ornl

ORNL/TM-11127

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Development of a Smart Temperature Measurement System Based on a Self-Calibrating Thermocoupl

F. R. Ruppel

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A05 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11127
Dist. Category UC-506

Instrumentation and Controls Division

DEVELOPMENT OF A SMART TEMPERATURE MEASUREMENT SYSTEM BASED ON A SELF-CALIBRATING THERMOCOUPLE*

F. R. Ruppel

Date Published - April 1989

*Thesis presented for the Master of Science Degree,
The University of Tennessee, Knoxville, 1988.

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0299437 5

CONTENTS

	Page
LIST OF FIGURES	v
LIST OF SYMBOLS	vii
ABSTRACT	ix
1. INTRODUCTION	1
1.1 Objectives	1
1.2 Self-Calibrating Thermocouple Description	1
1.3 Smart Sensor Background	3
2. THERMOCOUPLE THEORETICAL RELATIONS	5
2.1 The Thermoelectric Law	5
2.2 Transient Heat Transfer Analysis	5
2.3 Numerical Solution	10
2.3.1 Explicit Solution	10
2.3.2 Implicit Solution	11
2.3.3 Iteration Methods	13
2.3.4 Phase Change Implementation	14
3. PHASE CHANGE PLATEAU ANALYSIS	15
3.1 Heating/Cooling Rate	16
3.2 Location of the Thermojunction	18
3.3 Heat Transfer Coefficient and Mass and Heat of Fusion of the Metal	18
3.4 Degree of Alloy of the Metal	18
3.5 Homogeneity of the Metal	22
3.6 Phase Change Determination Methods	23
3.6.1 Phase Change Detection	24
3.6.2 Initial Plateau Point Method	25
3.6.3 Intersection of Slopes Method	25
3.6.4 Average Plateau Temperature Method	25
4. EXPERIMENTAL PROCEDURES AND RESULTS	27
4.1 Fast Ramp Test	27
4.2 Data Acquisition System for Drift Tests and Algorithm Testing	27
4.3 Drift Tests	28
4.3.1 Temperature Drift Tests	30
4.3.2 Phase Change Drift Tests	30
4.4 Physical Evaluations	33
4.4.1 X Ray	33
4.4.2 Metallography	33
4.4.3 Energy Dispersive Spectrometer	35

CONTENTS (continued)

Page

4.5 Simulation Program Verification 35
4.6 System Integration 42

5. CONCLUSIONS AND RECOMMENDATIONS 45

ACKNOWLEDGMENTS 46

REFERENCES 47

SELECTED BIBLIOGRAPHY 49

APPENDIX A. Simulation Program 51
APPENDIX B. Phase Change Detection Program 77

LIST OF FIGURES

Figure		Page
1.1	Cross-Sectional View of Self-Calibrating Thermocouple	2
1.2	Typical Process Heating Curve	2
1.3	Possible Smart Temperature Measurement System Configurations	4
2.1	Nodal Structure of Self-Calibrating Thermocouple	6
3.1	Typical Inverse Cooling Rate Curve	16
3.2	Summary of Cited Melting and Freezing Detection Points	17
3.3	Simulated Effect of Heating Rate on Melting Curve	19
3.4	Simulated Effect of Thermojunction Node Location on Melting Curve	20
3.5	Simulated Effect of Heat Transfer Coefficient on Melting Curve	21
3.6	Hypothetical Alloy Equilibrium Diagram and Heating Curve	23
3.7	Experimental Comparison of Melting and Freezing Curves of Zinc Thermocouple	24
3.8	Determination of Melting Point by the Intersection of Slopes Method	26
4.1	Fast Ramp Test Equipment Arrangement	28
4.2	Data Acquisition System Arrangement	29
4.3	Drift Tests Equipment Arrangement	30
4.4	Temperature Drift History	31
4.5	Tin Thermocouple Melting Point Drift History	32
4.6	Zinc Thermocouple Melting and Freezing Point Drift History	32
4.7	Gold Thermocouple Melting Point Drift History	33
4.8	X Ray Photograph of Self-Calibrating Thermocouples	34
4.9	Micrograph of Defective Self-Calibrating Thermocouple	36
4.10	Micrograph of Tested Type K with Tin Thermocouple	36
4.11	Micrograph of Tested Type S with Gold Thermocouple	37
4.12	Elemental Spectrum of Defective Type K with Zinc Thermoelement Wire	37
4.13	Elemental Spectrum of Tested Type K with Tin Thermocouple, Chromel Wire	38
4.14	Elemental Spectrum of Tested Type K with Tin Thermocouple, Alumel Wire	38
4.15	Elemental Spectrum of Tested Type S with Gold Thermocouple, Pt-10% Rh Wire	39
4.16	Elemental Spectrum of Tested Type S with Gold Thermocouple, Pt Wire	39
4.17	Comparison of Model Simulation with Experimental Run	40
A.1	Simplified Flowchart of Simulation Program	73
A.2	Typical Numeric Output of Simulation Program	74
A.3	Typical Graphical Output of Simulation Program	75
B.1	Simplified Flowchart of Phase Change Detection Program	83
B.2	Typical Output from Phase Change Detection Program	84

SYMBOLS

A	Heat transfer area, m^2
A_b	Thermocouple boundary surface area, m^2
A_s	Radiating surface area, m^2
$\mathbf{A}_e$	Explicit square system matrix containing thermal capacitances and thermal resistances between internal nodes
$\mathbf{A}_i$	Implicit square system matrix containing thermal capacitances and thermal resistances between internal nodes (this matrix contains zeroes along its diagonal)
$\mathbf{A}_{melt}$	Square melt matrix, used to determine when the total heat of fusion has been added to the melting nodes
b	Temperature axis intersect, °C
$\mathbf{b}$	Slope and intersect vector
$\mathbf{B}$	Diagonal matrix used in the implicit solution that contains D factors from equation 2.19
$\mathbf{C}$	Linear equation matrix used to find least squares fit of time-temperature data
C_p	Heat capacity, J/kg–°C
D	Denominator factor used in implicit solution, Eq. (2.19)
E	Energy variable that contains the heat input to the melting nodes, J
emf	Electromotive force (thermoelectric voltage), V
$\mathbf{F}$	Forcing function column vector containing capacitances and thermal resistances between boundary nodes and ambient forcing temperature
h	Convective heat transfer coefficient, W/m^2–°C
h_r	Radiative heat transfer coefficient, W/m^2–°C
ΔH_f	Heat of fusion, J/kg
i	Row index in nodal mesh
j	Column index in nodal mesh
$\mathbf{I}$	Identity matrix
k	Thermal conductivity, W/m–°C
k	Time increment index (i.e., k = present time increment, k–1 = past time increment, k+1 = future time increment)
m	Mass, kg; or slope of time-temperature line, °C/s
mp	Melting point, °C
q	Heat rate, W
Q	Energy generated per unit volume, W/m^3
$\mathbf{Q}$	Internal self-heat generation column vector that contains the QV terms for self-heated nodes
R	Thermal resistance, °C/W
$R_1(i,j)$	Thermal resistance between node i,j and node $i,j+1$, °C/W
$R_2(i,j)$	Thermal resistance between node i,j and node $i+1,j$, °C/W
$R_3(i,j)$	Thermal resistance between node i,j and node $i,j-1$, °C/W
$R_4(i,j)$	Thermal resistance between node i,j and node $i-1,j$, °C/W
S_i	Absolute Seebeck coefficient of thermoelement i (function of position and temperature), where i is normally either thermoelement A or B, V/°C
t	Time, s
Δt	Time increment, s

SYMBOLS (continued)

T	Temperature, °C (or K in radiation equations)
T	Column vector containing temperature of each node in the mesh
T_a	Ambient temperature, °C
T_b	Thermocouple boundary temperature, °C (or K in radiation equations)
T_s	Temperature of radiating surface, K
$T(x,y)$	Temperature of node x,y, °C
$T(x,y)^{k-1}$	Past temperature of node x,y, °C
$T(x,y)^{k}$	Present temperature of node x,y, °C
$T(x,y)^{k+1}$	Future temperature of node x,y, °C
ΔT	Temperature difference between nodes, °C
V	Volume of a self-heated node, m^3
x	Length or, in thermoelectric law, position along thermoelement path, m
Δx	Distance between centers of adjacent nodes, m
y	Linear equation vector

Greek Letters

σ	Stefan-Boltzmann constant, $5.669 * 10^{-8}$ $W/m^2\text{-}°C^4$
ϵ	Emissivity, dimensionless
ϵ_b	Emissivity of thermocouple boundary material, dimensionless
ϵ_s	Emissivity of radiating surface material, dimensionless
ρ	Density, kg/m^3

ABSTRACT

A smart temperature measurement system was developed that consists of a commercially available self-calibrating thermocouple (a thermocouple with a high-purity, low melting-point metal encapsulated near the thermojunction) connected to a data acquisition system with a specially designed algorithm capable of automatically detecting the calibration temperature of the self-calibrating thermocouple. A variety of self-calibrating thermocouples withstood long-term integrity testing, proving their worthiness as a front end to a smart temperature measurement system. A computer simulation program was written to explain the thermodynamic behavior of the system. Based on a literature review and simulation analysis, a method was developed to recognize which point on the melting or freezing plateau curve is the phase transition temperature of the encapsulated metal. The actual phase transition temperature is compared with the experimentally reported melting or freezing point of the encapsulated metal of the self-calibrating thermocouple to determine the magnitude of error in the thermocouple output.

1. INTRODUCTION

The work described in this report deals with the development of a smart temperature measurement system comprising a self-calibrating thermocouple and an algorithm running on a microprocessor-based data acquisition unit. The algorithm can automatically determine the calibration point of the self-calibrating thermocouple. This work is being carried out to advance the technology available in industrial smart sensors. Although the sensor used in this research is a thermocouple, a resistance temperature detector (RTD), thermistor, or other primary temperature-sensing device could be used in this application.

1.1 OBJECTIVES

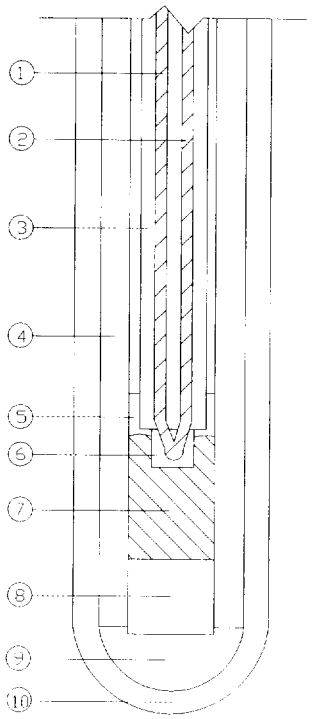
This work has three major objectives: (1) to test the commercially available self-calibrating thermocouple to see whether it is worthy as a front end to a smart temperature measurement system, (2) to design an algorithm to run on a microprocessor-based data acquisition unit that will automatically detect the calibration point of the self-calibrating thermocouple, and (3) to theoretically explain the behavior of the system.

1.2 SELF-CALIBRATING THERMOCOUPLE DESCRIPTION

The thermocouple used in this experiment is a special type, called a self-calibrating thermocouple, available from Isothermal Technology, Ltd., in England. It is special because it contains an encapsulated high-purity metal near the thermocouple junction, as shown in Fig. 1.1. If the thermocouple is heated through the melting point of the metal, the temperature record will indicate a plateau during the transition from solid to liquid state. A typical temperature record through the melting point is shown in Fig. 1.2. Of course, the melting point of the encapsulated metal is significantly lower than the melting points of the thermoelements and the ceramic materials of the thermocouple.

After melting is complete, the temperature record will recover to the original heating slope. By noting the temperature of the plateau point and comparing it with the known melting point of the metal, a single point calibration check can be performed.

The thermocouple junction is not in direct physical contact with the metal. A ceramic wall between the thermocouple junction and the encapsulated metal metallurgically isolates the two components. This is done so that the two components will not form an alloy with each other. An alloy could pose two detrimental threats. First, if the encapsulated metal alloyed significantly, its melting point could no longer be used as a reference because of the change in the alloy melting point from the pure metal melting point. Second, if the thermocouple junction alloyed significantly, its Seebeck coefficient would no longer be known. This



1. Thermoelement A
2. Thermoelement B
3. Two-bore ceramic tube
4. Ceramic housing
5. Ceramic paste plug
6. Ceramic insulator
7. Encapsulated metal
8. Ceramic cap
9. Air gap
10. Protection tube

Fig. 1.1 Cross-sectional view of self-calibrating thermocouple.

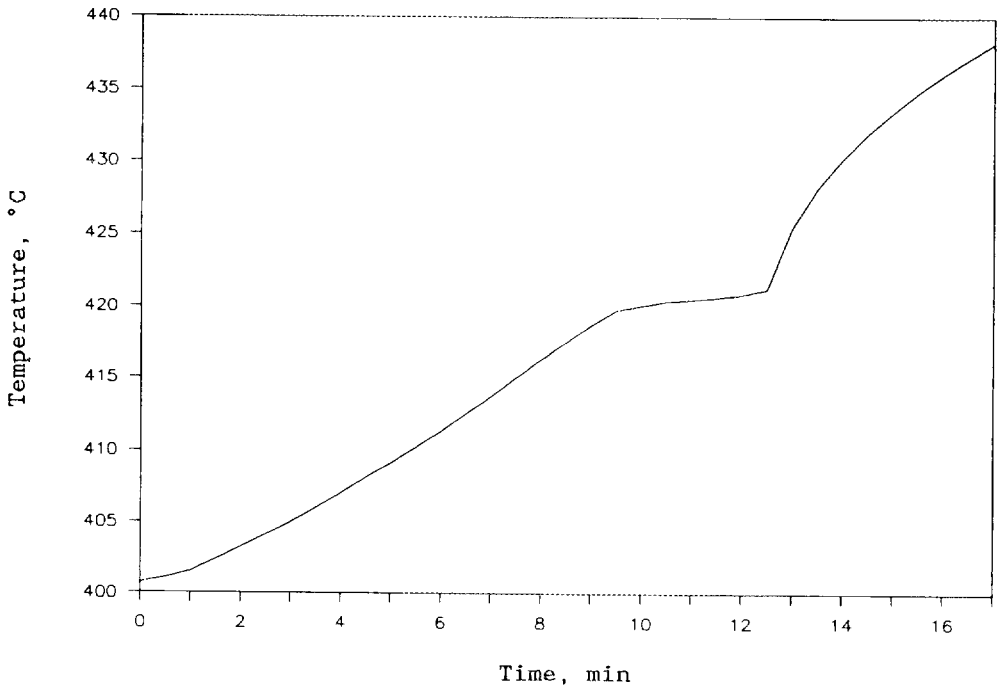


Fig. 1.2 Typical process heating curve.

could be a problem if the alloying proceeded far enough into the thermoelements such that the alloyed elements were not isothermal along the affected path. The thermocouple electromotive force (emf) vs temperature curve would no longer match the original. The problem of diffusion of metals at the thermojunction is not unique to this type of thermocouple. It has even been postulated that small errors can be caused by the simple interdiffusion between thermoelements at elevated temperatures.[1]

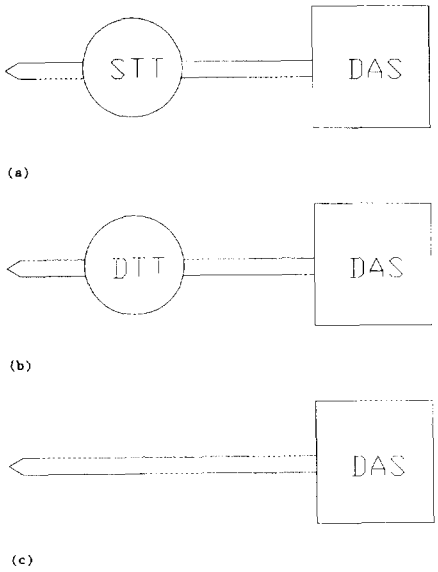
Although the thermocouple junction and the encapsulated metal are not in physical contact, they are thermally coupled because of their proximity. Therefore, the reading from the thermocouple will also indicate a plateau. The closeness of fit of the thermocouple record to that of the metal is discussed in Sect. 3.

The thermocouple is intended to be used in a process in which the normal operating temperature is somewhat above the melting point of the metal. This requirement defines the melting point criterion of the metal: an appropriate metal must be chosen for a given process. The calibration point can be checked as the process is ramping up to temperature, and if the metal does not supercool significantly, it can be checked when the process temperature is ramping down by using the freezing point of the metal.

1.3 SMART SENSOR BACKGROUND

A smart sensor is one in which a microprocessor is used to perform up to five levels of sophistication: (1) engineering unit conversion, (2) environmental compensation, (3) communications, (4) diagnostics, and (5) logic/actuation.[2] Smart sensors started to enter the measurement and control market when the microprocessor became rugged enough to withstand industrial environments and cheap enough to make the total cost of the system reasonable. The main feature of the proposed smart temperature measurement system is that it can determine whether the primary sensing element is operating properly. It is one of the first such units in which the primary sensor is located external from the microprocessor unit. Common functions of other smart sensor systems include automatic zeroing and scaling, tagging, and units conversion. However, almost always it is assumed that the primary sensing element is providing a proper signal. If the primary sensor is malfunctioning, there is no way for the conventional smart transmitter to be informed. The proposed smart temperature system would add the capability to detect whether the primary sensor, the thermocouple, is malfunctioning. In addition, every component in series with the temperature signal is checked (i.e., thermocouple extension wire, temperature transmitter if used, signal wire, analog to digital converter on data acquisition system, and software residing in the data acquisition system).

The smart thermocouple system can be configured in three ways, as depicted in Fig. 1.3. In the first configuration, Fig. 1.3a, the self-calibrating thermocouple is connected to a smart microprocessor-based field transmitter. The algorithm runs at the field unit. Calibration



Legend

DAS:	Data acquisition system
DTT:	Dumb temperature transmitter
STT:	Smart temperature transmitter

Fig. 1.3 Possible smart temperature measurement system configurations.

information may be available locally, depending on the design of the unit, and may also apprise the data acquisition system of the sensor diagnostics. A commercially available smart temperature transmitter can be modified to contain the algorithm.

In Fig. 1.3b, the algorithm runs in the data acquisition system. A conventional transmitter is placed between the thermocouple and the data acquisition system, transmitting a signal proportional to the temperature input.

In Fig. 1.3c, the thermocouple is connected directly to the data acquisition system using thermocouple extension wires. The algorithm runs in the data acquisition system in this configuration also.

2. THERMOCOUPLE THEORETICAL RELATIONS

2.1 THE THERMOELECTRIC LAW

One of the most fundamental laws relating thermoelectric voltage to temperature for a pair of thermoelements is given by the line integral in Eq. (2.1).[3]

$$\text{emf} = -\int^{\ell} S_i(x,T)\ \vec{\nabla}T\ \vec{dx} \quad . \tag{2.1}$$

This law illustrates that the emf produced by a thermocouple should not be treated like a battery at the thermojunction, as is commonly done in many applications, but rather, that the emf produced is developed along the entire path of the thermoelements. Any differences in the Seebeck coefficient along the path, provided that sufficient thermal gradient (∇T) exists between the points, will generate thermoelectric emf.

As mentioned before, an RTD or other primary sensor could possibly be used instead of a thermocouple in the smart temperature measurement system. However, the thermoelectric law demonstrates one of the reasons that it is necessary to restrict the use of the self-calibrating thermocouple to processes in which the calibration point is near the normal operating temperature. After a thermocouple has been in service for an extended period, especially at elevated temperatures, an inhomogeneous section of thermoelement wire usually develops. It is easy to show that, hypothetically, the inhomogeneous section may not be located within a substantial temperature gradient at the calibration temperature. Therefore, at this temperature, no error is introduced to the thermoelectric voltage because the thermal gradient at the inhomogeneous section is zero. However, if the temperature is increased significantly from the calibration temperature, hypothetically the thermal gradient could travel down the thermocouple and encompass the inhomogeneous section of thermoelements. The contribution of the Seebeck coefficient multiplied by the thermal gradient at the inhomogeneity will produce an error in the temperature reading that was not detected at the calibration temperature.

2.2 TRANSIENT HEAT TRANSFER ANALYSIS

Nodal heat transfer analysis can be performed on a thermocouple to study its transient behavior. The self-calibrating thermocouple can be divided into a mesh with thermal nodes, as shown in Fig. 2.1. Pertinent thermophysical properties of the self-calibrating thermocouple are listed in Table 2.1 with three encapsulated metal properties—those of tin, zinc, and gold. The heat transfer is analyzed as two-dimensional, axial and radial. Therefore, it is only necessary to look at half of the thermocouple when viewed radially. The temperature of the other half is identical to that shown. In fact, the other half uses the same node as that shown because the grid volume wraps completely around the

Notes:

1. Dimensions are given in millimeters.
2. Hatching represents encapsulated metal grids; Grids 25 and 26 are air gap; All other grids are alumina.
3. Dashed line represents actual shape of thermocouple sheath.
4. Dots represent node locations within mesh.
5. Rows and columns are indexed by i and j, respectively.

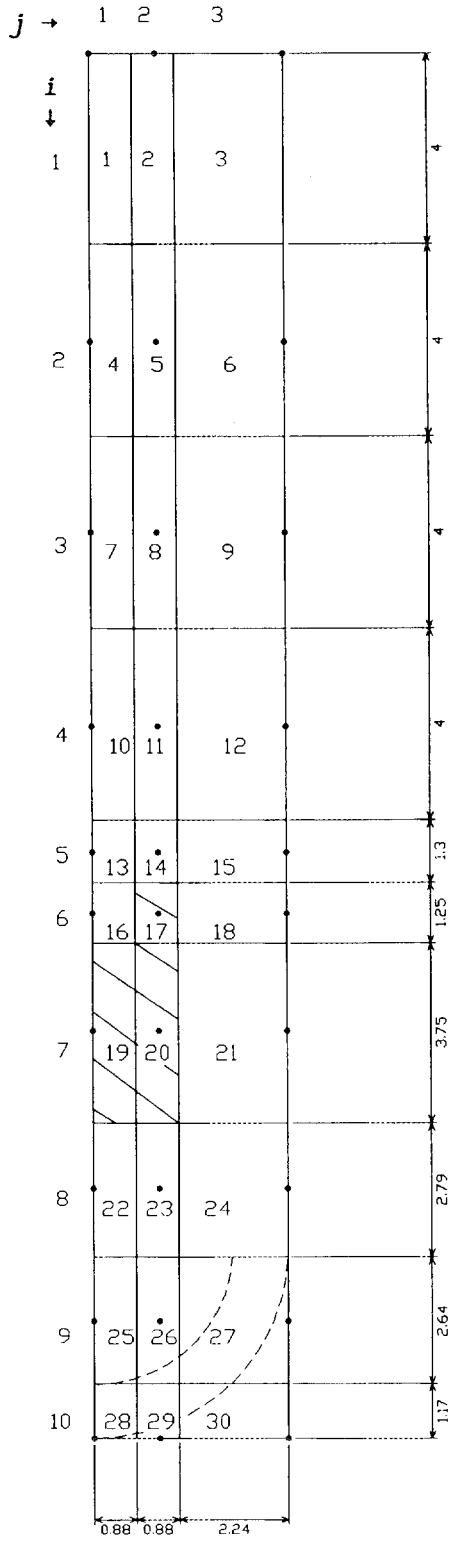


Fig. 2.1 Nodal structure of self-calibrating thermocouple.

Table 2.1 Self-calibrating thermocouple material properties.

Material	k (W/m-°C)	ρ (kg/m^3)	C_p (J/kg-°C)	ϵ	mp (°C)	ΔH_f (J/kg)
Alumina						
@232°C	19.7[a]	5,540[b]	1,046[b]	~0.3[c]	-	-
@420°C	13.2[a]	5,210[b]	1,150[b]	~0.3[c]	-	-
@1,064°C	5.8[a]	3,560[b]	1,260[b]	~0.3[c]	-	-
Air						
@232°C	0.0407[c]	0.698[c]	1,031[c]	-	-	-
@420°C	0.0519[c]	0.509[c]	1,074[c]	-	-	-
@1,064°C	0.0857[c]	0.264[c]	1,203[c]	-	-	-
Tin	57[c]	7,304[c]	263[d]	-	231.97[e]	60,627[d]
Zinc	96[f]	6,830[f]	457[f]	-	419.58[e]	113,003[f]
Gold	251[g]	19,300[h]	160[g]	-	1,064.43[e]	64,352[g]

[a]E. U. Schlunder, ed., *Heat Exchanger Design Handbook*, Vol. 5, Hemisphere Publishing Corp., New York, 1986.

[b]Y. S. Touloukian, ed., *Thermophysical Properties of High Temperature Solid Materials*, Vol. 4, MacMillan, New York, 1967.

[c]J. P. Holman, *Heat Transfer*, McGraw-Hill, New York, 1976.

[d]R. H. Perry and C. H. Chilton, 1973, *Chemical Engineers' Handbook*, Fifth Edition, McGraw-Hill, New York, 1973.

[e]Comite International des Poids et Mesures (International Committee of Weights and Measures), "International Practical Temperature Scale of 1968," *Metrologia* 5, 35-44 (1969).

[f]R. E. Kirk and D. F. Othmer, *Encyclopedia of Chemical Technology*, Third Edition, Vol. 24, Wiley, New York, 1984.

[g]R. C. Weast, ed., *Handbook of Chemistry and Physics*, 56th Edition, CRC Press, Inc., Cleveland, Ohio, 1975.

[h]H. S. Carslaw and J. C. Jaeger, *Conduction of Heat in Solids*, Clarendon Press, Oxford, 1959.

axis. Many nodes are used above the encapsulated metal because they are necessary to describe the abrupt temperature change after undergoing phase transition. At that point, the heat transfer to the newly melted (or frozen) metal is significantly axial. If fewer nodes are used, the heat transfer will be less because the boundary condition at the top of the nodal structure is such that no heat transfer occurs between the top of the structure and the unshown portion of the thermocouple. The shape of the curve, then, would not be true to that found experimentally because less heat would be axially transferred.

The nodes are arranged into rows and columns, which are indexed by the variables i and j respectively. Each node is also numbered starting from the upper left. The encapsulated metal is located at nodes 17, 19, and 20. The thermocouple junction appears at node 16 or node 13, based on whether or not the thermojunction is only partially enclosed by the encapsulated metal. (See Sect. 3.2 for a discussion of the effect of thermojunction location on the phase change plateau and Sect. 4.4.1 for a discussion of the actual thermojunction location.) The dashed curved line at the bottom indicates the actual shape of the thermocouple approximated by the rectangular structure. See Fig. 1.1 for a diagram of the actual shape of the thermocouple.

Each node is considered to have a uniform temperature. The solution would approach the exact solution as the dimensions of the grids approach zero. To study the transient heat transfer characteristics using nodal analysis, the generalized balance equation is used as shown in Eq. (2.2).

$$\text{Accumulation} = \text{Generation} + \text{Input} - \text{Output} \quad . \tag{2.2}$$

The accumulation of heat in a material produces a change in temperature, as shown in Eq. (2.3). The mC_p term is referred to as the capacitance term.

$$q = mC_p \, dT/dt \quad . \tag{2.3}$$

The conduction heat rate is given by Eq. (2.4). This equation can be used to calculate the conduction heat-rate input and output terms of the generalized balance equation.

$$q = kA \, dT/dx \quad . \tag{2.4}$$

In nodal analysis, the term dT/dx is approximated by the difference in temperature between nodes (ΔT) divided by the length between the nodes (Δx). To find the length, the exact location of the node within the mesh is required. This location is indicated by dots in Fig. 2.1. Usually the $kA/\Delta x$ term is grouped into a conductance term, or the reciprocal of resistance, so that the heat rate is equal to the thermal potential divided by the thermal resistance (analogous to Ohm's law in electricity), as shown in Eq. (2.5).

$$q = \Delta T/R \quad . \tag{2.5}$$

For a boundary node that experiences convective heat transfer with the ambient temperature, Eq. (2.6) holds.

$$q = hA(T_b - T_a) \quad . \tag{2.6}$$

This equation can also be expressed as in Eq. (2.5). The resistance term will then equal $1/(hA)$.

