AND TEMPERATURE VALUES WANTED - INPUT
1145 ' MODEL ="MODEL NUMBER TO BE USED - INPUT
1150 ' ************************************************************************
1155 ' bcsqba ="COOLING HYPERBOLA CONSTANT - SEC
1160 ' bhsqba ="HEATING HYPERBOLA CONSTANT - SEC
1165 ' dtimeba ="time! INTERVAL FOR EVALUATING TIME AND TEMPERATURE - SEC
1170 ' dummxba ="DUMMY VARIABLE
1175 ' rba ="DUMMY VARIABLE FOR COOLING HYPERBOLA
1180 ' sba ="DUMMY VARIABLE FOR HEATING HYPERBOLA
1185 ' tcmaxba ="MAXIMUM TEMPERATURE ATTAINED AT CONTAINER CENTRE - DEG C
1190 ' tcstarba ="HYPERBOLIC/LOGARITHMIC CURVES SWITCHOVER TEMPERATURE - DEG C
1195 ' tistarba ="HYPERBOLIC/LOGARITHMIC CURVES SWITCHOVER TIME - SEC
1200 ' yjcba ="DUMMY VARIABLE FOR jc!
1205 ' ************************************************************************
1210 ' OTHER VARIABLES, INTEGER:
1215 ' Iba ="DUMMY
1220 ' Jba ="DUMMY
1225 ' Mba ="DUMMY VARIABLE FOR MODEL
1230 ' ncoolba ="NUMBER OF VALUES CALCULATED FOR TOTAL COOLING CURVE
1235 ' ncoolhba ="NUMBER OF VALUES CALCULATED FOR HYPERBOLIC COOLING CURVE
1240 ' ncoollba ="NUMBER OF VALUES CALCULATED FOR LOGARITHMIC COOLING CURVE
1245 ' nheatba ="NUMBER OF VALUES CALCULATED FOR TOTAL HEATING CURVE
1250 ' nheathba ="NUMBER OF VALUES CALCULATED FOR HYPERBOLIC HEATING CURVE
1255 ' nheatlba ="NUMBER OF VALUES CALCULATED FOR LOGARITHMIC HEATING CURVE
1260 ' ************************************************************************
1265 '
1270 'INITIALIZATION AND DIMENSIONING
1275 DEFINT I-N
```

```
1280 DIM TIME(1001),TEMP(1001)
1285 '
1290 ' GETTING ARGUMENTS THAT WERE STORED IN COMM1 BY THE CALLING PROGRAM
1295 OPEN "c:comm1" FOR INPUT AS 1
1305 INPUT #1,MODEL,TINIT!,TRETRT!,TWATER!,NBA,TOTIME!,COOLTI!,FH!,FC!,JH!,JC!,
nsubmain
1310 close 1
1315 '
1320 ' INPUT PARAMETER CHECKING SECTION AND ERROR MESSAGE WRITING
1325 IF((TINIT!<=TRETRT!) AND (TWATER!<=TRETRT!)) THEN GOTO 1345
1330 PRINT "THERE IS A PROBLEM WITH INPUT DATA TINIT!,TRETRT! OR TWATER! "
1335 PRINT "THE PROGRAM WAS STOPPED"
1340 STOP
1345 IF (FH!>0!) AND (FC!>0!) THEN GOTO 1365
1350 PRINT "THERE IS AN ERROR IN FH! OR FC! INPUT DATA"
1355 PRINT "THE PROGRAM WAS STOPPED"
1360 STOP
1365 IF ((JH!>0!) OR (MODEL=1)) AND ((JC!=>0!) OR (MODEL<>3)) THEN GOTO 1385
1370 PRINT "THERE IS AN ERROR IN JH! OR JC!"
1375 PRINT "THE PROGRAM WAS STOPPED"
1380 STOP
1385 IF(TOTIME!>0!) AND (COOLTI!>0!) AND (TOTIME!>=COOLTI!) THEN GOTO 1405
1390 PRINT "THERE IS AN ERROR IN TOTIME! OR COOLTI!"
1395 PRINT "THE PROGRAM WAS STOPPED"
1400 STOP
1405 IF ((ABS(JH!-1.866)>.01) OR (MODEL=1))   AND   ((ABS(JC!-1.866)>.01) OR (M
ODEL<>3)) THEN GOTO 1425
1410 PRINT "THERE IS AN ERROR IN JH! OR JC!"
1415 PRINT "THE PROGRAM WAS STOPPED"
1420 STOP
1425 IF (NBA>2) THEN GOTO 1445
1430 PRINT "THERE IS AN ERROR IN THE NUMBER OF TIME OR TEMP VALUES GIVEN,NBA"
1435 PRINT "THE PROGRAM WAS STOPPED"
1440 STOP
1445 IF (MODEL>=1) AND (MODEL<=3) THEN GOTO 1470
1450 PRINT "THERE IS AN ERROR IN THE MODEL NUMBER SPECIFIED, MODEL"
1455 PRINT "THE PROGRAM WAS STOPPED"
1460 STOP
1465 '
1470 DEF FNZ(X)=(1!-.453461/X+.0514065/X/X)/(.453461/X/X/X-1!/X/X)
1475 '
1480 ' FINDING tcmaxba, g! AND m!
1485 TCMAXBA=TRETRT!-(TRETRT!-TINIT!)*JH!*10!^(-COOLTI!/FH!)
1490 G!=TRETRT!-TCMAXBA
1495 M!=TCMAXBA-TWATER!
1500 ' CALCULATING time! VALUES
1505 DTIMEBA=TOTIME!/CSNG(NBA-1)
1510 FOR IBA=1 TO NBA
1515 TIME!(IBA)=CSNG((IBA-1)*DTIMEBA)
1520 NEXT
1525 ' DETERMINE NUMBER OF VALUES ON HEATING AND COOLING CURVES
1530 NHEATBA=INT(COOLTI!/DTIMEBA)+1
1535 NCOOLBA=NBA-NHEATBA
1540 'DETERMINE NUMBERS OF VALUES ON HYPERBOLIC & LOGARITHMIC HEATING CURVES
1545 IF MODEL>1  THEN GOTO 1575
1550 ' LOGARITHMIC HEATING ONLY
1555 NHEATLBA=NHEATBA
```

```
1560 NHEATHBA=0
1565 GOTO 1655
1570 'HYPERBOLIC/LOGARITHMIC HEATING
1575 TCSTARBA=TINIT!+.343*(TRETRT!-TINIT!)
1580 IF TCSTARBA<TCMAXBA  THEN GOTO 1600
1585'PRINT " HYPERBOLIC MODEL NOT POSSIBLE FOR HEATING CURVE"
1590'PRINT " LOGARITHMIC MODEL ONLY USED"
1595 GOTO 1555
1600 SBA=.434294482#*LOG(JH!/.657)
1605 TISTARBA=FH!*SBA
1610 NHEATHBA=INT(TISTARBA/DTIMEBA)+1
1615 NHEATLBA=NHEATBA-NHEATHBA
1620 AHBA=(.343*(SBA-.22673))/(.45346-SBA)*(TRETRT!-TINIT!)
1625 BHSQBA=FNZ(SBA)*FH!^2
1630 ' HYPERBOLIC HEATING CURVE CALCULATION
1635 FOR IBA=1 TO NHEATHBA
1640 TEMP!(IBA)=TINIT!+AHBA*(SQR(1!+TIME!(IBA)^2/BHSQBA)-1!)
1645 NEXT
1650 ' LOGARITHMIC HEATING CURVE CALCULATION
1655 IF NHEATLBA<=0  THEN GOTO 1690
1660 DUMMXBA=(TRETRT!-TINIT!)*JH!
1665 JBA=NHEATHBA+1
1670 FOR IBA=JBA TO NHEATBA
1675 TEMP!(IBA)=TRETRT!-DUMMXBA*10!^(-TIME!(IBA)/FH!)
1680 NEXT
1685 ' COOLING CURVE CALCULATIONS
1690 IF NCOOLBA=0  THEN GOTO 1850
1695 'FINDING COOLING CURVE CONSTANTS
1700 YJCBA=1.41
1705 BCSQBA=(.173*FC!)^2
1710 ACBA=.3*(TCMAXBA-TWATER!)
1715 RBA=.434294482#*LOG(YJCBA/.657)
1720 IF MODEL<>3  THEN GOTO 1745
1725 YJCBA=JC!
1730 RBA=.434294482#*LOG(YJCBA/.657)
1735 BCSQBA=FNZ(RBA)*FC!^2
1740 ACBA=(.343*(RBA-.22673))/(.45346-RBA)*(TCMAXBA-TWATER!)
1745 TISTARBA=FC!*RBA
1750 IF TISTARBA<(TOTIME!-COOLTI!)  THEN GOTO 1770
1755 NCOOLLBA=0
1760 KBA=NBA
1765 GOTO 1790
1770 NCOOLHBA=INT((COOLTI!+TISTARBA)/DTIMEBA)+1-NHEATBA
1775 NCOOLLBA=NBA-NHEATBA-NCOOLHBA
1780 IF NCOOLHBA<=0  THEN GOTO 1810
1785 KBA=NHEATBA+NCOOLHBA
1790 JBA=NHEATBA+1
1795 FOR IBA=JBA TO KBA
1800 TEMP!(IBA)=TCMAXBA-ACBA*(SQR(1!+(TIME!(IBA)-COOLTI!)^2/BCSQBA)-1!)
1805 NEXT
1810 IF NCOOLLBA<=0  THEN GOTO 1850
1815 JBA=NHEATBA+1+NCOOLHBA
1820 DUMMXBA=(TCMAXBA-TWATER!)*YJCBA
1825 FOR IBA=JBA TO NBA
1830 TEMP!(IBA)=TWATER!+DUMMXBA*10!^(-1!/FC!*(TIME!(IBA)-COOLTI!))
1835 NEXT
1840 '
```

```
1845 '
1850 'PRINTING THE TIME,TEMP INFO INTO COMM2 FILE
1855 OPEN "c:comm2" FOR OUTPUT AS 2
1860 FOR IBA=1 TO NBA
1865 PRINT#2,TIME!(IBA);TEMP!(IBA)
1870 NEXT IBA
1875 CLOSE 2
1880 '
1885 ' CHAINING BACK TO THE SUBMAIN PROGRAM THAT CALLED BALLST
1890 IF NSUBMAIN=1 THEN CHAIN "maina"
1895 IF NSUBMAIN=2 THEN CHAIN "mainb"
1900 IF NSUBMAIN=3 THEN CHAIN "mainc"
1905 if nsubmain=4 then chain "maind"
1910 if nsubmain=5 then chain "maine"
1915 print "Unreasonable nsubmain value"
1920 print "Can't chain back to a submain"
1925 print "Program SUB1 was terminated"
1930 stop
```

APPENDIX 3.9

Program Listing of SUB2 (CAN1)

