AN ABSTRACT OF THE THESIS OF

Abdullah S. Al-Kheliewi for the degree of Doctor of Philosophy in Nuclear Engineering presented on Sept. 20, 1993.

Title: Modeling Transient Thermalhydraulic Behavior of a Thermionic Fuel Element For Nuclear Space Reactors

Abstract Approved: Redacted for Privacy
Andrew C. Klein

A transient code (TFETC) for calculating the temperature distribution throughout the radial and axial positions of a thermionic fuel element (TFE) has been successfully developed. It accommodates the variations of temperatures, thermal power, electrical power, voltage, and current density throughout the TFE as a function of time as well as the variations of heat fluxes arising from radiation, conduction, electron cooling, and collector heating. The thermionic fuel element transient code (TFETC) is designed to calculate all the above variables for three different cases namely: 1) Start-up; 2) Loss of flow accident; and 3) Shut down.

The results show that this design is suitable for space applications and does not show any deficiency in the

performance. It enhances the safety factor in the case of a loss of flow accident (LOFA). In LOFA, it has been found that if the mass flow rate decreases exponentially by a -0.033t, where t is a reactor transient time in seconds, the fuel temperature does not exceed the melting point right after the complete pump failures but rather allows some time, about 34 seconds, before taking an action. If the reactor is not shut down within 34 seconds, the fuel temperature may keep increasing until the melting point of the fuel is attained. On the other hand, the coolant temperature attains its boiling point, 1057 °K, in the case of a complete pump failure and may exceed it unless a proper action to trip the reactor is taken. For 1/2, 1/3, and 1/4 pump failures, the coolant temperatures are below the boiling point of the coolant.

Modeling Transient Thermalhydraulic Behavior of a Thermionic Fuel Element for Nuclear Space Reactors

by

Abdullah S. Al-Kheliewi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed September 20, 1993

Commencement June, 1994

APPROVED:

Redacted for Privacy

Andrew C. Klein, Ph.D.
Assoc. Professor of Nuclear Engineering in charge of major

Redacted for Privacy

Alan H. Robinson, Ph.D.
Head of department of Nuclear Engineering

Redacted for Privacy

for Thomas J. Maresh, Ph.D.
Dean of Graduate School

Date thesis is presented September 20, 1993

Typed by Abdullah S. Al-Kheliewi

ACKNOWLEDGEMENT

First, Thanks to [Allah], the Almighty, who guided me to the right path and peace and blessings of Allah be upon his messenger Mohammad, the greatest man ever known in the human history.

Second, This thesis is dedicated to my parents who devoted their life to my best. Without their love, care, and sacrifice I would not be who I am.

Third, I would like to express my gratitude to my "boss", Dr. Andrew C. Klein who deserves credit for the bulk of this work. Thank you, Andy, for your invaluable support during the course of my study, for encouraging and helping me doing this immense work, and for teaching me. I could not have found a better major professor, teacher, mentor, and friend had I searched with intention.

Fourth, thanks to the closest friend to my soul; My wife. Her encouragement, help, patience and unlimited support will never be forgotten. Gratitude is also extended to my lovely kids; Haifa and Salih.

I am also indebted to my advisor, in charge of minor, Dr. Brian Dodd and to my graduate committee members: Dr. Jack F. Higginbotham, Dr. M. M. Kulas, and Dr. John Peterson for their time and assistance.

Table of Contents

Table of Contents(Continued)

Chapter | Page

Table of Contents(Continued)

List of Figures

List of Figures(Continued)

List of Figures(Continued)

List of Figures(Continued)

List of Figures(Continued)

List of Figures(Continued)

List of Figures(Continued)

Modeling Transient Thermalhydraulic Behavior of a Thermionic Fuel Element for Nuclear Space Reactors

Chapter 1

Literature Review

1.1 Space Reactors

Nuclear power reactors play an important role in every aspect of today's technology not only on our planet but in outer space. For space missions, it has been found that nuclear technology can be useful in providing power for systems operation in earth orbit, on the moon, on Mars, and in deep space.

In early 1961 the United States Atomic Energy Commission initiated the SNAP-10A (Systems for Nuclear Auxiliary Power) program. It was then developed by Atomic International Division of North American Aviation, Inc., and the conversion unit by RCA. SNAP-10A is a liquid metal cooled reactor designed and developed to provide a minimum of 500 watts for one year in space. The goals were (1) to prove that thermoelectric reactors are reliable in space, (2) to provide sufficient data for designing another system of high performance and excellent integrity, (3) to verify that this type of reactor can generate power and can be controlled by remote command from the ground, and (4) to demonstrate safety criteria for reactors in outer space. The SNAP-10A was connected to the forward end of an Atlas-Agena rocket (see Figure 1.1) and the launch took place at 1:24 p.m. on April 3, 1965 from Point Arguello, California on a 700 n.m. (nautical mile) target circular orbit and achieved a 717 n.m. apogee and 699 n.m. perigee. A start command for reactor operation was given at 5:05 p.m., and the reactor reached criticality at 11:15 p.m.

The SNAP-10A reactor functioned for 43 days before being permanently shutdown by a voltage regulator malfunction. Although it remains in a long-lived orbit, portions of the satellite have begun to break up. [2,8,9,11,12,13,15].

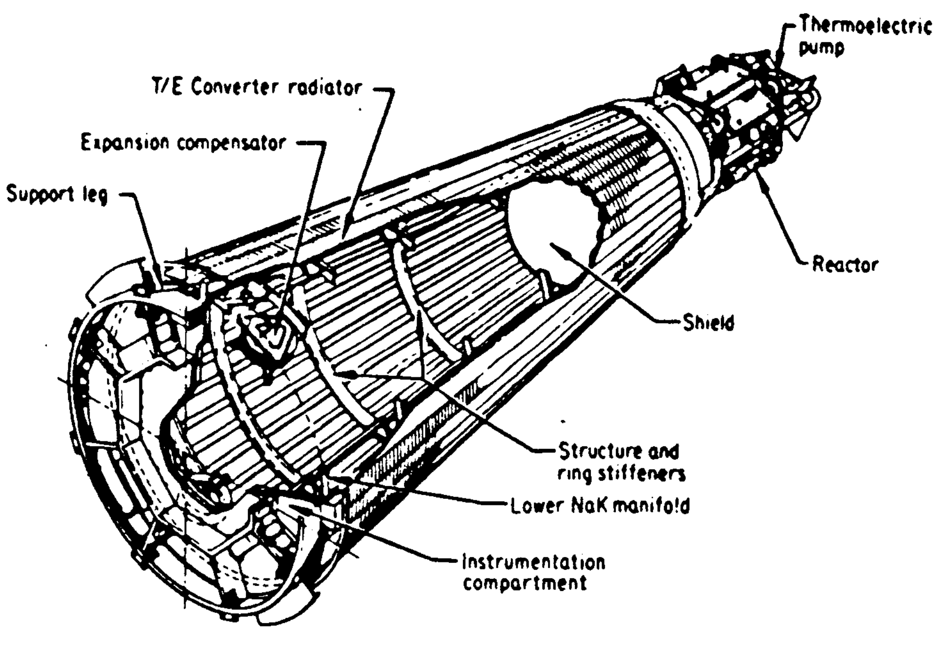

Figure 1.1. SNAP-10A Nuclear Space Reactor

Between 1961 and 1971, the U.S. launched a total of 23 spacecraft powered by more than thirty six radioisotope thermoelectric generators (RTG's) and one nuclear reactor, SNAP-10A. The former USSR has launched about 35 nuclear reactor-powered satellites and several RTG-powered satellites and is currently considered to be the only nation to use nuclear satellites in orbits.

Current U.S. space reactor development effort is focused on the SP-100 reactor, a joint program of the Defense Advanced Research Projects Agency, the Department of Energy's Office of Nuclear Energy, and NASA's Office of Aeronautics and Space Technology [2,55]. The SP-100, as shown in Figure 1.2, is a thermoelectric reactor designed to generate 100 KW of electricity continuously for seven years. The SP-100 is a fast spectrum reactor, fueled with about 190 Kg of uranium nitride fuel enriched to an average of 96% U-235 and cooled by liquid lithium metal. The reactor core is small (less than 1 m^3) [2].

Two types of nuclear power systems were implemented by the former USSR, "TOPAZ" and "TOPAZ-II" [7,52,56]. TOPAZ depends in its operation on multicell thermionic converters, while TOPAZ-II depends in its operation on single cell thermionic converters. The Soviets have sold TOPAZ reactors to the U.S. The former Soviet Union has moved far ahead of the U.S. in operational use of space nuclear power. TOPAZ thermionic reactors, each providing 10 Kw of power, were launched in 1987 into high orbits of about 800 naut. mi.

altitude to ensure safe operation. TOPAZ and TOPAZ-II operated for six months and one year respectively.