For a boundary node that experiences radiative heat transfer with its surroundings, Eq. (2.7) holds, assuming that the thermocouple is surrounded by a large cylindrical surface.[4]

$$q = \frac{\sigma A_b(T_b^4 - T_s^4)}{1/\epsilon_b + A_b/A_s(1/\epsilon_s - 1)} \quad . \tag{2.7}$$

It is convenient to convert this equation to one like Eq. (2.6). To do so, a radiative heat transfer coefficient is introduced:

$$q = h_r A_b(T_b - T_s) \quad . \tag{2.8}$$

Setting Eq. (2.8) equal to Eq. (2.7) and solving for h_r, we see that

$$h_r = \frac{\sigma(T_b^2 + T_s^2)(T_b + T_s)}{1/\epsilon_b + A_b/A_s(1/\epsilon_s - 1)} \quad . \tag{2.9}$$

Or as $A_b/A_s \to 0$,

$$h_r = \epsilon_b \sigma(T_b^2 + T_s^2)(T_b + T_s) \quad . \tag{2.10}$$

The radiative heat transfer equation can also be expressed as in Eq. (2.5). The value of R is $1/(h_r A)$. The radiative heat transfer coefficient, and consequently the resistance term, are severe functions of temperature. However, if solved by a digital computer, these terms may be calculated in a simulation algorithm at each time increment.

The generation term, as might be produced by a loop current step response test of a thermocouple,[5] is shown in Eq. (2.11):

$$q = QV \quad . \tag{2.11}$$

By using Eq. (2.3), (2.5), and (2.11), a heat balance in the form of Eq. (2.2) of a node relative to surrounding nodes in a rectangular mesh gives the following equation:

$$mC_p \frac{dT(i,j)}{dt} = QV + \left[\frac{T(i,j+1) - T(i,j)}{R_1(i,j)} + \frac{T(i+1,j) - T(i,j)}{R_2(i,j)} \right]$$

$$- \left[\frac{T(i,j) - T(i,j-1)}{R_3(i,j)} + \frac{T(i,j) - T(i-1,j)}{R_4(i,j)} \right] . \qquad (2.12)$$

Rearranging, this yields the following equation:

$$\frac{dT(i,j)}{dt} = \frac{1}{mC_p} \left[QV + \frac{T(i,j+1)}{R_1(i,j)} + \frac{T(i+1,j)}{R_2(i,j)} + \frac{T(i,j-1)}{R_3(i,j)} + \frac{T(i-1,j)}{R_4(i,j)} \right]$$

$$- \frac{T(i,j)}{mC_p} \left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)} \right] . \qquad (2.13)$$

2.3 NUMERICAL SOLUTION

A thermocouple simulation computer program was written by Natour.[6] This versatile program allows the operator to interactively enter the mesh arrangement of the nodes and the physical properties of the thermocouple materials. The program was modified to simulate the self-calibrating thermocouple. A radiative boundary condition and ramp forcing function were added to simulate experimental thermocouple data, the solution method was optimized, and a routine was added to reflect the performance of the thermocouple during phase change of the encapsulated metal. The radiative boundary condition was explained earlier in this section. Explanations of the other modifications follow. For further details of the program, refer to the source code listing given in Appendix A.

2.3.1 Explicit Solution

If the temperature and time variables of heat transfer Eq. (2.13) are made discrete using the backward difference approximation for the derivative of temperature, the following equation results:

$$\frac{T(i,j)^k - T(i,j)^{k-1}}{\Delta t} =$$

$$\frac{1}{mC_p} \left[QV + \frac{T(i,j+1)^{k-1}}{R_1(i,j)} + \frac{T(i+1,j)^{k-1}}{R_2(i,j)} + \frac{T(i,j-1)^{k-1}}{R_3(i,j)} + \frac{T(i-1,j)^{k-1}}{R_4(i,j)} \right]$$

$$- \frac{T(i,j)^{k-1}}{mC_p} \left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)} \right] . \qquad (2.14)$$

Multiplying each side of the equation by the time increment Δt and adding the $T(i,j)^{k-1}$ term to each side yields the following equation:

$$T(i,j)^{k} = \tag{2.15}$$

$$\frac{\Delta t}{mC_p}\left[QV + \frac{T(i,j+1)^{k-1}}{R_1(i,j)} + \frac{T(i+1,j)^{k-1}}{R_2(i,j)} + \frac{T(i,j-1)^{k-1}}{R_3(i,j)} + \frac{T(i-1,j)^{k-1}}{R_4(i,j)}\right]$$

$$+ T(i,j)^{k-1}\left[1 - \frac{\Delta t}{mC_p}\left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)}\right]\right] .$$

This is the explicit solution of the finite difference nodal heat transfer equation. When the remaining temperature nodes are added to the equations and a node is included for the ambient temperature, a matrix equation of the form shown below results.

$$\mathbf{T}^{k} = \mathbf{A}_e\ \mathbf{T}^{k-1} + \mathbf{Q} + \mathbf{F}\ T_a^{k-1} \quad . \tag{2.16}$$

When solved numerically, the explicit solution has the advantage that it can be solved recursively for each time increment. The new nodal temperature is a simple function of the past surrounding nodal temperatures and its own past temperature. The disadvantage is that to ensure stability of the numerical solution the time increment chosen must be very, very small (as low as 1 ms for cases in this work). The maximum allowed time increment is a function of the dimensions of the nodes and the thermal capacitances and resistances of the materials comprising the thermocouple. As the dimensions and the thermal capacitances and resistances decrease, the maximum allowed time increment decreases also. The stability of the numeric solution is determined by the coefficient of the $T(i,j)^{k-1}$ term in Eq. (2.15). This term occurs along the diagonal of the $\mathbf{A}_e$ system matrix. If the time increment becomes too large, the coefficient becomes negative. The effect of this phenomenon is that a higher $T(i,j)^{k-1}$ term will produce a lower $T(i,j)^{k}$ term. This will in turn produce a higher $T(i,j)^{k+1}$ term, and so on. The resulting numeric output will wildly swing negative and positive until the computer is finally overloaded.[7]

2.3.2 Implicit Solution

If the temperature and time variables of Eq. (2.13) are made discrete using the forward difference approximation for the derivative of temperature, the following equation results:

$$\frac{T(i,j)^{k+1} - T(i,j)^{k}}{\Delta t} =$$

$$\frac{1}{mC_p}\left[QV + \frac{T(i,j+1)^{k+1}}{R_1(i,j)} + \frac{T(i+1,j)^{k+1}}{R_2(i,j)} + \frac{T(i,j-1)^{k+1}}{R_3(i,j)} + \frac{T(i-1,j)^{k+1}}{R_4(i,j)}\right]$$

$$- \frac{T(i,j)^{k+1}}{mC_p}\left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)}\right] . \quad (2.17)$$

Rearranging Eq. (2.17) yields the following equation:

$$T(i,j)^{k+1}\left[1 + \frac{\Delta t}{mC_p}\left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)}\right]\right] =$$

$$\frac{\Delta t}{mC_p}\left[QV + \frac{T(i,j+1)^{k+1}}{R_1(i,j)} + \frac{T(i+1,j)^{k+1}}{R_2(i,j)} + \frac{T(i,j-1)^{k+1}}{R_3(i,j)} + \frac{T(i-1,j)^{k+1}}{R_4(i,j)}\right]$$

$$+ \; T(i,j)^{k} \quad . \quad (2.18)$$

If Eq. (2.18) is divided by factor D as shown in Eq. (2.19),

$$D = \left[1 + \frac{\Delta t}{mC_p}\left[\frac{1}{R_1(i,j)} + \frac{1}{R_2(i,j)} + \frac{1}{R_3(i,j)} + \frac{1}{R_4(i,j)}\right]\right], \quad (2.19)$$

then the equation can be expressed as shown in Eq. (2.20):[8]

$$T(i,j)^{k+1} =$$

$$\frac{\Delta t}{DmC_p}\left[QV + \frac{T(i,j+1)^{k+1}}{R_1(i,j)} + \frac{T(i+1,j)^{k+1}}{R_2(i,j)} + \frac{T(i,j-1)^{k+1}}{R_3(i,j)} + \frac{T(i-1,j)^{k+1}}{R_4(i,j)}\right]$$

$$+ \frac{T(i,j)^{k}}{D} \quad . \quad (2.20)$$

This is the implicit solution of the nodal heat transfer equation. When the remaining temperature nodes are added to the equations and a node is added for the ambient forcing temperature, a matrix equation of the form shown in Eq. (2.21) results.

$$\mathbf{T}^{k+1} = \mathbf{A}_i \; \mathbf{T}^{k+1} + \mathbf{B} \; \mathbf{T}^{k} + \mathbf{Q} + \mathbf{F} \; T_a^{k+1} \quad . \quad (2.21)$$

When solved numerically, the implicit solution has the advantage that it is stable for all values of the time increment chosen because, unlike

the explicit solution, it has no negative terms. (However, relatively small increments must still be chosen to ensure a solution close to actual.) The disadvantage is that the equation has to be solved by an iteration method or matrix inversion for each element in the temperature vector. Notice that Eq. (2.21) can be represented as follows:

$$\mathbf{T}^{k+1} = [\mathbf{I} - \mathbf{A}_i]^{-1} (\mathbf{B}\ \mathbf{T}^k + \mathbf{Q} + \mathbf{F}\ T_a^{k+1}) \quad . \tag{2.22}$$

Eq. (2.22) would produce a recursive solution similar to the explicit solution.

With an iteration technique, the number of computations at each time increment is larger than in the explicit method. However, because larger time increments can be chosen, the implicit method is usually obtained faster even with the iteration solution.

2.3.3 Iteration Methods

The matrices encountered in the solution of the heat transfer equations of the thermocouple are relatively sparse, except near the diagonal. As such, inversion solution methods such as that shown in Eq. (2.22) would prove to be somewhat tedious. Our attention is given to iteration methods because (1) the iterations tend to converge rapidly due to the sparse nature of the system matrices, and (2) codes can be placed in simulation routines such that only the required terms have to be calculated in the matrix multiplication. For example, when multiplying the $\mathbf{A}_i$ matrix by the $\mathbf{T}^{k+1}$ vector in Eq. (2.21), it should be remembered that only a maximum of five terms [those shown in Eq. (2.20)] need to be multiplied regardless of the size of the $\mathbf{A}_i$ matrix. However, the $[\mathbf{I} - \mathbf{A}_i]^{-1}$ matrix is not sparse. Every element has to be multiplied.

To solve Eq. (2.21), an approximation of the $\mathbf{T}^{k+1}$ vector is made by using the previous value of that vector. This vector is then used on the right-hand side of Eq. (2.21), and the equation is solved for the new value of $\mathbf{T}^{k+1}$. The old and new values of $\mathbf{T}^{k+1}$ are compared, and if their difference is more than a prespecified tolerance, then the process is repeated until the difference is within tolerance.

Two basic iteration methods that can be used to solve matrix equations are (1) the Gauss-Seidel iteration, and (2) the Jacobi iteration.[9] When using the Gauss-Seidel iteration method, each element in the iteration vector ($\mathbf{T}^{k+1}$) is computed, one element at time, from top to bottom. As each element is solved, its new value is used to solve the subsequent element iterations. This method is also called the "successive corrections method" because vector elements are replaced as soon as their new values have been computed. After all elements have been computed, the difference between the present and past iteration values is computed. If the difference is greater than a prespecified tolerance, the iteration continues.

In contrast to the Gauss-Seidel iteration, the Jacobi iteration method proceeds through the complete iteration vector each iteration, calculating new approximations for each element based entirely on the

past values of the iteration vector. When all elements of the vector are within tolerance, then the complete vector is updated with the new values. This method is also called the "simultaneous corrections method" because all elements are updated simultaneously.

The Jacobi iteration method is now used in the self-calibrating thermocouple simulation. The solution converges when the spectral radius of $\mathbf{A}_i$ is less than one, which is assured by the way the equations are generated.

2.3.4 Phase Change Implementation

When the temperature of an encapsulated metal node reaches the melting point, the temperature of the melting node is held steady during the melting duration.

As melting continues, an energy variable is monitored for the melting nodes to determine whether the heat of fusion has been provided. The energy term is calculated by using Eq. (2.4) to monitor heat into the melting nodes at each time increment.

$$E = \int q \, dt \quad . \tag{2.23}$$

The **A** matrix, whether in the explicit solution or the implicit solution, can be modified slightly to produce an $\mathbf{A}_{melt}$ matrix. Then the heat into the melting nodes can be calculated as follows:

$$E = \Sigma \, \mathbf{A}_{melt} \, \mathbf{T} \quad . \tag{2.24}$$

When melting is complete, the energy variable is equal to the total heat of fusion as shown in Eq. (2.25):

$$E = \Delta H_f \, m \quad . \tag{2.25}$$

Once the total heat of fusion has been added to all melting nodes, the melted node is liberated, capable of changing temperature again.

The same energy variable that is used in the explicit solution is monitored in the implicit solution also. When the heat of fusion has been added to all melting nodes, like the explicit solution, the node is free to change temperature once again.

Comparing simulation results with the actual experimental results of the self-calibrating thermocouple shows that all melting nodes must be held at their melting point temperature until the last melting node is melted rather than allowing melting-complete nodes to rise in temperature before the remaining melting nodes are through melting. This phenomenon is presumably caused by the small amount of metal encapsulated in the thermocouple. If a larger mass were present, it would be possible to detect the outer nodes melting before the inner nodes.

3. PHASE CHANGE PLATEAU ANALYSIS

Ideally, the reported phase change of a pure metal should produce a flat, horizontal plateau. In practice, this phenomenon is rarely realized. Which value of temperature on the melting and freezing curves should be considered the phase transition temperature? Over the years, a number of methods have been used to identify the melting point or freezing point of a metal based on its heating or cooling curve, respectively. Rosenhain[10] was one of the earliest to look at this problem. He studied recalescence curves of cooling metals using inverse rate curves. Incremental time per unit temperature on the abscissa is plotted against temperature on the ordinate. See Fig. 3.1 for an example. The resulting curve is vertical during most of the cooling duration, except during freezing, when it breaks out horizontally and then returns to a vertical line after the metal is completely frozen. This method was used because automatic temperature recorders were not available. It was easier to time the duration between equal temperature intervals than to take temperature readings at equal time intervals. Also, much information could be compressed into a small area of graph paper. Rosenhain interpreted the freezing point as the point at highest deflection in the horizontal direction that corresponds to the middle of the plateau in a time-temperature curve.

Weber[11] suggested that the melting point, as determined from a time-temperature heating curve, is the last point on the plateau before the rapid change that occurs after the metal is completely melted. The freezing point is the first point on the plateau of the cooling curve. No thermal analysis or experimentation was used to support this hypothesis.

Hume-Rothery, Christian, and Pearson[12] contend that the liquidus of the heating curve and the solidus of the cooling curve are stretched out farther than normal because equilibrium phase changes are never realized in practice. Therefore, the melting point is indicated by the first plateau point on a heating curve, and the freezing point is indicated by the first plateau point on a cooling curve.

McClaren[13] used the liquidus point on cooling curves to determine the freezing points and purity of metals to a high degree of accuracy.

The National Institute of Standards and Technology (formerly called the U.S. National Bureau of Standards, or NBS) uses the liquidus point on cooling curves to calibrate temperature measuring devices.[14]

See Fig. 3.2 for a comparison of the various melting and freezing determination points cited.

The advantage of the recent tests by McClaren and the National Institute of Standards and Technology (NIST) is that the metal and furnace are allowed to reach equilibrium, and the temperature is changed very little over a long length of time. The intent of the self-calibrating

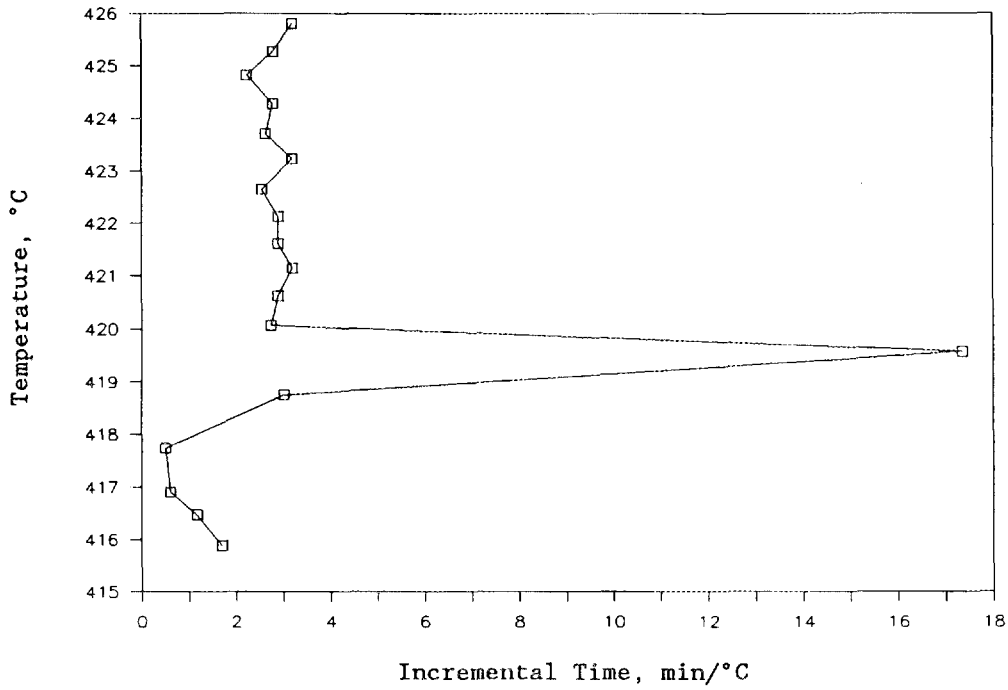


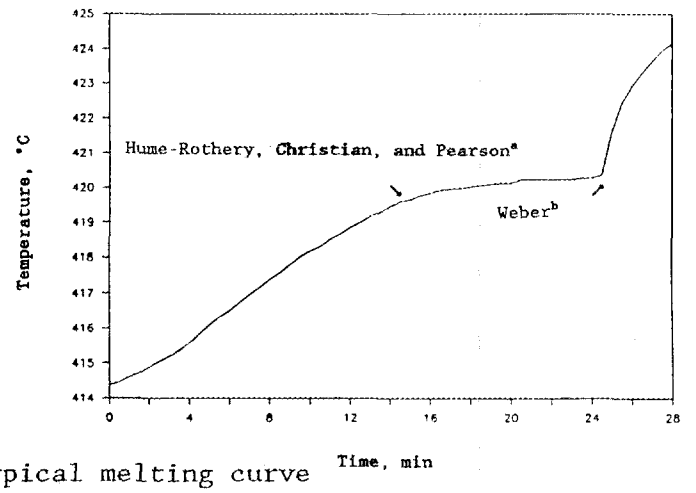
Fig. 3.1 Typical inverse cooling rate curve.

thermocouple is that it be used on an industrial process that does not reach equilibrium at the calibration point of the thermocouple. It is also not always possible to use the freezing points of metals in this thermocouple because, at the rates of cooling found in industrial processes, the metal may supercool significantly. Of the three metals tested in this work, only the freezing point of zinc could be used reliably. Gold and tin consistently supercool. The heat given off by the small mass of these metals during recalescence is not enough to bring the temperature of the metal back up to the normal freezing point; therefore, the supercooled metal cannot be used as a repeatable measure of the normal freezing point.

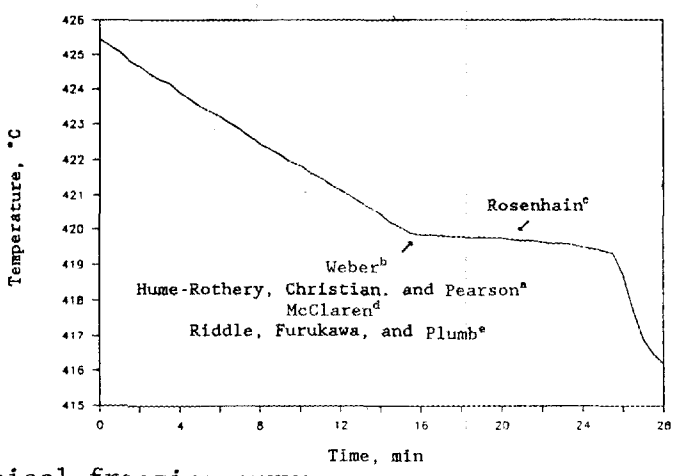
Several factors affect the plateau shape. Among these, listed somewhat in order of importance in the self-calibrating thermocouple, are (1) heating or cooling rate; (2) location of the thermojunction; (3) heat transfer coefficient, mass of metal, and heat of fusion of the metal; (4) degree of alloy of the metal; and (5) homogeneity of the metal.

3.1 HEATING/COOLING RATE

The self-calibrating thermocouple will probably never be used in a process in which the temperature through the phase change proceeds slowly (<1°C/h). Therefore, the heat transfer describing equations will always be transient, complicating the shape of the phase change plateau curve. As the heating rate is increased, the phase change plateau slope



(a) Typical melting curve



(b) Typical freezing curve

Fig. 3.2 Summary of cited melting and freezing detection points.

Sources: [a]W. Hume-Rothery, J. W. Christian, and W. B. Pearson, *Metallurgical Equilibrium Diagrams*, Unwin Brothers Ltd., London, 1952; [b]R. L. Weber, *Temperature Measurement and Control*, the Blakiston Company, Philadelphia, 1941; [c]W. Rosenhain, "Observations on Recalescence Curves," *Proc. Phys. Soc. Lond.* 21, 180-190 (1908); [d]E. H. McClaren, "The Freezing Points of High Purity Metals as Precision Temperature Standards," in *Temperature, Its Measurement and Control in Science and Industry*, Vol. 3, pp. 185-198, Reinhold Publishing Corp., New York, 1982; [e]J. L. Riddle, G. T. Furukawa, and H. H. Plumb, *Platinum Resistance Thermometry*, U.S. National Bureau of Standards Monogr. 126, Washington, D.C., 1973.

will increase, and the apparent melting range will increase also. Figure 3.3 shows the effect of varying the heating rate on simulated runs of the self-calibrating thermocouple while holding other conditions constant. Notice that the beginning of each plateau occurs at roughly the same temperature, while the end point does not. Also, we can see that at higher heating rates, the slope of the plateau is higher.

3.2 LOCATION OF THE THERMOJUNCTION

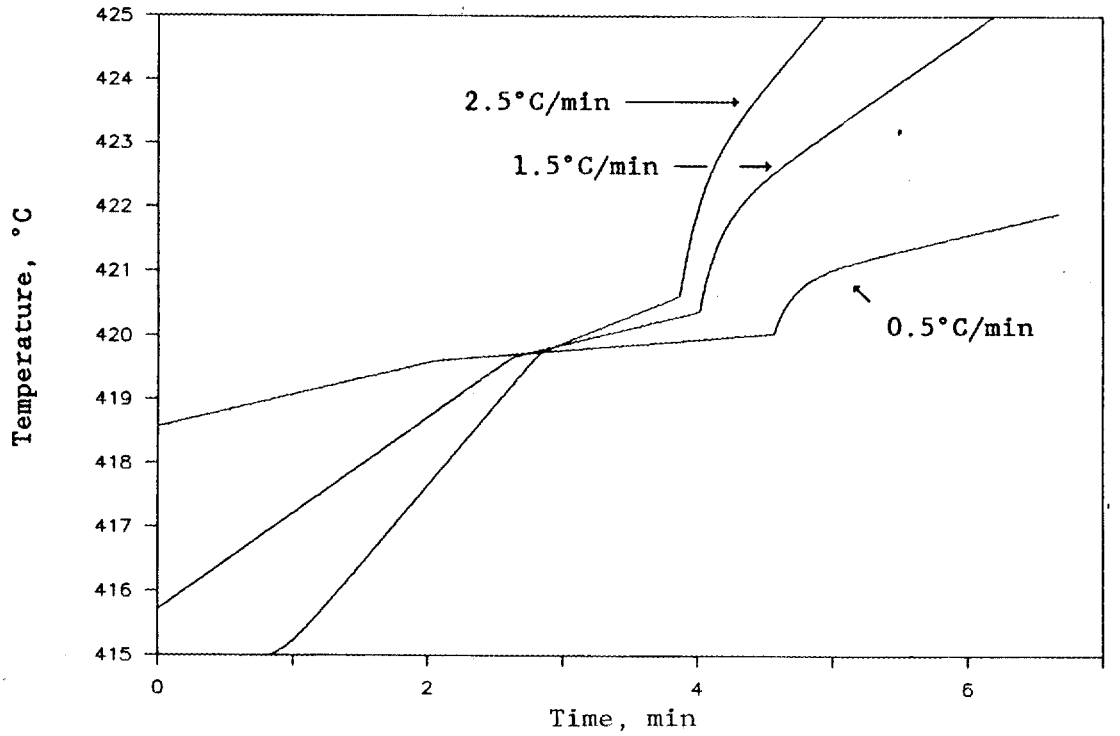
If the thermojunction is not in intimate contact with the metal, some degree of temperature lag will be associated with the temperature record. The reported temperature is not actually that of the metal, but that of the thermojunction, which is located within a ceramic sheath. The sheath extends only slightly into the metal. Therefore the thermojunction is not in intimate thermal contact with the metal. As the thermojunction node is moved away from the metal, it reproduces the plateau more and more poorly. Figure 3.4 shows the effect of moving the thermojunction node in and out of the metal nodes. The steepest curve results from using node 13 (see Fig. 2.1) as the sensing node. The intermediate curve results from using node 16 as the sensing node. The flattest curve results from using node 16 as the sensing node and setting node 14 as a metal node, effectively moving the sensing node farther inside the encapsulated metal. Surprisingly, the plateau of some of the tested self-calibrating thermocouples mirror the steepest curve in Fig. 3.4. Physical evaluations, discussed in Sect. 4, verify that the thermojunction is not completely surrounded by metal in these thermocouples. Note again that the beginning of the plateau occurs at roughly the same temperature.

3.3 HEAT TRANSFER COEFFICIENT AND MASS AND HEAT OF FUSION OF THE METAL

By increasing the heat transfer coefficient of the thermocouple, more heat is transferred to the metal in a shorter period, resulting in a shorter plateau period. The plateau period will also shorten with less metal mass and lower values of the heat of fusion of the metal. Figure 3.5 shows the effect of increasing the heat transfer coefficient from 500 to 5,000 to 50,000 W/m^2-°C while holding other conditions constant. Varying the mass or the heat of fusion of the metal would produce similar curves, except that the plateau slope would not change. Again we see that the initial plateau point occurs at nearly the same temperature for all three curves.