```
1000 '
1005 '
1010 '
1015 '
1020 ' ************************************************************************
1025 ' SUBROUTINE CAN1 CALCULATES A MATRIX OF TEMPERATURE VALUES INSIDE A
1030 ' FINITE CYLINDRICAL CONTAINER OF FOOD. THE CONTAINER IS INITIALLY AT A
1035 ' UNIFORM TEMPERATURE AND FROM TIME=0 IS SUBJECTED AT ALL ITS FACES TO
1040 ' A NEW TEMPERATURE. THE HEAT TRANSFER COEFFICIENT IS ASSUMED TO BE
1045 ' INFINITE.  SUBROUTINE CAN1 REQUIRES FUNCTION SUBPROGRAMS XJ0 AND XJ1
1050 '
1055 ' CAUTION !..... CAN1 OPERATES ENTIRELY IN DOUBLE PRECISION, EXCEPT
1060 '
1065 '      FOR ARGUMENTS WHICH ARE INPUT AND OUTPUT IN REAL*4
1070 '
1075 ' ************************************************************************
1080 ' SUBROUTINE PARAMETERS:
1085 ' tbath! ="TEMPERATURE TO WHICH SURFACE IS SUBJECTED - DEG C - INPUT
1090 ' tinit! ="INITIAL TEMPERATURE OF CONTAINER CONTENTS - DEG C - INPUT
1095 ' alpha! ="THERMAL DIFFUSIVITY OF CONTENTS - M**2/SEC - INPUT
1100 ' halfl! ="HALF OF TOTAL LENTH OF FINITE CYLINDER - M - INPUT
1105 ' radius! ="radius OF FINITE CYLINDER - M - INPUT
1110 ' NZ ="# OF z! VALUES AT WHICH TEMPERATURE IS TO BE EVALUATED - INPUT
1115 ' NR ="# OF RADII AT WHICH TEMPERATURE IS TO BE EVALUATED - INPUT
1120 ' NTIME ="# OF time! VALUES AT WHICH TEMP IS TO BE EVALUATED - INPUT
1125 ' z! ="VECTOR OF z! VALUES FOR TEMPERATURE EVALUATION - M - INPUT
1130 ' R! ="VECTOR OF radius! VALUES FOR TEMPERATURE EVALUATION - M - INPUT
1135 ' time! ="VECTOR OF TIME VALUES FOR TEMP EVALUATION - DEG C - INPUT
1155 ' resultc1! ="THREE DIMENSIONAL result ARRAY - DEG C - OUTPUT
1160 ' ************************************************************************
1165 ' INTERNAL PROGRAM VARIABLES USED; ARRAYS
1170 ' ZF1c1,ZF2c1,zl1c1%c1                              ) FUNCTIONS OUTLINED I
N
1175 ' RF1c1,RF2c1,RL2c1                                 )  PROGRAM DESCRIPTION
1180 ' ROOTc1, ROOTSc1                                   )
1185 ' ************************************************************************
1190 ' INTERNAL PROGRAM VARIABLES USED; REAL
1195 ' DUMM1c1,DUMM2c1,SUM1c1,SUM1Xc1,SUM2c1,SUM2Xc1  ) FUNCTIONS OUTLINED IN
1200 ' TERMc1,TERMMc1,TERMNc1                          ) PROGRAM DESCRIPTION
1205 ' ************************************************************************
1210 ' INTERNAL PROGRAM VARIABLES USED; INTEGER
1215 ' Ic1,Jc1,IRc1,ISUM1Xc1,ISUM2Xc1,ITIIMEc1,IZc1   ) FUNCTIONS OUTLINED IN
1220 ' Mc1,Nc1                                         ) PROGRAM DESCRIPTION
1225 ' ************************************************************************
1230 ' INTERNAL PROGRAM VARIABLES USED; LOGICAL
1231 ' IFLAGc1%= UNDERFLOW MEMORY FLAG
1232 ' ************************************************************************
1233 ' DECLARATION SECTION
1234 DEFDBL A-H,O-Z
1235 DEFINT I-N
1236 OPTION BASE 1
1237 DIM ZF1C1(100),ZF2C1(100),ZF3C1(100)   'subroutine CAN1
1238 DIM RF1C1(100),RF2C1(100)
1239 DIM ZL1C1%(100),RL1C1%(100)
1240 DIM ROOTSC1(30),ROOTC1(100),UZC1(100),URC1(100)
1241 DIM Z!(10),R!(10),TIME!(21),RESULTC1!(10,10,21)
1242 DIM A1X0(6),A2X0(7),A3X0(7)   'subroutine XJ0
```

```
1243 DIM B1X1(6),B2X1(7),B3X1(7)    'subroutine XJ1
1250 '
1255 RESTORE 1272
1260 FOR IC1=1 TO 30
1265 READ ROOTSC1(IC1)
1270 NEXT
1272 DATA 2.4048256,5.5200781,8.6537279,11.7915344,14.9309177
1274 DATA 18.0710640,21.2116366,24.3524715,27.4934791,30.6346065
1276 DATA 33.7758202,36.9170984,40.0584258,43.1997917,46.3411884
1278 DATA 49.4826099,52.6240518,55.7655108,58.9069839,62.0484692
1280 DATA 65.1899648,68.3314693,71.4729816,74.6145006,77.7560256
1282 DATA 80.8975559,84.0390908,87.1806298,90.3221726,93.4637188
1284 '
1286 PI=3.141592654#
1288 '
1289 '
1290 'Getting arguments that were stored in COMM1 by calling program
1292 OPEN "c:comm1" FOR INPUT AS 1
1293 INPUT#1,TBATH!,TINIT!,ALPHA!,HALFL!,RADIUS!,NZ,NR,NTIME,TOTIME!,NSUBMAIN
1294 CLOSE 1
1295 '
1296 '
1297 'Calculating Z!,R!,TIME! arrays
1298 IF (1<NZ) AND (1<NR) GOTO 1299 ELSE 1300
1299 IF (NZ<=10) AND (NR<=10) THEN GOTO 1301
1300 PRINT"Problem with NZ or NR or NTIME in CAN!- Program was stopped":STOP
1301 '
1302 DZ=HALFL!/CSNG(NZ-1)
1303 FOR IZC1=1 TO NZ
1304 Z!(IZC1)=CSNG(IZC1-1)*DZ
1305 NEXT IZC1
1306 '
1307 DR=RADIUS!/CSNG(NR-1)
1308 FOR IRC1=1 TO NR
1309 R!(IRC1)=CSNG(IRC1-1)*DR
1310 NEXT IRC1
1311 '
1312 DTIME=TOTIME!/CSNG(NTIME-1)
1313 FOR LC1=1 TO NTIME
1314 TIME!(LC1)=CSNG((LC1-1)*DTIME)
1315 NEXT LC1
1316 '
1318 '
1320 ' PARAMETER CHECKING SECTION
1325 IF ALPHA!>0!  GOTO 1350
1330 PRINT "PROBLEM DETECTED WITH NON-VECTOR INPUT DATA TO SUBROUTINE CAN1:"
1335 PRINT "alpha! ,  halfl! , radius! ,  NR , NZ , NTIME"
1340 PRINT "PROGRAM WAS STOPPED"
1345 STOP
1350 IF (HALFL!>0!) AND (RADIUS!>0!)GOTO 1515
1355 GOTO 1330
1510 '
1515 ' INITIALIZING COUNTERS AND LOGIC ARRAYS
1520 FOR IC1=1 TO 100
1525 ZL1C1%(IC1)=0
1530 RL1C1%(IC1)=0
1535 NEXT
```

```
1540 FOR JC1=1 TO 30
1545 ROOTC1(JC1)=ROOTSC1(JC1)
1550 XJ1 = ROOTC1(JC1)
1555 GOSUB 6000        'subroutine XJ1
1560 RF1C1(JC1)=2!/ROOTC1(JC1)/XJ1
1565 NEXT
1570 ' START OF TIME LOOP
1575 FOR ITIMEC1=1 TO NTIME
1580 IF TIME!(ITIMEC1)>0!  GOTO 1620
1585 FOR IRC1=1 TO NR
1590 URC1(IRC1) =SQR(PI/4!)
1595 NEXT
1600 FOR IZC1=1 TO NZ
1605 UZC1(IZC1) =SQR(PI/4!)
1610 NEXT
1615 GOTO 2005
1620 DUMM1C1 = CDBL(-ALPHA!*TIME!(ITIMEC1))
1622 '
1625 ' Z-LOOP
1630 FOR IZC1=1 TO NZ
1635 NC1=0
1640 SUM1C1 = 0#
1645 IFLAGC1%=0
1650 SUM1XC1=0#
1655 FOR ISUM1XC1=1 TO 2
1660 NC1=NC1+1
1665 IF N<=100THEN GOTO 1690
1670 PRINT "NUMBER OF ROOTS REQUIRED IN Z LOOP GREATER THAN 100"
1675 PRINT "ITIMEC1 was";ITIMEC1,"  NC1 was";NC1
1680 PRINT "PROGRAM WAS STOPPED"
1685 STOP
1690 DUMM2C1 = CDBL(2*NC1-1)
1695 IF (ZL1C1%(NC1)<>0) THEN GOTO 1710
1700 ZF2C1(NC1) = ((-1!)^(NC1+1))/DUMM2C1
1705 ZL1C1%(NC1) = -1
1710 TERMC1=((DUMM2C1*PI/2!/CDBL(HALFL!))^2) * DUMM1C1
1715 IF (TERMC1>-88#) THEN GOTO 1730
1720 TERMC1=-88#
1725 IFLAGC1%=-1
1730 ZF1C1(NC1) = COS(DUMM2C1*PI/2!* CDBL(Z!(IZC1)/HALFL!))
1735 TERMNC1=ZF2C1(NC1) * ZF1C1(NC1) * EXP(TERMC1)
1740 SUM1XC1=SUM1XC1+ABS(TERMNC1)
1745 SUM1C1=SUM1C1+TERMNC1
1750 NEXT ISUM1XC1
1755 IF (ABS(SUM1XC1/SUM1C1)>.0001#) AND (SUM1XC1>.0000000001#) AND (IFLAGC1%=0
) THEN GOTO 1650
1760 IF (ABS(SUM1C1)<.0000000001#) THEN SUM1C1=0#
1765 UZC1(IZC1)=SUM1C1
1770 NEXT IZC1
1772 '
1775 ' R-LOOP
1780 FOR IRC1=1 TO NR
1785 MC1=0
1790 SUM2C1=0#
1795 IFLAGC1%=0
1800 SUM2XC1=0#
1805 FOR ISUM1RC1=1 TO 2
```

```
1810 MC1=MC1+1
1815 IF (MC1<=100) THEN GOTO 1840
1820 PRINT " MORE THAN 100 VALUES REQUIRED IN R LOOP"
1825 PRINT "ITIMEC1 IS";ITIMEC1,"  MC1 is";MC1
1830 PRINT "THE PROGRAM WAS STOPPED"
1835 STOP
1840 IF (MC1<=30) THEN GOTO 1930
1845 IF (RL1C1%(MC1)<>0) THEN GOTO 1930
1850 ROOTC1(MC1) = ROOTC1(MC1) + PI
1855 DROOTC1 = -.0001#
1860 XO=ROOTC1(MC1)
1865 GOSUB 5000        'subroutine XJ0
1870 Y2C1 = XJ0
1875 Y1C1 = Y2C1
1880 ROOTC1(MC1) = ROOTC1(MC1) + DROOTC1
1885 XO=ROOTC1(MC1)
1890 GOSUB 5000        'subroutine XJ0
1895 Y2C1 = XJ0
1900 DROOTC1 = -DROOTC1* Y2C1/(Y2C1-Y1C1)
1905 IF (ABS(DROOTC1)>.0000000001#) THEN GOTO 1875
1910 X1=ROOTC1(MC1)
1915 GOSUB 6000        'subroutine XJ1
1920 RF1C1(MC1) = 2!/ROOTC1(MC1)/XJ1
1925 RL1C1%(MC1) = -1
1930 XO=ROOTC1(MC1)/ CDBL(R!(IRC1)/RADIUS!)
1935 GOSUB 5000        'subroutine XJ0
1940 RF2C1(MC1) = XJ0
1945 TERMC1=((ROOTC1(MC1)/CDBL(RADIUS!))^2) * DUMM1C1
1950 IF (TERMC1>-88#) THEN GOTO 1965
1955 TERMC1=-.88#
1960 IFLAGC1%=-1
1965 TERMMC1=RF1C1(MC1) * RF2C1(MC1) * EXP(TERMC1)
1970 SUM2XC1=SUM2XC1+ABS(TERMMC1)
1975 SUM2C1=SUM2C1+TERMMC1
1980 NEXT ISUM1RC1
1985 IF (ABS(SUM2XC1/SUM2C1)>.0001#) AND (SUM2XC1>.0000000001#) AND (IFLAGC1%=0
) THEN GOTO 1800
1990 IF (ABS(SUM2C1)<.0000000001#) THEN SUM2C1=0#
1995 URC1(IRC1)= SUM2C1
2000 NEXT IRC1
2002 '
2005 FOR IZC1=1 TO NZ
2010 FOR IRC1=1 TO NR
2015 RESULTC1!(IZC1,IRC1,ITIMEC1)=TBATH!- (TBATH!-TINIT!)*4!/CSNG(PI)*CSNG(UZC1
(IZC1)* URC1(IRC1))
2020 NEXT IRC1
2025 NEXT IZC1
2030 NEXT ITIMEC1
2035 '
2040 '
2050 'Printing Time, Position and Temp data into COMM2 file
2055 OPEN "c:comm2" FOR OUTPUT AS 2
2060 FOR LC1=1 TO NTIME
2065 FOR IC1=1 TO NZ
2070 FOR JC1=1 TO NR
2075 PRINT#2,Z!(IC1),R!(JC1),TIME!(LC1),RESULTC1!(IC1,JC1,LC1)
2080 NEXT JC1,IC1,LC1
```