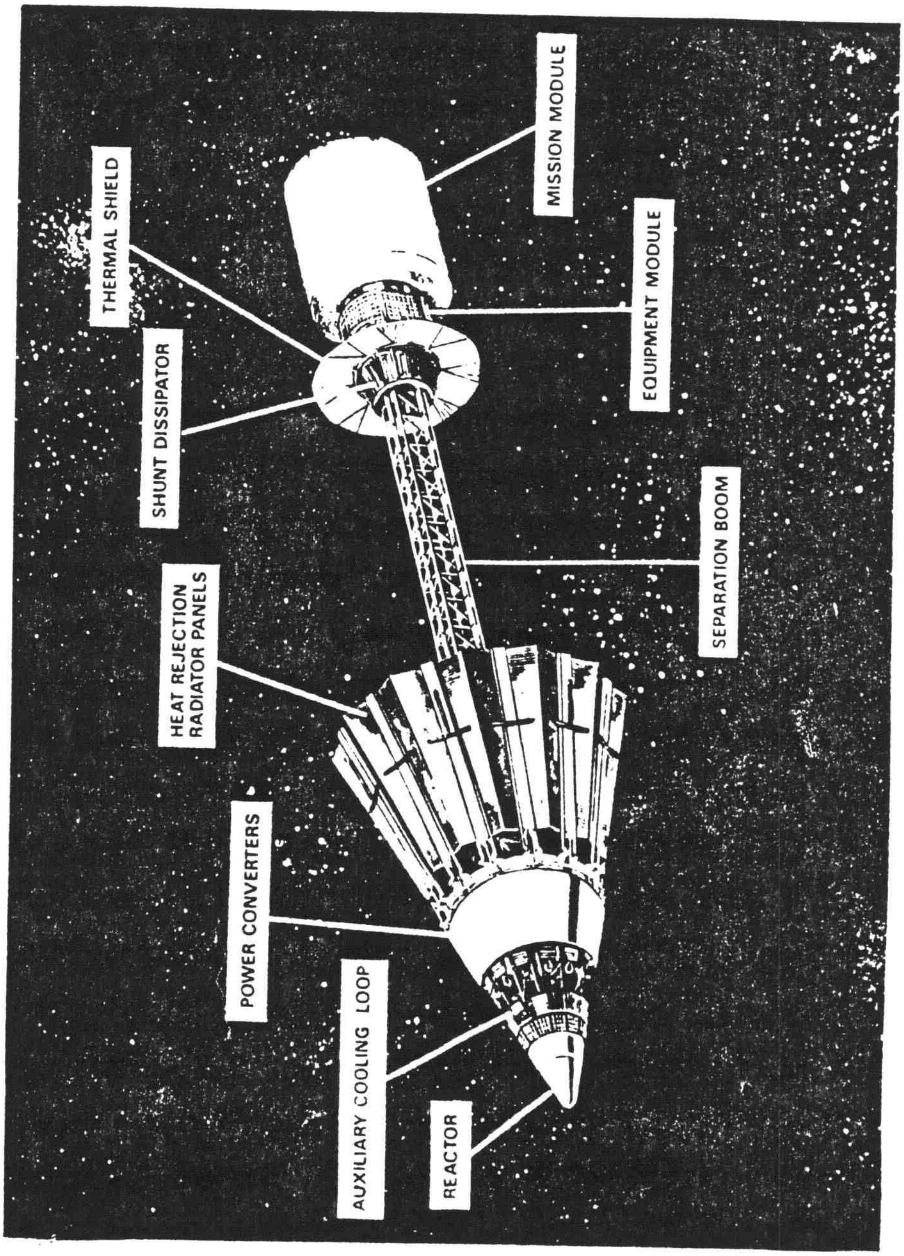

Figure 1.2 SP-100 reactor deployed configuration
(Source: Jet Propulsion Laboratory)

Most of nuclear space reactors depend in their operation on thermionic converters [1] because of the following advantages listed below:

1. No moving parts connected to the reactor and modular structure, which gives high reliability performance.
2. High rejection temperatures that allows a reduction in overall size of the power system.
3. The conversion of heat to electricity is of higher efficiency.
4. Quiet operation.

1.2 Thermionic Converter

1.2.1 Historical Introduction:

Thermionic emission phenomenon was first known by Edison in 1883, according to his patent application " I have discovered that if a conducting substance is interposed anywhere in the vacuous space within the globe of an incandescent electric lamp and said conducting substance is connected outside the lamp with one terminal, preferably the positive one of the incandescent conductor, a portion of the current will, when the lamp is in operation, pass through the shunt circuit thus formed, which shunt includes a portion of the vacuous space within the lamp. The current I have found to be proportional to the degree of incandescence of the conductor or candle power of the lamp."[13]. Further studies were extended by Schlicter in 1915. His efforts were focused on one type of thermionic converter called a vacuum thermionic converter. Surprisingly, no further studies had been conducted in the thermionic area until 1933 when Longmuir achieved considerable insight in understanding the methodology and physics of thermionic emission. During these efforts he constructed several types of thermionic converters. The progress in this area of research went slowly until 1956 when Hatsopoulos described two types of thermionic converters in his doctoral thesis at the Massachusetts Institute of Technology. However, in 1956 and after, many studies have been conducted and received more attention than before. In 1956, Moss [16] published a very good paper on using thermionic diodes as energy converters. Wilson, also, published a paper about thermionic phenomenon and converters in 1958. Several dozens of papers and tenfold times this number of surveys, digests, proceedings, etc., have been published exclusively from the U.S.A and the former U.S.S.R. The U.S. and former U.S.S.R. [62] took different approaches in thermionic reactor development. By 1973 the U.S. had achieved its thermionic fuel element lifetime and performance objectives and was planning to construct a test reactor. The former Soviet Union began ground testing its low power TOPAZ thermionic reactor in 1970, and ground-tested eight versions by 1983. In 1973 the U.S. discontinued its thermionic reactors program as well as space nuclear power program but resumed them again in 1983. In 1987 and 1988 the former Soviet Union announced operation and testing of two of its 6-KW TOPAZ thermionic reactor systems. In 1992, the former Soviet Union sold the TOPAZ and TOPAZ-II reactors to the U.S. Recently, the U.S. has conducted very good efforts for developing the technology and the operation of thermionic reactors and converters as well.

1.2.2 Basic Physical Principles:

The thermionic conversion system is a device in which heat is converted directly to electricity. Thermionic conversion phenomenon is based on a device called a thermionic converter (see Figure 1.3) which consists of a metal surface connected to the heat source and a secondary surface acting as an electron collector. The emitter emits electrons upon heating by a heat source and all emitted electrons transfer through the interelectrode space between the emitter and collector. Upon reaching the collector surface, which is kept at a temperature lower than that of the emitter to prevent any back emission toward the emitter that may affect the output power and efficiency of the thermionic converter, the electrons condense and return to the hot electrode via the electrical leads and the electrical load connected between the emitter and the collector. The flow of electrons through the electrical load is sustained by the temperature difference between the emitter and the collector [1,4,9-14].

To understand the operation of a thermionic converter, it is important to discuss several surface and solid phenomena, such as conduction electron energies, thermionic emission, and surface ionization; as well as space phenomena, such as negative space charge and plasma transport properties.

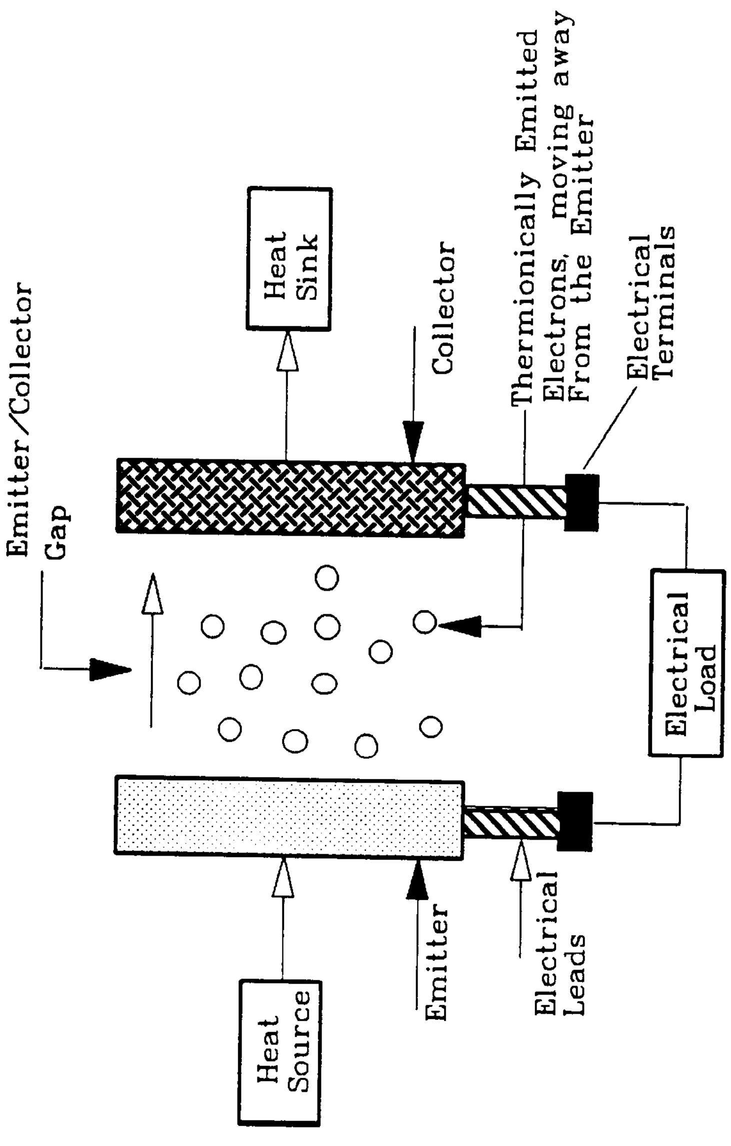

Figure 1.3 Thermionic Energy Converter

1.2.2.1 Solid Phenomena: The thermionic properties of any thermionic converter depend greatly on the crystallographical distribution of the surface of the emitter and the collector. The atom is made up of a positively charged nucleus surrounded by a different number of negatively charged electrons. The number of orbits and the number of electrons in each orbit depend actually on the type of the atom and consequently on the type of material. There are attractive forces between the nucleus and the surrounding electrons due to the opposite charges they carry. The valence (free) electrons are those types of electrons that are located usually in the outer or the far orbit from the nucleus so that they are weakly bound to the nucleus and free to move around inside the metal, while the nearby electrons are tightly bound to the nucleus. The valence (conduction) electrons are responsible for the mechanism of heat and electric conduction in metals. At the surface boundary, a potential energy barrier exists, since there are no positive ions on one side of the boundary to give the free electrons equal attractive forces. The electrons are attracted then by their image forces. The free electrons need more energy to boil them out of the metal into free space [9-14].