3.4 DEGREE OF ALLOY OF THE METAL

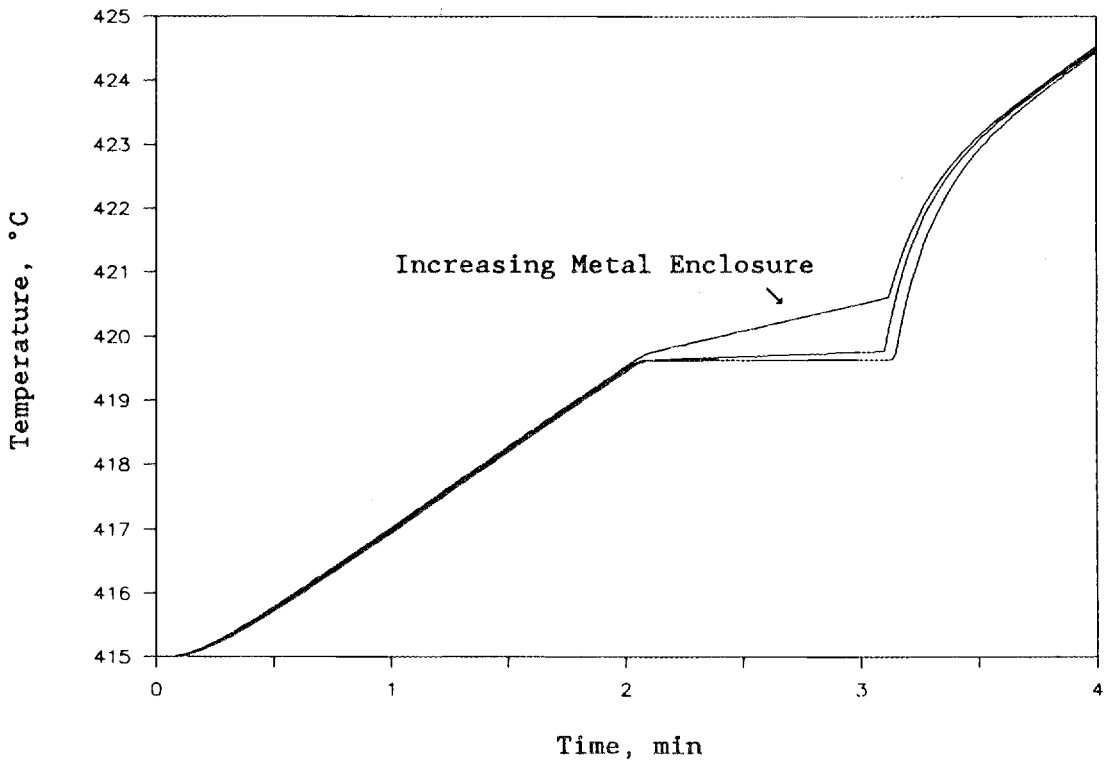
The time-temperature phase change curve of an alloy is never flat, even under perfect equilibrium conditions. The range of temperatures encountered during the phase change increases with higher degrees of alloying. Therefore, the melting or freezing temperature span is an indication of the purity of the metal. Because the metal encapsulated



Simulation Data

Model:	Fig. 2.1
Sensing node:	13
Encapsulated metal:	zinc
Physical properties:	Table 2.1
Forcing function:	ramp
Heating rate:	0.5, 1.5, and 2.5°C/min
Boundary condition:	convective
h:	5000 W/m^2-°C
Δt:	1 s
Solution method:	implicit
Iteration tolerance:	0.05°C

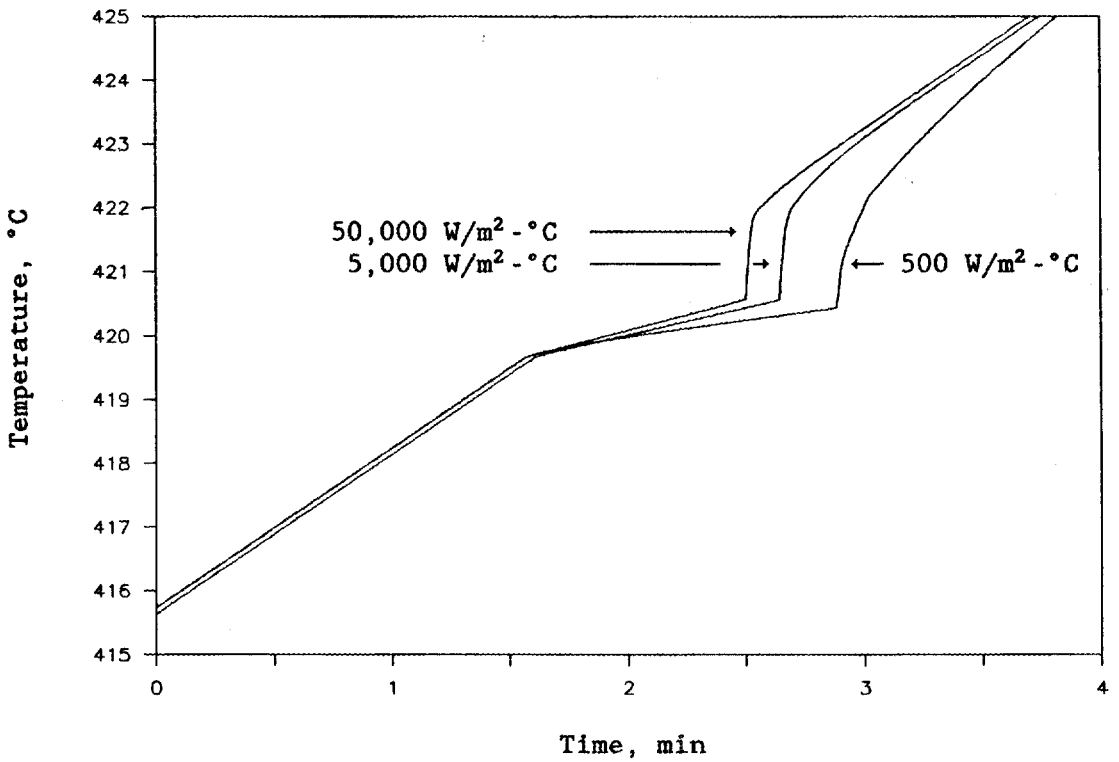
Fig. 3.3 Simulated effect of heating rate on melting curve.



Simulation Data

Model:	Fig. 2.1
Sensing node:	13, 16, and 16 with node 14 modeled as encapsulated metal
Encapsulated metal:	zinc
Physical properties:	Table 2.1
Forcing function:	ramp
Heating rate:	2.5°C/min
Boundary condition:	convective
h:	5000 W/m^2-°C
Δt:	1 s
Solution method:	implicit
Iteration tolerance:	0.05°C

Fig. 3.4 Simulated effect of thermojunction node location on melting curve.



Simulation Data

Model:	Fig. 2.1
Sensing node:	13
Encapsulated metal:	zinc
Physical properties:	Table 2.1
Forcing function:	ramp
Heating rate:	2.5°C/min
Boundary condition:	convective
h:	500; 5,000; and 50,000 W/m²-°C
Δt:	0.5 s
Solution method:	implicit
Iteration tolerance:	0.05°C

Fig. 3.5 Simulated effect of heat transfer coefficient on melting curve.

in the self-calibrating thermocouple is probably not 100% pure, the plateau will have some degree of slope. This effect would also become more pronounced if alloying of the metal and thermoelements occurred.

A composition diagram of a hypothetical metal alloyed with small concentrations of impurity is shown on the left side of Figure 3.6. (The abscissa represents zero only to a small fraction of impurity; i.e., the complete equilibrium diagram is not shown.) The melting point of the impurity is higher than that of the metal. As the temperature is raised along a constant composition path, the alloy begins to melt at the solidus point. The first liquid will have the composition shown at point L_1. As the alloy is further heated, the composition of the liquid will follow a path from L_1 to L_2 along the liquidus curve. The composition of the solid will follow a path from S_1 *to* S_2 along the solidus curve. When the metal is completely melted, the composition of the metal mixture will again be that of the original. The graph on the right indicates the time-temperature relationship that is observed during the melting of the alloy. Note that parameters discussed earlier in this section are ignored, and the nonzero slope of the plateau is caused totally by the alloy effect. However, unless extreme alloying occurs, this effect is negligible compared with the first effects. It is possible that if the point used to determine the melting point is the initial plateau point and the alloy melting point has increased from the original, the reported melting point will be close to that of the original metal. Note that the system depicted in Fig. 3.6 represents an alloy with increased melting point, which is plausible because the surrounding materials in the thermocouple have higher melting points than the encapsulated metal. However, a decreased melting point should not be ruled out.

It should be noted that these diagrams are called "equilibrium diagrams." The key word is "equilibrium." In practice, the liquid and solid phases of an alloy are never in equilibrium during a melting process. The phases of the encapsulated metal in the self-calibrating thermocouple are even less in equilibrium during its melting process. Using an equilibrium diagram to study the phase change of an alloy requires an intrinsic necessity: the metal must undergo its phase change in equilibrium, meaning that the concentration of the liquid *and* solid must be homogeneous at all times during the phase change. Even in the most carefully executed phase change, the concentration of the solid will hardly ever be homogeneous throughout the duration. The effect of the nonequilibrium phase change is that the liquidus will be stretched out on heating and the solidus will be stretched out on cooling, indicating that the solidus should be used on heating curves and the liquidus should be used on cooling curves to determine the melting and freezing points, respectively.[12]

3.5 HOMOGENEITY OF THE METAL

Impurities in the metal may be concentrated because of the zone melting effect.[15] Depending on whether the impurities are more soluble in

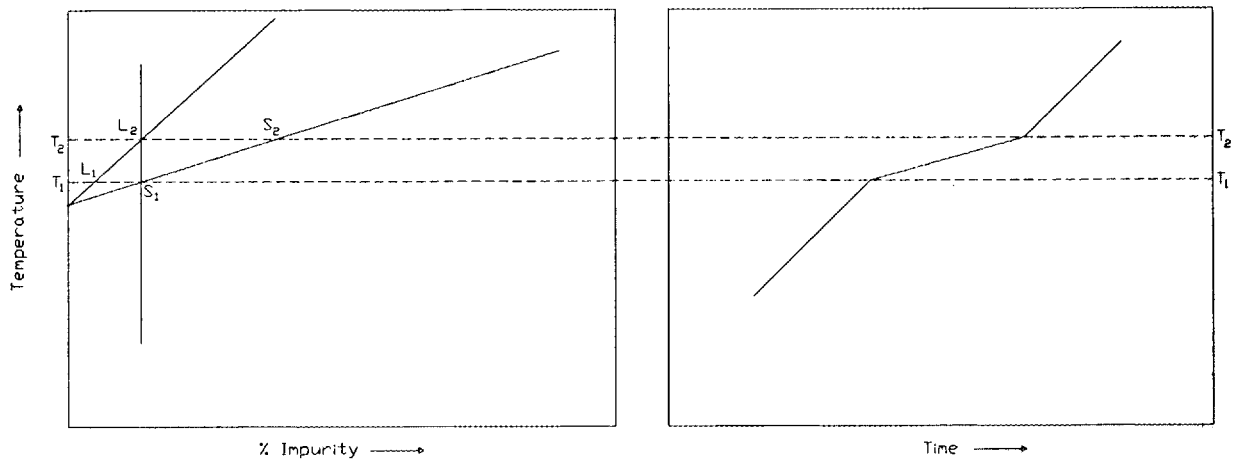


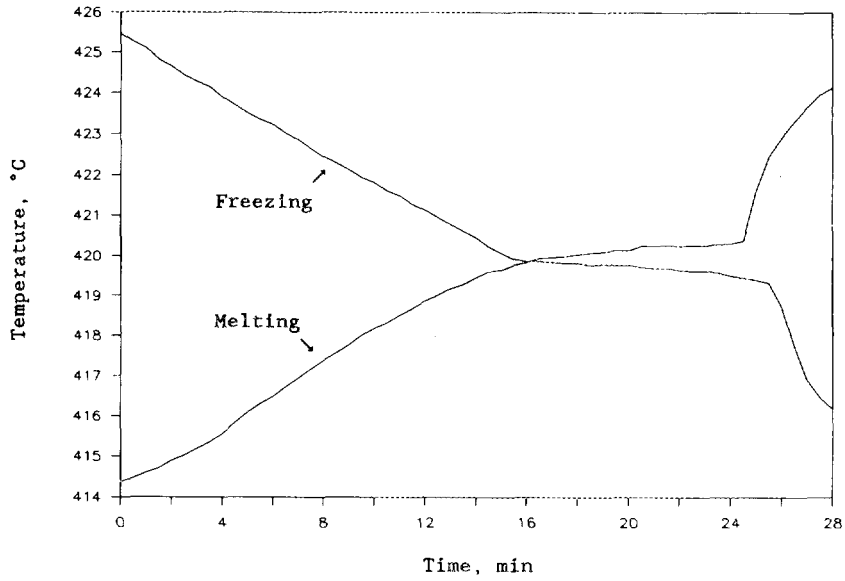
Fig. 3.6 Hypothetical alloy equilibrium diagram and heating curve.

liquid metal or solid metal, the impurities may concentrate near the thermoelements or near the outside wall surrounding the metal. If they accumulate near the thermoelements, the temperature during phase change of the material nearest the thermoelements will be different from the remainder of metal, thus adversely affecting the phase change plateau curve. This effect is difficult to monitor because it depends on the melting or freezing history of the metal. However, it is probably negligible compared with the first effects listed, considering the small amount of mass involved.

3.6 PHASE CHANGE DETERMINATION METHODS

Comparison test runs were made of the zinc thermocouple melting and freezing curves. As shown in Fig. 3.7, the tests show that the intersection of overlapped melting and freezing curves under similar conditions is the initial plateau point on each curve. In conclusion, given the factors discussed in Sects. 3.1 through 3.5 and the experimental data shown in Fig. 3.7, when using the self-calibrating thermocouple in industrial environments, the initial plateau points on both the heating and cooling curves will be interpreted as the phase transition temperature in this work. However, one caveat should be observed: Although the initial plateau point provides a relatively consistent number for the phase transition temperature, it is not mathematically equal to the phase transition temperature. In general, the reported melting point will be higher and the reported freezing point will be lower than the actual phase transition temperature.

Two methods can be used to detect the initial plateau point, and a third method can be used to report the phase transition temperature when it is not possible to find the initial plateau point. All three methods require that a phase change detected before analyzing the data for the melting or freezing point. The method used is explained in the next section, and a flowchart depicting the method is given in Appendix B.



Experimental Conditions

Thermocouple:	type K with zinc
Heating/cooling rate:	~0.4°C/min
Physical arrangement:	Fig. 4.3

Fig. 3.7 Experimental comparison of melting and freezing curves of zinc thermocouple.

3.6.1 Phase Change Detection

The temperature slope is stored in an array, and the second derivative is calculated using the backward finite difference method. The recovery from plateau temperature produces a discontinuity in the slope of the time-temperature curve, which produces a high finite difference second derivative. This process provides a good method to detect phase change. If the second derivative is not greater than some threshold value, say twice the maximum past second derivative, then no further processing is done. If the current second derivative is greater than twice the maximum past second derivative, then the slope array is evaluated to determine whether the immediate past slope was at or near zero. If the immediate past slope is not at or near zero, then this second derivative is saved as the new maximum and the next sample is taken as usual. However, if the immediate past slope is at or near zero, then a phase change event is declared. By checking for these two conditions, (1) high second derivative and (2) previous slope at or near zero, false phase change indications are avoided. Symptoms that could cause a false indication if both conditions were not checked are (1) noisy signal with corresponding high second derivative and (2) process temperature leveling out so that slope is zero. See Appendix B for the source code to the algorithm used in this work.

3.6.2 Initial Plateau Point Method

The first method involves looking at the finite-difference second derivative of temperature near the initial plateau point. For a heating curve, the point with the lowest second derivative is declared the initial plateau point. The advantage of this method is that only simple calculations are required, but the disadvantage is that to ensure that the correct temperature is chosen, the sampling rate has to be increased. However, with increased sampling rate comes increased signal noise, which erodes the validity of the second derivative data.

3.6.3 Intersection of Slopes Method

The second method involves finding the lowest second derivative, as explained previously, and constructing two least-squares-fit lines before and after the point of lowest second derivative. The intersection of these two points is declared the melting or freezing point. See Fig. 3.8 for an example of this method. The advantage of this method is that relatively slow sample rates are allowed, yielding relatively noise-free signals. The disadvantage is that much *a posteriori* computation is required.

3.6.4 Average Plateau Temperature Method

Both methods described previously have a common deficiency: The initial plateau point must be found. With slower heating or cooling rates, the initial plateau point becomes harder to find because it becomes more rounded off, as shown in Fig. 3.3. The method chosen to combat this problem is to use a third method for slow heating or cooling rates. With the third method, the average temperature of the plateau is reported as the melting or freezing point. The reported point will be somewhat above the melting point or somewhat below the freezing point, but with slower heating or cooling rates come lower apparent melting or freezing ranges. Hence, the reported point is not too much in error.

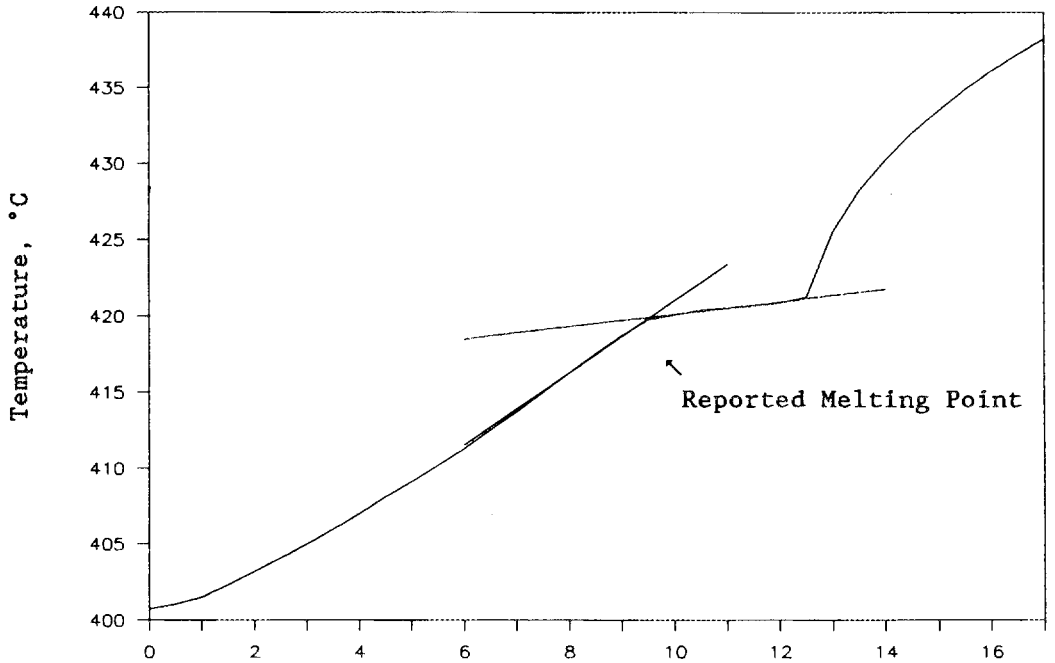


Fig. 3.8 Determination of melting point by the intersection of slopes method.

4. EXPERIMENTAL PROCEDURES AND RESULTS

A series of experimental tests and physical evaluations was performed on the self-calibrating thermocouple to determine whether it is an acceptable front end to a smart temperature measurement system. Three types of self-calibrating thermocouples were tested: (1) Type K with tin, (2) Type K with zinc, and (3) Type S with gold. In addition, the simulation program and phase change detection algorithm were tested.

4.1 FAST RAMP TEST

The automatic determination of the phase change of a metal depends on the duration of the phase change plateau. A suitable duration must be ensured so that the automatic algorithm will work properly. This requirement is especially applicable to a digitally sampled system. If the heating rate is too high and the sampling interval is too long, it would be possible for the phase change to occur between samples and not be noticed by the sampling system. Therefore, a test was performed to determine the phase change duration for increasing heating rates.

A tin self-calibrating thermocouple was inserted into an Inconel tube furnace (see Fig. 4.1). The thermocouple was connected to a recorder with built-in cold junction compensation. A variac autotransformer was connected to a primary transformer coil, as shown. A secondary coil wrapped around the primary coil was connected to the Inconel tube, and current from this secondary coil was passed through the Inconel tube. By varying the number of turns on the secondary coil, the voltage to the Inconel tube could be increased or decreased. Thus the heating rate of the tube could be varied by increasing the number of secondary coil turns. Power to the coils was applied instantaneously. No attempt was made to ramp the power. Several different heating ramp rate tests were done, and temperature records were made of each test to log the phase change plateau duration.

At a maximum ramp rate of 3.1°C/s, the duration of the phase change plateau was ~15 s. This duration was considered to be suitable for a majority of industrial applications. Because the simulation program appeared to provide an acceptable reflection of the behavior of the thermocouple, and because different heat transfer coefficients affect the plateau duration, the zinc and gold thermocouples were not tested in the fast ramp test. Rather, simulations can be performed to determine conservative numbers for the phase change duration with heating rates and heat transfer coefficients of interest.

4.2 DATA ACQUISITION SYSTEM FOR DRIFT TESTS AND ALGORITHM TESTING

The data acquisition system has been described by Cooper, Anderson, and Mossman.[16] It consists of a Hewlett-Packard 3495 scanner with low

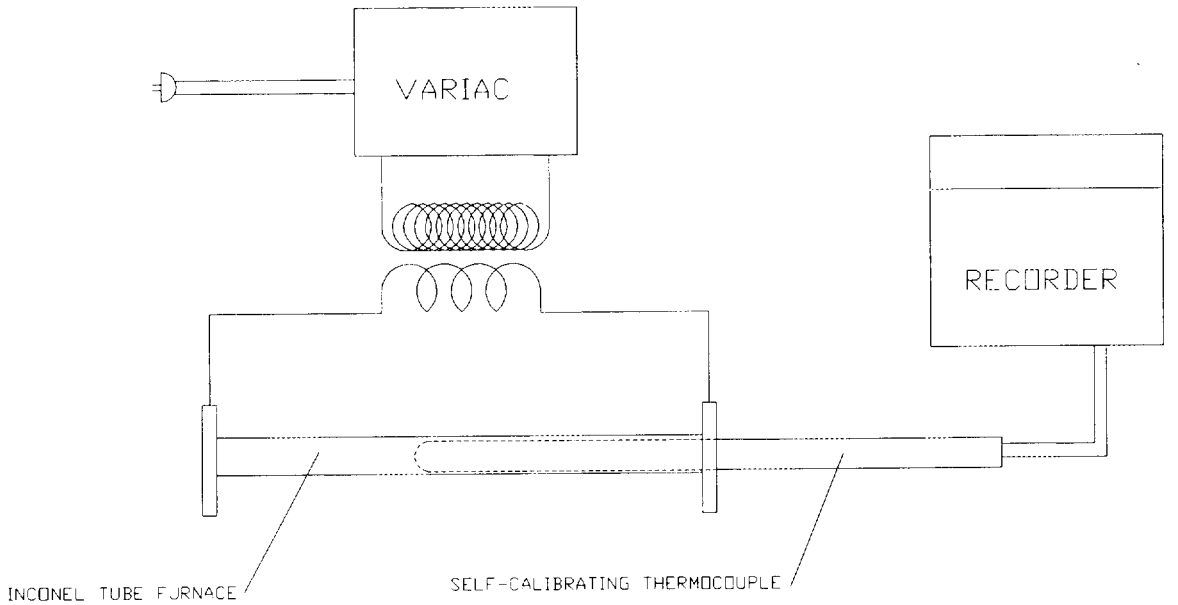


Fig. 4.1 Fast ramp test equipment arrangement.

thermal switches, a Hewlett-Packard 3455A digital multimeter, a Hewlett-Packard 59309A digital clock, and a Hewlett-Packard 9825A computer. Refer to the configuration drawing in Fig. 4.2. The scanner is used to multiplex several thermocouples to the digital multimeter, which passes data to the computer via an IEEE-488 bus. The digital clock, used to time daily recordings with the computer, is attached to the IEEE-488 bus.

The loop resistance of each thermocouple was calculated by directing a small amount of current through the loop and taking the average of the forward and reverse emf across the thermoelement leads to negate the effects of the Seebeck emf. This itself is a method of thermocouple self-diagnosis.[17] The temperature was calculated by taking the emf and using it to solve for temperature by using NBS Monograph 125 equations of emf as a function of temperature.[18] Because the equations are expressed as emf as a function of temperature, a Newton-Raphson algorithm was used to find values of temperature from the measured emf.

4.3 DRIFT TESTS

The original thermocouple obtained from Isothermal Technology, Ltd., showed severe signs of an encapsulated metal-to-thermojunction interaction. The metal and thermojunction components showed signs of alloying, and electrical conductivity was measured between them. Apparently the ceramic wall between the components had broken down or somehow become damaged. This interaction cannot be tolerated in an industrial-grade device that must perform reliably for thousands of hours. Therefore, it was decided that drift tests would be done on the thermocouples to characterize the interaction between the encapsulated

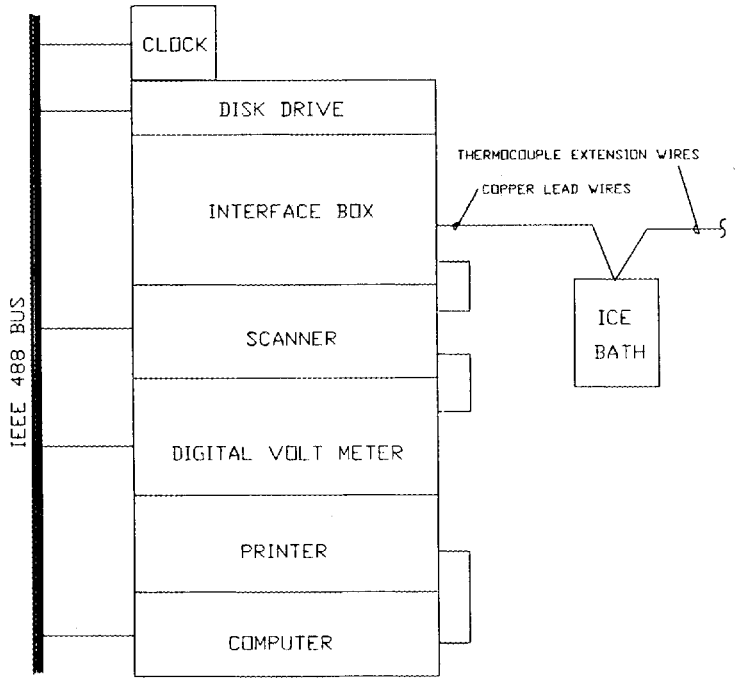


Fig. 4.2 Data acquisition system arrangement.

metal and the thermojunction. The tests were designed to report two characteristics of the thermocouples, (1) thermojunction emf drift and (2) phase change drift.

The test thermocouple and an end-closed ceramic tube were placed end to end, as in Fig. 4.3. A strip of platinum foil ~5 cm wide and 12 cm long was wrapped around the test thermocouple and ceramic tube to help keep the test section as isothermal as possible. The entire assembly was slid into a 40-cm-long tubular furnace, with the leads from the test thermocouple extending from one side and the open end of the ceramic tube extending from the other. The purpose of the ceramic tube was to provide a receptacle for a standard Type S thermocouple with calibration traceable to NIST. The standard thermocouple was slid into the ceramic tube only when readings were to be taken, thus preventing the prolonged high temperature of the furnace from aggravating a detrimental oxidation of the thermocouple junction.