```
2085 CLOSE 2
2090 '
2095 '
2100 'Chaining back to the submain program that called CAN1
2105 IF NSUBMAIN=1 THEN CHAIN "MAINA"
2110 IF NSUBMAIN=2 THEN CHAIN "MAINB"
2115 IF NSUBMAIN=3 THEN CHAIN "MAINC"
2120 IF NSUBMAIN=4 THEN CHAIN "MAIND"
2125 IF NSUBMAIN=5 THEN CHAIN "MAINE"
2130 PRINT "unreasonable nsubmain value in CAN1- can't chain back to subMAIN"
2135 PRINT "program was stopped":STOP
5000 '
5010 '
5020 '
5030 'subroutine XJ0
5040 IF ICOUNTX0=1 GOTO 5200
5050 RESTORE 5090
5060 FOR IX0 = 1 TO 6
5070 READ A1X0(IX0)
5080 NEXT
5090 DATA -2.2499997,1.2656208,-.3163866,.0444479,-.0039444,.0002100
5100 FOR IX0 = 1 TO 7
5110 READ A2X0(IX0)
5120 NEXT
5130 DATA .79788456,-.00000077,-.00552740,-.00009512,.00137237,-.00072805
5140 DATA .00014476
5150 FOR IX0 = 1 TO 7
5160 READ A3X0(IX0)
5170 NEXT
5180 DATA -.78539816,-.04166397,-.00003954,.00262573,-.00054125,-.00029333
5190 DATA .00013558
5200 ' Testing for negative or zero x0
5210 XJ0 = 1#
5220 ON SGN(X0)+2 GOTO 5230,5270,5290
5230 CLS
5240 PRINT "Message from subroutine xj0:"
5250 PRINT "argument was -ve"
5260 STOP
5270 ICOUNTX0=1
5280 RETURN
5290 ' Choosing method of calculation
5300 IF X0 > 3! GOTO 5400
5310 ' Method of calculating xj0 for x0 < or = 3.
5320 XX0 = X0*X0/9#
5330 YX0 = 1#
5340 FOR IX0 = 1 TO 6
5350 YX0 = YX0 * XX0
5360 XJ0 = XJ0 + A1X0(IX0)*YX0
5370 NEXT
5380 ICOUNTX0=1
5390 RETURN
5400 ' Method of calculating xj0 for x0 > 3.
5410 XX0 = 3#/X0
5420 YX0 = 1#
5430 FOX0 = A2X0(1)
5440 THETA0X0 = X0 + A3X0(1)
5450 FOR JX0 = 2 TO 7
```

```
5460 YX0 = YX0*XX0
5470 FOX0 = FOX0 + A2X0(JX0)*YX0
5480 THETAOX0 = THETAOX0 + A3X0(JX0)*YX0
5490 NEXT
5500 XJ0 = FOX0* COS(THETAOX0)/SQR(X0)
5510 ICOUNTX0=1
5520 RETURN
6000 '
6010 '
6020 '
6030 'subroutine XJ1
6040 IF ICOUNTX1=1 GOTO 6200
6050 RESTORE 6090
6060 FOR IX1 = 1 TO 6
6070 READ B1X1(IX1)
6080 NEXT
6090 DATA -.56249985,.21093573,-.03954289,.00443319,-.00031761,.00001109
6100 FOR IX1 = 1 TO 7
6110 READ B2X1(IX1)
6120 NEXT
6130 DATA .79788456,.00000156,.01659667,.00017105,-.00249511,.00113653
6140 DATA -.00020033
6150 FOR IX1 = 1 TO 7
6160 READ B3X1(IX1)
6170 NEXT
6180 DATA -2.35619449,.12499612,.00005650,-.00637879,.00074384,.00079824
6190 DATA -.00029166
6200 ' Testing for negative or zero x1
6210 XJ1=1#
6220 ON SGN(X1)+2 GOTO 6230,6270,6300
6230 CLS
6240 PRINT "Message from subroutine xj1 : "
6250 PRINT "argument was -ve"
6260 STOP
6270 ICOUNTX1=1
6280 RETURN
6290 '
6300 ' Choosing method of calculation
6310 IF X1 > 3! GOTO 6430
6320 'Method of calculating xj1 for x1 < or = 3.0
6330 XJ1 = .5#
6340 WX1 = X1*X1/9#
6350 ZX1 = 1#
6360 FOR KX1 = 1 TO 6
6370 ZX1 = ZX1*WX1
6380 XJ1 = XJ1 + B1X1(KX1)*ZX1
6390 NEXT
6400 XJ1 = XJ1*X1
6410 ICOUNTX1=1
6420 RETURN
6430 'Method of calculating xj1 for x1 > 3.
6440 WX1 = 3#/X1
6450 ZX1 = 1#
6460 F1X1 = B2X1(1)
6470 THETA1X1 = X1 + B3X1(1)
6480 FOR LX1= 2 TO 7
6490 ZX1 = ZX1*WX1
```

```
6500 F1X1 = F1X1 + B2X1(LX1)*ZX1
6510 THETA1X1 = THETA1X1 + B3X1(LX1)*ZX1
6520 NEXT
6530 XJ1 = F1X1* COS(THETA1X1) / SQR(X1)
6540 ICOUNTX1=1
6550 RETURN
```

APPENDIX 3.10

Program Listing of SUB3 (CAN2)

```
1000 '
1005 '
1010 '
1015 '
1020 ' ************************************************************************
1025 '                          CAN2
1030 ' SUBROUTINE CAN2 CALCULATES A MATRIX OF TEMPERATURE VALUES INSIDE A
1035 ' FINITE CYLINDRICAL CONTAINER OF FOOD. THE CONTAINER IS INITIALLY AT A
1040 ' UNIFORM TEMPERATURE AND FROM TIME=0 IS SUBJECTED AT ALL ITS FACES TO
1045 ' A NEW TEMPERATURE. A FINITE HEAT TRANSFER COEFFICIENT H EXISTS AT ALL
1050 ' THE FACES. SUBROUTINE CAN2 REQUIRES SUBROUTINES BETART AND GAMMRT;
1055 ' IN TURN GAMMRT REQUIRES THE SUBROUTINES XJ0 AND XJ1.
1060 '
1065 ' CAUTION !..... CAN2 OPERATES ENTIRELY IN DOUBLE PRECISION, EXCEPT
1070 '
1075 '       FOR ARGUMENTS WHICH ARE INPUT AND OUTPUT IN REAL*4
1080 '
1085 ' ************************************************************************
1090 ' SUBROUTINE PARAMETERS:
1095 ' tbath! ="TEMPERATURE TO WHICH SURFACE IS SUBJECTED - DEG C - INPUT
1100 ' tinit! ="INITIAL TEMPERATURE OF CONTAINER CONTENTS - DEG C - INPUT
1105 ' alpha! ="THERMAL DIFFUSIVITY OF CONTENTS - M**2/SEC - INPUT
1110 ' rho! ="DENSITY OF CONTENTS - KG/M**3 - INPUT
1115 ' cp! ="SPECIFIC HEAT OF CONTENTS - J/KG-DEG C - INPUT
1120 ' halfl! ="HALF OF TOTAL LENTH OF FINITE CYLINDER - M - INPUT
1125 ' radius! ="radius$OF FINITE CYLINDER - M - INPUT
1130 ' H!="HEAT TRANSFER COEFFICIENT ON SURFACE - J/M**2-SEC-DEG C - INPUT
1135 ' NZ ="# OF Z VALUES AT WHICH TEMPERATURE IS TO BE EVALUATED - INPUT
1140 ' NR ="# OF RADII AT WHICH TEMPERATURE IS TO BE EVALUATED - INPUT
1145 ' NTIME ="# OF TIME VALUES AT WHICH TEMP IS TO BE EVALUATED - INPUT
1147 ' TOTIME! ="TOTAL PROCESSING TIME - SEC - INPUT
1150 ' z! ="VECTOR OF Z VALUES FOR TEMPERATURE EVALUATION - M - INPUT
1155 ' R! ="VECTOR OF RADIUS VALUES FOR TEMPERATURE EVALUATION - M - INPUT
1160 ' time! ="VECTOR OF TIME VALUES FOR TEMP EVALUATION - DEG C - INPUT
1180 ' resultc2! ="THREE DIMENSIONAL resultc2! ARRAY - DEG C - OUTPUT
1185 ' ************************************************************************
1190 ' INTERNAL PROGRAM VARIABLES USED;$ARRAYS
1195 ' BETA,BETAF1c2,BETAF2c2,BETAF3c2,betal1c2%,betal2c2% )FUNCTIONS OUTLINED
IN
1200 ' GAMM,GAMMF1c2,GAMMF2c2,GAMMF3c2,gamml1c2%,gamml2c2% ) PROGRAM DESCRIPTIO
N
1205 ' ************************************************************************
1210 ' INTERNAL PROGRAM VARIABLES USED;$REAL
1215 ' BC ="CYLINDER BIOT NUMBER
1220 ' BP ="PLATE BIOT NUMBER
1225 ' DUMM1c2,DUMM2c2,SUM1c2,SUM1Xc2,SUM2c2,SUM2Xc2   ) FUNCTIONS OUTLINED IN
1230 ' TERMc2,TERMMc2,TERMNc2                          ) PROGRAM DESCRIPTION
1233 ' DRC2,DZC2,DTIME                              )
1235 ' ************************************************************************
1240 ' INTERNAL PROGRAM VARIABLES USED;$INTEGER
1245 ' Ic2,IRc2,ISUM1Xc2,ISUM2Xc2,MTIMEc2,IZc2   ) FUNCTIONS OUTLINED IN
1250 ' nb,Ng,NBETA2c2,NGAMM2c2                   ) PROGRAM DESCRIPTION
1255 ' ************************************************************************
1260 ' INTERNAL PROGRAM VARIABLES USED;$LOGICAL
1265 ' iflagc2% = UNDERFLOW MEMORY FLAG
1270 ' ************************************************************************
1272 '
```

```
1273 'Initialization and Dimensioning
1275 DEFDBL A-H,O-Z
1280 DEFINT I-N
1285 OPTION BASE 1
1290 DIM BETAF1C2(100),BETAF2C2(100),BETAF3C2(100)      'subroutine CAN2
1295 DIM GAMMF1C2(100),GAMMF2C2(100),GAMMF3C2(100)
1300 DIM BETAL1C2%(100),BETAL2C2%(100),GAMML1C2%(100),GAMML2C2%(100)
1305 DIM RESULTC2!(10,10,21)
1310 DIM Z!(10),R!(10),TIME!(21)
1315 DIM UZC2(100),URC2(100)
1320 DIM XB(40),ROOTSB(40),BETA(100)      'subroutine BETART