1.2.2.2 Surface Phenomena: The electron leaving a solid surface experiences a net positive charge inside the metal at the boundary. The electron needs energy to overcome the potential barrier and to be released from the emitter surface. This needed energy must be equal to the work required to raise it from the Fermi level, which is the highest energy level occupied by free electrons at absolute zero temperature (0°K) at which none of the electrons can escape, to a point outside the metal. This energy is called the work function of the metal and varies according to the type of material and some other factors. The work function can also be defined as the energy required to overcome the force exerted on the electron from its image force of positive charge of magnitude e as shown in Figure 1.4. The work function of a material depends somewhat on the crystallographic face exposed. The work function for most materials falls in the range from 1 to 6 ev. At low temperatures, some electrons possess enough initial

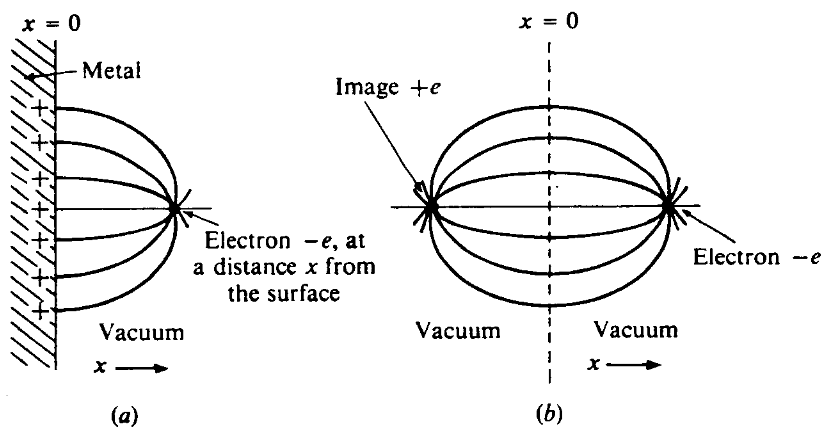

Figure 1.4 Image forces exerted on electron in surface metal [13].

kinetic energy to exceed the potential barrier of the emitter, which is equal to the product of the electron charge e and the work function in volts ($e\phi$ V), and get into the emitter-collector gap and reach the collector surface, while others do not. The situation is different at high temperatures due to an increase in the number of electrons that possess enough kinetic energy to leave the emitter surface.

The rate of electron emission is given by the Richardson-Dushman equation,

$$J=\frac{4\pi e m_e k^2}{h^3}T^2\exp\frac{-\Phi}{KT} \tag{1.1}$$

where

J = Rate of electrons emitted in amp/cm^2

e = Electron charge

m_e= Mass of electron (9.10909 x 10^{-28} gm)

h = Planck's constant (4.13576 x 10^{-15} ev sec)

T = Surface temperature, ^{o}K

ϕ = Work function, volt

K = Boltzman constant = 8.62 x 10^{-5} ev/^{o}K

Equation 1.1 can be written as

$$J=AT^2\exp\frac{-\Phi}{kT} \tag{1.2}$$

where

A = Richardson's constant

$= 4\pi e m_e k^2/h^3$

$= 120$ amp/cm^2.K^2

The Richardson-Dushman equation is only valid in a vacuum, and in a gas when the electron mean free path (mfp) is considerably greater than the distance from the emitter to the potential barrier [25]. The electrodes (emitter and collector) in a thermionic converter have different Fermi levels; the emitter has a low Fermi level whereas the collector has a relatively high Fermi level. The electron [13] in the emitter surface needs a larger energy to be lifted out of the emitter than would a corresponding electron to be lifted out of the collector. Thus the emitter work function is greater than the collector work function.

1.2.2.3 Space (Gap) Phenomena: There are two phenomena that better describe the operation of thermionic converters. The first one is the emission phenomenon which depends mainly on the emitter-collector materials, properties of the surface, and crystallographic structure of the surface. The second one is the transport phenomenon which describes the processes in which electrons migrate from the emitter and interact in the emitter/collector space.

In the interelectrode space between the emitter and the collector, the electrons (charged particles act as a working fluid in the emitter/collector space) are emitted from the refractory metal that possesses a high electron emission rate (usually tungsten) and condense on the collector surface. The speed of these electrons is limited in which they take some time (in terms of nano-seconds) to reach the collector. During the electrons' travel, they form a cloud of free negative electrons called "negative space charge".

This cloud of electrons will repel electrons emitted later back toward the emitter unless they have sufficient initial kinetic energy to overcome the repulsion and reach the collector surface. There is no doubt that the negative space charge affects the output current and consequently the efficiency of the thermionic converter and some precautions must be taken to suppress the electrostatic effect of this negative space charge. The classification of thermionic converters is based mainly on the type of suppression of the negative charges. Suppression can be achieved by several methods. These methods are described as follows:

1.2.3 Close-Space Vacuum Thermionic Converter:

In a vacuum thermionic converter, heat is supplied to the emitter surface and some electrons gain energy that raise them up from Fermi level until they reach the minimum potential or the emitter work function, ϕ_E as shown in Figure 1.5. The electrons still need an extra potential to overcome the space charge potential barrier so that they may not return to the emitter surface. The potential required is $(V_E - \phi_E)$ which is the potential difference between the top of the potential barrier [40] and the Fermi level of the emitter. Therefore the effective emitter (cathode) work function V_C is given by

$$V_E = \phi_E + V_{ES}$$

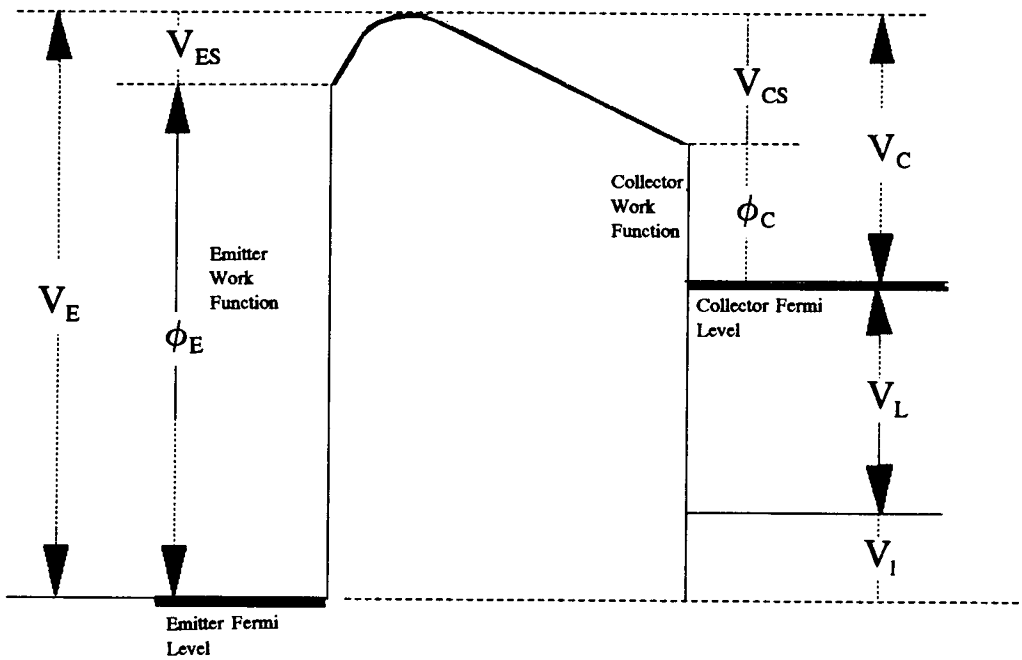

Figure 1.5 Potential diagram of a vacuum thermionic converter.

The electron that possesses a potential, equivalent to the effective work function, overcomes the hump or the potential peak and is accelerated towards the collector (anode) surface. Upon reaching the collector surface, the electron falls down on a potential energy scale by an amount equal to the work function of the collector surface and releases an effective collector potential V_C and an energy eV_C until it reaches the collector Fermi level. This energy appears as heat in the collector surface and is given by

$$V_C = \phi_C + V_{CS}$$

$$eV_C = e(\phi_C + V_{CS})$$

It is extremely important that the collector work function should be smaller than the emitter work function to allow a net potential difference which can be connected to a useful load, V_l between the emitter/collector surfaces. The energy loss through electrical leads, V_L, as a result of their electrical resistance should be subtracted from the useful (electrical) energy before reaching the emitter Fermi level.

The space between the two electrodes in a vacuum thermionic converter is very narrow so that no appreciable space charge can build up in the evacuated space between them. It has been found that a spacing of 0.001 cm (10μ) or less is standard for these types of converters (Figure 1.6) as was confirmed experimentally by Hatsopoulos and Kaye [14] in 1958. They obtained an estimated 12-13% efficiency at this spacing.

It has been concluded [1] that a close-space vacuum converter is not practical and has some disadvantages such as:

1. Difficulty of manufacturing prevents the attainment of interelectrode gap (spacing) of less than about 10 μ.

2. No materials have been found to be usable as an emitter in a vacuum converter because all materials produce excessive evaporation which is not desirable because it (a)

limits the useful life of the emitter, (b) causes an electrical short between the emitter and the collector, and (c) alters the work function of the collector and makes it approach that of the emitter. All these undesirable effects can be avoided by introduction of a suitable rarefied vapor such as cesium.