The leads from all thermocouples were spliced with copper wires. The thermocouple-to-copper junction of about half the wires was placed in a Kaye electronic ice point reference cell; the other half was placed in an Omega ice point reference cell. The free ends of the copper wires were terminated with thermocouple plugs that mated with corresponding receptacles on the data acquisition system interface box.

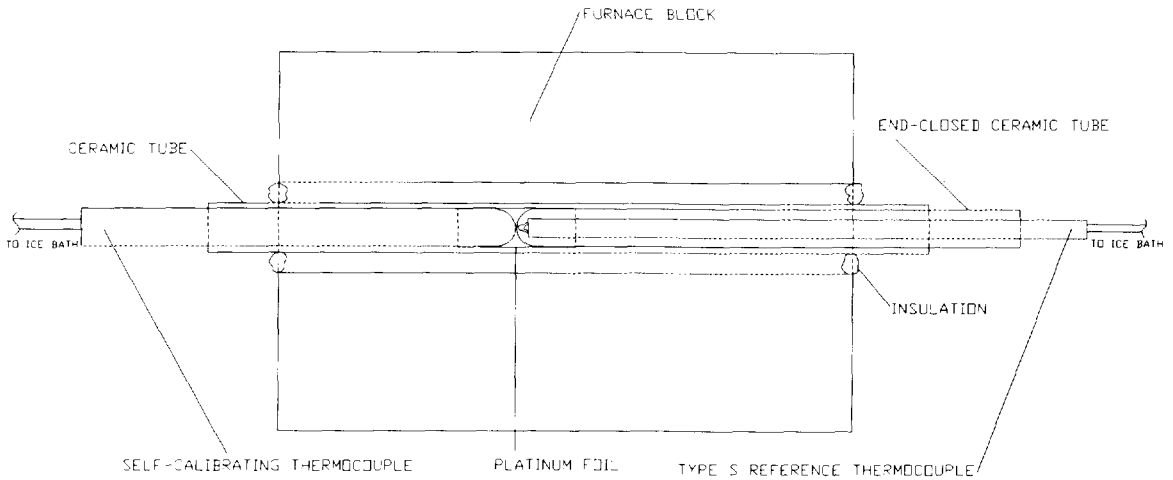


Fig. 4.3 Drift tests equipment arrangement.

4.3.1 Temperature Drift Tests

The set point of each furnace temperature controller was ~30°C above the melting point of the encapsulated metal, thus ensuring an aggressive environment for decalibration caused by the metal being in its liquid state. Daily recordings were made of the test thermocouple temperatures and loop resistances. About once a week, the standard thermocouple was slid into the ceramic tube. Its temperature was allowed to equilibrate with the test assembly, then temperature readings were taken of the test thermocouple and the standard thermocouple to record the thermocouple emf drift.

Figure 4.4 shows the history of the test thermocouples as compared with the Type S standard thermocouple. The first 40 d are not shown because a problem was found with the Omega electronic ice bath, yielding a cyclic ±2°C error in the data during that period. For the tin thermocouple, the difference between the test thermocouple reading and the standard reference thermocouple reading varied considerably throughout the test. Evidently, there was significant thermal resistance between the test thermocouple junction and the standard reference thermocouple junction. The zinc thermocouple held closest to the reference thermocouple of the three tested, no more than 2°C variation throughout the test. The gold thermocouple varied more, ~4°C. Because the test and reference thermocouples were not ideally thermally coupled, the results of the temperature drift test are inconclusive. The following section discusses the correlation between the temperature drift tests and the phase change drift tests.

4.3.2 Phase Change Drift Tests

At 1-week intervals, the temperature of the furnace was dropped below the freezing point of the encapsulated metal. A freezing point calibration test was performed on the zinc thermocouple. Because of

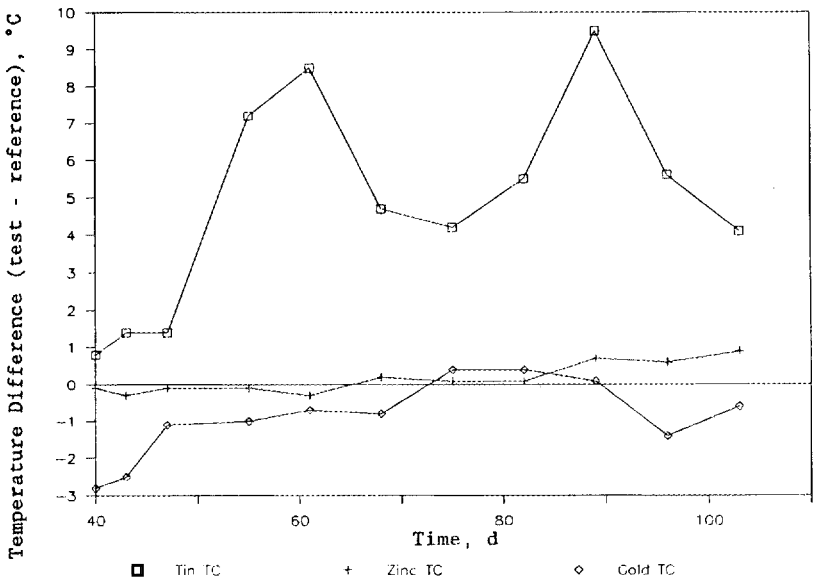


Fig. 4.4 Temperature drift history.

excessive supercooling, the freezing point of the tin and gold thermocouples could not be taken. The tin thermocouple supercooled as much as 25°C, and the gold thermocouple supercooled more than 100°C at times. (Even then, the thermocouple had to be vibrated before the gold would freeze.) When the thermocouple temperature had stabilized below the fusion temperature, the furnace temperature was raised above the fusion temperature again. A melting point calibration was performed on all three test thermocouples.

Figures 4.5 through 4.7 show the history of the test thermocouple phase change drift tests. The temperature and phase change drifts, as computed by a least-squares-type fit through the experimental data, are tabulated below.

Thermocouple	Temperature Drift (°C/h)	Phase Change Drift (°C/h)
Tin	0.0027	0.000081 (melt)
Zinc	0.00069	-0.00005 (melt)
		0.0002 (freeze)
Gold	0.0012	-0.00028 (melt)

The melting point drift of the tin thermocouple is negligible. The zinc thermocouple shows two different phase change drift trends based on whether the melting or freezing point trend is used. The freezing point trend appears to correlate somewhat with the temperature drift test for

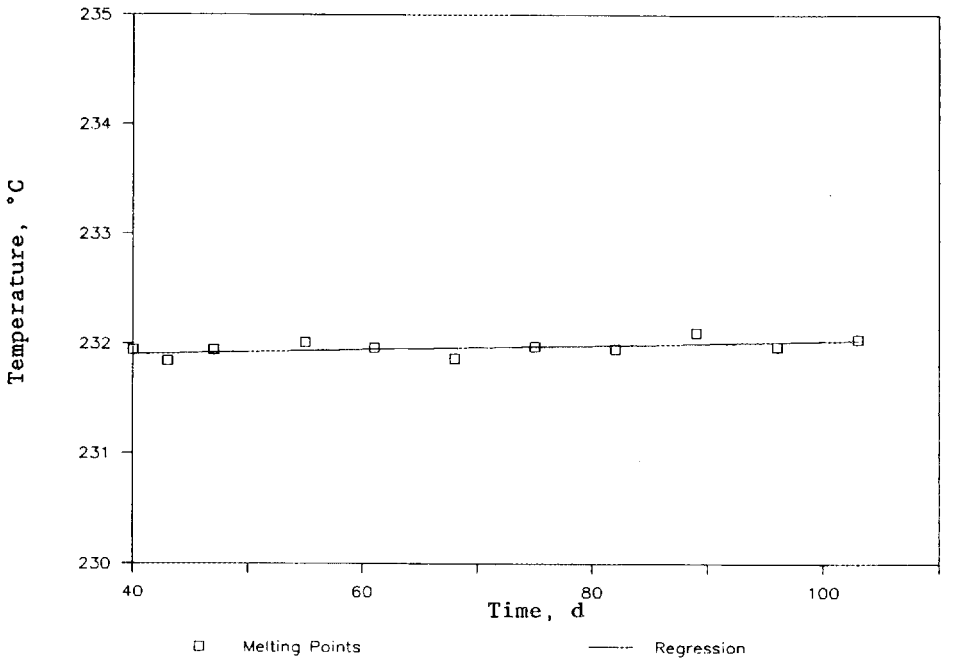


Fig. 4.5 Tin thermocouple melting point drift history.

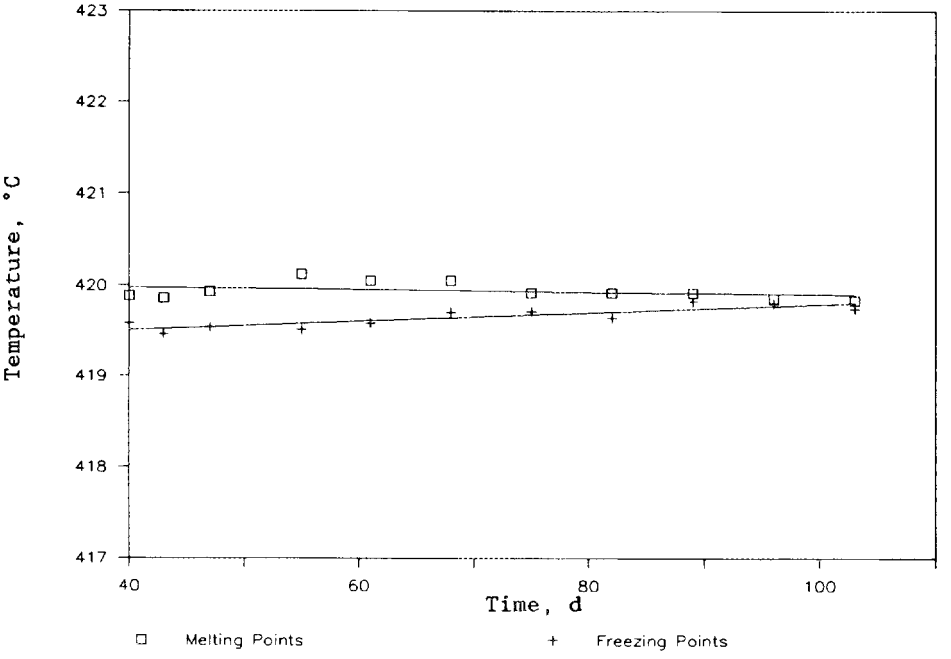


Fig. 4.6 Zinc thermocouple melting and freezing point drift history.

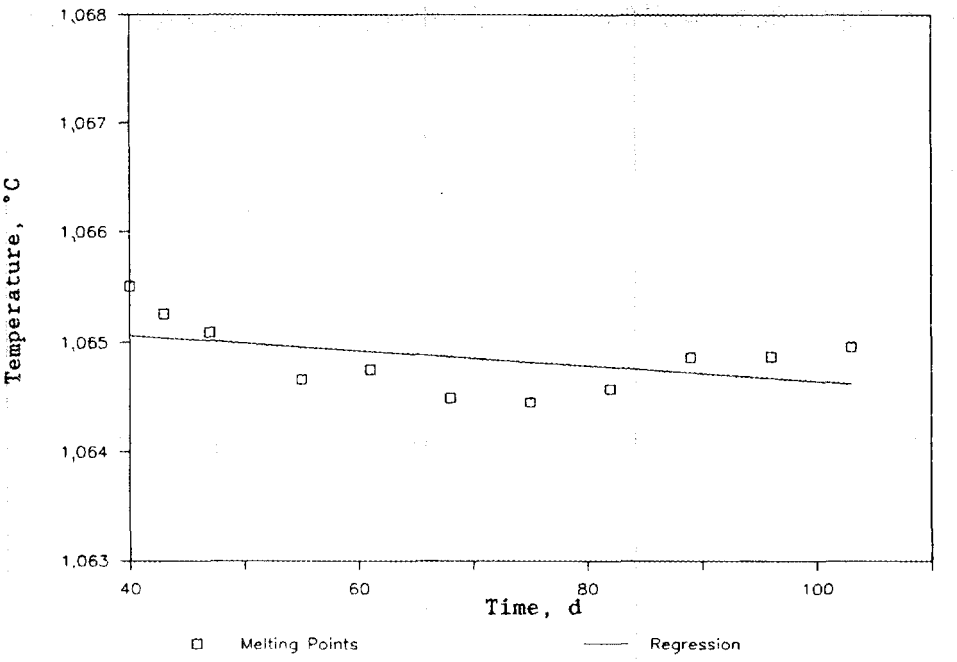


Fig. 4.7 Gold thermocouple melting point drift history.

zinc. In general, the freezing point is a more accurate determination than the melting point. The trend of the gold melting points shows only a slight downward drift. The melting points varied within roughly ±0.5°C but appeared to have a reverse correlation with the temperature drift tests.

4.4 PHYSICAL EVALUATIONS

Both nondestructive and destructive physical evaluations were performed on the test thermocouples.

4.4.1 X Ray

An X ray of six test thermocouples taken before any tests is shown in Fig. 4.8. Starting with thermocouple No. 1 at the left and going to the right, Nos. 1 and 2 are Type K with zinc, Nos. 3 and 4 are Type K with tin, and Nos. 5 and 6 are Type S with gold. Unfortunately, it is not possible to determine how far the thermoelements extend into the surroundings of the encapsulated metal. However, on thermocouples Nos. 3, 4, and 5, it is possible to note that the thermojunction is not hidden by the metal, therefore indicating that it is not ideally thermally coupled to the metal. It would be better if the metal totally enclosed the thermojunction.

4.4.2 Metallography

After the drift tests were complete, the junction ends of the test thermocouple assemblies were broken apart from the remainder of the

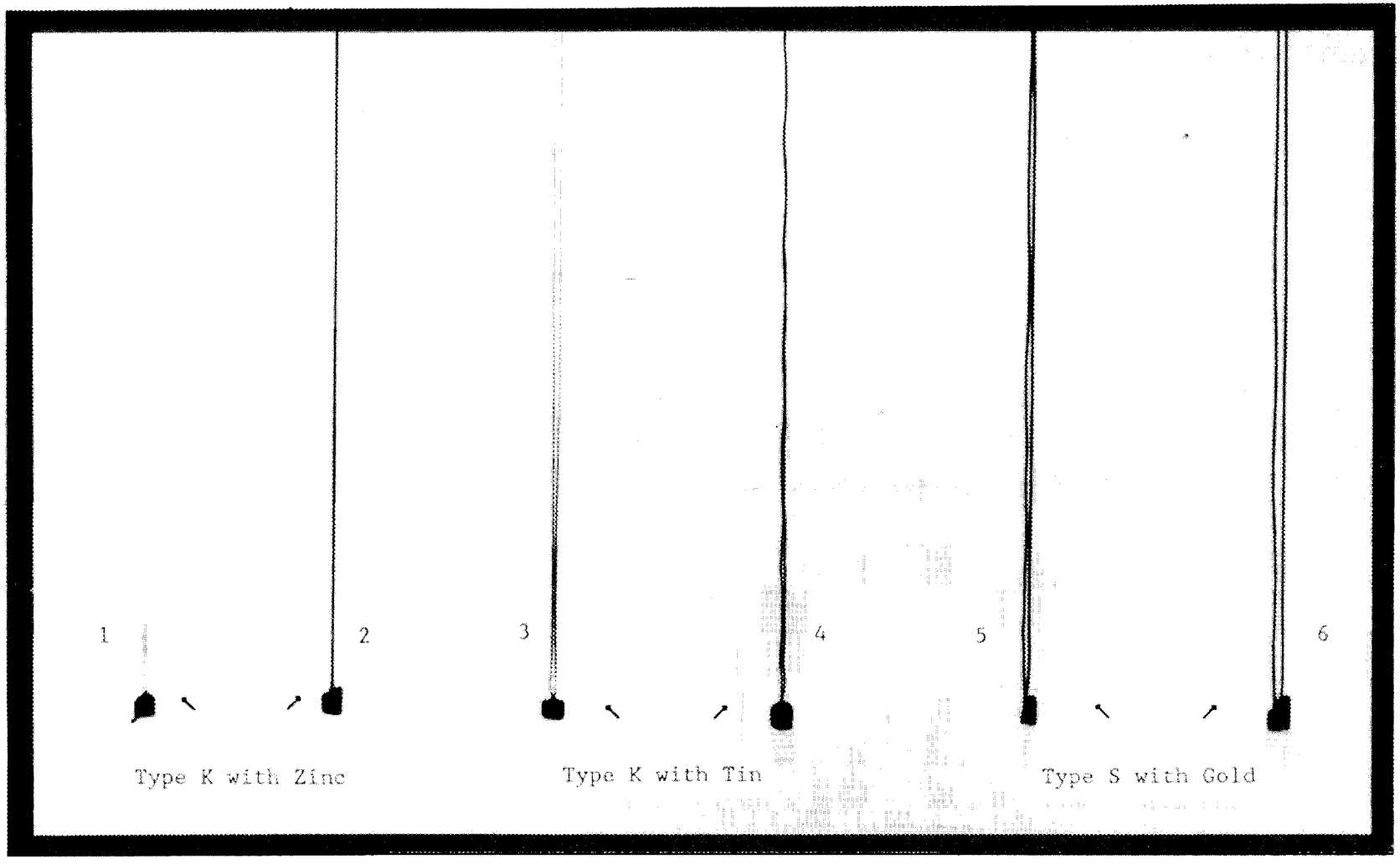


Fig. 4.8 X ray photograph of self-calibrating thermocouples.

assembly. As mentioned before, the first thermocouple obtained before testing began showed visual signs of a thermojunction-to-metal reaction. The thermoelement leads were embrittled. Electrical conductivity was measured between the thermoelements and the metal. A visual examination of the tested thermocouples did not indicate embrittled thermoelements with these specimens, as with the former defective specimen. It appeared that the encapsulated metal was well insulated from the thermoelements by the ceramic shroud. In fact, the zinc metal fell away from the ceramic shroud and thermoelements when it was disassembled. The test specimens were axially ground to the thermojunction point, polished, mounted, and micrographed to study the crystalline nature of the thermojunction-to-metal reaction, if there was any. Figure 4.9 shows a micrograph of the initial defective thermocouple. The ceramic insulator is cracked, and the surrounding metal, zinc, has infiltrated the thermoelements. Unfortunately, this thermocouple was not tested in a furnace to see whether the reported melting point had changed much. Figures 4.10 and 4.11 show micrographs of the Type K thermocouple with tin and the Type S thermocouple with gold, respectively. These micrographs do not indicate a metal-to-thermoelement interaction.

4.4.3 Energy Dispersive Spectrometer

The specimens were also evaluated using an energy-dispersive spectrometer on a scanning electron microscope to study the qualitative elemental composition of the metal-to-thermoelement interface. Figure 4.12 shows the spectrum of the initial defective thermocouple. The spectrum was taken of one of the thermoelements, but the elemental composition is largely zinc, indicating a severe interaction between the encapsulated zinc metal and the thermoelements. Figures 4.13 and 4.14 show similar scans of the tested Type K thermocouple with tin. Figure 4.13 shows the Chromel wire scan and Fig. 4.14 the Alumel wire scan. The composition of Chromel is 90% nickel and 10% chromium; the composition of Alumel is 94% nickel, 3% manganese, 2% aluminum, and 1% silicon.[19] The scan is absent of the surrounding material, tin.

Figures 4.15 and 4.16 show similar scans of the tested Type S thermocouple with gold. Figure 4.15 shows the Pt-10% Rh wire scan and Fig. 4.16 the platinum wire scan. These scans are absent of the surrounding metal, gold. It is expected, although not confirmed, that similar noninteraction would have been reported of the test Type K with zinc thermocouple because the ceramic insulator freely fell away from the metal.

4.5 SIMULATION PROGRAM VERIFICATION

The simulation program described in Sect. 2 was used to make most of the parametric studies in Sect. 3. The program appeared to respond appropriately to all the varying parameters. However, a comparison of experimental data vs simulation data is required to verify the program. Such a comparison is shown in Fig. 4.17. The experimental data were obtained from one of the zinc thermocouple drift tests described in Sect. 4 with a nominal heating rate of 2.5°C/min. The simulation was

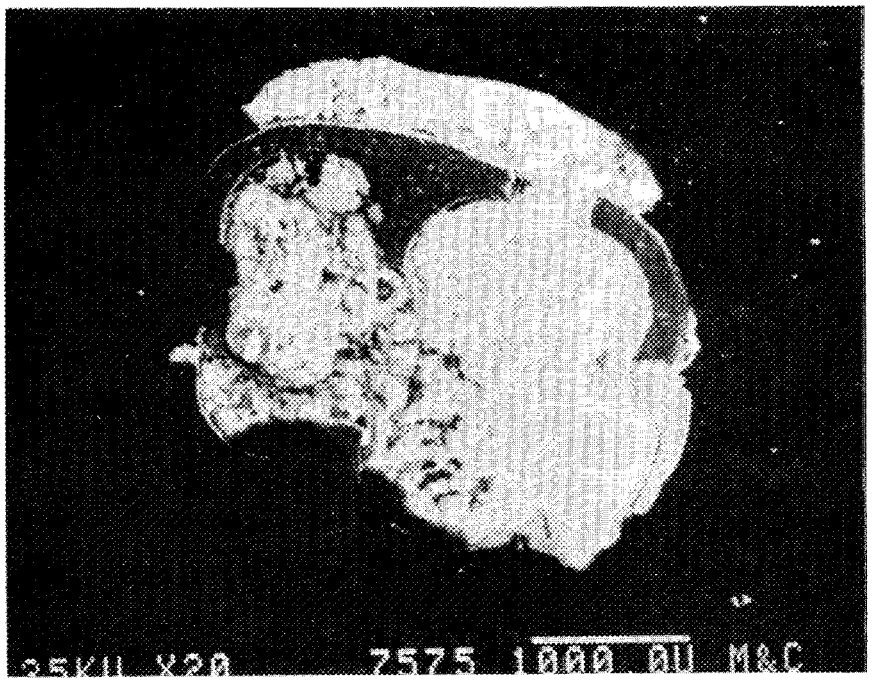


Fig. 4.9 Micrograph of defective self-calibrating thermocouple (20X).

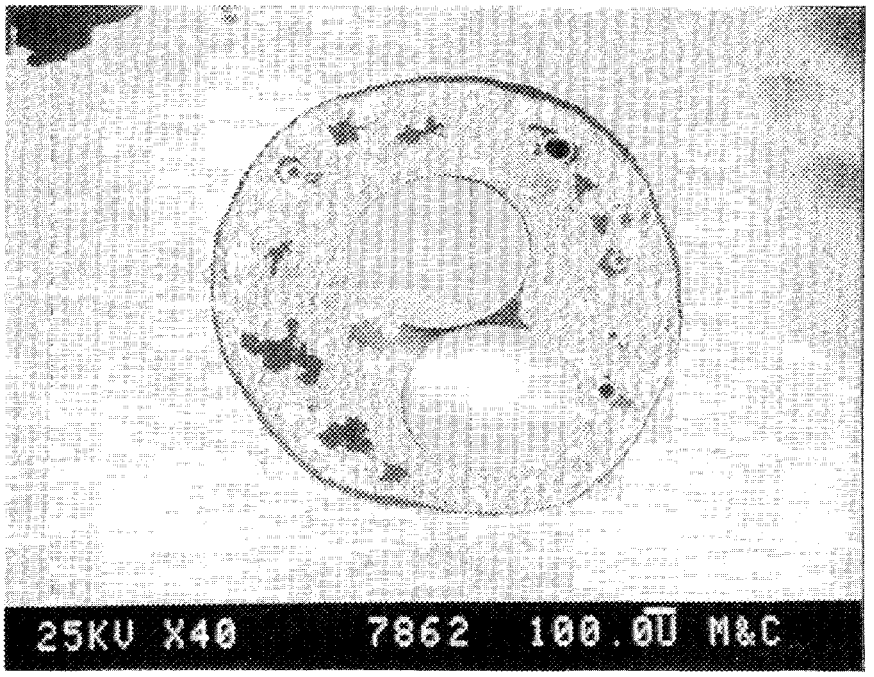


Fig. 4.10 Micrograph of tested Type K with tin thermocouple (40X).

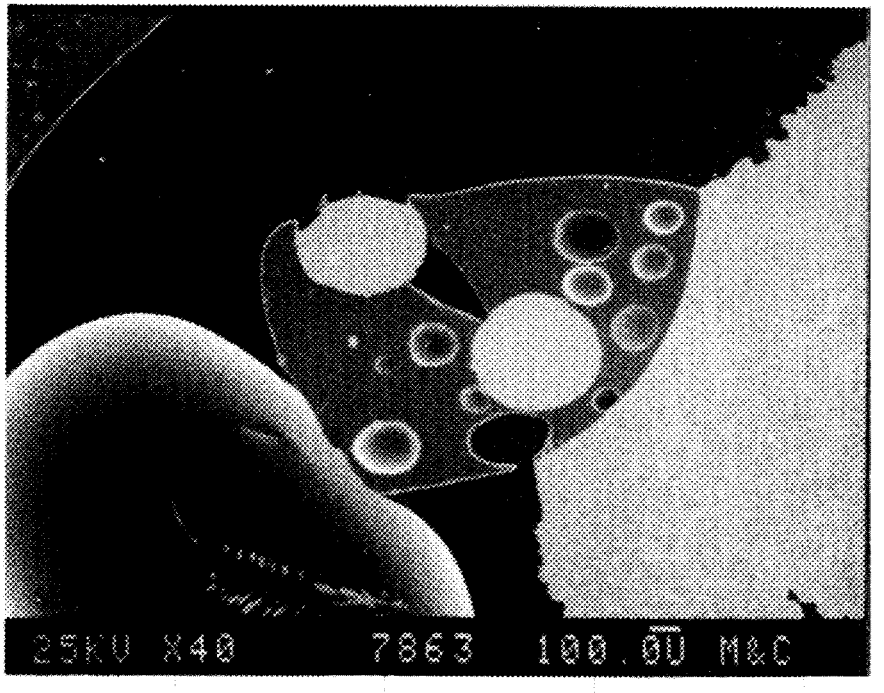


Fig. 4.11 Micrograph of tested Type S with gold thermocouple (40X).