1325 DIM XG(36),ROOTSG(36),GAMM(100)      'subroutine GAMMRT

1330 DIM A1X0(6),A2X0(7),A3X0(7)      'subroutine XJ0
1335 DIM B1X1(6),B2X1(7),B3X1(7)      'subroutine XJ1
1340 '
1345 '
1350 'Getting arguments that were stored in COMM1 by the calling submain
1355 OPEN"c:comm1" FOR INPUT AS 1
1360 INPUT#1,TBATH!,TINIT!,ALPHA!,HALFL!,RADIUS!,NZ,NR,NTIME,TOTIME!,RHO!,CP!,H
!,NSUBMAIN
1365 CLOSE 1
1370 '
1375 '
1530 'SETTING UP INPUT ARRAYS
1540 DZC2=HALFL!/CSNG(NZ-1)
1550 FOR IZC2=1 TO NZ
1560 Z!(IZC2)=CSNG((IZC2-1)*DZC2)
1570 NEXT IZC2
1580 DRC2=RADIUS!/CSNG(NR-1)
1590 FOR IRC2=1 TO NR
1600 R!(IRC2)=CSNG((IRC2-1)*DRC2)
1610 NEXT IRC2
1620 DTIME=TOTIME!/CSNG(NTIME-1)
1630 FOR LC2=1 TO NTIME
1640 TIME!(LC2)=CSNG((LC2-1)*DTIME)
1650 NEXT LC2
1660 '
1670 'PARAMETER CHECKING SECTION
1680 IF (ALPHA!>0!) AND (RHO!>0!) AND (CP!>0!)GOTO 1730
1690 PRINT " PROBLEM DETECTED WITH NON-VECTOR INPUT DATA IN CAN2: "
1700 PRINT " ALPHA!, RHO!, CP!, HALFL!, RADIUS!, H!, NR, NZ, NTIME "
1710 PRINT " PROGRAM HAS BEEN STOPPED "
1720 STOP
1730 IF (HALFL!>0!) AND (RADIUS!>0!) AND (H!>0!) GOTO 1740 ELSE GOTO 1690
1740 IF (NZ>1) AND (NR>1) AND (NTIME>1) GOTO 1750 ELSE GOTO 1690
1750 FOR IC2= 1 TO NZ
1760 IF (Z!(IC2)<0!) OR (Z!(IC2)>HALFL!)  GOTO 1780 ELSE GOTO 1820
1770 NEXT
1780 PRINT "PROBLEM WITH VECTOR DATA IN CAN2: "
1790 PRINT " TIME!, Z!,OR R!"
1800 PRINT " THE PROGRAM HAS BEEN STOPPED"
1810 STOP
1820 FOR IC2= 1 TO NR
1830 IF (R!(IC2)<0!) OR (R!(IC2)>RADIUS!)  GOTO 1780 ELSE GOTO 1850
1840 NEXT
```

```
1850 FOR IC2= 1 TO NTIME
1860 IF (TIME!(IC2)<0!) GOTO 1780 ELSE GOTO 1880
1870 NEXT
1880 '
1930 '
1940 'INITIALIZING COUNTERS AND LOGIC ARRAYS
1950 FOR IC2= 1 TO 100
1960 BETAL1C2%(IC2)=0
1970 BETAL2C2%(IC2)=0
1980 GAMML1C2%(IC2)=0
1990 GAMML2C2%(IC2)=0
2000 NEXT
2010 '
2020 'CALCULATING AND CHECKING BIOT NUMBERS
2030 BP=CDBL(H!*HALFL!/ALPHA!/RHO!/CP!)
2040 IF(BP<.02#) GOTO 2050 ELSE GOTO 2080
2050 PRINT " MESSAGE FROM SUBPROGRAM CAN2 :"
2060 PRINT " VERY SMALL PLATE BIOT NUMBER DETECTED:"
2070 PRINT " BIOT NUMBER WAS: ";BP
2080 IF(BP>200#) GOTO 2090 ELSE GOTO 2120
2090 PRINT " MESSAGE FROM SUBPROGRAM CAN2 :"
2100 PRINT " VERY LARGE PLATE BIOT NUMBER DETECTED:"
2110 PRINT " BIOT NUMBER WAS: ";BP
2120 BC=CDBL(H!*RADIUS!/ALPHA!/RHO!/CP!)
2130 IF(BC<.02#) GOTO 2140 ELSE GOTO 2170
2140 PRINT " MESSAGE FROM SUBPROGRAM CAN2 :"
2150 PRINT " VERY SMALL CYLINDER BIOT NUMBER DETECTED:"
2160 PRINT " BIOT NUMBER WAS: ";BC
2170 IF(BC>200#) GOTO 2180 ELSE GOTO 2220
2180 PRINT " MESSAGE FROM SUBPROGRAM CAN2 :"
2190 PRINT " VERY LARGE CYLINDER BIOT NUMBER DETECTED:"
2200 PRINT " BIOT NUMBER WAS: ";BC
2210 '
2220 ' START OF TIME LOOP
2230 FOR ITIMEC2=1 TO NTIME
2240 IF(TIME!(ITIMEC2)>0!) GOTO 2320
2250 FOR IRC2=1 TO NR
2260 URC2(IRC2) = .5#
2270 NEXT IRC2
2280 FOR IZC2=1 TO NZ
2290 UZC2(IZC2) = .5#
2300 NEXT IZC2
2310 GOTO 3050
2320 DUMM1C2 = CDBL(-ALPHA!*TIME!(ITIMEC2))
2330 '
2340 ' Z-LOOP
2350 FOR IZC2=1 TO NZ
2360 NB=0
2370 SUM1C2 = 0#
2380 IFLAGC2%=0
2390 SUM1XC2=0#
2400 FOR ISUM1XC2= 1 TO 2
2410 NB=NB+1
2420 IF (NB<=100) GOTO 2480
2430 PRINT " NUMBER OF ROOTS REQUIRED IN Z-LOOP GREATER THAN 100 "
2440 PRINT " ITIMEc2 WAS :";ITIMEC2, "     NB WAS: ";NB
2450 PRINT " THE PROGRAM WAS STOPPED"
```

```
2460 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
2470 STOP
2480 IF (BETAL1C2%(NB)<>0) GOTO 2550
2490 GOSUB 3120 ' subroutine BETART
2500 DUMM2C2=SIN(BETA(NB))
2510 BETAF1C2(NB)=DUMM2C2/(BETA(NB)+DUMM2C2*COS(BETA(NB)))
2520 BETAF2C2(NB)=BETA(NB)/CDBL(HALFL!)
2530 BETAF3C2(NB)=BETAF2C2(NB)^2
2540 BETAL1C2%(NB)=-1
2550 TERMC2=DUMM1C2*BETAF3C2(NB)
2560 IF (TERMC2>-88#) GOTO 2590
2570 TERMC2=-88#
2580 IFLAGC2%=-1
2590 TERMNC2=BETAF1C2(NB)*COS(BETAF2C2(NB)*CDBL(Z!(IZC2)))*EXP(TERMC2)
2600 SUM1XC2=SUM1XC2+ABS(TERMNC2)
2610 SUM1C2=SUM1C2+TERMNC2
2620 NEXT ISUM1XC2
2630 IF (ABS(SUM1XC2/SUM1C2)>.0001#) AND (SUM1XC2>.0000000001#) AND (IFLAGC2%=0
) GOTO 2390
2640 IF (ABS(SUM1C2)<.0000000001#)  THEN SUM1C2=0#
2650 UZC2(IZC2)=SUM1C2
2660 NEXT IZC2
2670 '
2680 'R-LOOP
2690 FOR IRC2=1 TO NR
2700 NG=0
2710 SUM2C2=0#
2720 IFLAGC2%=0
2730 SUM2XC2=0#
2740 FOR ISUM2YC2=1 TO 2
2750 NG=NG+1
2760 IF (NG<=100) GOTO 2810
2770 PRINT " MESSAGE FROM SUBROUTINE CAN2: "
2780 PRINT " MORE THAN 100 VALUES REQUIRED IN R LOOP "
2790 PRINT " ITIMEc2 IS: ";ITIMEC2," NG IS : ";NG
2800 PRINT " PROGRAM WAS STOPPED"
2810 GOSUB 3590    ' subroutine GAMMRT
2820 XO = GAMM(NG)'PRINT "GAMM(NG)",GAMM(NG)
2830 GOSUB 4180   ' subroutine XJ0
2840 IF (GAMML1C2%(NG)=-1) GOTO 2880
2850 GAMMF1C2(NG)=(BC^2 + GAMM(NG)^2)*XJ0
2860 GAMMF3C2(NG)=(GAMM(NG)/CDBL(RADIUS!))^2
2870 GAMML1C2%(NG)=-1
2880 XO = GAMM(NG)*CDBL(R!(IRC2)/RADIUS!)'PRINT"GAMM(NG),IRC2,R!,RADIUS!",GAMM(
NG),IRC2,R!(IRC2),RADIUS!
2890 GOSUB 4180
2900 GAMMF2C2(NG)=XJ0/GAMMF1C2(NG)
2910 GAMML2C2%(NG)=-1
2920 TERMC2=DUMM1C2*GAMMF3C2(NG)
2930 IF (TERMC2>-88#) GOTO 2960
2940 TERMC2=-88#
2950 IFLAGC2%=-1
2960 TERMMC2=GAMMF2C2(NG)*EXP(TERMC2)
2970 SUM2XC2=SUM2XC2+ABS(TERMMC2)
2980 SUM2C2=SUM2C2+TERMMC2
2990 NEXT ISUM2YC2
3000 IF (ABS(SUM2XC2/SUM2C2)>.0001#) AND (SUM2XC2>.0000000001#) AND (IFLAGC2%=0
```

```
) GOTO 2730
3010 IF (ABS(SUM2C2)<.0000000001#) THEN SUM2C2=0#
3020 URC2(IRC2)=BC*SUM2C2
3030 NEXT IRC2
3035 '
3037 '
3040 'Calculating Temperature result
3050 FOR IZC2=1 TO NZ
3060 FOR IRC2=1 TO NR
3070 RESULTC2!(IZC2,IRC2,ITIMEC2)=TBATH!- (TBATH!-TINIT!)*4!*CSNG(UZC2(IZC2)*UR
C2(IRC2))
3072 NEXT IRC2
3074 NEXT IZC2
3076 NEXT ITIMEC2
3078 '
3080 '
3082 ' Writing results into COMM2
3084 OPEN "c:comm2" FOR OUTPUT AS 2
3086 FOR LC2=1 TO NTIME
3088 FOR IC2 = 1 TO NZ
3090 FOR JC2 = 1 TO NR
3092 PRINT#2,Z!(IC2),R!(JC2),TIME!(LC2),RESULTC2!(IC2,JC2,LC2)
3094 NEXT JC2,IC2,LC2
3096 CLOSE 2
3097 '
3098 IF NSUBMAIN=1 THEN CHAIN"MAINA"
3100 IF NSUBMAIN=2 THEN CHAIN"MAINB"
3102 IF NSUBMAIN=3 THEN CHAIN"MAINC"
3104 IF NSUBMAIN=4 THEN CHAIN"MAIND"
3106 IF NSUBMAIN=5 THEN CHAIN "MAINE"
3110 RETURN
3120 '
3130 '
3140 '
3150 '
3160 '
3170 'BETART subroutine
3180 PI=3.1415926654#
3190 IF ICOUNTB=1 GOTO 3360
3200 RESTORE 3240
3210 FOR IB = 1 TO 40
3220 READ XB(IB)
3230 NEXT
3240 DATA 0.0,0.001,0.002,0.004,0.006,0.008,0.01,0.02,0.04,0.06
3250 DATA 0.08,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
3260 DATA 1.0,1.5,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0
3270 DATA 10.0,15.0,20.0,30.0,40.0,50.0,60.0,80.0,100.0,1.0D20
3280 RESTORE 3320
3290 FOR IB = 1 TO 40
3300 READ ROOTSB(IB)
3310 NEXT
3320 DATA 0.0,0.0316,0.0447,0.0632,0.0774,0.0893,0.0998,0.1410,0.1987,0.2425
3330 DATA 0.2791,0.3111,0.4328,0.5218,0.5932,0.6533,0.7051,0.7506,0.7910,0.8274
3340 DATA 0.8603,0.9882,1.0769,1.1925,1.2646,1.3138,1.3496,1.3766,1.3978,1.4149
3350 DATA 1.4289,1.4729,1.4961,1.5202,1.5325,1.5400,1.5451,1.5514,1.5552,1.5708
```