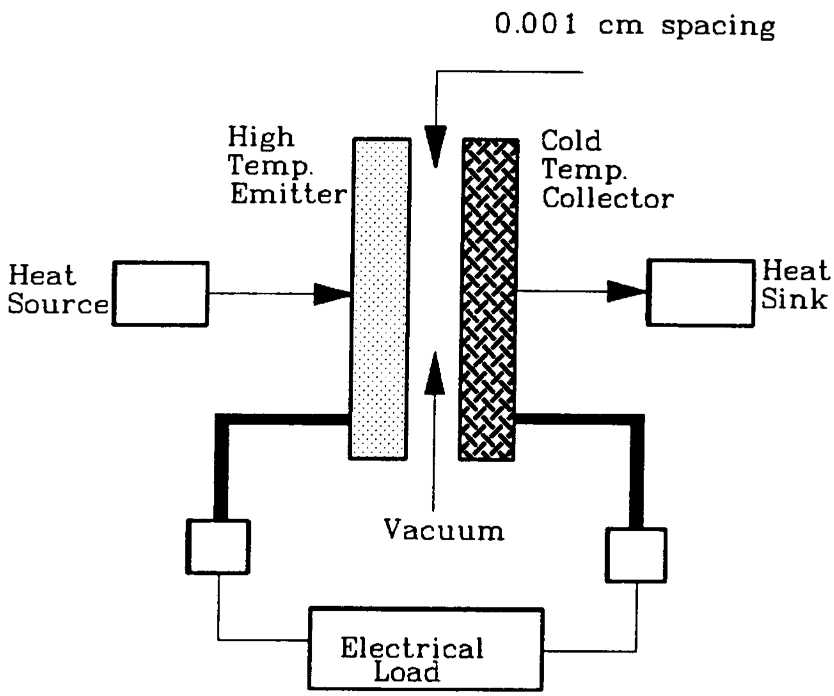

Figure 1.6 Close-space vacuum thermionic converter

1.2.4 Cesium Vapor Thermionic Converter:

The best way to overcome the negative space charge in the emitter/collector gap is to introduce a rarefied cesium vapor. The reasons for choosing this kind of vapor are because of 1) its low ionization potential (3.89 ev), lower than that of the emitter, to completely neutralize the cesium atoms which impinge on the emitter surface and lose their outermost electrons then evaporate as positive ions, and 2) it is the most easily ionizable of all the stable gases. (see Figure 1.7).

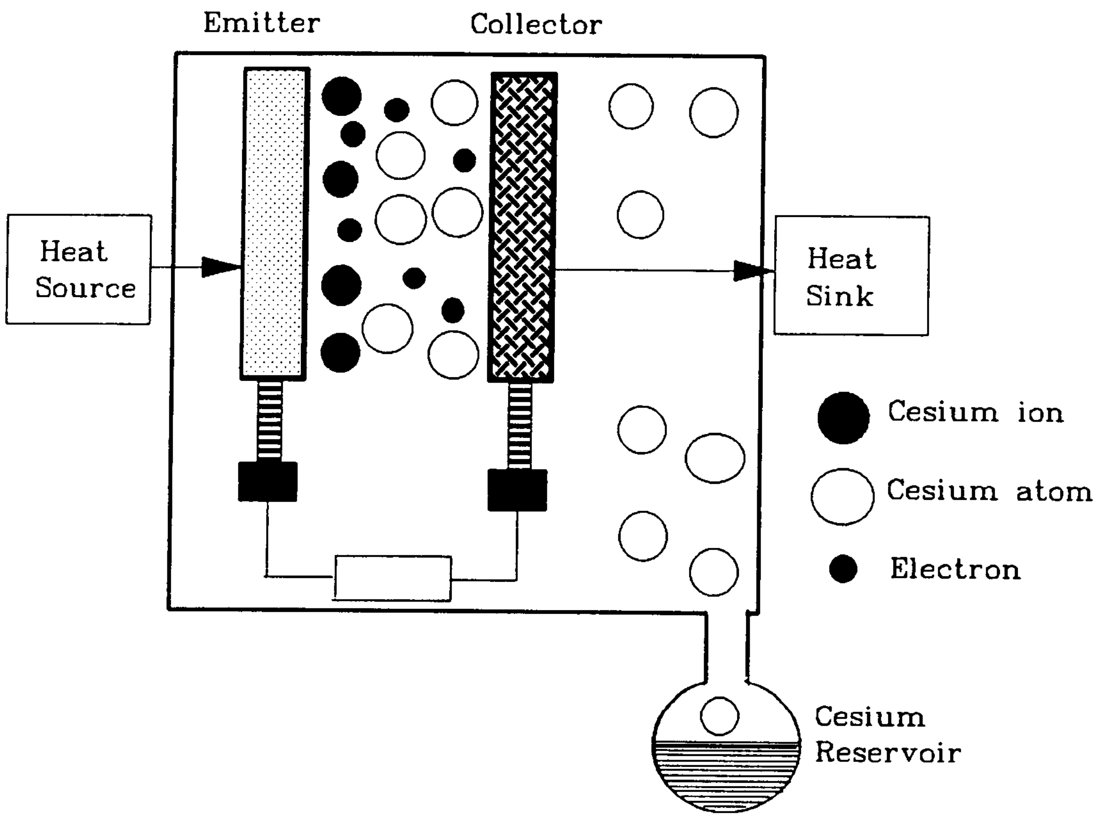

Figure 1.7 Cesium Thermionic Converter

The cesium atoms will be partially ionized when touching the hot emitter surface and consequently some ions are formed. The positive charge of the cesium ions will neutralize the negative charge of the electron cloud.

There are two modes for the operation of thermionic converters. These modes are 1) ignited (ball of fire) mode and 2) unignited mode. In the latter, a cesium atom comes into contact with a hot surface (contact ionization) if the ionization potential of the atom is lower than the work function of the surface. The valence electron of the gas atom detaches from the atom and attaches instead to the surface material. If the surface is hot enough, the electron is then emitted, and an electron ion- pair are produced at the surface. The plasma (a mixture of positive and negative charged particles) is maintained entirely by thermionic emission of positive ions from the emitter. The rate of production depends mainly upon the cesium vapor pressure, which in turn depends upon the cesium reservoir temperature. It has been found that for the most effective rate of electrical power the emitter temperature must be at least 3.6 times the cesium reservoir temperature [9,33,47]. The motive diagram for the unignited plasma is shown in Figure 1.8. In the unignited mode, at low cesium vapor pressure (10^{-4} mm Hg), the mean free path of electrons in the emitter/collector gap is larger than the gap itself so the inelastic collisions are negligible. Also the negative space charge is partially neutralized, while at high cesium pressure, where the collisional processes are considered, it is completely neutralized. This mode of operation is impractical because 1) it requires high emitter surface temperatures (>1900 °K) that may cause some metallurgical problems and 2) the output power densities and currents are small.

In the ignited mode as illustrated in Figure 1.9, part of the electric power generated by the converter [33] is dissipated internally in the interelectrode gas by collisional processes. This mode of operation is more efficient than the unignited mode because of the high power densities output and efficiencies. The cesium vapor pressure is relatively high (1 mm Hg or higher) and the electron collisions are taken into consideration. The electron mean free path is much smaller than the emitter/collector space. The majority of all thermionic converters in operation today operates in the ignited mode [13]. The so called ball of fire mode refers to an external power source, whereas the arc, or ignited, mode refers to internal heating by the emission current. This mode of operation can be classified into two regions: one of bright plasma and the second of dark plasma. In the dark region the electrons do not possess enough energy to ionize significant number of cesium atoms but neutralization occurs due to the ion flow from the bright region which is caused by the inelastic collisions. Ions produced in this mode are capable not only of neutralization of cesium vapor, but also of producing a strong positive space charge. The ideal performance in the ignited mode can be achieved by firstly complete reduction of the negative space charge and secondly by reduction of the large internal voltage drop. This reduction as shown in Figure 1.10, is simply to minimize the product of the cesium vapor pressure times the emitter/collector gap.