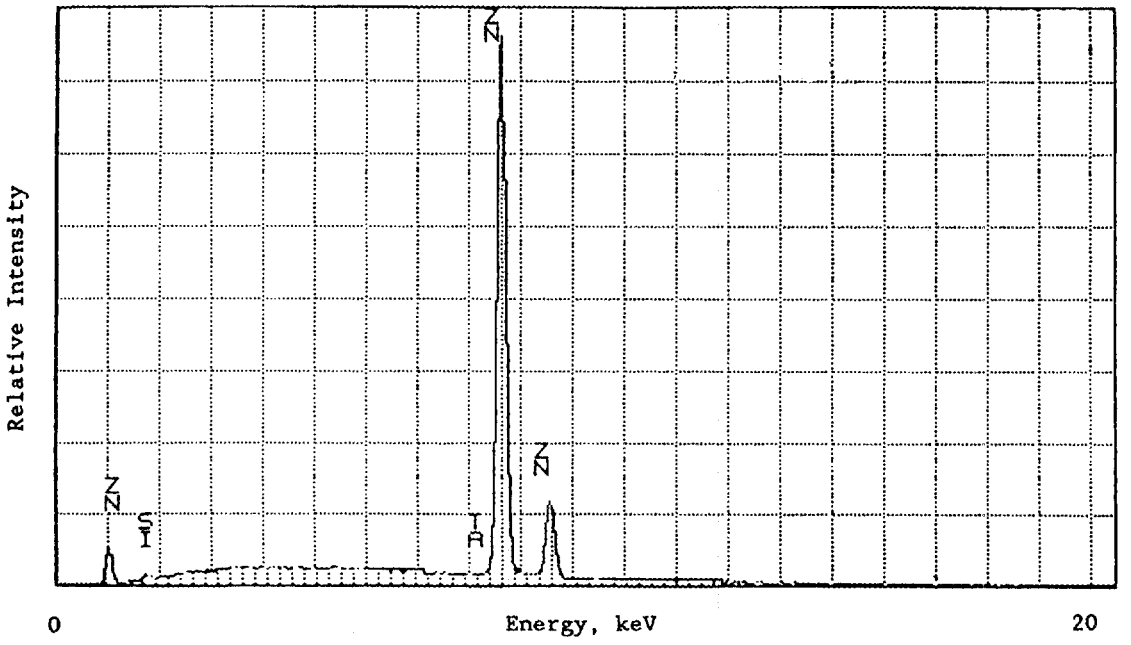


Fig. 4.12 Elemental spectrum of defective Type K with zinc thermoelement wire.

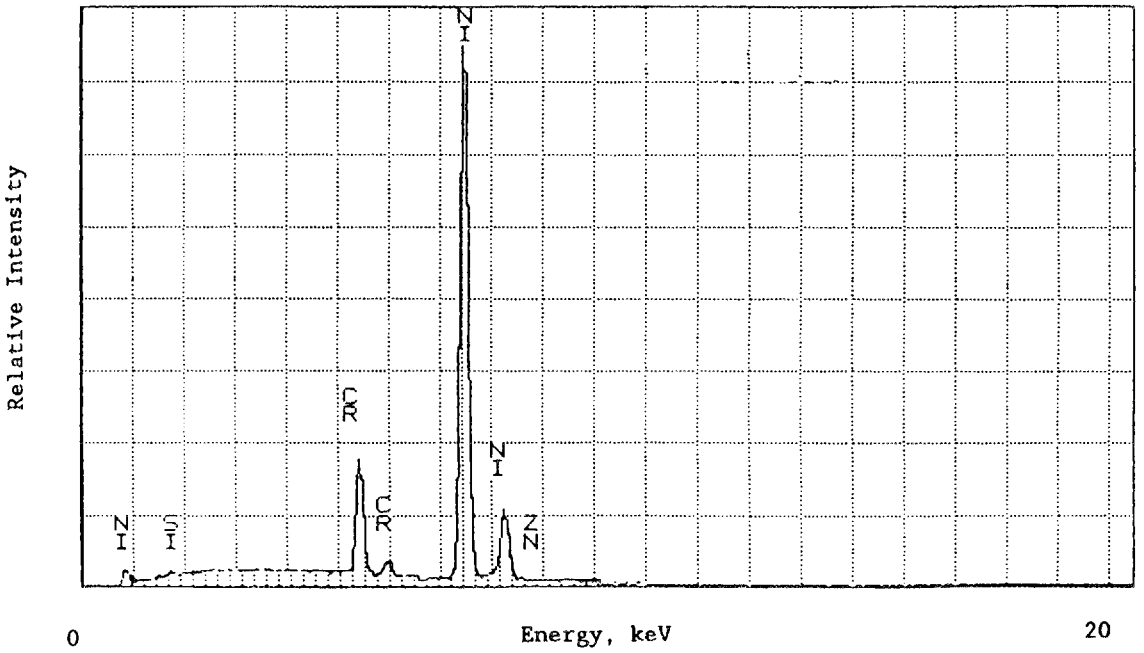


Fig. 4.13 Elemental spectrum of tested Type K with tin thermocouple, Chromel wire.

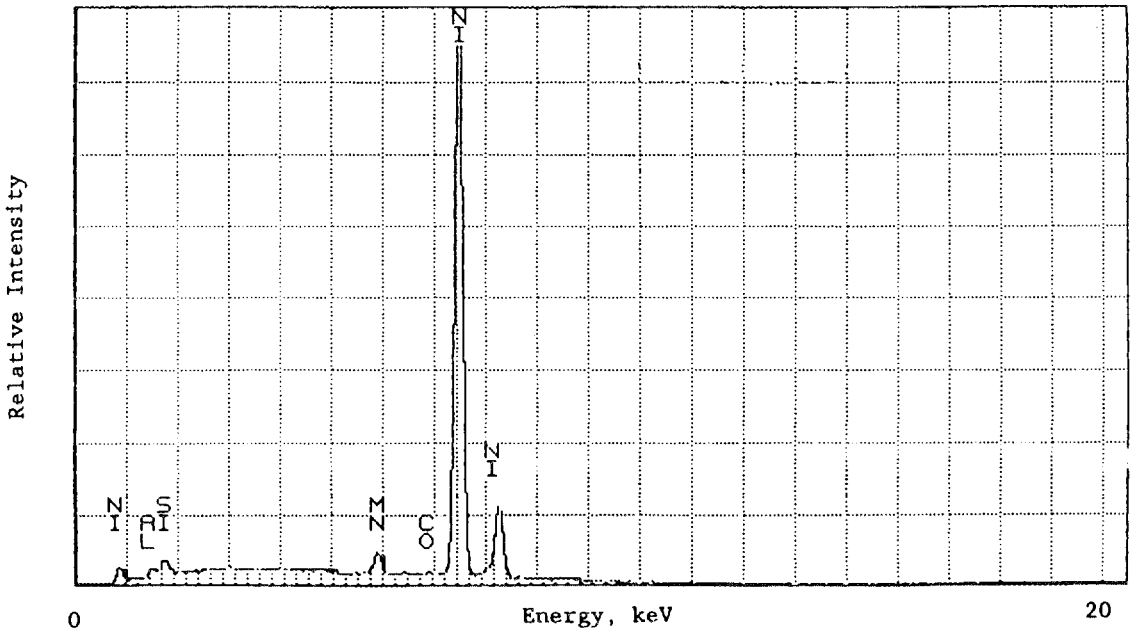


Fig. 4.14 Elemental spectrum of tested Type K with tin thermocouple, Alumel wire.

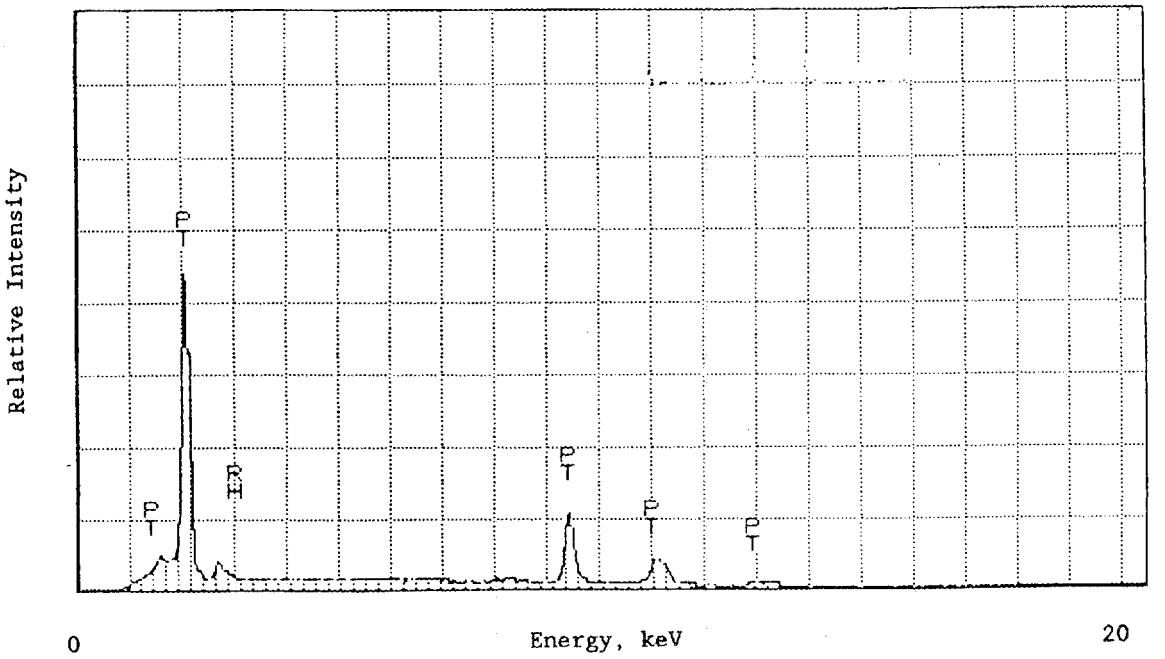


Fig. 4.15 Elemental spectrum of tested Type S with gold thermocouple, Pt-10% Rh wire.

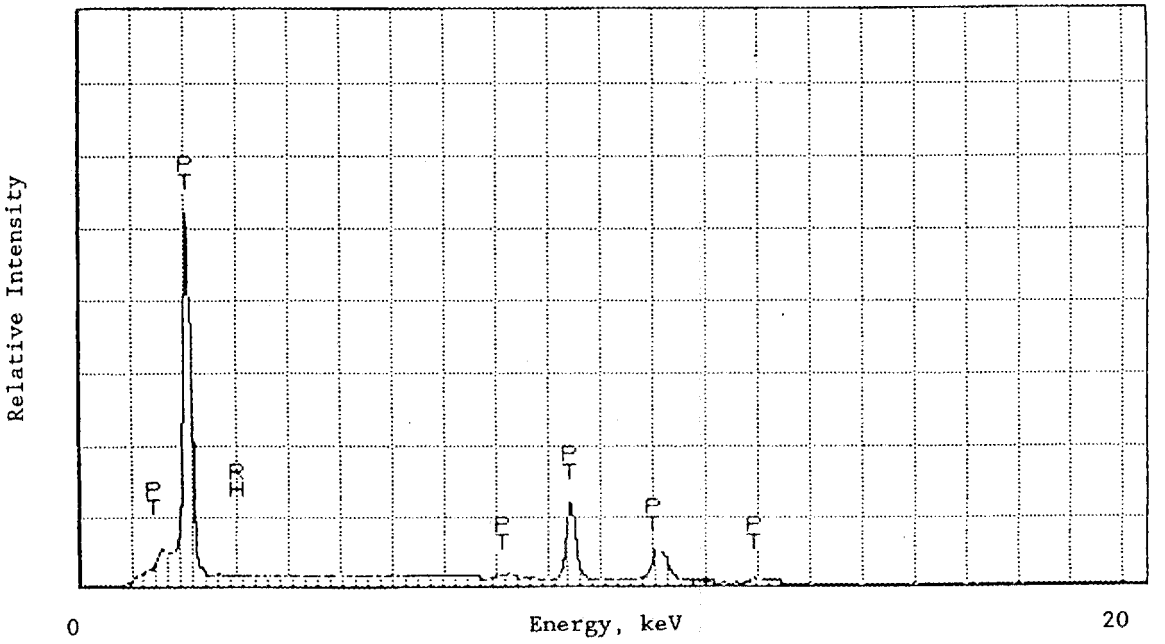
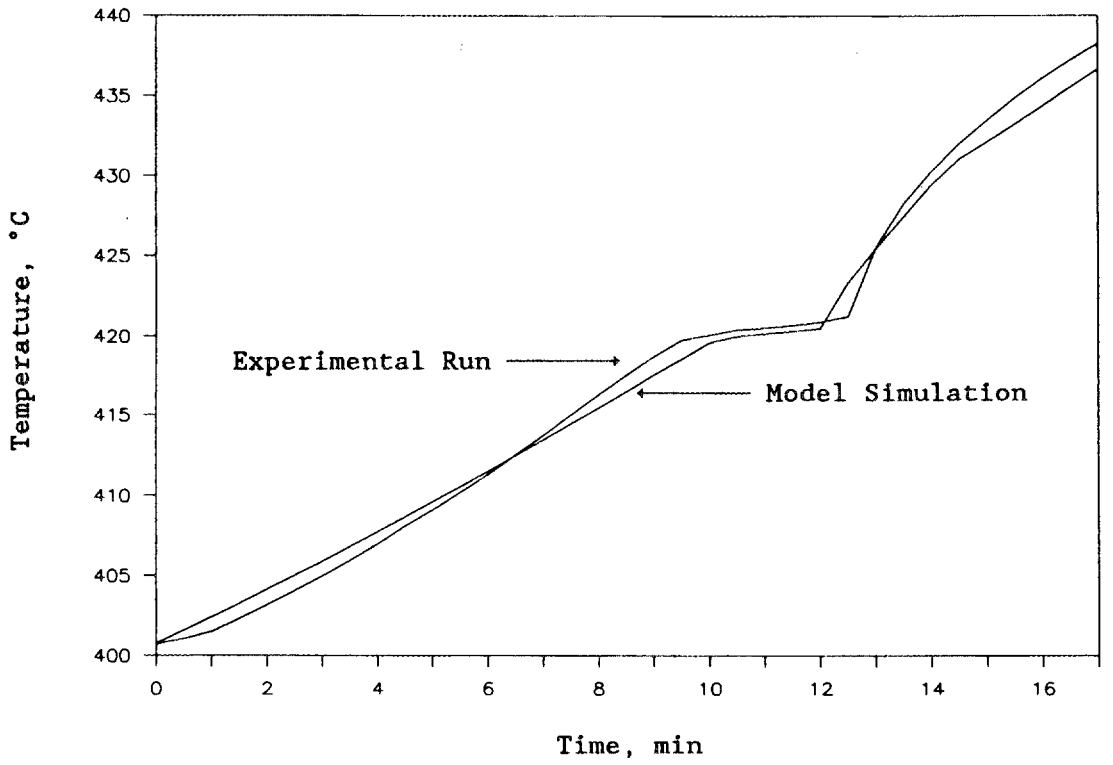


Fig. 4.16 Elemental spectrum of tested Type S with gold thermocouple, Pt wire.



Experimental Conditions

Thermocouple:	Type K with zinc
Heating rate:	~2.5°C/min
Physical arrangement:	Fig. 4.3

Simulation Data

Model:	Fig. 2.1
Sensing node:	13
Encapsulated metal:	zinc
Physical properties:	Table 2.1
Forcing function:	ramp
Heating rate:	2.5°C/min
Boundary condition:	radiative
A_b/A_s:	0.2
Δt:	0.5 s
Solution method:	implicit
Iteration tolerance:	0.05°C

Fig. 4.17 Comparison of model simulation with experimental run.

done using the self-calibrating thermocouple model, as shown in Fig. 2.1, with the sensing node located at node 13. The thermophysical data for the zinc model as well as the tin and gold models are found in Table 2.1. The primary mode of heat transfer from the furnace to the thermocouple was analyzed as radiation.

The initial plateau point of the experimental run and the model simulation both appear to occur at the same temperature. However, the melting duration of the model simulation is somewhat shorter than the experimental run. One possible reason for this discrepancy is that the heat transfer coefficient of the model is higher than actual (see Sect. 3.3). Since a radiation boundary condition was used for the model, the greatest source of error in the radiation heat transfer coefficient is probably the emissivity of the thermocouple sheath. As seen in Eq. (2.10), with a diminishing ratio of boundary surface area to radiating surface area, the heat transfer coefficient is directly proportional to the emissivity of the thermocouple sheath. The emissivity of the model sheath, alumina, seems to be a nebulous figure that varies greatly with porosity. The emissivity has been cited to be as low as 0.18 (ref. 4) and as high as 0.70.[20] The porosity of the self-calibrating thermocouple sheath was not determined. The uncertainty in emissivity could possibly contribute to the shorter than actual melting duration of the model, but in reviewing Fig. 3.5, it is evident that a decrease of at least one order in magnitude in the heat transfer coefficient is required to make the melting duration increase by the amount needed. The emissivity is capable of making only a proportional change to the heat transfer coefficient.

A second factor causing a less than actual melting duration could be that the mass of the encapsulated metal—zinc in this case—is higher than what is used in the model. Calculations of mass were based on X-ray determinations of the metal's dimensions, assuming a partially hollowed cylinder of metal. While the X-ray method should yield a good approximation, a better method would have been to weigh the metal during physical evaluation. However, it would probably take a 50% error or more in the mass calculation to account for the difference in melting durations seen.

A third factor that could cause the short simulated melting duration is that the heat of fusion used is less than actual. However, the largest cited heat of fusion was used for the simulation. Other references give a value of 102,088 J/kg for the heat of fusion of zinc.[21,22] However, the value chosen was 113,003 J/kg.[23]

The final and most probable reason for the less than actual melting duration is that there is an intrinsic problem in either the model or the simulation program. The encapsulated metal is modeled as if its surface is in intimate contact with its retaining cell. Perhaps because of a void in the cell, the heat transfer area between cell wall and encapsulated metal is not as great as modeled. Notice too that the melting algorithm is not a very sophisticated routine. The properties of the melting material in the program do not change after melting as they do in the real system. For example, the thermal conductivity of

liquid zinc is about 60% of that of solid zinc, both referenced to the melting temperature.[23] Therefore, when the outer surface of the encapsulated zinc metal melts, it does not conduct heat through to the inside volume as well as it would if it were still in the solid state. On the other hand, the heat rate may not change too much because a second mode of heat transfer is present due to the convective currents within the liquid zinc.

In summary, other than the melting duration, the simulation program provides a sufficient tool in analyzing the performance of the self-calibrating thermocouple in different situations. It can be a powerful tool for the thermocouple designer by allowing him to see the effects of the thermocouple node location and mass of metal, and for the applications engineer by allowing him to note the performance of the thermocouple in varying conditions of heat transfer and heating rates. But the previous paragraphs illustrate that it cannot do all the work. The properties of the thermocouple have to be determined correctly, and by far the most difficult task is to determine the correct heat transfer coefficient for a given application.

4.6 SYSTEM INTEGRATION

The phase change detection algorithm was programmed into the data acquisition unit before the onset of drift testing. It was tested somewhat at that time. The algorithm was also used to determine the melting and freezing points in the phase change drift tests. Thus, the algorithm was tested regularly throughout these tests. Over the many weeks of testing, the algorithm evolved to its present state. Notice that the algorithm was used before phase change plateau analysis was performed. Thus, it was assumed that the plateau would be flat, and the first phase change detection method used was the average temperature method. The method referred to in the Appendix B typical output as the "new method" is the intersection of slopes method.

A few trials were made at incorporating a variable scan rate into the system. At first a constant scan rate of 1 scan every 30 s was used, but it was thought that, to produce a more robust system that would respond equally well to varying heating rates, the scan rate should be capable of increasing with higher heating rates. An algorithm was designed to do this, basing its scan rate on temperature increments up to the point that the metal begins to change phase. At this point, the scan rate retains the previous scan rate before phase change.

The resulting variable scan-rate code appeared to work satisfactorily. It was set up to take temperature readings of 1°C at every step. Since the intersection of slopes method was used to determine the melting and freezing points, this temperature interval was sufficient. But if the initial plateau point method were to be used to determine melting or freezing point, the temperature interval should be decreased—depending on how precise the melting or freezing point calibration is desired.

The data acquisition system contained a matrix PROM (programmable read-only memory) which simplified analyzing the least-squares fit of lines for the intersection of slopes method of determining melting and freezing points. The equation for a straight time-temperature line is as follows:

$$T = mt + b \quad . \tag{4.1}$$

At the intersection of the time-temperature line before phase change with the line during phase change, the intersection point (the reported phase change temperature as seen in Fig. 3.8) is given by Eq. (4.2).

$$T = \frac{b_1 - m_1/m_2 \; b_2}{1 - m_1/m_2} \quad . \tag{4.2}$$

For several samples of time and temperature, Eq. (4.1) can be expressed in matrix form as follows:

$$\begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ . & . \\ . & . \\ . & . \\ 1 & t_N \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \\ . \\ . \\ . \\ T_N \end{bmatrix} \quad . \tag{4.3}$$

This is a typical linear matrix equation that can be represented as follows:

$$\mathbf{C}\ \mathbf{b} = \mathbf{y} \quad . \tag{4.4}$$

To solve for the least-squares solution of the slope and intersect vector, **b**, we first premultiply by the transposition of the C matrix.[24]

$$\mathbf{C}^T\ \mathbf{C}\ \mathbf{b} = \mathbf{C}^T\ \mathbf{y} \quad . \tag{4.5}$$

Next we premultiply each side of the equation by the inverse of $[\mathbf{C}^T\ \mathbf{C}]$:

$$\mathbf{b} = [\mathbf{C}^T\ \mathbf{C}]^{-1}\ \mathbf{C}^T\ \mathbf{y} \quad . \tag{4.6}$$

The resulting **b** vector contains the appropriate slope and intersect for one time-temperature line. A similar data manipulation of the second line will yield a second slope and intersect set. Given the slope and intersect for each of the two lines, Eq. (4.2) is used to determine the reported phase change temperature. It should be noted that matrix equations were used for convenience only. If a particular system does not support matrix algebra, then the normal equations may be used for regression analysis.

In an industrial system, it may also be desirable to incorporate a digital filter for the raw temperature input into the system. A first- or second-order recursive digital filter algorithm could be added to the

phase change detection algorithm and the scan-rate algorithm code very simply. The algorithm used in this work at first contained a digital filter, but that was found not to be necessary for relatively noise-free laboratory environments.

Thus the algorithms necessary to incorporate phase change detection into a data acquisition system along with the self-calibrating thermocouple to produce a *bona fide* smart temperature measurement system are not overwhelming if they are done in the data acquisition system itself (as in Figs. 1.3b and 1.3c). They may, however, be somewhat tedious if matrix algebra least-squares analysis of the intersection of slopes method is used for a remote smart temperature transmitter-based system, as shown in Fig. 1.3a.

5. CONCLUSIONS AND RECOMMENDATIONS

Based on the drift tests and physical evaluations of the self-calibrating thermocouple, it appears that the thermocouple is a viable industrial-grade assembly and therefore a good front end for a smart temperature measurement system. However, the results are based only on the testing of three self-calibrating thermocouples, which is not a statistically large sample. Further testing would be advised before embarking on a critical project using self-calibrating thermocouples. Based on results of this work, it is recommended that the self-calibrating thermocouple manufacturer take steps to ensure that the thermojunction is completely enclosed by the encapsulated metal to provide a flatter plateau.

The phase detection algorithm works well in the test furnaces. The postplateau jump provides a near foolproof method of detecting phase change. Experience will tell whether it will work as well in other applications, but the simulation program can be used to predict its performance there.

The simulation program provides insight into the thermodynamic behavior of the system. Except for the phase change duration, the program appears to mimic the behavior of the actual thermocouple well. Because the simulated plateau duration is shorter than actual, the program can provide conservative numbers for the plateau duration to aid in system design.

The smart temperature measurement system is designed to provide a single calibration point. Although a system may be functioning with good agreement with a single calibration, there is no guarantee that the system is not out of calibration at other points in the system. Furthermore, no method can adjust the emf/temperature relation based on a single calibration point. However, if the calibration temperature is close to the operating temperature, then the calibration error can probably be used for the operating temperature error.

The results of this work show that, unless confirmed beforehand, the freezing point of the encapsulated metal cannot be used as a calibration point because of excessive metal supercooling.

One foreseeable use of these thermocouples would be in nuclear reactor facilities. Because of the stringent certification requirements of these facilities, a self-calibrating thermocouple seems to be a natural fit. However, the materials used in the thermocouple must be compatible with such a facility. The only material that is strictly forbidden from these facilities is mercury. The test thermocouple has no mercury in the housing, and it is unlikely that the thermoelements would incorporate mercury because no standard thermocouples contain mercury. It is also unlikely that the chosen encapsulated metal would contain mercury because of its low melting point. However, care should be taken that an exotic eutectic alloy, if chosen for the encapsulated metal, not contain mercury.

ACKNOWLEDGMENTS

I would like to express appreciation to Majed Natour for providing the core of the simulation program; Douglas Hergert for providing computer routines in his book *Microsoft QuickBASIC*,[25] some of which I used in the simulation program; and Bob Crouse for performing and interpreting the metallography.

REFERENCES

1. A. J. Mortlock, "Error in Temperature Measurements Due to the Interdiffusion at the Hot Junction of a Thermocouple," *J. Sci. Instrum.* **35**, 283-284 (August 1958).

2. T. F. Mabrey, "Smart Sensing," *Meas. Control* **21**, 124-127 (December 1987).

3. C. A. Mossman, J. L. Horton, and R. L. Anderson, "Testing of Thermocouples for Inhomogeneities: A Review of Theory, with Examples," pp. 923-929 in *Temperature, Its Measurement and Control in Science and Industry*, Vol. 5, American Institute of Physics, New York, 1982.

4. J. P. Holman, *Heat Transfer*, McGraw-Hill, New York, 1976.

5. T. W. Kerlin, *Analytical Methods for Interpreting In-Situ Measurement of Response Times in Thermocouples and Resistance Thermometers*, ORNL/TM-4912, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1976.

6. M. N. Natour, *Modelling and Testing of Industrial Thermometers*, Master's Thesis, The University of Tennessee, Knoxville, 1986.

7. D. R. Pitts, and L. E. Sissom, *Heat Transfer*, McGraw-Hill, New York, 1977.

8. M. N. Ozisik, *Boundary Value Problems of Heat Conduction*, International Text Book Company, Scranton, Pa., 1968.

9. E. Kreyzig, *Advanced Engineering Mathematics*, Wiley, New York, 1983.

10. W. Rosenhain, "Observations on Recalescence Curves," *Proc. Phys. Soc. Lond.* **21**, 180-209 (1908).

11. R. L. Weber, *Temperature Measurement and Control*, The Blakiston Company, Philadelphia, 1941.

12. W. Hume-Rothery, J. W. Christian, and W. B. Pearson, *Metallurgical Equilibrium Diagrams*, Unwin Brothers Ltd., London, 1952.

13. E. H. McClaren, "The Freezing Points of High Purity Metals as Precision Temperature Standards," pp. 185-198 in *Temperature, Its Measurement and Control in Science and Industry*, Vol. 3, Reinhold, New York, 1982.

14. J. L. Riddle, G. T. Furukawa, and H. H. Plumb, *Platinum Resistance Thermometry*, U.S. National Bureau of Standards Monogr. 126, Washington, D.C., 1973.

15. W. G. Pfann, *Zone Melting*, Wiley, New York, 1966.

16. M. H. Cooper Jr., R. L. Anderson, and C. A. Mossman, "Automated Temperature Measurement from -183 to 2300°C," pp. 1287-1292, in *Temperature, Its Measurement and Control in Science and Industry*, Vol. 5, American Institute of Physics, New York, 1982.

17. R. P. Reed, *Improving the Dependability of Critical Thermocouple Thermometry for Fossil Fuel Gasifiers and Retorts*, SAND82-0341, Sandia National Laboratories, Albuquerque, N.M., 1982.

18. R. L. Powell, et al., *Thermocouple Reference Tables Based on the IPTS-68*, U.S. National Bureau of Standards Monogr. 125, Washington, D.C., 1974.

19. P. A. Kinzie, *Thermocouple Temperature Measurement*, Wiley, New York, 1973.

20. Y. S. Touloukian, ed., *Thermophysical Properties of High Temperature Solid Materials*, Vol. 4, MacMillan, New York, 1967.