```
3900 ROOT1G=GAMM(NG-1)+TI
3910 IF BC<.375# THEN DROOTG=-DROOTG
3920 X0=ROOT1G'x0 is the input value for xj0 subroutine
3930 GOSUB 4180
3940 X1=ROOT1G'x1 is the input value for xj1 subroutine
3950 GOSUB 4730
3960 F1G=FNFUNCG(ROOT1G)
3970 ROOT2G=ROOT1G+DROOTG
3980 X0=ROOT2G
3990 GOSUB 4180
4000 X1=ROOT2G
4010 GOSUB 4730
4020 F2G=FNFUNCG(ROOT2G)
4030 SOLUTNG=ROOT1G-F1G*(ROOT2G-ROOT1G)/(F2G-F1G)
4040 X0=SOLUTNG
4050 GOSUB 4180
4060 X1=SOLUTNG
4070 GOSUB 4730
4080 FSOLG=FNFUNCG(SOLUTNG)
4090 IF ABS(FSOLG)<.000001# GOTO 4150
4100 ROOT1G=ROOT2G
4110 F1G=F2G
4120 ROOT2G=SOLUTNG
4130 F2G=FSOLG
4140 GOTO 4030
4150 GAMM(NG)=SOLUTNG
4160 ICOUNTG=1
4170 RETURN
4180 '
4190 '
4200 '
4210 'subroutine XJ0
4220 IF ICOUNTX0=1 GOTO 4380
4230 RESTORE 4270
4240 FOR IX0 = 1 TO 6
4250 READ A1X0(IX0)
4260 NEXT
4270 DATA -2.2499997,1.2656208,-.3163866,.0444479,-.0039444,.0002100
4280 FOR IX0 = 1 TO 7
4290 READ A2X0(IX0)
4300 NEXT
4310 DATA .79788456,-.00000077,-.00552740,-.00009512,.00137237,-.00072805
4320 DATA .00014476
4330 FOR IX0 = 1 TO 7
4340 READ A3X0(IX0)
4350 NEXT
4360 DATA -.78539816,-.04166397,-.00003954,.00262573,-.00054125,-.00029333
4370 DATA .00013558
4380 ' Testing for negative or zero x0
4390 XJ0 = 1#
4400 ON SGN(X0)+2 GOTO 4410,4450,4490
4410 CLS
4420 PRINT "Message from subroutine xj0:"
4430 PRINT "argument was -ve"
4440 STOP
4450 ICOUNTX0=1
4460 RETURN
```

```
3360 DEF FNFB(ROOTB)=ROOTB*TAN(ROOTB)-BP
3370 DROOTB=-.005
3380 IF NB<>1 GOTO 3440
3390 FOR IB=1 TO 39
3400 IF BP<XB(IB) GOTO 3450
3410 ROOT1B=ROOTSB(IB+1)
3420 NEXT
3430 GOTO 3450
3440 ROOT1B=BETA(NB-1)+TI
3450 F1B=FNFB(ROOT1B)
3460 ROOT2B=ROOT1B+DROOTB
3470 F2B=FNFB(ROOT2B)
3480 SOLUTNB=ROOT1B-F1B*(ROOT2B-ROOT1B)/(F2B-F1B)
3490 FSOLB=FNFB(SOLUTNB)
3500 IF ABS(FSOLB)<.000001# GOTO 3560
3510 ROOT1B=ROOT2B
3520 F1B=F2B
3530 ROOT2B=SOLUTNB
3540 F2B=FSOLB
3550 GOTO 3480
3560 BETA(NB)=SOLUTNB
3570 ICOUNTB=1
3580 RETURN
3590 '
3600 '
3610 '
3620 '
3630 '
3640 'GAMMRT subroutine
3650 PI=3.1415926541#
3660 IF ICOUNTG=1 GOTO 3830
3670 RESTORE 3710
3680 FOR IG = 1 TO 36
3690 READ XG(IG)
3700 NEXT
3710 DATA 0.0,0.01,0.02,0.04,0.06,0.08,0.1,0.15,0.2,0.3
3720 DATA 0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.5,2.0,3.0
3730 DATA 4.0,5.0,6.0,7.0,8.0,9.0,10.0,15.0,20.0,30.0
3740 DATA 40.0,50.0,60.0,80.0,100.0,1.0D+20
3750 FOR JG = 1 TO 36
3760 READ ROOTSG(JG)
3770 NEXT
3780 DATA 0.0,0.1412,0.1995,0.2814,0.3438,0.3960,0.4417,0.5376,0.6170,0.7465
3790 DATA 0.8516,0.9408,1.0184,1.0873,1.1490,1.2048,1.2558,1.4569,1.5994,1.7887

3800 DATA 1.9081,1.9898,2.0490,2.0937,2.1286,2.1566,2.1795,2.2509,2.2880,2.3261

3810 DATA 2.3455,2.3572,2.3651,2.3750,2.3809,2.4048
3820 DEF FNFUNCG(ROOTG)=ROOTG*XJ1-BC*XJ0     'actually xj1(root) and xj0(root)
3830 DROOTG=-.005
3840 IF NG<>1  GOTO 3900
3850 FOR IG = 1 TO 35
3860 IF BC<XG(IG) GOTO 3920
3870 ROOT1G=ROOTSG(IG+1)
3880 NEXT
3890 GOTO 3920
```

```
4490 ' Choosing method of calculation
4500 IF X0 > 3! GOTO 4600
4510 ' Method of calculating xj0 for x0 < or = 3.
4520 XX0 = X0*X0/9#
4530 YX0 = 1#
4540 FOR IX0 = 1 TO 6
4550 YX0 = YX0 * XX0
4560 XJ0 = XJ0 + A1X0(IX0)*YX0
4570 NEXT
4580 ICOUNTX0=1
4590 RETURN
4600 ' Method of calculating xj0 for x0 > 3.
4610 XX0 = 3#/X0
4620 YX0 = 1#
4630 FOX0 = A2X0(1)
4640 THETA0X0 = X0 + A3X0(1)
4650 FOR JX0 = 2 TO 7
4660 YX0 = YX0*XX0
4670 FOX0 = FOX0 + A2X0(JX0)*YX0
4680 THETA0X0 = THETA0X0 + A3X0(JX0)*YX0
4690 NEXT
4700 XJ0 = FOX0* COS(THETA0X0)/SQR(X0)
4710 ICOUNTX0=1
4720 RETURN
4730 '
4740 '
4750 '
4760 'subroutine XJ1
4770 IF ICOUNTX1=1 GOTO 4930
4780 RESTORE 4820
4790 FOR IX1 = 1 TO 6
4800 READ B1X1(IX1)
4810 NEXT
4820 DATA -.56249985,.21093573,-.03954289,.00443319,-.00031761,.00001109
4830 FOR IX1 = 1 TO 7
4840 READ B2X1(IX1)
4850 NEXT
4860 DATA .79788456,.00000156,.01659667,.00017105,-.00249511,.00113653
4870 DATA -.00020033
4880 FOR IX1 = 1 TO 7
4890 READ B3X1(IX1)
4900 NEXT
4910 DATA -2.35619449,.12499612,.00005650,-.00637879,.00074384,.00079824
4920 DATA -.00029166
4930 ' Testing for negative or zero x1
4935 XJ1=1#
4940 ON SGN(X1)+2 GOTO 4950,4990,5030
4950 CLS
4960 PRINT "Message from subroutine xj1 : "
4970 PRINT "argument was -ve"
4980 STOP
4990 ICOUNTX1=1
5000 RETURN
5020 '
5030 ' Choosing method of calculation
5040 IF X1 > 3! GOTO 5160
5050 'Method of calculating xj1 for x1 < or = 3.0
```

```
5060 XJ1 = .5#
5070 WX1 = X1*X1/9#
5080 ZX1 = 1#
5090 FOR KX1 = 1 TO 6
5100 ZX1 = ZX1*WX1
5110 XJ1 = XJ1 + B1X1(KX1)*ZX1
5120 NEXT
5130 XJ1 = XJ1*X1
5140 ICOUNTX1=1
5150 RETURN
5160 'Method of calculating xj1 for x1 > 3.
5170 WX1 = 3#/X1
5180 ZX1 = 1#
5190 F1X1 = B2X1(1)
5200 THETA1X1 = X1 + B3X1(1)
5210 FOR LX1= 2 TO 7
5220 ZX1 = ZX1*WX1
5230 F1X1 = F1X1 + B2X1(LX1)*ZX1
5240 THETA1X1 = THETA1X1 + B3X1(LX1)*ZX1
5250 NEXT
5260 XJ1 = F1X1* COS(THETA1X1) / SQR(X1)
5270 ICOUNTX1=1
5280 RETURN
```

APPENDIX 3.11

Program Listing of SUB4 (POUCH1)

```
1000 '
1010 '
1020 '
1030 '
1040 '
1050 ' ************************************************************************
1060 'SUBROUTINE POUCH1 CALCULATES THE TEMPERATURE-TIME CURVES AT A 3-D GRID
1070 ' OF X,Y,Z LOCATIONS FOR A NUMBER OF TIME VALUES IN A PARALLELLOPIPED-
1080 ' SHAPED BLOCK OF CONDUCTION-HEATING FOOD. THE FOOD IS ASSUMED TO
1090 ' INITIALLY BE AT A UNIFORM TEMPERATURE THROUGHOUT AND IS SUBJECTED TO
1100 ' A SUDDEN AND SUSTAINED TEMPERATURE CHANGE AT ITS SURFACE.
1110 ' IN POUCHN THE HEAT TRANSFER COEFFICIENT AT THE BLOCK'S SURFACE IS
1120 ' ASSUMED TO BE INFINITE FOR THE CALCULATIONS.
1130 ' ************************************************************************
1140 ' SUBROUTINE PARAMETERS:
1150 ' tbath! ="TEMPERATURE APPLIED AT SURFACE - DEG C - INPUT
1160 ' tinit! ="INITIAL TEMPERATURE OF BLOCK - DEG C - INPUT
1170 ' alpha! ="THERMAL DIFFUSIVITY OF MATERIAL - DEG C -INPUT
1180 ' halfx! ="HALF THE BLOCK DIMENSION IN THE X-DIRECTION - M - INPUT
1190 ' halfy! ="HELF THE BLOCK DIMENSION IN THE Y-DIRECTION - M - INPUT
1200 ' halfz! ="HALF THE BLOCK DIMENSION IN THE Z-DIRECTION - M - INPUT
1210 ' NX ="NUMBER OF X-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1220 ' NY ="NUMBER OF Y-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1230 ' NZ ="NUMBER OF Z-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1240 ' x! ="VECTOR OF X-COORDINATE VALUES FOR CALCULATIONS - M - INPUT
1250 ' y! ="VECTOR OF Y-COORDINATE VALUES FOR CALCULATIONS - M - INPUT
1260 ' z! ="VECTOR OF Z-COORDINATE VALUES FOR CALCULATIONS - M -INPUT
1270 ' NTIME ="NUMBER OF TIME VALUES FOR resultp1! CALCULATION - INPUT
1280 ' TOTIME! ="TOTAL PROCESSING TIME - S - INPUT
1290 ' time! ="VECTOR OF TIME VALUES FOR resultp1! CALCULATIONS - INPUT
1300 ' resultp1! ="FOUR-DIMENSIONAL resultp1! ARRAY - OUTPUT
1310 ' ************************************************************************
1320 ' INTERNAL ARRAYS:
1330 ' x!, y!, z!, time!                          )
1340 ' xl1p1%, yl1p1%, zl1p1%                     )
1350 ' xf1p1, xf2p1, xf3p1                        ) AS OUTLINED IN PROGRAM
1360 ' yf1p1, yf2p1p1, yf3p1                      ) DESCRIPTION
1370 ' zf1p1, zf2p1, zf3p1                        )
1380 ' uxp1,uyp1,uzp1                             )
1390 ' ************************************************************************
1400 ' INTERNAL REAL VARIABLES:
1410 ' DUMM1p1,DUMM2p1,DUMM3p1,DUMM4p1                    ) AS OUTLINED IN PROGRAM
1420 ' SUM1p1,SUM1Xp1,SUM2p1,SUM2Xp1,SUM3p1,SUM3Xp1       ) DESCRIPTION
1430 ' TERMp1,TERMLp1,TERMMp1,TERMNp1                     )
1440 ' DX,DY,DZ,DTIME                                     )
1450 ' ************************************************************************
1460 ' INTERNAL INTEGER VARIABLES:
1470 ' Ip1,IXp1,IYp1,IZp1,ITIMEp1          )
1480 ' ISUM1Xp1,ISUM2Yp1,ISUM3Zp1          ) AS OUTLINED IN PROGRAM DESCRIPTION
1490 ' Np1,Mp1,Lp1 = COUNTERS FOR X,Y AND Z ROOTS
1500 ' ************************************************************************
1510 ' INTERNAL LOGICAL VARIABLES:
1520 ' IFLAGp1% = EXPONENT UNDERFLOW TRUTH FLAG
1530 ' ************************************************************************
1540 '
1550 '
1560 'Initialization and Dimensioning
```