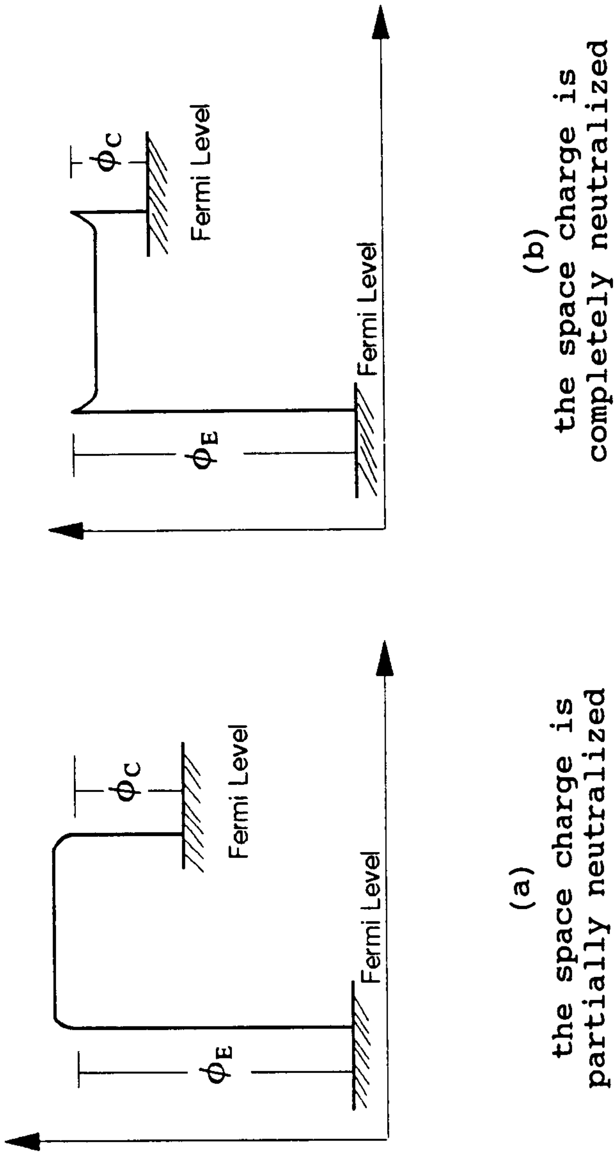

Figure 1.8 Motive Diagram (Unignited mode)

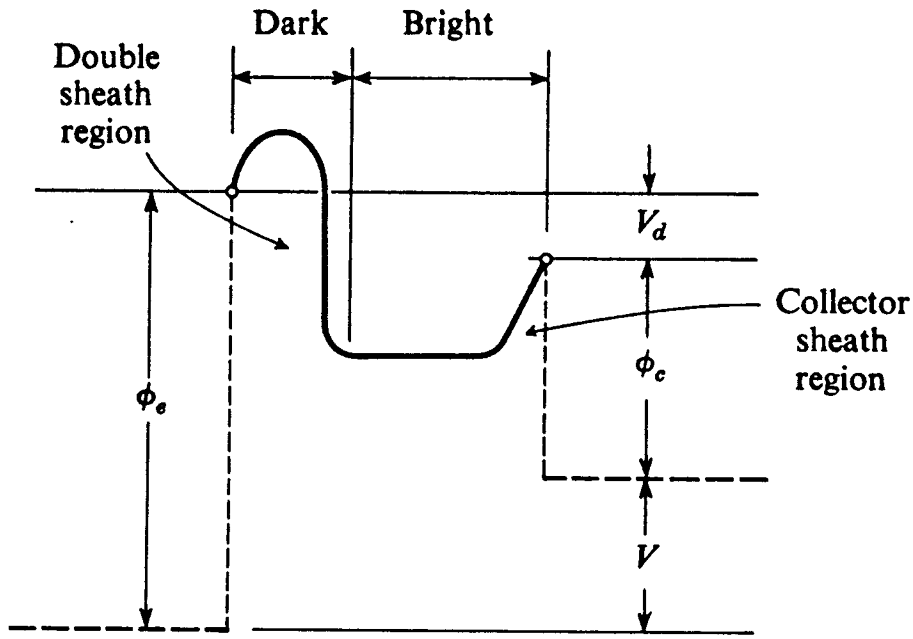

Figure 1.9 Motive Diagram (Ignited mode)[15]

1.2.5 The Ideal Thermionic Converter:

The ideal thermionic converter assumes that there is no negative space charge that may affect the transmission of electrons from the emitter to the collector. The potential between the barrier heights of the electrodes (emitter and collector) must be continuous [33]. The motive diagram for the ideal diode thermionic converter is illustrated in Figure 1.10. For an electron to move into the interelectrode gap, it must experience forces that overcome the potential energy barrier or the emitter work function ϕ_E. An energy barrier V + ϕ_C must be overcome to allow an electron to move into the gap and reach the collector surface when the electrode potential energy difference (output voltage) V is greater than the contact potential energy difference $V_o = \phi_E - \phi_C$. When V is less than V_o, a barrier ϕ_E must be overcome. Neglecting electron emission from the collector, the output current density of the ideal diode thermionic converter is given by the Richardson-Dushman equation:

$$J = AT_E^2 \exp\left(-\frac{V+\Phi_C}{kT_E}\right) \qquad \text{for } V>V_o \qquad (1.3)$$

$$J = AT_E^2 \exp\left(-\frac{\Phi_E}{kT_E}\right) \equiv J_{s_E} \qquad \text{for } V<V_o \qquad (1.4)$$

where J_{Se} is the saturation current density for the emitter

The total heat that must be supplied to the emitter is

$q_E = q_e + q_r + q_{el}$

where

$q_e = J(\phi_E + 2kT_E)$ = Emitter electron cooling

$q_r = \sigma\varepsilon(T_E^4 - T_C^4)$ = Heat removed by radiation

q_{el} = Heat conducted down the emitter lead

The optimum ideal performance for the ideal thermionic converter depends mainly on the optimum choice of thermionic properties values that allows the attainment of the maximum possible ideal efficiency [1]. Emitter temperatures between about (1500 to 2000 °K) define the region of most attractive operation of ideal thermionic converter. It has been found that [1] at an emitter temperature of 2300 °K, the output current density is about 100 amp/cm^2 which seems attractive but in reality it is impractical because of the difficulty of handling high current densities and because of the extreme difficulty and expense of operating the heat source at very high temperatures. An ideal current between 5 and 50 amp/cm^2 can be achieved in the presence of suitable materials. The heat radiation flux term, Q_{Rad}, reduces the efficiency of the ideal thermionic converter at higher temperatures because the emissivity of refractory metals increases with temperature. The optimum emissivity value falls in the range (0.1 to 0.2). The <0.1 emissivity is not maintainable and >0.2 emissivity is not desirable [1,25]. For the collector work function, ϕ_c is restricted to values greater than about 1.5 ev. The collector temperature should not exceed 1000 °K. At the same time the collector temperature can not be taken at very low temperatures because of the need to reject heat at a reasonable temperature level.

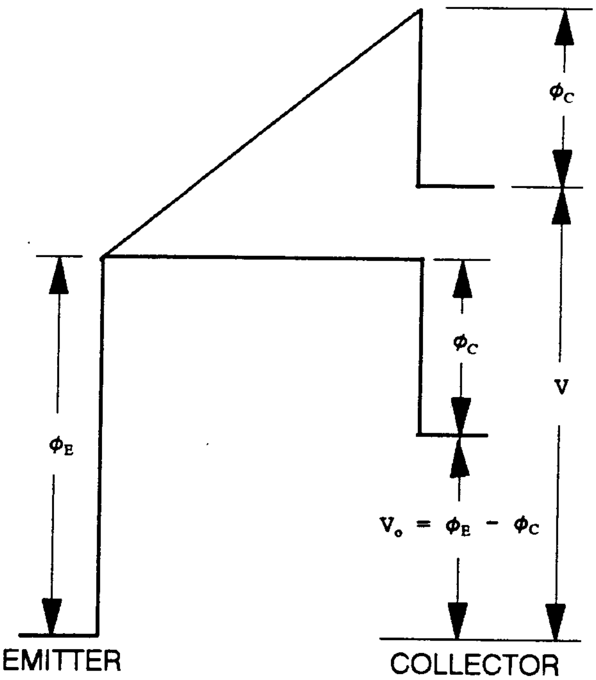

Figure 1.10 The ideal Motive Diagram of Thermionic Converter.

1.2.6 Heat Sources:

The emitter in a thermionic reactor needs to be heated in order to emit electrons into the emitter/collector gap. There are many kinds of heat sources that may be of use for this purpose. The choice of the heat source depends mainly on the type of application, time of operation, space, cost, and several other factors.

For thermionic converters, there are three kinds of heat sources to be listed as: 1) Chemical source; 2) Solar source; and 3) Nuclear source.

1. Chemical source: Fossil fuel can be used but can not be recommended as a heat source for thermionic reactors due to the following deficiencies:

a. Large mass that takes large space which is not desirable for space applications.

b. Limited life due to the fast rate of burn-up of the chemical feed stock.

c. Regular maintenance is always needed to avoid poisoning converter elements by their products and corrosion.

d. Ventilation is required to expel the undesirable smoke into space which may, in turn, cause some hazards.

2. Solar source: Solar energy is a very cheap source of energy and is not life-limited as in the case of chemical source. Parabolic reflectors are required to concentrate the heat on the emitter surface. This type of heat sources is not practical due to its high cost and large size.

3. Nuclear source: Nuclear fuel is the most efficient source of energy for thermionic reactors for several considerations:

a. Long life in space due to the long half live of uranium-235 (i.e., 7.13×10^8 years). The fuel burn-up rate is so small because the electrical power produced in thermionic systems is so small.

b. Low maintenance requirements due to the safety precautions for these types of reactors. In the case of any unexpected failure in the operating system, the shut down and emergency systems overcome the problem.

c. Small size core. The fission of a single uranium-235 nucleus is accompanied by the release of about 200 MeV of energy, while the energy released by a combustion of one carbon-12 atom is 4 ev. Hence, the fission of uranium yields something like 3 million times as much energy as the combustion of the same mass of carbon. In other words, the energy produced by 1 kg of uranium is equivalent to the energy produced by 2,700 metric tons of coal[57].

The only disadvantage of a nuclear fuel is the requirements for heavy masses of shielding to prevent any radioactive release in space.

1.2.7 Efficiency:

The efficiency of a thermionic converter depends on many factors such as: 1) The temperature of the emitter and collector, 2) The cesium reservoir temperature, 3) The type of materials used as emitter or collector, 4) The suppression of the negative electron space charge, 5) The pressure of the cesium vapor, 6) The work function of both the emitter and the collector, 7) The size of the emitter/collector gap, 8) emissivity characteristics of the emitter and collector surfaces, 9) The electrical power output, and finally 10) the impurities on the emitter and collector surfaces [1]. The efficiency can be defined as the electrical power output per unit area of emitter divided by the emitter heat input per unit area of emitter.