21. R. H. Perry, and C. H. Chilton, *Chemical Engineers' Handbook*, Fifth Edition, McGraw-Hill, New York, 1973.

22. R. C. Weast, ed., *Handbook of Chemistry and Physics*, 56th Edition, CRC Press, Inc., Cleveland, Ohio, 1975.

23. R. E. Kirk and D. F. Othmer, *Encyclopedia of Chemical Technology*, Third Edition, Vol. 24, Wiley, New York, 1984.

24. N. Neter, W. Wasserman, and M. Kutner, *Applied Linear Regression Models*, Richard D. Irwin, Inc., Homewood, Ill., 1983.

25. D. Hergert, *Microsoft QuickBASIC*, Microsoft Press, Redmond, Wash., 1987.

SELECTED BIBLIOGRAPHY

R. L. Anderson et al., "Decalibration of Sheathed Thermocouples," pp. 977-1007 in *Temperature, Its Measurement and Control in Science and Industry*, Vol. 5, American Institute of Physics, New York, 1982.

R. P. Benedict, *Fundamentals of Temperature, Pressure, and Flow Measurements*, Wiley, New York, 1969.

H. S. Carslaw, H. S. and J. C. Jaeger, *Conduction of Heat in Solids*, Clarendon Press, Oxford, 1959.

Comite International des Poids et Mesures (International Committee of Weights and Measures), "International Practical Temperature Scale of 1968," *Metrologia*, 5, 35-44 (1969).

M. B. Herskovitz, "Failure Mechanisms in Metal-Sheathed Thermocouples," presented at The American Society for Testing Materials Committee E20, Temperature, Los Angeles, Calif., May 22, 1985.

M. B. Herskovitz et al., *Long Term Drift of Commercial Sheathed Chromel/Alumel Thermocouples at 1600°F*, ORNL/TM-3802, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1974.

T. W. Kerlin and R. L. Shepard, *Industrial Temperature Measurement*, Instrument Society of America, Research Triangle Park, N.C., 1982.

T. G. Kollie et al., "Temperature Measurement Errors with Type-K (Chromel vs Alumel) Thermocouples Due to Short-Range Ordering in Chromel," *Rev. Sci. Instrum.* 46, 1447-1461 (November 1975).

E. U. Schlunder, ed., *Heat Exchanger Design Handbook*, Vol. 5, Hemisphere Publishing Corp., New York, 1986.

J. F. Schooley, *Thermometry*, CRC Press, Inc., Boca Raton, Fla., 1987.

APPENDIX A

SIMULATION PROGRAM

Simulation Program Source Code

```
'Programe Name:  SIMULATE.BAS
'Author:         Francis R. Ruppel
'Date:           September 28, 1988
'Purpose:        This program simulates the response of self-calibrating
'                thermocouples to different forcing functions.

'This section declares functions and subprograms to be used within the program.
DECLARE FUNCTION Upper$(textVal$)
DECLARE SUB Menu(title$, menuChoices$(1), numChosen%)
DECLARE SUB Frame(leftCol%, rightCol%, topRow%, bottomRow%)
DECLARE SUB Query1(Prompt$,Reply)
DECLARE SUB Query2(Prompt$,Number,Reply)
DECLARE SUB Query3(Prompt1$,Number,Prompt2$,stringReply$)
DECLARE SUB Query4(Prompt1$,Number1,Prompt2$,Number2,row,column)
DECLARE SUB Query5(Prompt$,Row,Column)
DECLARE SUB Query6(Prompt$,stringReply$)

'Set Boolean and Pi constants
true%=-1
false%=0
Pi=3.1415926#

'Display introduction menu.
REDIM Option$(2)
Option$(1)="Preview Instructions"
Option$(2)="Run Program"
CALL Menu("Thermocouple Simulation Program",Option$(),Choice%)

IF Choice%=1 THEN
     'Display help menu.
     DO
          REDIM Option$(8)
          Option$(1)="General Instructions"
          Option$(2)="Physical Properties Data Input"
          Option$(3)="Reading Data Files"
          Option$(4)="Forcing Function"
          Option$(5)="Heat Transfer Coefficient"
          Option$(6)="Monitor Type"
          Option$(7)="Output Options"
          Option$(8)="Start Program"
          CALL Menu("Choose a help screen or start the program.", Option$(), Choice%)

          SELECT CASE Choice%
               CASE 1
                    GOSUB Instructions
               CASE 2
                    GOSUB dataHelp
               CASE 3
                    GOSUB readFileHelp
               CASE 4
                    GOSUB forcingFunctionHelp
               CASE 5
                    GOSUB heatXfrCoeffHelp
               CASE 6
                    GOSUB outputTypeHelp
               CASE 7
                    GOSUB outputOptionsHelp
              CASE 8
                    GOTO Start
          END SELECT
     LOOP
END IF

Start:

'Display physical properties menu.

REDIM Option$(3)
Option$(1)="Read from Data File"
Option$(2)="Input Interactively"
Option$(3)="Help"
CALL Menu("Choose physical properties-entry method.", Option$(), Choice%)

SELECT CASE Choice%
```

```
     CASE 1
          readFile%=true%
     CASE 2
          readFile%=false%
          CALL Query6("Do you want to create a file from your inputs (y/n)",ans$)
          IF ans$="Y" THEN
               createFile%=true%
               CALL Query6("What is the name of the file to be created",fileName$)
               OPEN fileName$ FOR OUTPUT AS #1
          ELSE createFile%=false%
          END IF
     CASE 3
          GOSUB dataHelp
          GOTO Start
END SELECT

IF readFile% THEN
     readFileOptions:
     'Display data file menu.
     REDIM Option$(5)
     Option$(1)="Tin Self-Calibrating Thermocouple"
     Option$(2)="Zinc Self-Calibrating Thermocouple"
     Option$(3)="Gold Self-Calibrating Thermocouple"
     Option$(4)="Other Data File"
     Option$(5)="Help"
     CALL Menu("Choose the data file.", Option$(),Choice%)
     SELECT CASE Choice%
          CASE 1
               OPEN "TIN.DAT" FOR INPUT AS #1
          CASE 2
               OPEN "ZINC.DAT" FOR INPUT AS #1
          CASE 3
               OPEN "GOLD.DAT" FOR INPUT AS #1
          CASE 4
               CALL Query6("What is the name of the data file",fileName$)
               OPEN fileName$ FOR INPUT AS #1
          CASE 5
               GOSUB readFileHelp
               GOTO readFileOptions
     END SELECT
END IF

IF readFile% THEN
     INPUT #1, Ni,Nj
ELSE
     CALL Query1("How many axial sections are there in the sensor",Ni)
     CALL Query1("How many radial sections are there in the sensor",Nj)
     IF createFile% THEN WRITE #1, Ni,Nj
END IF

N=Ni*Nj
REDIM A(N,N),R(Nj),Z(Ni),K(Ni+1,Nj),D(Ni,Nj),Cp(Ni,Nj),T(N)
REDIM meltingNode%(N),Told(N),Tnew(N),Amelt(N,N)
REDIM C(Ni,Nj),R1(Ni,Nj),R2(Ni,Nj),R3(Ni,Nj),R4(Ni,Nj),R5(Ni,Nj),F(N)
REDIM Tzi(N),melted%(N),Q(N),Mass(Ni,Nj),B(N)

GOSUB PhysicalProperties

forcingFunctionOptions:
'Display forcing function menu.
REDIM Option$(4)
Option$(1)="Ramp ambient temperature"

Option$(2)="Internal step heat generation"
Option$(3)="Step ambient temperature"
Option$(4)="Help"
CALL Menu("Choose the forcing function type.", Option$(),forcingFunction%)

SELECT CASE forcingFunction%
     CASE 1
          'ramp forcing function
          CALL Query1("What is the ramp rate ("+CHR$(248)+"C/min)",minRamp)
          Ramp=minRamp/60
          FOR K=1 TO N
               Q(K)=0
          NEXT K
     CASE 2
```

```
          'self-heating forcing function
          CALL Query1("What is the rate of joule heating (W/m^3)",Heat)
          Y6=(Y1*Nj)-(Nj-Y2)
          FOR I=Y6 TO Nj*Y3 STEP Nj
               Q(I)=Heat
          NEXT I
     CASE 3
          'step ambient change forcing function
          CALL Query1("What is the magnitude ("+CHR$(248)+"C) of the step change",Tstep)
          FOR K=1 TO N
               Q(K)=0
          NEXT K
     CASE 4
          GOSUB forcingFunctionHelp
          GOTO forcingFunctionOptions
END SELECT

heatXfrCoeffOptions:
'Display heat transfer coefficient menu.
REDIM Option$(4)
Option$(1)="Enter convective heat transfer coefficient"
Option$(2)="Find convective heat transfer coefficient for fluids"
Option$(3)="Use radiation boundary condition"
Option$(4)="Help"
CALL Menu("Choose the heat transfer coefficient.", Option$(),Choice%)

SELECT CASE Choice%
     CASE 1
          'user enters coefficient
          CALL Query1("What is the heat transfer coefficient (W/m^2-"+CHR$(248)+"C)",H)
          radiationMode%=false%
     CASE 2
          'program calculates coefficient
          GOSUB findCoefficient
          radiationMode%=false%
     CASE 3
          'radiation boundary condition
          radiationMode%=true%
          CALL Query1("What is the emissivity of the thermocouple sheath",E1)
          CALL Query1("What is the emissivity of the heating surface",E2)
          CALL Query1("What is the sensor/heating surface area ratio",Aratio)
          Krad=5.669E-08/(1/E1+Aratio*(1/E2-1))
     CASE 4
          GOSUB heatXfrCoeffHelp
          GOTO heatXfrCoeffOptions
END SELECT

GOSUB Capacitances
GOSUB Resistances

'Calculate the maximum allowed time increment for explicit solution.
FOR I=1 TO Ni
     FOR J=1 TO Nj
          deltaT=C(I,J)/(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J))
          IF J=1 AND I=1 THEN
               deltaTmax = deltaT
          ELSEIF deltaT < deltaTmax THEN
               deltaTmax = deltaT
          END IF
     NEXT J
NEXT I

'Display maximum explicit time increment, use implicit if time increment too large.
CALL Frame(2,78,5,15)
LOCATE 8,7
PRINT "Maximum allowed time increment (sec) for explicit solution = ";
PRINT USING "#.#####";deltaTmax
LOCATE 10,7
INPUT"Input the time increment (sec) -> ", deltaT
IF deltaT>deltaTmax THEN
     LOCATE 12,7
     PRINT"Time increment too large for explicit solution, using implicit."
     LOCATE 13,7
     PRINT "Press any key to continue."
     WHILE INKEY$=""
     WEND
     Implicit%=true%
```

```
     CALL Query1("What is the iteration tolerance ("+CHR$(248)+"C)",Ttol)
ELSE Implicit%=false%
END IF

CALL Query1("How long should the simulation last (sec)",Duration)

GOSUB Amatrix
GOSUB ForcingVectors

CALL Query1("What is the initial temperature ("+CHR$(248)+"C)",Tinit)
Tinit=Tinit+273.15

Energy=0

FOR K=1 TO N
     T(K)=Tinit

     IF Tinit<Tmelt THEN
          melted%(K)=false%
     ELSE
          melted%(K)=true%
     END IF
NEXT K

outputTypeOptions:
'Display output type menu.
REDIM Option$(6)
Option$(1)="Text Output Only"
Option$(2)="CGA Graphics Output"
Option$(3)="Hercules Graphics Output"
Option$(4)="EGA Graphics Output"
Option$(5)="Output Options"
Option$(6)="Assistance (Help)"
CALL Menu("Enter the monitor type or select output options.", Option$(),Monitor%)

IF Monitor%=6 THEN
     GOSUB outputTypeHelp
     GOTO outputTypeOptions
END IF

IF Monitor%=5 THEN
outputOptions:
     quitOptions%=false%
     DO
          REDIM Option$(9)
          Option$(1)="1 Add non-sensing nodes traces"
          Option$(2)="2 Add ambient temperature trace"
          Option$(3)="3 Add axis numeric scaling (graphics only)"
          Option$(4)="4 Add grids (graphics only)
          Option$(5)="5 Change temperature range display (graphics only)"
          Option$(6)="6 Change time range display (graphics only)"
          Option$(7)="7 Reset options to default settings"
          Option$(8)="Return to Output Type Menu"
          Option$(9)="Help"
          CALL Menu("Enter the output options or return to previous menu.", Option$(),Choice%)

          SELECT CASE Choice%
               CASE 1
                    CALL Query1("How many non-sensing nodes should be shown",numberOfTraces)
                    REDIM Trace(numberOfTraces)

                    FOR K=1 TO numberOfTraces
                         CALL Query2("What is the node # for trace",K,Trace(K))
                    NEXT K
               CASE 2
                    CALL Query6("Should the ambient temperature be shown (y/n)",ans$)
                    IF ans$="Y" THEN showAmbient%=true%
               CASE 3
                    CALL Query1("How many temperature scale points",tempPoints)
                    CALL Query1("How many time scale points",timePoints)
                    showScale%=true%
               CASE 4
                    CALL Query6("Should horizontal grids be added",response$)
                    IF response$="Y" THEN CALL Query1("How many horizontal grids",horizGrids)
                    CALL Query6("Should vertical grids be added",response$)
                    IF response$="Y" THEN CALL Query1("How many vertical grids",vertGrids)
               CASE 5
```

```
                    CALL Query1("What is the new lower range ("+CHR$(248)+"C)",lrTemp)
                    CALL Query1("What is the upper range ("+CHR$(248)+"C)",urTemp)
                    lrTemp=lrTemp+273.15
                    urTemp=urTemp+273.15
                    tempRangeChanged%=true%
               CASE 6
                    CALL Query1("What is the new lower range (sec)",lrTime)
                    CALL Query1("What is the new upper range (sec)",urTime)
                    timeRangeChanged%=true%
               CASE 7
                    tempRangeChanged%=false%
                    timeRangeChanged%=false%
                    showAmbient%=false%
                    showScale%=false%
                    numberOfTraces=0
                    horizGrids=0
                    vertGrids=0
                    CALL Frame(5,75,5,11)
                    LOCATE 8,15
                    PRINT"Options reset; press any key to continue."
                    WHILE INKEY$=""
                    WEND
               CASE 8
                    quitOptions%=true%
               CASE 9
                    GOSUB outputOptionsHelp
                    GOTO outputOptions
          END SELECT
     LOOP UNTIL quitOptions%
     GOTO outputTypeOptions
END IF

IF NOT tempRangeChanged% THEN
     'set default temperature range
     lrTemp=Tinit
     SELECT CASE forcingFunction%
          CASE 1
               'ramp forcing function
               urTemp=Tinit+Ramp*Duration
          CASE 2
               'self-heating forcing function
               urTemp=Tinit+5
          CASE 3
               'step forcing function
               urTemp=Tinit+Tstep
     END SELECT
END IF

numberOfSteps=Duration/deltaT

IF NOT timeRangeChanged% THEN
     'set default time range
     lrTime=0
     urTime=Duration
END IF
lrSteps=lrTime/deltaT
urSteps=urTime/deltaT

'set screen for appropriate display.
graphicsMonitor% = true%
SELECT CASE Monitor%
     CASE 1
          'text output
          graphicsMonitor%=false
     CASE 2
          'CGA monitor
          CLS
          SCREEN 2
          Horiz = 639
          Vert = 199
     CASE 3
          'Hercules monitor
          CLS
          SCREEN 3
          Horiz = 719
          Vert = 347
     CASE 4
```

```
        'EGA monitor
        CLS
        SCREEN 10
        Horiz = 639
        Vert = 349
END SELECT

IF graphicsMonitor% THEN
     'This section displays the scale if requested.
     IF showScale% THEN
          'set the graphics window
          VIEW (7/80*Horiz,1)-(Horiz-1,21/25*Vert-1),,1

          'print the temperature scale points
          FOR I=0 TO tempPoints
               LOCATE 21-I*20/tempPoints,1
               PRINT USING "####.#"; I*(urTemp-lrTemp)/tempPoints+lrTemp-273.15
          NEXT I

          'print the time scale points
          FOR I=0 TO timePoints
               LOCATE 22,4+I*73/timePoints
               PRINT USING "####";I*(urTime-lrTime)/timePoints+lrTime
          NEXT I

          LOCATE 11,1: PRINT "  "+CHR$(248)+"C  "
          LOCATE 23,41: PRINT "SECONDS"
     ELSE
          'If scales are not requested, then a plain box is drawn.
          VIEW (1,1)-(Horiz-1,Vert-1),,1
     END IF

     WINDOW (lrSteps,lrTemp)-(urSteps,urTemp)

     'Grids will be drawn if requested.
     FOR I=1 TO horizGrids-1
          yaxisGrid=I/horizGrids*(urTemp-lrTemp)+lrTemp
          LINE (lrSteps,yaxisGrid)-(urSteps,yaxisGrid),,,&H8888
     NEXT I

     FOR I=1 TO vertGrids-1
          xaxisGrid=I/vertGrids*(urSteps-lrSteps)+lrSteps
          LINE (xaxisGrid,lrTemp)-(xaxisGrid,urTemp),,,&H8888
     NEXT I
END IF

headerPrinted%=false%

IF forcingFunction%=2 THEN Tamb=Tinit   'self-heating forcing function
IF forcingFunction%=3 THEN Tamb=Tinit+Tstep 'step forcing function

'This section enables aborting the simulation by striking the escape key
KEY 15, CHR$(0)+CHR$(1)   'trap plain escape key
KEY 16, CHR$(32)+CHR$(1)  'trap escape key with num lock depressed
KEY 17, CHR$(64)+CHR$(1)  'trap escape key with caps lock depressed
KEY 18, CHR$(96)+CHR$(1)  'trap escape key with both num and caps lock depressed
ON KEY(15) GOSUB abort : ON KEY(16) GOSUB abort : ON KEY(17) GOSUB abort : ON KEY(18) GOSUB abort
KEY(15) ON : KEY(16) ON : KEY(17) ON : KEY(18) ON

'This section will loop for the duration of the simulation.
FOR incrementNumber = 0 TO numberOfSteps

     Toldamb=Tamb
     'Increase temperatue if ramp forcing function
     IF forcingFunction%=1 THEN Tamb=Tinit+Ramp*incrementNumber*deltaT

     'If there is a radiation boundary condition, then H will change with
     'temperature.  Resistances, Amatrix, and ForcingVectors have to
     'be recalculated.
     IF radiationMode% THEN
          H=Krad*(Tamb^2+T(N)^2)*(Tamb+T(N))
          GOSUB Resistances
          GOSUB Amatrix
          GOSUB ForcingVectors
     END IF

     FOR K = 1 TO N
          Told(K)=T(K)
```

```
     IF meltingNode%(K) AND T(K)>=Tmelt AND NOT melted%(K) THEN
          'This routine runs during the melting of each node.
          'The Energy variable contains the energy input to the melting nodes. If
          'the energy is less than Hf (the heat of fusion), then melting continues.
          IF Energy<Hf*meltMass THEN
               FOR L=1 TO N
                    Energy=Energy+Amelt(K,L)*T(L)
               NEXT L
          ELSE
               melted%(K)=true%
          END IF
     END IF
NEXT K

IF Implicit% AND incrementNumber<>0 THEN
     'This loop will run in the implicit solution until the new
     'temperature is solved iteratively.
     DO
          Complete%=true%
          FOR K=1 TO N
               IF meltingNode%(K) AND T(K)>=Tmelt AND NOT melted%(K) THEN
                    Tnew(K)=Tmelt

               ELSE
                    Tzi(K)=B(K)*Told(K)
                    'The Tzi() vector is the zero input vector (response of system when
                    'there is no input).
                    'Multiply the Aimplicit matrix by the Temperature vector
                    FOR termCounter%=1 TO 4
                         skip%=false%
                         SELECT CASE termCounter%
                              CASE 1
                                   L=K-Nj
                                   IF L<1 THEN skip%=true%
                              CASE 2
                                   L=K-1
                                   IF L<1 THEN skip%=true%
                              CASE 3
                                   L=K+1
                                   IF L>N THEN skip%=true%
                              CASE 4
                                   L=K+Nj
                                   IF L>N THEN skip%=true%
                         END SELECT
                         IF NOT skip% THEN Tzi(K)=Tzi(K)+A(K,L)*T(L)
                    NEXT termCounter%
                    Tnew(K)=Tzi(K)+F(K)*Tamb +Q(K)
               END IF
          NEXT K

          'Compare new temperature to old, if it is within tolerance then proceed,
          'otherwise re-iterate.
          FOR K=1 TO N
               IF ABS(Tnew(K)-T(K))>Ttol THEN Complete%=false%
               T(K)=Tnew(K)
          NEXT K
     LOOP UNTIL Complete%

ELSEIF incrementNumber<>0 THEN
     'This code will run for the explicit solution.
     FOR K=1 TO N
          IF meltingNode%(K) AND T(K)>=Tmelt AND NOT melted%(K) THEN
               T(K)=Tmelt
          ELSE
               Tzi(K)=0
               'The Tzi() vector is the zero input vector (response of system when
               'there is no input).
               'Multiply the Aexplicit matrix by the Temperature vector

               FOR termCounter%=1 to 5
                    skip%=false%
                    SELECT CASE termCounter%
                         CASE 1
                              L=K-Nj
                              IF L<1 THEN skip%=true%
                         CASE 2
                              L=K-1
```

```
                    IF L<1 THEN skip%=true%
                CASE 3
                    L=K
                CASE 4
                    L=K+1
                    IF L>N THEN skip%=true%
                CASE 5
                    L=K+Nj
                    IF L>N THEN skip%=true%
            END SELECT
            IF NOT skip% THEN Tzi(K)=Tzi(K)+A(K,L)*T(L)
        NEXT termCounter%
        T(K)=Tzi(K)+F(K)*Tamb+Q(K)
    END IF
NEXT K

END IF

IF incrementNumber=0 THEN
    Toldsense=Tinit
ELSE
    Toldsense=Tsense
END IF

'This code calculates the average sensing temperature. Y variables are the
'sensing node's row and column coordinates.
X=0
K=0
Y6=(Y1*Nj)-(Nj-Y2)
FOR J=Y6 TO Nj*Y3 STEP Nj
    K=K+1
    X=X+T(J)
NEXT J
Tsense=X/K

IF graphicsMonitor% THEN
    'Points will be plotted for a graphics terminal.
    'The sensing element will always be plotted.
    LINE  (incrementNumber-1,Toldsense) - (incrementNumber,Tsense)

    FOR K=1 TO numberOfTraces
        'Requested nodes will also be plotted.
        LINE (incrementNumber-1,Told(Trace(K))) - (incrementNumber,T(Trace(K)))
    NEXT K

    IF showAmbient%  THEN
        'The ambient temperature will be plotted if requested.
        LINE (incrementNumber-1,Toldamb) - (incrementNumber,Tamb)
    END IF
ELSE
    'Data will be printed for a text terminal.
    IF NOT headerPrinted% THEN
        CLS
        PRINT "Time in seconds, Temperature in "+CHR$(248)+"C"
        PRINT
        PRINT "   Time   Tsense ";

        FOR K=1 TO numberOfTraces
            PRINT "  T(";
            PRINT USING "##";Trace(K);
            PRINT ")  ";
        NEXT K

        IF showAmbient% THEN PRINT "  Tamb";
        PRINT
        VIEW PRINT 4 TO 24
        headerPrinted%=true%
    END IF

    PRINT USING "####.##";incrementNumber*deltaT;
    PRINT USING "######.##";Tsense-273.15;

    FOR K=1 TO numberOfTraces
        PRINT USING "######.##";T(Trace(K))-273.15;
    NEXT K

    IF showAmbient% THEN PRINT USING "######.##";Tamb-273.15;
```

```
            PRINT

        END IF

NEXT incrementNumber

'The simulation is terminated.
BEEP

WHILE INKEY$ = ""
WEND

quit:
'Turn off escape key abort option.
KEY(15) OFF : KEY(16) OFF : KEY(17) OFF : KEY(18) OFF

SCREEN 0
VIEW PRINT

'Display end of simulation menu.
REDIM Option$(3)
Option$(1)="Repeat same physical model with new conditions"
Option$(2)="Start over with new physical model"
Option$(3)="Quit"
CALL Menu("Simulation terminated:  select next route.",Option$(),Choice%)
SELECT CASE Choice%
     CASE 1
          GOTO forcingFunctionOptions
     CASE 2
          GOTO Start
     CASE 3
          END
END SELECT

END   'end of main program module

'Beginning of subroutine, function, and subprogram section of program

'If simulation was terminated with escape key, just go to quit label.
abort:
     GOTO quit
RETURN

PhysicalProperties:
     'This subroutine generates the nodes and assigns the physical properties
     'to the model.
     R0=0

     FOR J=1 TO Nj
          IF readFile% THEN
               INPUT #1,R
          ELSE
               CALL Query2("What is the radius increment (mm) for node",J,R)
               IF createFile% THEN WRITE #1, R
          END IF

          R(J)=R/1000
          R0=R0+R(J)
     NEXT J

     Z0=0

     FOR I=1 TO Ni
          IF readFile% THEN
               INPUT #1,Z
          ELSE
               CALL Query2("What is the length increment (mm) for axial node",I,Z)
               IF createFile% THEN WRITE #1, Z
          END IF

          Z(I)=Z/1000
          Z0=Z0+Z(I)
     NEXT I

     IF readFile% THEN
          INPUT #1,N3
     ELSE
```

```
        CALL Query1("How many different materials comprise the sensor",N3)
        IF createFile% THEN WRITE #1, N3
    END IF

    FOR L=1 TO N3
        IF readFile% THEN
            INPUT #1,Melt$
            IF Melt$="Y" THEN INPUT #1,Tmelt,Hf
            INPUT #1,K,D,Cp,N4
        ELSE
            CALL Query3("Does material",L,"melt in the temperature range of interest (y/n)",Melt$)
            IF createFile% THEN WRITE #1, Melt$
            IF Melt$="Y" THEN
                CALL Query2("What is the melting point ("+CHR$(248)+"C) of material",L,Tmelt)
                CALL Query2("What is the heat of fusion (J/kg) of material",L,Hf)
                IF createFile% THEN WRITE #1, Tmelt,Hf
            END IF
            CALL Query2("What is the thermal conductivity (W/m-"+CHR$(248)+"C) of material",L,K)
            CALL Query2("What is the density (kg/m^3) of material",L,D)
            CALL Query2("What is the specific heat (J/kg-"+CHR$(248)+"C) of material",L,Cp)
            CALL Query2("How many zones are made of material",L,N4)
            IF createFile% THEN WRITE #1, K,D,Cp,N4
        END IF

        Tmelt=Tmelt+273.15

        FOR P=1 TO N4
            IF readFile% THEN
                INPUT #1, I1,J1,I2,J2
            ELSE
                NodeCoordinates:
                CALL Query4("Row,column of the top left node of zone",P,"of material",L,I1,J1)
                CALL Query4("Row,column of the bottom right node of zone",P,"of material",L,I2,J2)

                IF Melt$="Y" AND (I1=Ni OR I2=Ni OR J1=Nj OR J2=Nj) THEN
                    CALL Frame(5,75,5,12)
                    LOCATE 8,10
                    PRINT "Melting node not allowed on exterior surface."
                    LOCATE 9,10
                    PRINT "Press any key to continue."
                    WHILE INKEY$=""
                    WEND
                    GOTO NodeCoordinates
                END IF

                IF createFile% THEN WRITE #1, I1,J1,I2,J2
            END IF

            FOR I=I1 TO I2
                FOR J=J1 TO J2
                    K(I,J)=K
                    D(I,J)=D
                    Cp(I,J)=Cp
                    X=(I-1)*Nj+J
                    IF Melt$="Y" THEN meltingNode%(X)=true%
                NEXT J
            NEXT I
        NEXT P
    NEXT L

    ' Location of sensing elements
    IF readFile% THEN
        INPUT #1,Y1,Y2,Y3,Y4
    ELSE
        CALL Query5("Row,Column of the top left sensing node",Y1,Y2)
        CALL Query5("Row,Column of the bottom right sensing node",Y3,Y4)
        IF createFile% THEN WRITE #1, Y1,Y2,Y3,Y4
    END IF

    CLOSE #1

RETURN

Capacitances:
    'The Purpose of the Capacitances subroutine is to calculate the thermal
    'capacitance of each node.
    R6=0
```

```
     R7=0
     meltMass=0
     FOR I=1 TO Ni
          FOR J=1 TO Nj
               P=(I-1)*Nj+J
               R7=R6
               R6=R6+R(J)
               Mass(I,J)=Pi*(R6^2-R7^2)*Z(I)*D(I,J)
               C(I,J)=Mass(I,J)*Cp(I,J)
               IF meltingNode%(P) THEN meltMass=meltMass+Mass(I,J)
          NEXT J
          R6=0
          R7=0
     NEXT I
RETURN

Resistances:
     'The purpose of the Resistances subroutine is to calculate the thermal
     'resistance between each node.
     R6=0
     R7=0
     FOR I=1 TO Ni
          FOR J=1 TO Nj
               R7=R6
               R6=R6+R(J)
               R5(I,J)=2*Pi*(R6^2-R7^2)*K(I,J)*K(I+1,J)
               IF I=Ni THEN
                    IF J=Nj THEN
                         R1(I,J)=2*Pi*R6*Z(I)*H
                         R2(I,J)=H*(R6^2-R7^2)*Pi
                    ELSEIF J=Nj-1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+2*R(J+1)*K(I,J))
                         R2(I,J)=H*Pi*(R6^2-R7^2)
                    ELSEIF J>1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+R(J+1)*K(I,J))
                         R2(I,J)=H*(R6^2-R7^2)*Pi
                    ELSE
                         'J=1, I=Ni
                         R1(I,J)=4*Pi*R6*K(I,J)*K(I,J+1)*Z(I)/(2*R(J)*K(I,J+1)+R(J+1)*K(I,J))
                         R2(I,J)=H*(R6^2-R7^2)*Pi
                    END IF
               ELSEIF I=Ni-1 THEN
                    IF J=Nj   THEN
                         R1(I,J)=H*Pi*2*R6*Z(I)
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+2*Z(I+1)*K(I,J))
                    ELSEIF J=Nj-1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+2*R(J+1)*K(I,J))
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+2*Z(I+1)*K(I,J))
                    ELSEIF J>1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+R(J+1)*K(I,J))
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+2*Z(I+1)*K(I,J))
                    ELSE
                         'J=1, I=  Ni-1
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(2*R(J)*K(I,J+1)+R(J+1)*K(I,J))
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+Z(I+1)*2*K(I,J))
                    END IF
               ELSEIF I>1 THEN
                    '1<I<Ni-1
                    IF J=Nj THEN
                         R1(I,J)=H*R6*2*Pi*Z(I)
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
                    ELSEIF J=Nj-1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+2*R(J+1)*K(I,J))

                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
                    ELSEIF J>1 THEN
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J+1)*K(I,J)/((R(J)*K(I,J+1))+R(J+1)*K(I,J))
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
                    ELSE
                         'J=1 & 1<I<Ni-1
                         R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(2*R(J)*K(I,J+1)+R(J+1)*K(I,J))
                         R2(I,J)=R5(I,J)/(Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
                    END IF
               ELSE
                    'I=1
                    IF J=Nj THEN
                         R1(I,J)=H*2*Pi*Z(I)*R6
```

```
                    R2(I,J)=R5(I,J)/(2*Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
               ELSEIF J=Nj-1 THEN
                    R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+2*R(J+1)*K(I,J))
                    R2(I,J)=R5(I,J)/(2*Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
               ELSEIF J>1 THEN
                    R1(I,J)=4*Pi*R6*Z(I)*K(I,J)*K(I,J+1)/(R(J)*K(I,J+1)+R(J+1)*K(I,J))
                    R2(I,J)=R5(I,J)/(2*Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
               ELSE
                    'I=1 ,J=1
                    R1(I,J)=4*Pi*K(I,J)*K(I,J+1)*R6*Z(I)/(2*R(J)*K(I,J+1)+R(J+1)*K(I,J))
                    R2(I,J)=R5(I,J)/(2*Z(I)*K(I+1,J)+Z(I+1)*K(I,J))
               END IF
          END IF
          R3(I,J)=R1(I,J-1)
          R4(I,J)=R2(I-1,J)
     NEXT J
R6=0
NEXT I
RETURN

Amatrix:
     'The purpose of the Amatrix subroutine is to generate the system matrix
     FOR I=1 TO Ni
          FOR J=1 TO Nj
               P=(I-1)*Nj+J

               IF Implicit% THEN
                    'The implicit system matrix is calculated differently
                    'than the explicit system matrix.
                    Denominator=(1+deltaT/C(I,J)*(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J)))
               ELSE
                    'explicit system matrix
                    Denominator=1
               END IF

               IF I>1 THEN
                    IF I=Ni THEN
                         IF J=Nj THEN
                              A(P,P-Nj)=(deltaT*R4(I,J)/C(I,J))/Denominator
                              A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         ELSE
                              A(P,P+1)=(deltaT*R1(I,J)/C(I,J))/Denominator
                              A(P,P-Nj)=(deltaT*R4(I,J)/C(I,J))/Denominator
                              A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         END IF
                    ELSEIF P<>I*Nj THEN
                         A(P,P-Nj)=(deltaT*R4(I,J)/C(I,J))/Denominator
                         A(P,P+1)=(deltaT*R1(I,J)/C(I,J))/Denominator
                         A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         A(P,P+Nj)=(deltaT*R2(I,J)/C(I,J))/Denominator

                         IF meltingNode%(P) THEN
                              Amelt(P,P-Nj)=deltaT*R4(I,J)
                              Amelt(P,P+1)=deltaT*R1(I,J)
                              Amelt(P,P-1)=deltaT*R3(I,J)
                              Amelt(P,P+Nj)=deltaT*R2(I,J)
                         END IF

                    ELSE
                         A(P,P+1)=0
                         A(P,P-Nj)=(deltaT*R4(I,J)/C(I,J))/Denominator
                         A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         A(P,P+Nj)=(deltaT*R2(I,J)/C(I,J))/Denominator
                    END IF
               ELSEIF J>1 THEN
                    IF J<Nj THEN
                         A(P,P+1)=(deltaT*R1(I,J)/C(I,J))/Denominator
                         A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         A(P,P+Nj)=(deltaT*R2(I,J)/C(I,J))/Denominator
                    ELSE
                         A(P,P-1)=(deltaT*R3(I,J)/C(I,J))/Denominator
                         A(P,P+Nj)=(deltaT*R2(I,J)/C(I,J))/Denominator
                    END IF
               ELSE
                    A(P,P+1)=(deltaT*R1(I,J)/C(I,J))/Denominator
                    A(P,P+Nj)=(deltaT*R2(I,J)/C(I,J))/Denominator
               END IF
```

```
                IF meltingNode%(P) THEN
                     Amelt(P,P)=-deltaT*(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J))
                END IF

                IF Implicit% THEN
                     B(P)=1/Denominator
                     A(P,P)=0
                ELSE
                     A(P,P)=-deltaT*(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J))/C(I,J)+1
                END IF
           NEXT J
      NEXT I
RETURN

FindCoefficient:
     'The heat transfer coefficient can be calculated if requested.
     CALL Query1("What is the velocity (m/sec) of the fluid",V1)
     CALL Query1("What is the thermal conductivity (W/m-"+CHR$(248)+"C) of the fluid",K)
     CALL Query1("What is the density (kg/m^3) of the fluid",D)
     CALL Query1("What is the specific heat (J/kg-"+CHR$(248)+"C) of the fluid",Cp)
     CALL Query1("What is the viscosity (kg/m-sec) of the fluid",u)
     'Find Reynolds Number
     Reynolds=2*R0*V1*D/u
     IF Reynolds>5000 THEN
          LOCATE 9,10
          PRINT"Reynolds # is too high. User must input heat transfer coefficient"
          WHILE INKEY$=""
          WEND
          GOTO heatXfrCoeffOptions
     ELSEIF Reynolds>1000 THEN
          H=(K/(2*R0))*0.26*(Reynolds^0.6)*(Cp*u/K)^0.3
     ELSEIF Reynolds>.1 THEN
          H=(K/(2*R0))*((0.35+(0.47*Reynolds^0.52))*((Cp*u/K)^0.3))
     ELSE
          LOCATE 9,10
          PRINT"Reynolds # is too low. User must input heat transfer coefficient."
          WHILE INKEY$=""
          WEND
          GOTO heatXfrCoeffOptions
     END IF
RETURN

ForcingVectors:
     'The purpose of this subroutine is to generate the forcing vectors.
     FOR I=1 TO Ni
          FOR J=1 TO Nj
               IF Implicit% THEN
                    Denominator=(1+deltaT/C(I,J)*(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J)))
               ELSE
                    Denominator=1
               END IF
               P=(I-1)*Nj+J
               IF P=I*Nj AND P<N THEN
                    F(P)=(deltaT*R1(I,J)/C(I,J))/Denominator
               ELSEIF P>=Nj*(Ni-1)+1 THEN
                    IF P<>N THEN
                         F(P)=(deltaT*R2(I,J)/C(I,J))/Denominator
                    ELSEIF P=N THEN
                         F(N)=(deltaT*(R1(Ni,Nj)+R2(Ni,Nj))/C(Ni,Nj))/Denominator
                    END IF
               END IF
          NEXT J
     NEXT I
     IF forcingFunction%=2 THEN
          'Forcing function is a step change in joule heating
          Y9=0
          Y7=Y1*Nj-(Nj-Y2)
          Y8=Y3*Nj-(Nj-Y2)
          K1=0

          FOR J=1 TO Y2-1
               K1=R(J)+K1
          NEXT J

          FOR I=1 TO Ni
               FOR J=1 TO Nj
```

```
                    IF Implicit% THEN
                         Denominator=(1+deltaT/C(I,J)*(R1(I,J)+R2(I,J)+R3(I,J)+R4(I,J)))
                    ELSE
                         Denominator=1
                    END IF
                    P=(I-1)*Nj+J
                    Y6=Y7+Y9

                    IF P=Y6 AND Y6<= Y8 THEN
                         Q(P)=deltaT*(Q(P)*Pi*(((K1+R(J))^2)-K1^2)*Z(I))/C(I,J)/Denominator
                         Y9=Y9+Nj
                    END IF
               NEXT J
          NEXT I
     END IF
RETURN

DEF FN Upper$(textVal$)
     'The Upper$ function converts alphabetic characters in a string value to
     'upper case letters.
     STATIC i%, number%, character$
     'Find the length of the string value received.
     number% = LEN(textVal$)
     'Examine each character in the string and convert as necessary.
     FOR i% = 1 TO number%
          character$ = MID$(textVal$, i%, 1)
          IF (character$ >= "a" AND character$ <= "z") THEN
               MID$(textVal$, i%) = CHR$(ASC(character$) - 32)
          END IF
     NEXT i%
     FN Upper$ = textVal$
END DEF

SUB Menu(title$, menuChoices$(1), numChosen%) STATIC
     'The Menu subprogram displays a menu on the screen and elicits a menu
     'choice from the user.  Menu receives a string array (menuChoices$)
     'containing the menu choices and returns an integer (numChosen%)
     'indicating the user's selection from among those choices.

     'Find the number of choices (numOfChoices%); initialize variables.

     numOfChoices% = UBOUND(menuChoices$)
     prompt$ = " "
     okString$ = ""
     longString% = 0

     'Prepare the prompt string (prompt$) and the string of legal input
     'characters (okString$).  Also, find the length of the longest choice
     'string (longString$).
     FOR i% = 1 TO numOfChoices%
          first$ = FN Upper$(LEFT$(menuChoices$(i%), 1))
          okString$ = okString$ + first$
          prompt$ = prompt$ + first$ +"  "
          temp% = LEN(menuChoices$(i%))
          IF (temp% > longString%) THEN longString% = temp%
     NEXT i%

     IF (LEN(title$) > longString%) THEN longString% = LEN(title$)

     longString% = longString% + 1
     prompt$ = prompt$ + "-> "

     'Declare true% and false% as Boolean variables.  These will always hold
     'values of -1 and 0 respectively.
     true%=-1
     false%=0

     'Test to see if the prompt string is longer than longString%.
     IF (LEN(prompt$) >= longString%) THEN longString% = LEN(prompt$) + 1

     'Given longString% and numOfChoices%, determine the dimensions of the
     'menu frame.  Draw the frame, calling the Frame subprogram.
     lc% = 37 - (longString% \ 2)
     rc% = 80 - lc%
     tr% = 3
     br% = 10 + numOfChoices%
     CALL Frame(lc%, rc%, tr%, br%)
```

```
     'Display the menu choices.  The first letter of each choice is displayed
     'in uppercase, followed by a parenthesis character.
     FOR i% = 1 TO numOfChoices%
          LOCATE 6 + i%, lc% + 3
          PRINT FN Upper$(LEFT$(menuChoices$(i%), 1)) + ")" + _
          MID$(menuChoices$(i%), 2)
     NEXT i%

     LOCATE 4, 40-LEN(title$)/2
     PRINT title$
     line$ = STRING$(longString%, 196)
     LOCATE 5, lc% + 3:  PRINT line$
     LOCATE 7 + numOfChoices%,lc% + 3:  PRINT line$

     'Print the input prompt
     LOCATE 9 + numOfChoices%,lc% + 3:  PRINT prompt$;

     'Get a menu choice.  Validate and verify the choice.
     ctrlKey$ = CHR$(13) + CHR$(27)
     done% = false%
     WHILE NOT done%
          LOCATE ,, 1
          charPos% = 0
          WHILE charPos% = 0
               ans$ = INKEY$
               IF (ans$ <> "") THEN
                    ans$ = FN Upper$(ans$)
                    charPos% = INSTR(okString$, ans$)
                    IF (charPos% = 0) THEN BEEP
               END IF
          WEND
          PRINT ans$

          LOCATE 11 + numOfChoices%, 23, 0
          PRINT "<Enter> to confirm; <ESC> to redo."
          numChosen% = charPos%

          charPos% = 0
          WHILE charPos% = 0
               ans$ = INKEY$
               IF (ans$ <> "") THEN
                    charPos% = INSTR(ctrlKey$, ans$)
                    IF (charPos% = 0) THEN BEEP
               END IF
          WEND

          IF (charPos% = 1) THEN
               done% = true%
               CLS
          ELSE
               LOCATE 11 + numOfChoices%, 23:  PRINT SPACE$(35)
               LOCATE 9 + numOfChoices%, lc% + 3 + LEN(prompt$):  PRINT " ";
               LOCATE , POS(0) - 1:
          END IF
     WEND
END SUB

SUB Frame(leftCol%, rightCol%, topRow%, bottomRow%) STATIC
     'The Frame subprogram draws a rectangular double-line frame on
     'the screen, using text-graphics characters from the upper range
     'of the ASCII code.

     'Print the four corners.
     CLS
     LOCATE topRow%, leftCol%:  PRINT CHR$(201)
     LOCATE topRow%, rightCol%:  PRINT CHR$(187)
     LOCATE bottomRow%, leftCol%:  PRINT CHR$(200)
     LOCATE bottomRow%, rightCol%:  PRINT CHR$(188)

     'Print the vertical lines.
     FOR vertLine% = topRow% + 1 to bottomRow% - 1
          LOCATE vertLine%, leftCol%:  PRINT CHR$(186);
          LOCATE vertLine%, rightCol%:  PRINT CHR$(186);
     NEXT vertLine%

     'Print the horizontal lines.
```

```
     horizLength% = rightCol% - leftCol% - 1
     horizLine$ = STRING$(horizLength%, 205)
     LOCATE topRow%, leftCol% + 1:  PRINT horizLine$
     LOCATE bottomRow%, leftCol% + 1:  PRINT horizLine$;
END SUB

SUB Query1(Prompt$,Reply)
     'The purpose of this subprogram is to prompt the user for a numeric reply.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt$)+15
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+2
     PRINT Prompt$;
     INPUT Reply
END SUB

SUB Query2(Prompt$,Number,Reply)
     'The purpose of this subprogram is to prompt the user for a numeric reply.
     'The prompt line is split into a prompt and a number.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt$)+15
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+2
     PRINT Prompt$;
     PRINT Number;
     INPUT Reply
END SUB

SUB Query3(Prompt1$,Number,Prompt2$,stringReply$)
     'The purpose of this subprogram is to prompt the user for a string reply.
     'The prompt is split into two prompts and one number.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt1$)+LEN(Prompt2$)+15
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+5
     PRINT Prompt1$;
     PRINT Number;
     PRINT Prompt2$;
     INPUT stringReply$
     stringReply$=FNUpper$(stringReply$)
END SUB

SUB Query4(Prompt1$,Number1,Prompt2$,Number2,row,column)
     'The purpose of this subprogram is to prompt the user for a row & column.
     'The prompt is split into two prompts and two numbers.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt1$)+LEN(Prompt2$)+20
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+5
     PRINT Prompt1$;
     PRINT Number1;
     PRINT Prompt2$;
     PRINT Number2;
     INPUT row,column
END SUB

SUB Query5(Prompt$,Row,Column)
     'The purpose of this subprogram is to prompt the user for a row & column.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt$)+15
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+5
     PRINT Prompt$;
     INPUT Row,Column
END SUB

SUB Query6(Prompt$,stringReply$)
```

```
     'The purpose of this subprogram is to prompt the user for a string reply.
     'A frame is drawn around the prompt area.
     stringLength% = LEN(Prompt$)+15
     leftMargin% = 40-(stringLength%\2)
     rightMargin% = 80-leftMargin%
     CALL Frame(leftMargin%,rightMargin%,5,11)
     LOCATE 8,leftMargin%+2
     PRINT Prompt$;
     INPUT stringReply$
     stringReply$=FNUpper$(stringReply$)
END SUB

Instructions:
     PRINT
     PRINT"   This program uses a finite difference method to simulate temperature"
     PRINT"   responses of a thermocouple to selected forcing functions."
     PRINT"   Although the program is designed to simulate any thermocouple"
     PRINT"   geometry and physical properties in general, a special feature"
     PRINT"   is that it will also simulate the self-calibrating thermocouple."
     PRINT"   (A thermocouple with a low melting point metal encapsulated near"
     PRINT"   the measuring junction.)
     PRINT
     PRINT"   There are two methods of solution:  1) explicit, and 2) implicit."
     PRINT"   The explicit case is used when very small values for the time "
     PRINT"   increment are allowed.  The maximum time increment for this method"
     PRINT"   is calculated and presented to the user.  It is a function of the"
     PRINT"   diffusivities of the sensor materials and the length increments."
     PRINT"   The implicit method is used when time increments larger than the"
     PRINT"   maximum explicit time increment are desired.  It uses a simultaneous"
     PRINT"   substition iteration method, therefore the time spent on each time"
     PRINT"   increment is larger than the explicit method, but larger time"
     PRINT"   increments are allowed, so it will normally proceed faster than"
     PRINT"   the explicit method."
     PRINT:PRINT
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN

dataHelp:
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"     The physical properties of the temperature sensor may either be"
     PRINT"     read from a data file or input interactively.  If the user "
     PRINT"     chooses to input interactively, he may also choose to create a"
     PRINT"     data file for later use."
     PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT:PRINT:PRINT:PRINT:PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN

readFileHelp:
     'This subroutine print the instructions for the read file option.
     CLS
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"    The physical properties of the temperature sensor may be stored"
     PRINT"    in a data file.  There are three ready made data files of the"
     PRINT"    three self-calibrating thermocouples studied so far.  Any one of"
     PRINT"    these three files may be chosen, or a different file may be chosen"
     PRINT"    (be sure to specify path if it is not in the default drive)."
     PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN

forcingFunctionHelp:
     CLS
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"     There are three types of forcing inputs to the temperature"
     PRINT"     sensor system:  1)ambient temperature ramp, 2) ambient"
     PRINT"     temperature step, and 3) internal joule heating step."
     PRINT
     PRINT"     For a sensor in a typical process, the temperature near the"
```

```
     PRINT"     melting point of the encapsulated metal in the self-calibrating"
     PRINT"     thermocouple can be approximated by the ramp function."
     PRINT
     PRINT"     The ambient temperature step and the internal joule heating"
     PRINT"     step inputs are normally used for time constant evaluation."
     PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS

RETURN

heatXfrCoeffHelp:
     CLS
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"     There are two types of heat transfer boundary conditions"
     PRINT"     that can be simulated:  1) convective, and 2) radiative."
     PRINT
     PRINT"     For the convective case, the user may input the convective"
     PRINT"     heat transfer coefficient, or have the program calculate it"
     PRINT"     for the case of a moving fluid in a pipe."
     PRINT
     PRINT"     For the radiative case, the program will calculate the "
     PRINT"     radiative heat transfer coefficient as a function of temperature."
     PRINT"     (Note that this must be calculated each time step, therefore it"
     PRINT"     will slow down execution of the program.)  The program assumes"
     PRINT"     a cylindrical temperature sensor surrounded by a cylindrical"
     PRINT"     heating surface.  The program requests the emissivity of the"
     PRINT"     sensor and heating surface, and the ratio of sensor/heating"
     PRINT"     surface area."
     PRINT:PRINT
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN

outputTypeHelp:
     CLS
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"     The user may choose to have the program output appear numerically"
     PRINT"     by choosing text only output, or he may choose to have the output"
     PRINT"     appear graphically."
     PRINT
     PRINT"     The numeric output will work with any type of monitor.  The"
     PRINT"     graphic output will work for the displayed monitor types."
     PRINT
     PRINT"    NOTE:  DURING EXECUTION, THE USER MAY STRIKE THE ESCAPE KEY "
     PRINT"           TO HALT THE PROGRAM"
     PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN

outputOptionsHelp:
     CLS
     PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"     Two options are available for numeric output monitors; the"
     PRINT"     remainder are for graphics monitors only."
     PRINT
     PRINT"     An option is also available to reset the output options."
     PRINT"     This may be convient if the user is running more than one"
     PRINT"     simulation case."
     PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
     PRINT"Press any key to continue."
     WHILE INKEY$=""
     WEND
     CLS
RETURN
```