```
1570 DEFDBL A-H,O-Z
1580 DEFINT I-N
1590 OPTION BASE 1
1600 DIM XF1P1(100),XF2P1(100),XF3P1(100)
1610 DIM YF1P1(100),YF2P1(100),YF3P1(100)
1620 DIM ZF1P1(100),ZF2P1(100),ZF3P1(100)
1630 DIM XL1P1%(100),YL1P1%(100),ZL1P1%(100)
1640 DIM UXP1(100),UYP1(100),UZP1(100)
1650 DIM X!(5),Y!(5),Z!(5),TIME!(21)
1660 DIM RESULTP1!(5,5,5,21)
1670 '
1680 '
1690 'Getting arguments that were stored in COMM1 by calling program
1700 OPEN "c:comm1" FOR INPUT AS 1
1710 INPUT#1,TBATH!,TINIT!,ALPHA!,HALFX!,HALFY!,HALFZ!,NX,NY,NZ,NTIME,TOTIME!,N
SUBMAIN
1720 CLOSE 1
1730 '
1740 '
1750 'Calculating X!, Y!, Z! and TIME! arrays
1760 IF (1<NX) AND (1<NY) AND (1<NZ) AND (1<NTIME) GOTO 1770 ELSE 1780
1770 IF (NX<=5) AND (NY<=5) AND (NZ<=5) AND (NTIME<=21) GOTO 1830
1780 PRINT" PROBLEM WITH NX,NY,NZ OR NTIME"
1790 PRINT" SUB2 WAS STOPPED"
1800 PRINT"nx=";NX,"ny=";NY,"nz=";NZ,"ntime=";NTIME
1810 STOP
1820 '
1830 DX=HALFX!/CSNG(NX-1)
1840 FOR IXP1=1 TO NX
1850 X!(IXP1)=CSNG(IXP1-1)*DX
1860 NEXT IXP1
1870 '
1880 DY=HALFY!/CSNG(NY-1)
1890 FOR IYP1=1 TO NY
1900 Y!(IYP1)=CSNG(IYP1-1)*DY
1910 NEXT IYP1
1920 '
1930 DZ=HALFZ!/CSNG(NZ-1)
1940 FOR IZP1=1 TO NZ
1950 Z!(IZP1)=CSNG(IZP1-1)*DZ
1960 NEXT IZP1
1970 '
1980 DTIME=TOTIME!/CSNG(NTIME-1)
1990 FOR LP1=1 TO NTIME
2000 TIME!(LP1)=CSNG((LP1-1)*DTIME)
2010 NEXT LP1
2020 '
2030 '
2040 ' Parameter Checking Section
2050 '
2060 IF ALPHA!>0!  THEN GOTO 2120
2070 PRINT " PROBLEM WITH NON-VECTOR INPUT DATA:"
2080 PRINT "ALPHA!, HALFX!, HALFY! or HALFZ!"
2090 PRINT "THE PROGRAM WAS STOPPED"
2100 STOP
2110 '
2120 IF (HALFX!>0!) AND (HALFY!>0!) AND (HALFZ!>0!) THEN 2160 ELSE 2070
```

```
2130 '
2140 '
2150 '
2160 PI = 3.141592654#
2170 '
2180 '
2190 ' INITIALIZING LOGIC FLAGS
2200 FOR IP1=1 TO 100
2210 XL1P1%(IP1)=0
2220 YL1P1%(IP1)=0
2230 ZL1P1%(IP1)=0
2240 NEXT
2250 '
2260 ' START OF TIME LOOP
2270 FOR ITIMEP1=1 TO NTIME
2280 IF TIME!(ITIMEP1)>0!  THEN GOTO 2390
2290 FOR IXP1=1 TO NX
2300 UXP1(IXP1)=RI/4!
2310 NEXT
2320 FOR IYP1=1 TO NY
2330 UYP1(IYP1)=RI/4!
2340 NEXT
2350 FOR IZP1=1 TO NZ
2360 UZP1(IZP1)=RI/4!
2370 NEXT
2380 GOTO 3440
2390 DUMM2P1=CDBL(-ALPHA!*TIME!(ITIMEP1))
2400 DUMM4P1=CDBL(HALFX!)
2410 ' X-LOOP
2420 FOR IXP1=1 TO NX
2430 DUMM3P1=CDBL(X!(IXP1))
2440 NP1=0
2450 SUM1P1=0#
2460 IFLAGP1%=0
2470 SUM1XP1=0#
2480 FOR ISUM1XP1=1 TO 2
2490 NP1=NP1+1
2500 IF NP1<=100  THEN GOTO 2550
2510 PRINT "NUMBER OF ROOTS REQUIRED IN X LOOP GREATER THAN 100"
2520 PRINT "ITIMEP1 WAS";ITIMEP1,"IXP1 WAS";IXP1
2530 PRINT "THE PROGRAM WAS STOPPED"
2540 STOP
2550 DUMM1P1=CDBL(2*NP1 -1)
2560 IF XL1P1%(NP1)<>0THEN GOTO 2590
2570 XF3P1(NP1)=(DUMM1P1*PI/2!/DUMM4P1)^2
2580 XL1P1%(NP1)=-1
2590 TERMP1=DUMM2P1*XF3P1(NP1)
2600 IF TERMP1>-88#  THEN GOTO 2630
2610 TERMP1=-88#
2620 IFLAGP1%=-1
2630 XF1P1(NP1)=COS(DUMM1P1*PI*DUMM3P1/2!/DUMM4P1)
2640 XF2P1(NP1)=XF1P1(NP1)/DUMM1P1/((-1!)^(NP1+1))
2650 TERMNP1=XF2P1(NP1)*EXP(TERMP1)
2660 SUM1XP1=SUM1XP1+ABS(TERMNP1)
2670 SUM1P1=SUM1P1+TERMNP1
2680 NEXT ISUM1XP1
2690 IF (ABS(SUM1XP1/SUM1P1)>.0001#) AND (SUM1XP1>.0000000001#) AND (IFLAGP1%=0
```

```
) THEN GOTO 2470
2700 IF(ABS(SUM1P1)<.0000000001#) THEN SUM1P1=0#
2710 UXP1(IXP1)=SUM1P1
2720 NEXT IXP1
2730 '
2740 ' Y-LOOP
2750 DUMM4P1=CDBL(HALFY!)
2760 FOR IYP1=1 TO NY
2770 DUMM3P1=CDBL(Y!(IYP1))
2780 MP1=0
2790 SUM2P1=0#
2800 IFLAGP1% =0
2810 SUM2XP1=0#
2820 FOR ISUM2YP1=1 TO 2
2830 MP1=MP1+1
2840 IF MP1<=100  THEN GOTO 2890
2850 PRINT " MORE THAN 100 VALUES REQUIRED IN Y LOOP"
2860 PRINT "ITIMEP1 WAS";ITIMEP1,"IYP1 WAS";IYP1
2870 PRINT "THE PROGRAM WAS STOPPED"
2880 STOP
2890 DUMM1P1=CDBL(2*MP1 -1)
2900 IF YL1P1%(MP1)<>0THEN GOTO 2930
2910 YF3P1(MP1)=(DUMM1P1*PI/2!/DUMM4P1)^2
2920 YL1P1%(MP1)=-1
2930 TERMP1=DUMM2P1*YF3P1(MP1)
2940 IF TERMP1>-88#  THEN GOTO 2970
2950 TERMP1=-88#
2960 IFLAGP1%=-1
2970 YF1P1(MP1)=COS(DUMM1P1*PI*DUMM3P1/2!/DUMM4P1)
2980 YF2P1(MP1)=YF1P1(MP1)/DUMM1P1/((-1!)^(MP1+1))
2990 TERMMP1=YF2P1(MP1)*EXP(TERMP1)
3000 SUM2XP1=SUM2XP1+ABS(TERMMP1)
3010 SUM2P1=SUM2P1+TERMMP1
3020 NEXT ISUM2YP1
3030 IF (ABS(SUM2XP1/SUM2P1)>.0001#) AND (SUM2XP1>.0000000001#) AND (IFLAGP1%=0
) THEN GOTO 2810
3040 IF(ABS(SUM2P1)<.0000000001#) THEN SUM2P1=0#
3050 UYP1(IYP1)=SUM2P1
3060 NEXT IYP1
3070 '
3080 ' Z-LOOP
3090 DUMM4P1=CDBL(HALFZ!)
3100 FOR IZP1=1 TO NZ
3110 DUMM3P1=CDBL(Z!(IZP1))
3120 LP1=0
3130 SUM3P1=0#
3140 IFLAGP1%=0
3150 SUM3XP1=0#
3160 FOR ISUM3ZP1=1 TO 2
3170 LP1=LP1+1
3180 IF LP1<=100  THEN GOTO 3230
3190 PRINT "MORE THAN 100 VALUES REQUIRED IN Z LOOP"
3200 PRINT "ITIMEP1 WAS";ITIMEP1,"IZP1 WAS";IZP1
3210 PRINT "THE PROGRAM WAS STOPPED"
3220 STOP
3230 DUMM1P1=CDBL(2*LP1 -1)
3240 IF ZL1P1%(LP1)<>0THEN GOTO 3270
```