The power output = $(J_E - J_C)(V_E - V_C)$

where

J_E = Emitter current density (amp/cm^2).

J_C = Collector current density (amp/cm^2).

$(J_E - J_C)$ = Net current flow between emitter and collector (amp/cm^2).

$(V_E - V_C)$ = Output voltage (volt).

The efficiency of thermionic converter can be given as

$$\eta = \frac{(J_E - J_C)\,(V_E - V_C - V_L)}{Q^{Rad} + Q^{k} + [Q_L - \frac{Q_d}{2}] + Q^{EC} - Q^{CH}} \tag{1.5}$$

where

Q^{Rad} = Radiation heat flux (watt/cm^2).

Q^{EC} = Emitter electron cooling (watt/cm^2).

$= J_E\ (eV_E + 2kT_E)$

Q^{CH} = Collector electron heating (watt/cm^2).

$= J_C\ (eV_C + 2kT_C)$

Q^{k} = Heat conduction through cesium and structural components. (watt/cm^2).

V_L = Voltage drop across the leads (volt).

$(Q_L - Q_d/2)$ = Heat conduction through electrical leads (watt/cm^2).

$Q_d/2$ = One half of the Joulean heat generated in the leads that transfers back to the emitter.

The emitter surface temperature is very high with respect to the collector surface temperature so the current flow towards the emitter is very small because the back emission of electrons is very small so that it can be negligible (i.e. $J_c = 0$) so that the net current is J_E. Equation 1.5 can be rearranged and written as

$$\eta = \frac{J_E V}{Q^{Rad} + Q^{EC} + Q_k + [Q_L - \frac{Q_d}{2}]} \tag{1.6}$$

where

$V = V_E - V_C - V_L$

$J = J_E$

If the voltage drop across the leads is considered small, one can play with equation (1.6) by variation of many parameters. For example, if $V_E = V_C$, that leads to zero efficiency. As V_C is lowered, η increases until, at some point, the collector begins to back-emit. The efficiency goes through a maximum [25] at the V_C value given by ($V_C = V_E\, T_C/T_E$). At this optimum value of V_C the back emission is

$$J_a=\left(\frac{T_C}{T_E}\right)^2 J_E \tag{1.7}$$

If V_C is lowered further, the back emission rapidly increases, and η falls to zero when $J_C = J_E$.

1.2.8 Heat Transfer in the Emitter/Collector Gap [1]:

As shown in Figure 1.11, energy is transferred away in the radial direction from the emitter surface by the following three modes:

1. Heat conduction rate through the following media:
   a. Heat conduction rate, $(Q_L - Q_d/2)$ through the leads connected to the emitter and collector is:

$$Q_L=k_L\frac{S_L}{l_L}(T_E-T_C) \tag{1.8}$$

where k_L, s_L, and l_L are the thermal conductivity, the cross-sectional area and the length of the electrical leads respectively.

$$-\frac{1}{2}Q_d=-\frac{1}{2}SJV_L \tag{1.9}$$

where

Q_d = The Joulean heat rate

V_L = The voltage drop across the leads.

b. Heat conduction rate through the cesium vapor. Let Q_{Cs} be the heat conduction rate through the cesium vapor.

c. Heat conduction rate through the structural components.

Now, let Q_{k1} be the heat conduction rate through the structural components connected to the emitter.
The total heat conduction rate through the gap is given by:

$$Q_k = Q_{k1} + Q_{Cs} = g_k (T_E - T_C) \tag{1.10}$$

where g_k is the sum of the thermal conductances g_{k1} of structural materials connected to the emitter and g_v of the vapor.

2. Thermal radiation rate, Q_r

$$Q_r = S\sigma_o \varepsilon (T^4{}_E - T^4{}_C) \tag{1.11}$$

where σ_o is the Stephan-Boltzman constant(= 5.67 x 10^{-12} watt/cm^2-k^4) and ϵ is the net effective thermal emissivity.

3. Electron cooling rate, Q_E:

a. The Energy flux associated with electrons travelling from the emitter to the collector is

$$SJ_{EC}\frac{\Psi_{max}+2kT_E}{e} \tag{1.12}$$

where Ψ_{max} is the maximum value of the interelectrode motive.

b. The Energy flux associated with electrons returning to the emitter through the electrical load is given by:

$$-SJ_{EC}\frac{\mu_E}{e} \tag{1.13}$$

c. The energy flux associated with electrons flowing from the collector to the emitter in the emitter/collector gap is given by:

$$-SJ_{CE}\frac{\psi_{max}+2kT_C}{e} \tag{1.14}$$

d. The energy flux associated with electrons leaving the emitter through the electrical load is:

$$SJ_{CE}\frac{\mu_E}{e} \tag{1.15}$$

Thus the electron cooling rate, Q_E is:

$$Q_E=\frac{SJ_{EC}(\psi_{max}-\mu_E+2kT_E)-SJ_{CE}(\psi_{max}-\mu_E+2kT_C)}{e} \tag{1.16}$$

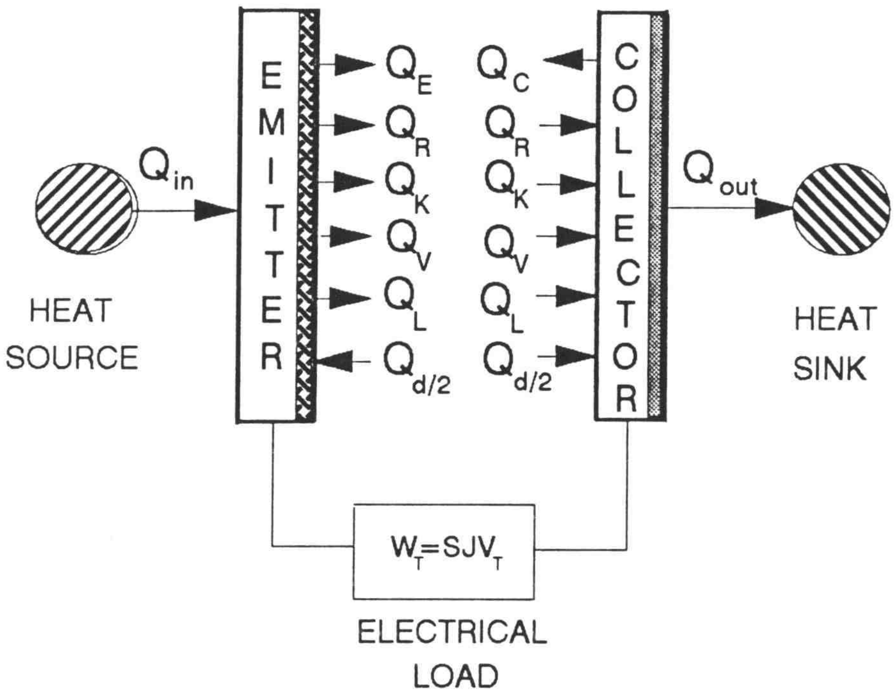

Figure 1.11 Energy Transfer Modes in a Thermionic Converter

1.3 Thermionic Fuel Element (TFE)

Thermionic fuel elements are used extensively in nuclear space reactors for power generation purposes. The heat source in the TFE is the nuclear fuel. The fuel is completely enclosed by the emitter material (Figure 1.12), and waste heat is removed from the collector by fluid convection.

Thermionic fuel elements (TFEs) for incore reactors can be either multicell or single cell. In the multicell type, also known as flashlight, all the thermionic cells are grouped in thermionic fuel elements. In the single cell configuration, the thermionic converter is enclosed in the TFE. Single fuel elements have many advantages:

1. Simulation task is possible due to using an electrical heater instead of nuclear fuel for ground base tests before launching to space.
2. Simplicity of removing gas fission fragments from fuel elements.
3. Possibility of additional TFEs in a fully assembled reactor [1].

The various components of a typical TFE include:

a. Void: The void is located at the center and extended along the axial direction of the TFE. It serves as a vent to expel the fission gas products which arise from the nuclear fission process in the nuclear fuel during operation. These may have an effect on the life span of the TFE. It also prevents any swelling in the fuel that may arise from trapping of fission products in the fuel lattice. Densification of fuel during reactor operation, when the fuel temperature reaches a maximum, can be prevented due to the existence of the void. The size of the void is directly proportional to the size and weight of the fuel.

b. Fuel: The heat source used in the TFE is uranium dioxide enriched with 95% uranium 235. The nuclear fuel used in the TFE has the following advantages:

1. High density (advantageous for size reduction of

reactor in space applications).

2. Solidarity and durability at high operational

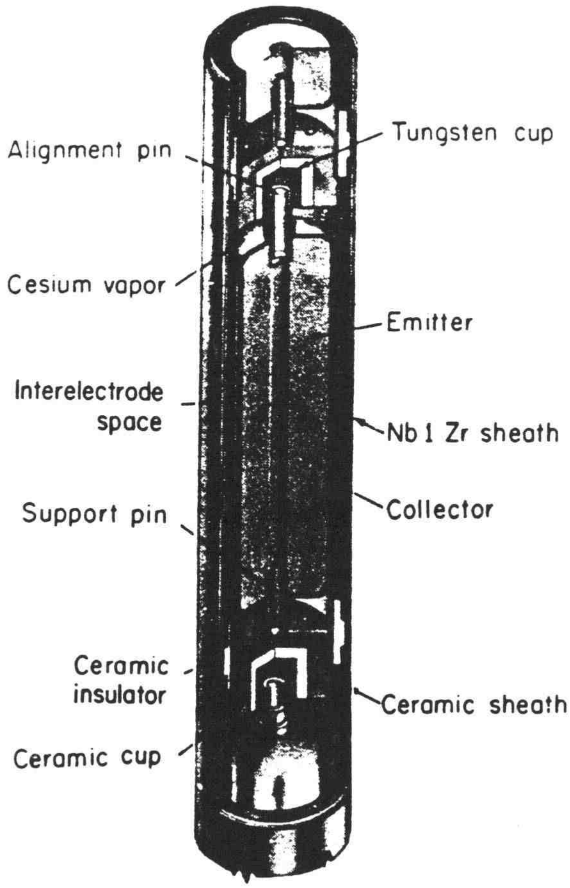

Figure 1.12 Thermionic Fuel Element

temperatures(due to the ceramic composition).