Simulation Program Data Dictionary

Boolean Arrays

	melted%()indicates that a melting node has completely melted
meltingNode%()	indicates that a node is a melting node (part of the encapsulated metal)

Boolean variables

Complete%	used to indicate when an iteration cycle is complete in the implicit solution
createFile%	indicates that a file is to be created to store the data that will be input interactively
graphicsMonitor%	indicates that a graphics monitor has been selected as the output display type
headerPrinted%	indicates that the header for numeric output has been printed
Implicit%	indicates that the implicit solution method is being used
quitOptions%	indicates that all options have been selected for the output options menu
radiationMode%	indicates that the heat transfer mode is radiation
readFile%	indicates that data is to be read from a data file
showAmbient%	indicates that the ambient temperature should be displayed on output
showScale%	indicates that scales should be displayed
skip%	used to indicate when to skip a calculation when multiplying the A() matrix by the T() vector
tempRangeChanged%	indicates that the temperature range has been changed from default
timeRangeChanged%	indicates that the time range has been changed from default

Floating Point Matrices

A()	system defining matrix that contains the thermal resistances and capacitances between nodes in the system mesh
Amelt()	matrix used to determine when all the heat of fusion has been provided to the melting metal
B()	used in the implicit solution as a coefficient of the past temperature vector
C()	contains the thermal capacitance of each node
Cp()	contains the heat capacity of each node
D()	contains the density of each node
F()	ambient temperature forcing function vector
K()	contains the thermal conductivity of each node
Mass()	contains the mass of each node
Q()	self-heating forcing function vector
R()	contains the radii of each nodal column
R1()	for R1(i,j), contains the resistance term between node i,j and node i,j+1
R2()	for R2(i,j), contains the resistance term between node i,j and node i+1,j
R3()	for R3(i,j), contains the resistance term between node i,j and node i,j-1
R4()	for R4(i,j), contains the resistance term between node i,j and node i-1,j
R5()	general matrix used to calculate thermal resistances
T()	contains the current temperature of each node
Tnew()	newly iterated temperature of each node
Told()	past temperature of each node
Trace()	contains the nodes of which the temperature will be displayed
Tzi()	zero input temperature vector (the response of the nodes when there is no forcing function)
Z()	contains the axial length of each nodal row

Floating Point Variables

Aratio	ratio of thermocouple surface area to radiating surface area
deltaT	time increment
deltaTmax	maximum allowable time increment for an explicit solution
Denominator	used to solve for temperature, it is set to different values depending on whether the explicit or implicit solution is being used
Duration	length of the simulation in seconds
E1	emissivity of the thermocouple sheath
E2	emissivity of the radiating surface
Energy	energy absorbed by the melting nodes
H	heat transfer coefficient
Heat	contains the self-heat per unit volume
Hf	heat of fusion of the encapsulated metal
Horiz	horizontal resolution of the graphics screen
horizGrids	number of horizontal grids to display
I	row counter used in loops
I1	general top left row coordinate of material in model
I2	general top left column coordinate of material in model
incrementNumber	current increment number of the simulation
J	column counter used in loops
J1	general bottom right row coordinate of material in model
J2	general bottom right column coordinate of material in model
K	general counter used in loops mostly to count up to the total number of nodes
Krad	radiation heat transfer coefficient

L	general counter used in loops mostly to count up to the number of nodes when K has already been used
lrTemp	lower range temperature to be displayed
lrTime	lower range time to be displayed
meltMass	mass of the encapsulated metal
minRamp	ambient temperature ramp rate in °C/min
N3	the total number of materials in the model
N4	the number of zones of a material in the model
Ni	number of rows in the nodal mesh
Nj	number of columns in the nodal mesh
numberOfSteps	number of time increments in the simulation
numberOfTraces	number of nodal traces that will be displayed
P	used to calculate the node number from the row and column number
R	radius of a nodal column
R0	total radius of model
R6	used to determine radius of node
R7	used to determine radius of node
Ramp	ambient temperature ramp rate in °C/sec
Reynolds	Reynolds number of the fluid
Tamb	ambient temperature
tempPoints	number of temperature scale points to be displayed
timePoints	number of time scale points to be displayed
Tinit	initial simulation temperature
Tmelt	melting point of the encapsulated metal
Toldsense	old sensing temperature as seen by the thermocouple
Tsense	sensing temperature as seen by the thermocouple
u	fluid viscosity
urTemp	upper rang temperature to be displayed
urTime	upper range time to be displayed
V1	fluid flowing velocity
Vert	vertical resolution of the graphics screen
vertGrids	number of vertical grids to display
Y1	top left row coordinate of sensing node
Y2	top left column coordinate of sensing node
Y3	bottom right row coordinate of sensing node
Y4	bottom right column coordinate of sensing node
Y6	node counter used in self-heating equations
Z	length of nodal row
Z0	total axial length of model

<u>Integers</u>

Choice%	general integer that is the response to a menu query
forcingFunction%	indicates which forcing function has been selected
termCounter%	used to identify the term when multiplying the A() matrix by the T() vector