```
3250 ZF3P1(LP1)=(DUMM1P1*PI/2!/DUMM4P1)^2
3260 ZL1P1%(LP1)=-1
3270 TERMP1=DUMM2P1*ZF3P1(LP1)
3280 IF TERM>-88#  THEN GOTO 3310
3290 TERMP1=-88#
3300 IFLAGP1%=-1
3310 ZF1P1(LP1)=COS(DUMM1P1*PI*DUMM3P1/2!/DUMM4P1)
3320 ZF2P1(LP1)=ZF1P1(LP1)/DUMM1P1/((-1!)^(LP1+1))
3330 TERMLP1=ZF2P1(LP1)*EXP(TERMP1)
3340 SUM3XP1=SUM3XP1+ABS(TERMLP1)
3350 SUM3P1=SUM3P1+TERMLP1
3360 NEXT ISUM3ZP1
3370 IF (ABS(SUM3XP1/SUM3P1)>.0001#) AND (SUM3XP1>.0000000001#) AND (IFLAGP1%=0
) THEN GOTO 3150
3380 IF(ABS(SUM3P1)<.0000000001#) THEN SUM3P1=0#
3390 UZP1(IZP1)=SUM3P1
3400 NEXT IZP1
3410 '
3420 '
3430 'Calculating the temperature result
3440 FOR IXP1=1 TO NX
3450 FOR IYP1=1 TO NY
3460 FOR IZP1=1 TO NZ
3470 RESULTP1!(IXP1,IYP1,IZP1,ITIMEP1)=TBATH!- (TBATH!-TINIT!)*64!/CSNG(PI^3)*C
SNG(UXP1(IXP1)*UYP1(IYP1)*UZP1(IZP1))
3480 NEXT IZP1,IYP1,IXP1
3490 '
3500 '
3510 NEXT ITIMEP1
3520 '
3530 '
3540 'Printing Time, Position and Temp data into COMM2 file
3550 OPEN "c:comm2" FOR OUTPUT AS 2
3560 FOR LP1=1 TO NTIME
3570 FOR IP1 = 1 TO NX
3580 FOR JP1=1 TO NY
3590 FOR KP1=1 TO NZ
3600 PRINT#2,X!(IP1),Y!(JP1),Z!(KP1),TIME!(LP1),RESULTP1!(IP1,JP1,KP1,LP1)
3610 NEXT KP1,JP1,IP1,LP1
3620 CLOSE 2
3630 '
3640 '
3650 ' Chaining back to the submain program that called POUCH1
3660 IF NSUBMAIN=1 THEN CHAIN "MAINA"
3670 IF NSUBMAIN=2 THEN CHAIN "MAINB"
3680 IF NSUBMAIN=3 THEN CHAIN "MAINC"
3690 IF NSUBMAIN=4 THEN CHAIN "MAIND"
3700 IF NSUBMAIN=5 THEN CHAIN "maine"
3710 PRINT "Unreasonable nsubmain value"
3720 PRINT "Can't chain back to a submain"
3730 PRINT "Program SUB2 was terminated"
3740 STOP
```

APPENDIX 3.12

Program Listing of SUB5 (POUCH2)

```
1000 '
1005 '
1010 '
1015 '
1020 '
1025 ' ************************************************************************
1030 '                          POUCH2
1035 ' SUBROUTINE POUCH2 CALCULATES THE TEMPERATURE-TIME CURVES AT A 3-D GRID
1040 ' OF X,Y,Z LOCATIONS FOR A NUMBER OF TIME VALUES IN A PARALLELLOPIPED-
1045 ' SHAPED BLOCK OF CONDUCTION-HEATING FOOD. THE FOOD IS ASSUMED TO
1050 ' INITIALLY BE AT A UNIFORM TEMPERATURE THROUGHOUT AND IS SUBJECTED TO
1055 ' A SUDDEN AND SUSTAINED TEMPERATURE CHANGE AT ITS SURFACE.
1060 ' IN POUCH2 THE HEAT TRANSFER COEFFICIENT AT THE BLOCK'S SURFACE CAN BE
1065 ' SPECIFIED AND IS TAKEN INTO ACCOUNT IN THE CALCULATIONS.
1070 ' ************************************************************************
1075 ' SUBROUTINE PARAMETERS:
1080 ' tbath! ="TEMPERATURE APPLIED AT SURFACE - DEG C - INPUT
1085 ' tinit! ="INITIAL TEMPERATURE OF BLOCK - DEG C - INPUT
1090 ' alpha! ="THERMAL DIFFUSIVITY OF MATERIAL - DEG C -INPUT
1095 ' rho! ="DENSITY OF MATERIAL - KG/M**3 - INPUT
1100 ' cp! ="SPECIFIC HEAT OF MATERIAL - J/KG-DEG C - INPUT
1105 ' halfx! ="HALF THE BLOCK DIMENSION IN THE X-DIRECTION - M - INPUT
1110 ' halfy! ="HELF THE BLOCK DIMENSION IN THE Y-DIRECTION - M - INPUT
1115 ' halfz! ="HALF THE BLOCK DIMENSION IN THE Z-DIRECTION - M -INPUT
1120 ' h! ="HEAT TRANSFER COEFFICIENT AT SURFACE - J/SEC-M**2-DEG C - INPUT
1125 ' NX ="NUMBER OF X-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1130 ' NY ="NUMBER OF Y-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1135 ' NZ ="NUMBER OF Z-COORDINATE VALUES IN LOCATION MATRIX - INPUT
1140 ' X! ="VECTOR OF X-COORDINATE VALUES FOR CALCULATIONS - M -INPUT
1145 ' Y! ="VECTOR OF Y-COORDINATE VALUES FOR CALCULATIONS - M -INPUT
1150 ' Z! ="VECTOR OF Z-COORDINATE VALUES FOR CALCULATIONS - M -INPUT
1155 ' NTIME ="NUMBER OF TIME VALUES FOR resultp2! CALCULATION - INPUT
1160 ' TOTIME! ="TOTAL PROCESSING TIME  - SEC - INPUT
1165 ' TIME! ="VECTOR OF TIME VALUES FOR resultp2! CALCULATIONS - INPUT
1170 ' RESULTP2! ="FOUR-DIMENSIONAL RESULT ARRAY - OUTPUT
1175 ' ************************************************************************
1180 ' INTERNAL ARRAYS:
1185 ' BETXF1p2,BETXF2p2,BETXF3p2,BETXL1p2% )
1190 ' BETYF1p2,BETYF2p2,BETYF3p2,BETYL1p2% ) AS OUTLINED IN PROGRAM
1195 ' BETZF1p2,BETZF2p2,BETZF3p2,BETZL1p2% ) DESCRIPTION
1200 ' UXp2,UYp2,UZp2,                      )
1205 ' ************************************************************************
1210 ' INTERNAL REAL VARIABLES:
1215 ' DUMM1p2,DUMM2p2,DUMM3p2,DUMMXp2              ) AS OUTLINED IN PROGRAM
1220 ' SUM1p2,SUM1Xp2,SUM2p2,SUM2Xp2,SUM3p2,SUM3Xp2 ) DESCRIPTION
1225 ' TERMp2,TERMLp2,TERMMp2,TERMNp2               )
1230 ' BP=PLATE BIOT NUMBER                         )
1235 ' DX,DY,DZ,DTIME                               )
1240 ' ************************************************************************
1245 ' INTERNAL INTEGER VARIABLES:
1250 ' Ip2,IXp2,IYp2,IZp2,ITIMEp2                   ) AS OUTINED IN
1255 ' ISUM1Xp2,ISUM2Yp2,ISUM3Zp2                   ) PROGRAM DESCRIPTION
1260 ' Nb = COUNTER FOR X,Y AND Z ROOTS             )
1265 ' ************************************************************************
1270 ' INTERNAL LOGICAL VARIABLES:
1275 ' IFLAGp2% = EXPONENT UNDERFLOW TRUTH FLAG
1280 ' ************************************************************************
```

```
1285 '
1290 '
1295 ' Initialization and Dimensioning
1300 DEFDBL A-H,O-Z
1305 DEFINT I-N
1310 OPTION BASE 1
1315 DIM BETXF1P2(100),BETXF2P2(100),BETXF3P2(100)
1320 DIM BETYF1P2(100),BETYF2P2(100),BETYF3P2(100)
1325 DIM BETZF1P2(100),BETZF2P2(100),BETZF3P2(100)
1330 DIM BETXL1P2%(100),BETYL1P2%(100),BETZL1P2%(100)
1335 DIM X!(5),Y!(5),Z!(5),TIME!(21)
1340 DIM XB(40),ROOTSB(40),BETA(100)
1345 DIM UXP2(100),UYP2(100),UZP2(100)
1350 DIM RESULTP2!(5,5,5,21)
1355 '
1360 '
1365 'Getting arguments that were stored in COMM1 by calling program
1370 OPEN "c:comm1" FOR INPUT AS 1
1375 INPUT#1,TBATH!,TINIT!,ALPHA!,HALFX!,HALFY!,HALFZ!,NX,NY,NZ,NTIME,TOTIME!,R
HO!,CP!,H!,NSUBMAIN
1380 CLOSE 1
1385 '
1390 '
1395 ' CALCULATING X!,Y!,Z!,TIME!
1400 IF (1<NX) AND (1<NY) AND (1<NZ) AND (1<NTIME) GOTO 1405 ELSE 1410
1405 IF (NX<=5) AND (NY<=5) AND (NZ<=5) AND (NTIME<=21) GOTO 1425 ELSE 14 10
1410 PRINT "PROBLEM WITH NX, NY, NZ OR NTIME"
1415 PRINT "SUB3 WAS STOPPED"
1420 STOP
1425 '
1430 DX=HALFX!/CSNG(NX-1)
1435 FOR IXP2=1 TO NX
1440 X!(IXP2)=CSNG(IXP2-1)*DX
1445 NEXT IXP2
1450 '
1455 DY=HALFY!/CSNG(NY-1)
1460 FOR IYP2=1 TO NY
1465 Y!(IYP2)=CSNG(IYP2-1)*DY
1470 NEXT IYP2
1475 '
1480 DZ=HALFZ!/CSNG(NZ-1)
1485 FOR IZP2=1 TO NZ
1490 Z!(IZP2)=CSNG(IZP2-1)*DZ
1495 NEXT IZP2
1500 '
1505 DTIME=TOTIME!/CSNG(NTIME-1)
1510 FOR LP2=1 TO NTIME
1515 TIME!(LP2)=CSNG((LP2-1)*DTIME)
1520 NEXT LP2
1525 '
1530 '
1535 ' PARAMETER CHECKING SECTION
1540 IF (ALPHA!>0!) AND (RHO!>0!) AND (CP!>0!)GOTO 1570
1545 PRINT "PROBLEM WITH NON-VECTOR INPUT DATA:"
1550 PRINT "ALPHA!, RHO!, CP!, HALFX!, HALFY!, HALFZ! or H!"
1555 PRINT "PROGRAM WAS STOPPED"
1560 STOP
```