3. Low neutron absorption cross section of

oxygen that prolongs the life time of the fuel.

4. Excellent chemical and mechanical integrity.

c. Emitter: The emitter in this design is adjacent to the fuel so that there is no fuel/emitter gap. Heat is transferred directly from the fuel to the emitter by the conduction mode.

The emitter is a refractory metal made of tungsten (W) material that is being used in many thermionic reactors for the following considerations:

1. High melting point (3700 °K). This high temperature is compatible to the fuel melting point temperature and is of great importance in case of the loss of flow accident (LOFA) in which all thermionic parts and reactor components can be prevented from any expected damage in case of fuel melt down.
2. High electron emission at higher temperatures (1900 °K). The higher the emitter temperature the higher the emission rate and the higher the reactor efficiency and the higher the reactor power output.
3. High work function.
4. Low emissivity rate that reduces the transferred heat loss by radiation.

Unfortunately, most of the tungsten isotopes are highly neutron absorbing materials and are not recommended for use in thermionic reactors due to the reduction of the fuel life and minimization of the reactor efficiency. Fortunately one isotope (^{184}W) is exceptional [47] due its low neutron absorption advantage. Hence the emitter should be highly enriched with this isotope. The only disadvantage is the fabrication cost but it is worthy for the benefits.

d. Emitter/Collector gap: The gap is filled with cesium vapor that neutralizes the negative charge and eases the transportation of emitted electrons from the emitter to the collector. It is considered to be the most important region in the TFE through which many energy transformations take place.

e. Collector: The collector works as a sink to collect the emitted electrons from the emitter. The collector material is made of niobium which has a low work function. This work function is lower than that of the emitter and is kept at low temperature lower than that of the emitter.

f. Insulator: The insulator sheath is made of Al_2O_3 to electrically insulate the collector and prevent any current leakage that may affect the efficiency of the thermionic converter. Also, the insulator in a thermionic converter should be a good heat conductor.

g. Cladding: To prevent any discharge of radioactive materials during the reactor operation. The cladding is usually made of niobium.

h. Coolant: The liquid metal coolant keeps the thermionic fuel element temperature within safe limits. It flows along the outside axial length of the cladding. The coolant used is eutectic NaK (78% K) which 1) possesses a very high thermal conductivity to transfer more heat from the contiguous surface of cladding and 2) has a wide useful range of temperature in the liquid phase. The 22% sodium in the coolant prevents any corrosion that may arise from any adjacent surface due to the long operation in space.

i. Liner: The main purpose of the liner is to retain the liquid metal coolant from discharging outside the TFE. The liner is made of stainless steel that withstands the elevated temperatures. It also protects the ZrH block (moderator) from the coolant.

References :

1. G. N. Hatsopoulos and E. P. Gyftopoulos, Thermionic Energy Conversion (MIT, Cambridge, MA, 1979), Vols. **I** and **II**.

2. David Buden, Nuclear Reactors For Space Power, Aerospace America, pp. 66-69, 1985.

3. M.S. El-Genk and M.D. Hoover, "Space Nuclear Power Systems", Vol. **9**, pp. 351-355, Orbit Book Company, Inc., FL, 1989.

4. V. C. Wilson, Conversion of Heat to Electricity by Thermionic Emission, J. Appl. Phys. **30**, pp. 475-481, 1959.

5. J. B. Taylor and I. Langmuir, The Evaporation of Atoms, Ions, and Electrons from Cesium Films on Tungsten, Phys. Rev. **44**, pp. 423-453, 1933.

6. N.N. Ponomarev-Stepnoi, et al, "NPS "TOPAZ-II" Description,"JV INERTEK report, Moscow, Russia, 1991.

7. N.N. Ponomarev-Stepnoi, et al, "Thermionic Fuel Element of Power Plant TOPAZ-2, "JV INERTEK report, Moscow, Russia, 1991. N.N. Ponomarev-Stepnoi, et al, "NPS "TOPAZ-II" Trials Results, "JV INERTEK report, Moscow, Russia, 1991.

8. T.M. Foley, Soviet Reveal Testing in Space of Thermionic Nuclear Reactor (TOPAZ), Aviation Week & Space Technology **130**, pp. 30, Jan. 16, 1989.

9. M.M. El-Wakil, Nuclear Energy Conversion, 2nd Edition, The American Nuclear Society Press., Illinois, 1978.

10. S. L. Soo, Direct Energy Conversion, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1968.

11. G. W. Sutton, Direct Energy Conversion, Vol. **3**, McGraw-Hill Book Company, N.Y., 1966.

12. K. H. Spring, Direct Energy Conversion, Academic Press, London and New York, 1965.

13. S. W. Angrist, Direct Energy Conversion, Allyn and Bacon, Inc., MT, 3rd Edition, 1976.

14. J. Kaye and J.A. Welsh, Direct Conversion of Heat To Electricity, John Wiley & Sons, Inc., NY, 1960.

15. R.F. Wilson and H. M. Dieckamp. "What Happened to SNAP10A," Astronautics and Aeronautics, pp. 60-65, Oct. 1965.

16. H. Moss, Thermionic Diodes as Energy Converters, Brit. J. Electron **2**, pp. 305-322, 1957.

17. V.C. Wilson, Conversion of Heat to Electricity by Thermionic Emission, Bull. Am. Phys. Soc. **3**, pp. 266, 1958.

18. G. M. Grover, D. J. Roehling, E. W. Salmi, and R. W. Pidd," Properties of a Thermoelectric Cell", J. Appl. Phys. **29**, pp. 1611-1612, 1958.

19. H. F. Webster and J. E. Beggs, High Vacuum Thermionic Energy Converter, Bull. Am. Phys. Soc. **3**, pp. 266., 1958.

20. M. A. Cayless, Thermionic Generation of Electricity, Brit. J. Appl. Phys. **12**, pp. 433-442, 1961.

21. L. N. Dobrestov, Thermoelectronic Converters of Thermal Energy into Electric Energy, Soviet Phys.Tech. Phys.(English Transl.), Vol. **5**, pp. 343-368, 1960.

22. K. G. Hernqvist, M. Kanefsky, and F. H. Norman, Thermionic Energy Converter, RCA Rev. **19**, pp. 244-258, 1959.

23. J. M. Houston and H. F. Webster, Thermionic Energy Conversion, Electronics and Electron Physics, Vol. **17**, pp. 125-206, Academic Press Inc., New York, 1962.

24. N. S. Rasor, Figure of Merit for Thermionic Energy Conversion, J. Appl. Phys. **31**, pp. 163-167, 1960.

25. J. M. Houston, Theoretical Efficiency of the Thermionic Energy Converter, J. Appl. Phys. **30**, pp. 481-487, 1959.

26. P. L. Auer and H. Hurwitz,Jr., Space Charge Neutralization by Positive Ions in Diodes, J. Appl. Phys. **30**, pp. 161-165, 1959.

27. J. A. Becker, "Thermionic Electron Emission and Absorption Part I, Thermionic Emission," Rev. Mod. Phys., Vol. **7**, pp. 95-128; April, 1935.

28. C. Herring and M. H. Nichols, Thermionic Emission, Rev. Mod. Phys. **21**, pp. 185-270; April 1949.

29. A. L. Reimann, Thermionic Emission, J. Wiley & Sons, Inc., New York, 1934.

30. E. Bloch, Thermionic Phenomena, Methuen & co. ltd., London, 1927.

31. M. M. El-Wakil, Nuclear Heat Transport, 3rd Edition, The American Nuclear Society Press., 1981.

32. S.S. Kitrilakis and M. Meeker, "Experimental Determination of the Heat Conduction of Cesium Gas", Advanced Energy Conversion, Vol. **3**, pp. 59-68, 1963.

33. N.S. Rasor,"Thermionic Energy Conversion," in Applied Atomic Collision Processes, Massey, H.S.W., McDaniel, E.W., and Bederson,B., eds. Vol. **5**, pp. 169-200, Academic Press, New York, 1982.

34. O. Faust,"Sodium-NaK Engineering Handbook", Vol.**1**, pp. 52-53, 1972.

35. W.A. Ranken, G.M. Grover, and E.W. Salmi, "Experimental Investigations of the Cesium Plasma Cell", J. Appl. Phys. **31**, pp. 2140, 1960.

36. H.W. Lewis and J.R. Reitz,"Efficiency of Plasma Thermocouple", J. Appl. Phys. **31**, pp. 723, 1960.

37. E.N. Carabateas, S.D. Pezaris, and G.N. Hatsopoulos,"Interpretation of Experimental Characteristics of Cesium Thermionic Converters", J. Appl. Phys. **32**, pp. 352, 1961.