<u>String Arrays</u>

Option$	general string array that contains options for the menu displays

<u>String Variables</u>

CHR$(248)	prints the degree symbol " ° "
fileName$	variable that contains the name of a file to be created or opened
INKEY$	input from the keyboard
Melt$	used to indicate that a model material melts in the temperature range of interest

<u>Subprograms</u>

Frame	draws a frame around a display
Menu	displays menus to the operator
Query	used to query the operator for information or a choice
Upper	converts lower case alphanumerics to upper case

<u>Subroutines</u>

Amatrix	calculates the A() system matrix
Capacitances	calculates the thermal capacitances of nodes
dataHelp	prints a physical properties data help screen
forcingFunctionHelp	prints a help screen about the different forcing functions available to the user
ForcingVectors	calculates the forcing vectors
heatXfrCoeffHelp	prints a help screen about the different heat transfer coefficients
Instructions	prints general instructions to the user
outputOptionsHelp	prints a help screen about the options available for data output
outputTypeHelp	prints a help screen about the different types of output available
PhysicalProperties	determines the physical properties of the model
readFileHelp	prints a help screen for reading physical properties from a data file
Resistances	calculates the thermal resistances between nodes

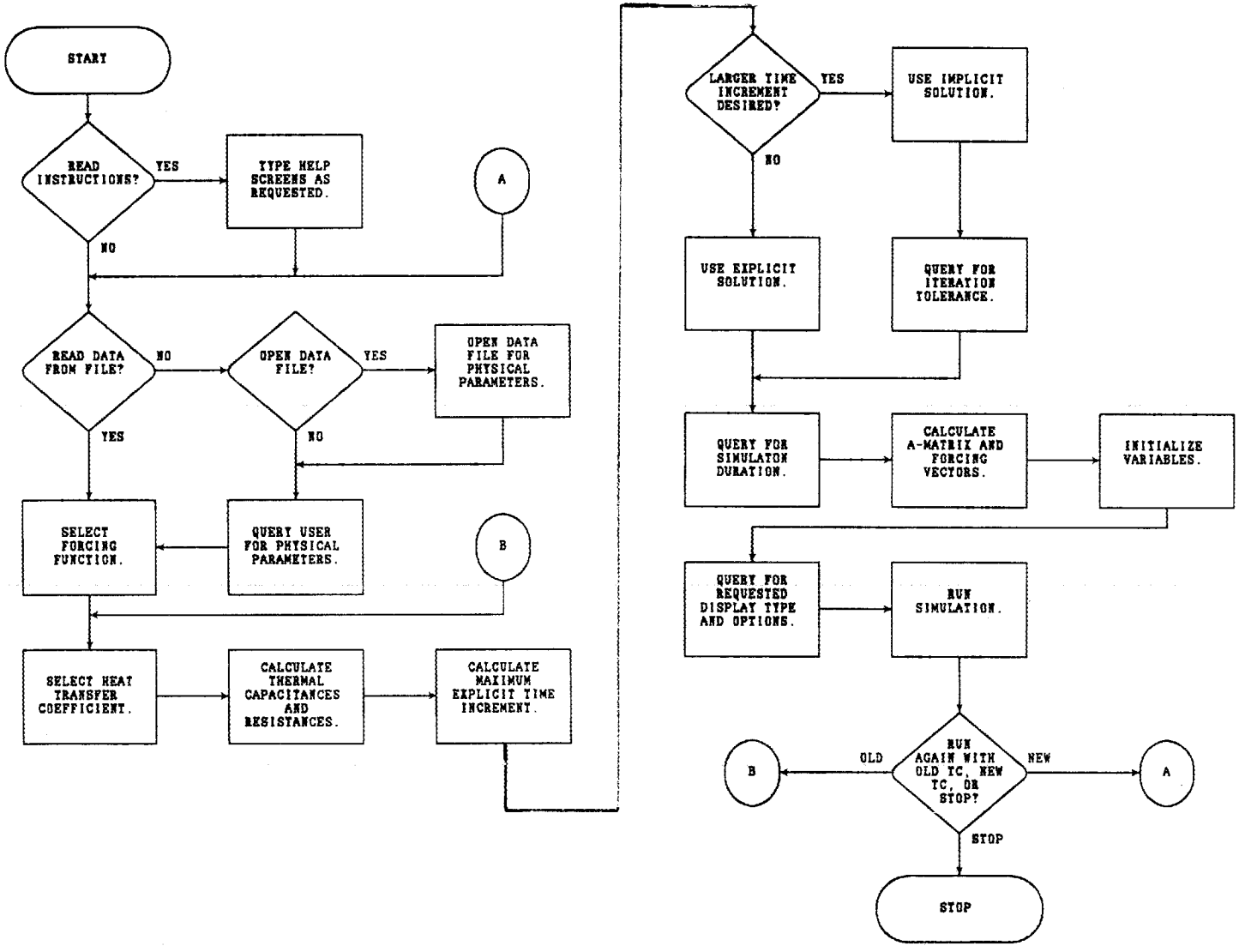


Fig. A.1 Simplified flowchart of simulation program.

Time in seconds, Temperature in °C

Time	Tsense	T(20)	T(30)	Tamb
0.00	415.00	415.00	415.00	415.00
1.00	415.00	415.00	415.00	415.04
2.00	415.00	415.00	415.01	415.08
3.00	415.00	415.00	415.02	415.13
4.00	415.00	415.00	415.04	415.17
5.00	415.00	415.00	415.05	415.21
6.00	415.00	415.00	415.07	415.25
7.00	415.01	415.01	415.08	415.29
8.00	415.01	415.01	415.10	415.33
9.00	415.01	415.01	415.12	415.38
10.00	415.02	415.02	415.15	415.42
11.00	415.02	415.02	415.17	415.46
12.00	415.03	415.03	415.19	415.50
13.00	415.04	415.03	415.22	415.54
14.00	415.04	415.04	415.24	415.58
15.00	415.05	415.05	415.27	415.63
16.00	415.06	415.06	415.30	415.67
17.00	415.07	415.06	415.32	415.71
18.00	415.08	415.07	415.35	415.75
19.00	415.09	415.08	415.38	415.79
20.00	415.10	415.09	415.41	415.83
21.00	415.12	415.11	415.44	415.88
22.00	415.13	415.12	415.47	415.92
23.00	415.15	415.13	415.50	415.96
24.00	415.16	415.14	415.53	416.00
25.00	415.18	415.16	415.57	416.04
26.00	415.19	415.17	415.60	416.08
27.00	415.21	415.19	415.63	416.13
28.00	415.23	415.20	415.66	416.17
29.00	415.24	415.22	415.70	416.21
30.00	415.26	415.24	415.73	416.25
31.00	415.28	415.25	415.76	416.29
32.00	415.30	415.27	415.80	416.33
33.00	415.32	415.29	415.83	416.38
34.00	415.34	415.31	415.86	416.42
35.00	415.36	415.33	415.90	416.46
36.00	415.38	415.35	415.93	416.50
37.00	415.41	415.37	415.97	416.54
38.00	415.43	415.39	416.00	416.58
39.00	415.45	415.41	416.04	416.63
40.00	415.48	415.44	416.07	416.67

Fig. A.2 Typical numeric output of simulation program. Time, sensing temperature, nodes Nos. 20 and 30 temperature, and ambient temperature are displayed.

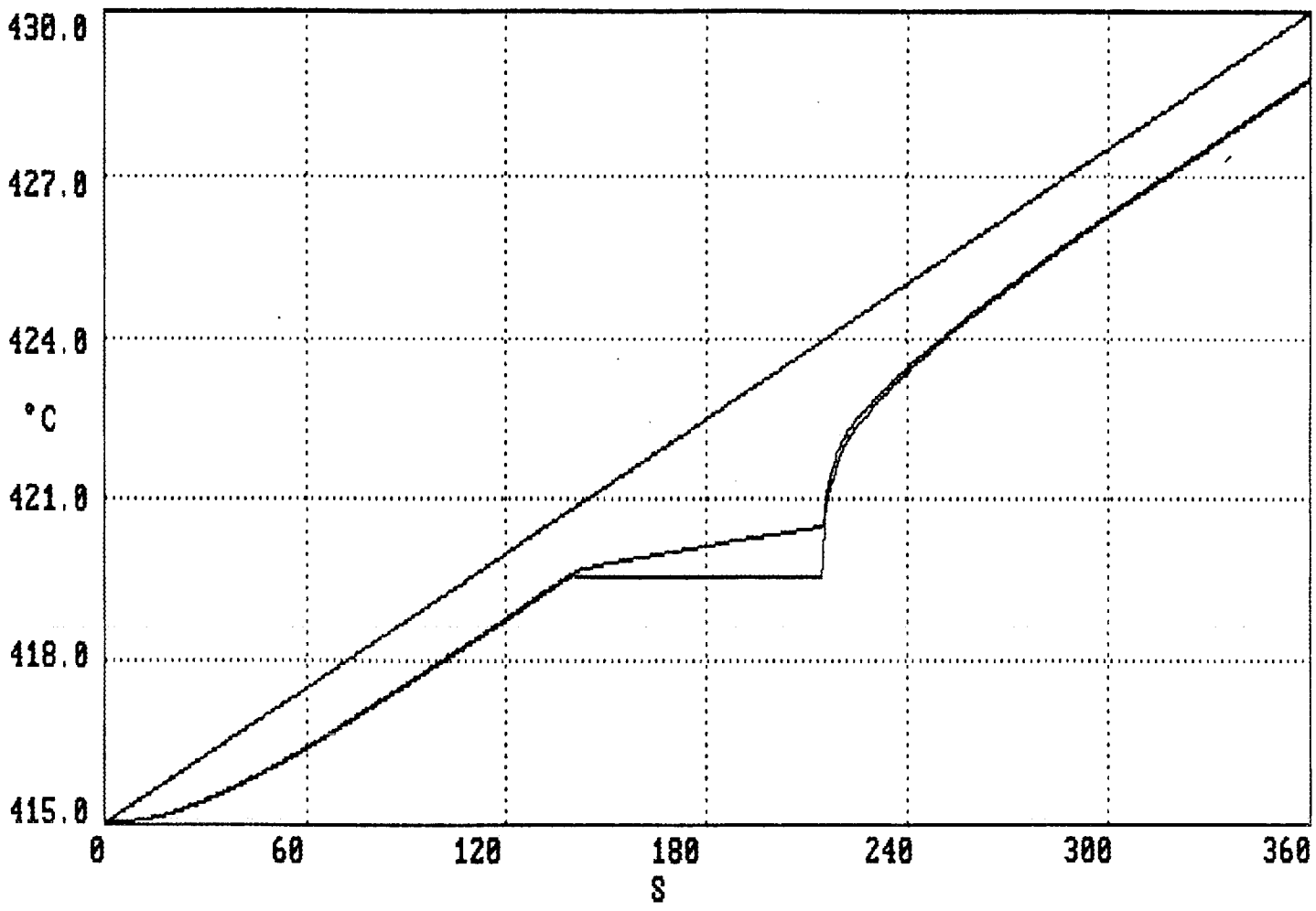


Fig. A.3 Typical graphical output of simulation program. Ambient temperature, node No. 20 temperature, and sensing temperature are shown.

APPENDIX B

PHASE CHANGE DETECTION PROGRAM

Phase Change Detection Program Source Code

```
0: "Smart Thermocouple Algorithm, Type K with Zinc, F.R. Ruppel":
1: dim S[10],T[10],M[10],X$[12],A[100],B[100],C[100],I[100,3],V[3]
2: dim F$[6],S$[1]
3: "Disk string byte req":0→r79
4: "Disk Numeric byte req. for W[X,Y]=X*Y*8+2":300*8+2→r89
5: -1→I
6: red 716,X$
7: wrt 6,X$[3,4],"/",X$[5,6],"/88   ",X$[7,8],":",X$[9,10]
8: ent "melt or freeze, m or f",R$
9: if R$="m";wrt 6,"Melting Point Determination - Zinc"
10: if R$="f";wrt 6,"Freezing Point Determination - Zinc"
11: 30→U
12: U/60→R
13: fmt 2,"T,degC",10x,"Slp,degC/min",10x,"2ndDer,degC/(min)↑2",5x,"Scan"
14: wrt 6.2
15: 1→Z
16: .3→F
17: 419.58→M
18: clr 709;wrt 709,"11,22"
19: wait 1000
20: "start":
21: for J=1 to 3
22: wait 21
23: red 722,V[J]
24: next J
25: (V[1]+V[2]+V[3])/3→V
26: 1e6*V→V
27: '* temp K'(V)→T
28: fxd 2
29: dsp T,"degC"
30: if C=0;gto "timer"
31: (T-C)/R→S
32: if T<M-30 or T>M+30;gto "scan rate"
33: I+1→I
34: S*(1-F)+F*E→D
35: D→E
36: fmt 3,f10.2,f20.1,f20.1,f18.1
37: "fill arrays":
38: for J=10 to 2 by -1
39: S[J-1]→S[J]
40: T[J-1]→T[J]
41: next J
42: S→S[1]
43: T→T[1]
44: if I<2;gto "reset"
45: (S[1]-S[2])/R→G
46: wrt 6.3,T,S,G,U
47: T→A[I];S→B[I];G→C[I];I→I[I,1];R→I[I,2];R+I[I-1,3]→I[I,3]
48: if R$="m" and G>2*H and I>10;gto "melting point?"
49: if R$="f" and G<2*H and I>10;gto "melting point?"
50: "reset":
51: if R$="m" and G>H;G→H
52: if R$="f" and G<H;G→H
53: if abs(S)<2*Z;jmp 4
54: "scan rate":
55: if abs(S)<.5;120→U;jmp 2
56: 1/abs(S)*60→U
57: if U<1;1→U
58: U/60→R
59: "timer":
60: T→C
61: int(U)→U
62: for J=1 to U
63: wait 1000
64: next J
65: gto "start"
66: "melting point?":
67: 0→K
68: for J=1 to 10
69: if abs(S[J])>Z;gto "next J"
70: K+1→K
71: T[J]→M[K]
72: "next J":
73: next J
```

```
74: if K=0;gto "reset"
75: 0→P
76: for J=1 to K
77: P+M[J]→P
78: next J
79: P/K→A
80: fmt 9,"Calibration Temperature: ",f6.2," Error: ",f5.2
81: wrt 6.9,A,A-M
82: for J=1 to I
83: if C[J]=min(C[*]) and R$="m";gto "out"
84: if C[J]=max(C[*]) and R$="f";gto "out"
85: next J
86: "out":
87: J→r10
88: I-(r10+2)→r11
89: dim D[r11,2],E[r11],F[2],G[r11,2],H[r11],J[2]
90: dim K[2,r11],L[2,2],U[2,2],N[2,r11]
91: dim O[2,r11],P[2,2],Q[2,2],R[2,r11]
92: for J=1 to r11
93: 1→D[J,1];1→G[J,1]
94: I[r10+J,3]→D[J,2];I[r10-J-1,3]→G[J,2]
95: A[r10+J]→E[J];A[r10-J-1]→H[J]
96: next J
97: trn D→K
98: mat KD→L
99: inv L→U
100: mat UK→N
101: mat NE→F
102: trn G→O
103: mat OG→P
104: inv P→Q
105: mat QO→R
106: mat RH→J
107: J[2]/F[2]→r12
108: (J[1]-r12*F[1])/(1-r12)→r13
109: fmt 8,"New Method: ",f6.2," Error: ",f5.2
110: wrt 6.8,r13,r13-M
111: cll 'disk record'
112: end
113: "* temp K":
114: if p1=0;ret p1
115: if p1>0;gto "K>ZERO"
116: "est temp":
117: 2.3783697e-2p1+-2.4382217e-6p1↑2→p2
118: p2-6.8203073e-10p1↑3-9.4854031e-14p1↑4→p2
119: 3.9475433139e1→p3
120: 2.7465251138e-2→p4
121: -1.6565406716e-4→p5
122: -1.5190912392e-6→p6
123: -2.4581670924e-8→p7
124: -2.4757917816e-10→p8
125: -1.5585276173e-12→p9
126: -5.9729921255e-15→p10
127: -1.2688801216e-17→p11
128: -1.1382797374e-20→p12
129: "K<ZERO":0→p13→p14
130: for L=3 to 12
131: p13+pLp2↑(L-2)→p13
132: p14+(L-2)pLp2↑(L-3)→p14
133: next L
134: p13-p1→p13
135: p2-p13/p14→p16
136: if abs(p16-p2)<.01;ret p16
137: p16→p2
138: gto "K<ZERO"
139: "K<ZERO":
140: 2.513278e-2p1-6.088342e-8p1p1+5.535820e-13p1↑3+9.372091e-18p1↑4→p2
141: -1.8533063272e1→p3;3.8918344612e1→p4
142: 1.6645154356e-2→p5;-7.8702374448e-5→p6
143: 2.2835785557e-7→p7;-3.5700231258e-10→p8
144: 2.9932909136e-13→p9;-1.2849848798e-16→p10
145: 2.2239974336e-20→p11
146: "K+ZERO":0→p12→p13
147: for L=0 to 8
148: L+3→A
149: p12+pAp2↑L→p12
150: p13+LpAp2↑(L-1)→p13
```

```
151: next L
152: p12+125*exp(-.5((p2-127)/65)↑2)→p12
153: p12-p1→p12
154: p2-p12/p13→p15
155: if abs(p15-p2)<.1;ret p15
156: p15→p2
157: gto "K+ZERO"
158:
159: "find null data file":
160: for F=r99 to 999;fxd 0;str(F)→F$;F$[2]→F$
161: dsp F$;asgn F$,1,1,X;if X=1;F→r99;ret
162: next F;prt "No file found for data";stp
163:
164: "disk record":
165: beep;wait 1000;beep;beep;ent "Record data y or n",S$;if S$#"y";ret
166: cll 'find null data file'
167: "CONVERT BYTES TO RECORDS":int((r79+r89)/256)+1→X;dsp X
168: drive 1,700;open F$[1],X
169: files *;asgn F$[1],1,1,X;rread 1,1;sprt 1,A[*],B[*],C[*]
170: wrt 6,"Data on DISK F.R.Ruppel, Data #2, File ",F$
171: drive 0,700;ret
172:
```

Phase Change Detection Program Data Dictionary

Device Numbers

6	printer
700	disk drives (0,700 and 1,700)
709	scanner
716	clock
722	digital volt meter

Matrices

A[]	general temperature array
B[]	general slope array
C[]	general second derivative of temperature array
D[]	consists of two columns, the first column is filled with ones (as shown in equation 4.3), the second column if filled with the time samples for the least squares fit of the plateau line
E[]	temperature samples to be used in the least squares fit of the plateau line
F[]	the slope and intersect vector for the least squares fit plateau line
G[]	consists of two columns, the first column is filled with ones (as shown in equation 4.3), the second column if filled with the time samples for the least squares fit of the line preceding the plateau
H[]	temperature samples to be used in the least squares fit of the line preceding the plateau
I[]	matrix consisting of three columns, column one contains the increment counter numbers, column two contains the scan times for each increment, and column three contains the total elapsed time for each increment
J[]	the slope and intersect vector for the least squares fit of the line preceding the plateau
K[]	transpose of the D[] matrix, used to find the least squares fit of the plateau line
L[]	product of K[]*D[] matrices, used to find the least squares fit of the plateau line
M[]	used to find average value of melting point
N[]	product of U[]*K[], used to find the lease squares fit of plateau line
O[]	transpose of the G[] matrix, used to find the least squares fit of the line preceding the plateau
P[]	product of O[]*G[] matrices, used to find the least squares fit of the line preceding the plateau
Q[]	inverse of P[], used to find least squares fit of plateau line per equation 4.6
R[]	product of Q[]*O[], used to find the lease squares fit of the line preceding the plateau
S[]	slopes of plateau points
T[]	temperatures of plateau points
U[]	inverse of L[], used to find least squares fit of plateau line per equation 4.6
V[]	voltage input from thermocouple

Strings

R$	general response to query
X$	date and time stamp
F$	file number
S$	record data query response

Subroutines

"* temp K":	Newton-Raphson routine to solve for temperature
"disk record":	records data on disk
"find null data file":	finds blank area on disk to record data

Variables

C	past thermocouple temperature
D	current filtered slope of temperature (not used now)
E	past filtered slope of temperature (not used now)
F	digital filter constant
G	second derivative of temperature
H	peak second derivative of temperature
I	increment counter
J	general counter used in loops
K	counter used to determine number of data points falling within the phase change plateau duration
M	melting point of encapsulated metal, °C
P	variable used to find average plateau temperature
p*	general variables used in the Newton-Raphson subroutine to calculate temperature from thermocouple emf, where * = 1-99
R	scan time in minutes
r10	increment number of lowest second derivative for melting or highest second derivative for freezing
r11	variable that determines how many points to include in the least squares fit of the line preceding plateau and the plateau line
r12	the ratio of the least squares fit slopes of lines before and during the plateau
r13	reported phase change temperature calculated as in equation 4.2
r79	string disk byte requirement
r89	disk numeric byte requirement
S	slope of time-temperature curve
T	thermocouple temperature
U	scan time in seconds
V	thermocouple input voltage
Z	maximum threshold value of phase change plateau slope, °C/min

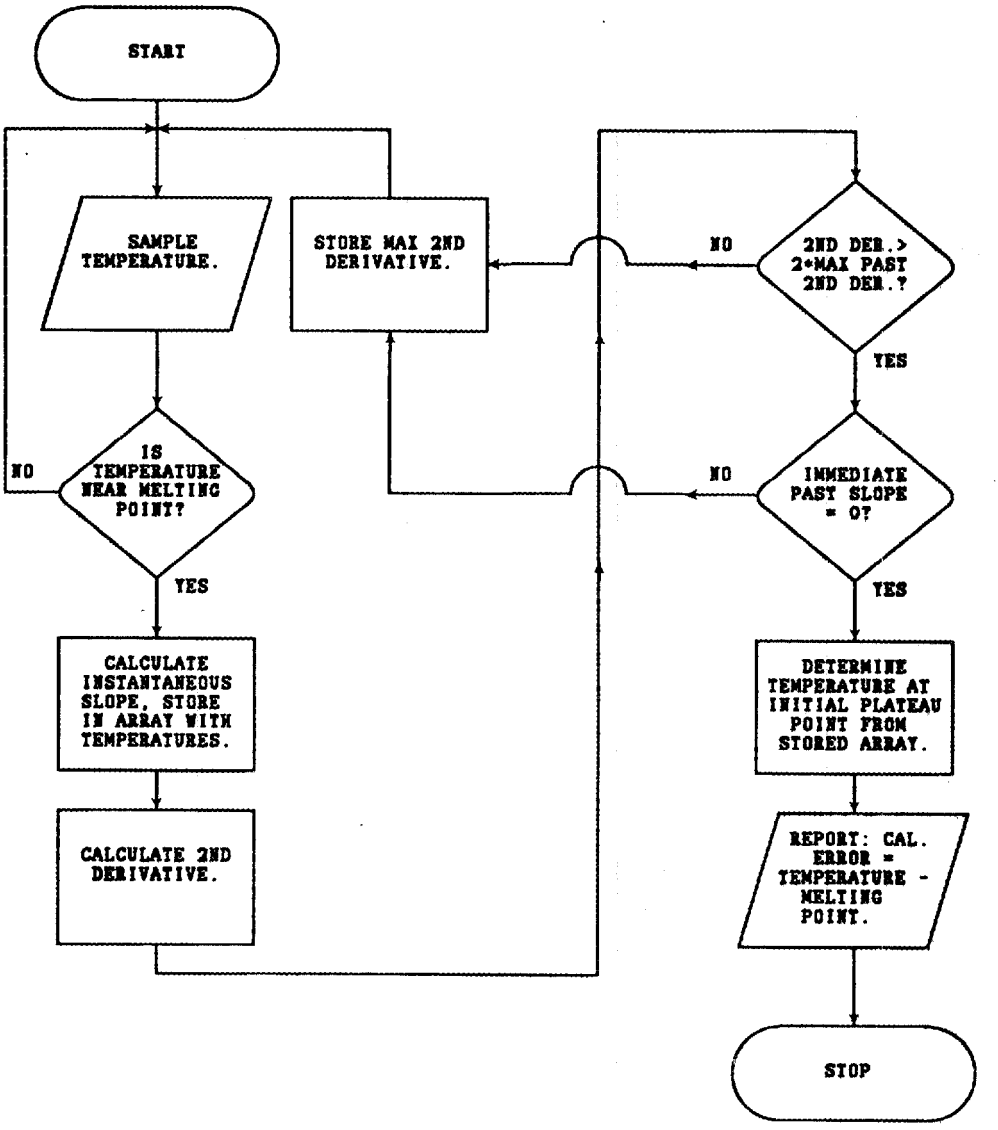


Fig. B.1 Simplified flowchart of phase change detection program.

03/17/88 09:35

Melting Point Determination - Zinc

T, degC	Slp, degC/min	2nd Der,degC/(min)↑2	Scan
401.61	0.3	0.6	30.0
402.01	0.8	0.9	30.0
402.52	1.0	0.4	30.0
403.13	1.2	0.4	30.0
403.88	1.5	0.6	30.0
404.69	1.6	0.2	30.0
405.55	1.7	0.2	30.0
406.35	1.6	-0.2	30.0
407.27	1.8	0.5	30.0
408.27	2.0	0.3	30.0
409.25	2.0	-0.0	30.0
410.26	2.0	0.1	30.0
411.28	2.1	0.1	29.0
412.36	2.2	0.3	29.0
413.45	2.4	0.4	27.0
414.49	2.5	0.2	24.0
415.46	2.4	-0.2	23.0
416.51	2.6	0.3	24.0
417.46	2.4	-0.4	23.0
418.47	2.5	0.1	24.0
419.37	2.2	-0.6	24.0
419.86	1.1	-2.5	27.0
420.00	0.3	-1.7	27.0
420.08	0.2	-0.3	27.0
420.18	0.2	0.1	27.0
420.48	0.7	1.0	27.0
420.92	1.0	0.7	27.0
421.44	1.2	0.4	27.0
425.50	9.0	17.5	27.0

Calibration Temperature: 420.33 Error: 0.75

New Method: 419.42 Error: -0.16

Data on DISK F.R.Ruppel, Data #2, File 45

Fig. B.2 Typical output from phase change detection program.

ORNL/TM-11127
Dist. Category UC-506

INTERNAL DISTRIBUTION

1. R. L. Anderson
2. H. R. Brashear
3. B. G. Eads
4. D. N. Fry
5. J. M. Googe
6. W. R. Hamel
7. D. W. McDonald
8. D. R. Miller
9. O. B. Morgan
10. L. C. Oakes
11-20. F. R. Ruppel
21. R. E. Uhrig
22. J. B. Ball, Advisor
23. P. F. McCrea, Advisor
24. T. B. Sheridan, Advisor
25-26. Central Research Library
27. Y-12 Technical Reference Section
28-29. Laboratory Records Department
30. Laboratory Records ORNL-RC
31. ORNL Patent Section
32. I&C Division Publications Office

EXTERNAL DISTRIBUTION

33. Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, Oak Ridge, TN 37831.
34-35. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831.
36. T. W. Kerlin, The University of Tennessee, Knoxville, TN 37996-2300.
37. L. L. Eisenmann, Leiko Industries, Inc., 250 W. 57th St., New York, NY 10019.
38. J. P. Tavener, Isothermal Technology, Ltd., Pine Grove, Southport, Merseyside PRQ 9AG England.
39-169. Given distribution under UC Category 506.