```
1565 '
1570 IF (HALFX!>0!) AND (HALFY!>0!) AND (HALFZ!>0!) AND (H!>0!) GOTO 1580
1575 GOTO 1545
1580 '
1585 '
1590 ' BIOT NUMBER CALCULATION
1595 DUMMXP2=CDBL(H!/ALPHA!/RHO!/CP!)
1600 BP=DUMMXP2*CDBL(HALFX!)
1605 '
1610 ' INITIALIZING LOGIC FLAGS
1615 FOR IP2 = 1 TO 100
1620 BETXL1P2%(IP2)=0
1625 BETYL1P2%(IP2)=0
1630 BETZL1P2%(IP2)=0
1635 NEXT
1640 '
1645 ' START OF TIME LOOP
1650 FOR ITIMEP2=1 TO NTIME
1655 IF (TIME!(ITIMEP2)>0!) GOTO 1710
1660 FOR IXP2=1 TO NX
1665 UXP2(IXP2)=.5#
1670 NEXT
1675 FOR IYP2=1 TO NY
1680 UYP2(IYP2)=.5#
1685 NEXT
1690 FOR IZP2 = 1 TO NZ
1695 UZP2(IZP2)=.5#
1700 NEXT
1705 GOTO 2245
1710 DUMM2P2=CDBL(-ALPHA!*TIME!(ITIMEP2))
1715 '
1720 ' x-LOOP
1725 BP=DUMMXP2*CDBL(HALFX!)
1730 FOR IXP2=1 TO NX
1735 DUMM3P2=CDBL(X!(IXP2))
1740 NB=0
1745 SUM1P2=0#
1750 IFLAGP2% =0
1755 SUM1XP2=0#
1760 FOR ISUM1XP2 =1 TO 2
1765 NB=NB+1
1770 IF(NB<=100) THEN GOTO 1795
1775 PRINT "MORE THAN 100 VALUES REQUIRED IN x LOOP"
1780 PRINT "ITIMEP2 WAS";ITIMEP2, "NB WAS";NB
1785 PRINT "THE PROGRAM WAS STOPPED"
1790 STOP
1795 IF(BETXL1P2%(NB)<>0) GOTO 1830
1800 GOSUB 3350      'subroutine BETART
1805 DUMM1P2=SIN(BETA(NB))
1810 BETXF1P2(NB)=DUMM1P2/(BETA(NB)+DUMM1P2*COS(BETA(NB)))
1815 BETXF2P2(NB)=BETA(NB)/CDBL(HALFX!)
1820 BETXF3P2(NB)=BETXF2P2(NB)^2
1825 BETXL1P2%(NB)=-1
1830 TERMP2=DUMM2P2*BETXF3P2(NB)
1835 IF(TERMP2>-88#) THEN GOTO 1850
1840 TERMP2=-88#
1845 IFLAGP2%=-1
```

```
1850 TERMNP2=BETXF1P2(NB)*COS(BETXF2P2(NB)*DUMM3P2)*EXP(TERMP2)
1855 SUM1XP2=SUM1XP2+ABS(TERMNP2)
1860 SUM1P2=SUM1P2+TERMNP2
1865 NEXT ISUM1XP2
1870 IF (ABS(SUM1XP2/SUM1P2)>.0001#) AND (SUM1XP2>.0000000001#) AND (IFLAGP2%=0
) THEN GOTO 1755
1875 IF(ABS(SUM1P2)<.0000000001#) THEN SUM1P2=0#
1880 UXP2(IXP2)=SUM1P2
1885 NEXT IXP2
1890 '
1895 ' Y-LOOP
1900 BP=DUMMXP2*CDBL(HALFY!)
1905 FOR IYP2=1 TO NY
1910 DUMM3P2=CDBL(Y!(IYP2))
1915 NB=0
1920 SUM2P2=0#
1925 IFLAGP2% =0
1930 SUM2XP2=0#
1935 FOR ISUM2YP2 =1 TO 2
1940 NB=NB+1
1945 IF(NB<=100) THEN GOTO 1970
1950 PRINT "MORE THAN 100 VALUES REQUIRED IN Y LOOP"
1955 PRINT "ITIMEP2 WAS";ITIMEP2, "NB WAS";NB
1960 PRINT "THE PROGRAM WAS STOPPED"
1965 STOP
1970 IF(BETYL1P2%(NB)<>0) GOTO 2005
1975 GOSUB 3350     !subroutine BETART
1980 DUMM1P2=SIN(BETA(NB))
1985 BETYF1P2(NB)=DUMM1P2/(BETA(NB)+DUMM1P2*COS(BETA(NB)))
1990 BETYF2P2(NB)=BETA(NB)/CDBL(HALFY!)
1995 BETYF3P2(NB)=BETYF2P2(NB)^2
2000 BETYL1P2%(NB)=-1
2005 TERMP2=DUMM2P2*BETYF3P2(NB)
2010 IF(TERMP2>-88#) THEN GOTO 2025
2015 TERMP2=-88#
2020 IFLAGP2%=-1
2025 TERMMP2=BETYF1P2(NB)*COS(BETYF2P2(NB)*DUMM3P2)*EXP(TERMP2)
2030 SUM2XP2=SUM2XP2+ABS(TERMMP2)
2035 SUM2P2=SUM2P2+TERMMP2
2040 NEXT ISUM2YP2
2045 IF (ABS(SUM2XP2/SUM2P2)>.0001#) AND (SUM2XP2>.0000000001#) AND (IFLAGP2%=0
) THEN GOTO 1930
2050 IF(ABS(SUM2P2)<.0000000001#) THEN SUM2P2=0#
2055 UYP2(IYP2)=SUM2P2
2060 NEXT IYP2
2065 '
2070 ' Z-LOOP
2075 BP=DUMMXP2*(HALFZ!)
2080 FOR IZP2=1 TO NZ
2085 DUMM3P2=CDBL(Z!(IZP2))
2090 NB=0
2095 SUM3P2=0#
2100 IFLAGP2%=0
2105 SUM3XP2=0#
2110 FOR ISUM3ZP2=1 TO 2
2115 NB=NB+1
2120 IF(NB<=100) THEN GOTO 2145
```

```
2125 PRINT " MORE THAN 100 VALUES REQUIRED IN Z LOOP"
2130 PRINT "ITIMEP2 WAS: ";ITIMEP2; '  NB WAS :"; NB
2135 PRINT "PROGRAM WAS STOPPED"
2140 STOP
2145 IF(BETZL1P2%(NB)<>0) THEN GOTO 2180
2150 GOSUB 3350        'subroutine BETART
2155 DUMM1P2=SIN(BETA(NB))
2160 BETZF1P2(NB)=DUMM1P2/(BETA(NB)+DUMM1P2*COS(BETA(NB)))
2165 BETZF2P2(NB)=BETA(NB)/CDBL(HALFZ!)
2170 BETZF3P2(NB)=BETZF2P2(NB)^2
2175 BETZL1P2%(NB)=-1
2180 TERMP2=DUMM2P2*BETZF3P2(NB)
2185 IF(TERMP2>-88#) THEN GOTO 2200
2190 TERMP2=-88#
2195 IFLAGP2%=-1
2200 TERMLP2=BETZF1P2(NB)*COS(BETZF2P2(NB)*DUMM3P2)*EXP(TERMP2)
2205 SUM3XP2=SUM3XP2+ABS(TERMLP2)
2210 SUM3P2=SUM3P2+TERMLP2
2215 NEXT ISUM3ZP2
2220 IF(ABS(SUM3XP2/SUM3P2)>.0001#) AND (SUM3XP2>.0000000001#) AND (IFLAGP2%=0)
 THEN GOTO 2105
2225 IF(ABS(SUM3P2)<.0000000001#) THEN SUM3P2=0#
2230 UZP2(IZP2)=SUM3P2
2235 NEXT IZP2
2240 '
2245 FOR IXP2=1 TO NX
2250 FOR IYP2=1 TO NY
2255 FOR IZP2=1 TO NZ
2260 RESULTP2!(IXP2,IYP2,IZP2,ITIMEP2)=TBATH!-(TBATH!-TINIT!)*8!*CSNG(UXP2(IXP2
)*UYP2(IYP2)*UZP2(IZP2))
2265 NEXT IZP2, IYP2,IXP2
2270 '
2275 '
2280 NEXT ITIMEP2
2285 '
2290 '
2295 'Printing Time, Position and Temp data into COMM2 file
2300 OPEN "c:comm2" FOR OUTPUT AS 2
2305 FOR LP2=1 TO NTIME
2310 FOR IP2=1 TO NX
2315 FOR JP2=1 TO NY
2320 FOR KP2=1 TO NZ
2325 PRINT#2,X!(IP2),Y!(JP2),Z!(KP2),TIME!(LP2),RESULTP2!(iP2,jP2,kT2,lP2)
2330 NEXT KP2,JP2,IP2,LP2
2335 CLOSE 2
2340 '
2345 '
2350 'Chaining back to the submain program"that called POUCH2
2355 IF NSUBMAIN=1 THEN CHAIN "MAINA"
2360 IF NSUBMAIN=2 THEN CHAIN "MAINB"
2365 IF NSUBMAIN=3 THEN CHAIN "MAINC"
2370 IF NSUBMAIN=4 THEN CHAIN "MAIND"
2375 IF NSUBMAIN=5 THEN CHAIN "maine"
2380 PRINT "UNREASONABLE NSUBMAIN VALUE"
2385 PRINT "CAN'T CHAIN BACK TO A SUBMAIN"
2390 PRINT "PROGRAM SUB3 WAS TERMINATED"
2395 STOP
```

```
3350 '
3355 '
3360 '
3365 '
3370 '
3375 'BETART subroutine
3380 PI=3.1415926543
3385 IF ICOUNTB=1 GOTO 3470
3390 RESTORE 3410
3395 FOR IB = 1 TO 40
3400 READ XB(IB)
3405 NEXT
3410 DATA 0.0,0.001,0.002,0.004,0.006,0.008,0.01,0.02,0.04,0.06
3415 DATA 0.08,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
3420 DATA 1.0,1.5,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0
3425 DATA 10.0,15.0,20.0,30.0,40.0,50.0,60.0,80.0,100.0,1.0D20
3430 RESTORE 3450
3435 FOR IB = 1 TO 40
3440 READ ROOTSB(IB)
3445 NEXT
3450 DATA 0.0,0.0316,0.0447,0.0632,0.0774,0.0893,0.0998,0.1410,0.1987,0.2425
3455 DATA 0.2791,0.3111,0.4328,0.5218,0.5932,0.6533,0.7051,0.7506,0.7910,0.8274

3460 DATA 0.8603,0.9882,1.0769,1.1925,1.2646,1.3138,1.3496,1.3766,1.3978,1.4149

3465 DATA 1.4289,1.4729,1.4961,1.5202,1.5325,1.5400,1.5451,1.5514,1.5552,1.5708

3470 DEF FNFB(ROOTB)=ROOTB*TAN(ROOTB)-BP
3475 DROOTB=-.005
3480 IF NB<>1 GOTO 3510
3485 FOR IB=1 TO 39
3490 IF BP<XB(IB) GOTO 3515
3495 ROOT1B=ROOTSB(IB+1)
3500 NEXT
3505 GOTO 3515
3510 ROOT1B=BETA(NB-1)+TI
3515 F1B=FNFB(ROOT1B)
3520 ROOT2B=ROOT1B+DROOTB
3525 F2B=FNFB(ROOT2B)
3530 SOLUTNB=ROOT1B-F1B*(ROOT2B-ROOT1B)/(F2B-F1B)
3535 FSOLB=FNFB(SOLUTNB)
3540 IF ABS(FSOLB)<.000001# GOTO 3570
3545 ROOT1B=ROOT2B
3550 F1B=F2B
3555 ROOT2B=SOLUTNB
3560 F2B=FSOLB
3565 GOTO 3530
3570 BETA(NB)=SOLUTNB
3575 ICOUNTB=1
3580 RETURN
```

APPENDIX 3.13

Program Listing of SESSREST

```
10 '
20 '
30 '          ************* SESSREST ******************
40 '
50 '
60 '
70 'Setting up the text screen
80 KEY OFF
90 SCREEN 0,0,0,0
100 WIDTH 80
110 COLOR 7,1,1
120 CLS
130 '
140 '
150 ' Opening and readying the ALPHA file
160 OPEN "c:alpha1" AS 1
170 FIELD 1,80 AS COMMENT$
180 '
190 '
200 'Printing the "OPTION NOT AVAILABLE" message
210 GET 1,1
220 PRINT COMMENT$
230 GET 1,12:PRINT COMMENT$
240 DUMMS$=INKEY$: IF DUMMS$="" GOTO 240
250 '
260 '
270 'Closing files so can chain back to MAIN
280 CLOSE 1
290 '
300 '
310 'Loading the please wait message
320 CLS: SCREEN 1: COLOR 0,0
330 DEF SEG=&HB800
340 BLOAD "b:fire.scn",0
350 '
360 '
370 'Chaining back to MAIN
380 CHAIN "main"
```

ALPHA1

```
 1 Sorry, this option is not yet available
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12 PRESS ANY KEY TO CONTINUE
```