38. R.K. Steinberg,"Hot-Cathode Arcs in Cesium Vapor", J. Appl. Phys. **21**, pp. 1028, 1950.

39. E.B. Hensley,"Thermionic Emission Constants and Their Interpretation", J. Appl. Phys. **32**, pp. 301, 1961.

40. J.H. Ingold,"Calculation of the Maximum Efficiency of the Thermionic Converter", J. Appl. Phys. **32**, pp. 769, 1961.

41. A. Schock,"Optimization of Emission-Limited Thermionic Generators", J. Appl. Phys. **32**, pp. 1564, 1961.

42. E.P. Gyftopoulos and J.D. Levine,"Work Function Variation of Metals Coated by Metallic Films", J. Appl. Phys. **33**, pp. 67, 1962.

43. J.M. Houston,"Thermionic Emission of Refractory Metals in Cesium", Vol. **6**, pp. 358, 1961.

44. E.S. Rittner,"On the Theory of the Close-Spaced Impregnated Cathode Thermionic Converter", J. Appl. Phys. **31**, pp. 1065, 1960.

45. H.F. Webster,"Calculation of the Performance of a High-Vacuum Thermionic Energy Converter", J. Appl. Phys. **30**, pp. 488, 1959.

46. A.F. Dugan,"Contribution of Anode Emission to Space Charge in Thermionic Power Converters", J. Appl. Phys. **31**, pp. 1397, 1960.

47. A.C. Klein, H.H. Lee, B.R. Lewis, R.A. Pawlowski and Shahab Abdul-Hamid, "Advanced Single Cell Thermionic Reactor System Design Studies", Oregon State University, OSU-NE-9209, Corvallis, OR Sept. 1992.

48. H. H. Lee,"System Modeling and Reactor Design Study of an Advanced Incore Thermionic Space Reactor", M.S. Thesis, Oregon State University, Corvallis, OR, Sept. 1992.

49. B.R. Lewis, R.A. Pawlowski, K.J. Greek and A.C. Klein, "Advanced Thermionic Reactor System Design Code", Proceedings of 8th Symposium on Space Nuclear Power Systems, CONF-910116, Albuquerque, NM, 1991.

50. R.A. Pawlowski and A.C. Klein,"Analysis of TOPAZ-II Thermionic Fuel Element Performance Using TFEHX", Proceedings of 10th Symposium on Space Nuclear Power Systems, Albuquerque, NM, 1993.

51. H.H. Lee, B.R. Lewis, R.A. Pawlowski and A.C. Klein,"Design Analysis Code for Space Nuclear Reactor Using Single Cell Thermionic Fuel Element", at Nuclear Technologies for Space Exploration, pp. 271, Jackson Hole, WY, 1992.

52. V.P. Nickitin, B.G. Ogloblin, A.N. Luppov, N.N. Ponomarev-Stepnoi, V.A. Usov, Y.V. Nicolaev, and J.R. Wetch,"TOPAZ-2 Thermionic Space Nuclear Power System and Perspectives of its Development," 8th Symposium on Space Nuclear Power Systems Proceedings, CONF-910116, Albuquerque, NM, Jan. 1991.

53. N.A. Deane, S.L. Stewart, T.F. Marcille, and D.W. Newkirk, "SP-100 Reactor and Shield Design Update," Proceedings of 9th Symposium on Space Nuclear Power Systems, CONF-920104, Albuquerque, NM, Jan. 1992.

54. J.B. McVey, "TECMDL-Ignited Mode Planar Converter Model", E-563-004-C-082988, Rasor Associates, Inc., Sunnyvale, CA, Aug. 1990.

55. J.B. McVey, G.L. Hatch, and K.J. Greek, Rasor Associates, Inc., and G.J. Parker, W.N.G. Hitchon, and J.E. Lawler, University of Wisconsin-Madison, "Comprehensive Time Dependent Semi-3D Modeling of Thermionic Converters In Core", NSR-53 / 92-1004, Sept. 30, 1992.

56. J.J. Duderstadt and L.J. Hamilton, Nuclear Reactor Analysis, 1st Edition, John Wiley and Sons, NY, 1976.

57. S. Glasstone and A. Sesonske, Nuclear Reactor Engineering, Van Nostrand Reinhold Company, NY, 1981.

58. G.H.M. Gubbels and R. Metselaar, A Thermionic Energy Converter with an Electrolytically Etched Tungsten Emitter, J. Appl. Phys. **68**, pp. 1883-1888, Aug. 1990.

59. A.C. Klein and R.A. Pawlowski, "Analysis of TOPAZ-II Thermionic Fuel Element Performance using TFEHX," pp. 1489-1494 Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1489-1494 Albuquerque, NM, Jan. 1993.

60. V.V. Skorlygin, T.O. Skorlygina, Y.A. Nechaev, and M.Y. Yermoshin, Kurchatov Institute, Moscow, Russia, Alexey N. Luppov, Central Design Bureau of Machine Building, St. Petersburg, Russia, and Norm Gunther, Space Power, Inc., San Jose, CA, "Simulation Behavior of TOPAZ-2 and Relation to the Operation of TSET System," Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1495-1498, Albuquerque, NM, Jan. 1993.

61. D.B. Morris, "The Thermionic System Evaluation Test (TSET): Description, Limitations, and the Invovment of the Space Nuclear Power Community," Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1251-1256, Albuquerque, NM, Jan. 1993.

62. N.S. Rasor, "Thermionic Energy Conversion Plasmas", IEEE Transactions on Plasma Science, vol. **19**, No. 6, Dec. 1991

63. J.R. Lamarsh, Introduction to Nuclear Reactor Theory, Addison-Wesley Publishing Company, NY, 1972.
D.B. Morris, "The Thermionic System Evaluation Test (TSET): Description, Limitations, and the Invovment of the Space Nuclear Power Community," Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1251-1256, Albuquerque, NM, Jan. 1993.

Chapter 2

Theory

2.1 Introduction

The temperature distribution throughout a thermionic fuel element (TFE) is a function of many factors:

1. The location of the node point along the radial and axial positions in the TFE. There are different materials which have different thermal conductivities and specific heat terms.
2. There are some nodes which lie on the interface between two layers, in this case, any temperature dependent physical properties terms can be averaged.
3. The thickness of each layer as well as the number of regions in each material (see Table 2.1).

A steady state computer code (TFEHX) for calculating the steady state temperature distribution along the axial and radial directions has been developed. The TFEHX computer code is one of the most complete descriptions of a thermionic system in existence, and the first combined thermionic-thermal-neutronic code developed in the United States [7]. This code needs to be developed to accommodate the transient thermalhydraulic behavior of the TFE. This task was accomplished using TFETC (Thermionic Fuel Element Transient

Code) which is a newer version of TFEHX that has been developed by the author of this thesis. The heat transfer mechanism varies throughout the TFE according to the physical properties of materials from region to region. Also the emitter/collector gap has a great effect on the heat transfer mechanism as well as the liquid metal coolant which is adjacent to the cladding surface. All these modes of heat transfer need to be taken care of by introducing a suitable partial differential equation.

Table 2.1 Thermionic Fuel Pin Parameters

Region	Inner Radius (cm)	Outer Radius (cm)	Thickness (cm)	Material
Fission Gas Plenum	------	0.15	0.15	Void
Fuel	0.15	0.60	0.45	UO_2
Emitter	0.60	0.75	0.15	Tungsten
Gap	0.75	0.80	0.05	Cesium Vapor
Collector	0.80	0.90	0.10	Niobium
Insulator	0.90	0.95	0.05	Al_2O_3
Cladding	0.95	1.00	0.05	Niobium
Coolant	1.00	1.25	0.25	NaK (Eutectic)
Liner	1.25	1.255	0.005	Stainless Steel

The unsteady state nonhomogeneous heat conduction partial differential equation (equation 2.1) is required and suitable for solving the temperature distribution along the TFE pin.

$$\nabla.k(r,z,t)\nabla T(r,z,t)+g(r,z,t) = \rho(r,z,t)C_p(r,z,t)\frac{\partial T(r,z,t)}{\partial t} \qquad (2.1)$$

where

k = Thermal conductivity of a material in the TFE, W/m.°K.

C_p = Specific heat of a material in the TFE, J/Kg.°K.

ρ = Density of a material in the TFE, Kg/cm^3.

g = Rate at which heat is generated in the fuel, watt.

T = Temperature at any point in the TFE, °K.

t = Transient time of reactor operation, sec.

Some physical properties such as thermal conductivity, density, and specific heat are location and time dependent and need to be determined at various temperatures. For some solid materials such as fuel, emitter, collector, and insulator the density has to be constant for each material (i.e., does not vary with temperature variation) and that is true due to the fact that thermal expansion for solids is very small. The exceptional case is for a coolant (NaK) in which the density changes at different temperatures. On the other hand, thermal conductivity and specific heat differ with temperature variation and should be calculated for all time steps as a function of temperature.

2.2 TFE Configuration

Figure 2.1 shows the top view of the TFE. The detailed description of all regions of the TFE is presented in chapter 1 of this thesis. The following describes the regions at which the only effective heat transfer mechanism is conduction. These regions are:

1. Fuel/fuel interface.
2. Fuel/emitter interface.
3. Collector/insulator interface.
4. Insulator/cladding interface.

The effective heat transfer mechanism in the cladding/coolant interface is convection. The most important modes of heat transfer that play an important role in the TFE operation are the ones that lie in the emitter/collector gap. The energy is transferred away from the emitter surface to the collector surface in the positive r-direction by the following three modes [4]:

1. Thermal conduction of cesium vapor.
2. Thermal radiation between the emitter and the collector.

3. Thermionic heat transfer processes which include:
   a. Energy transferred away by the emitted electrons which is greater than that converted into electricity.
   b. Thermal radiation from the ignited cesium plasma back to the emitter surface.

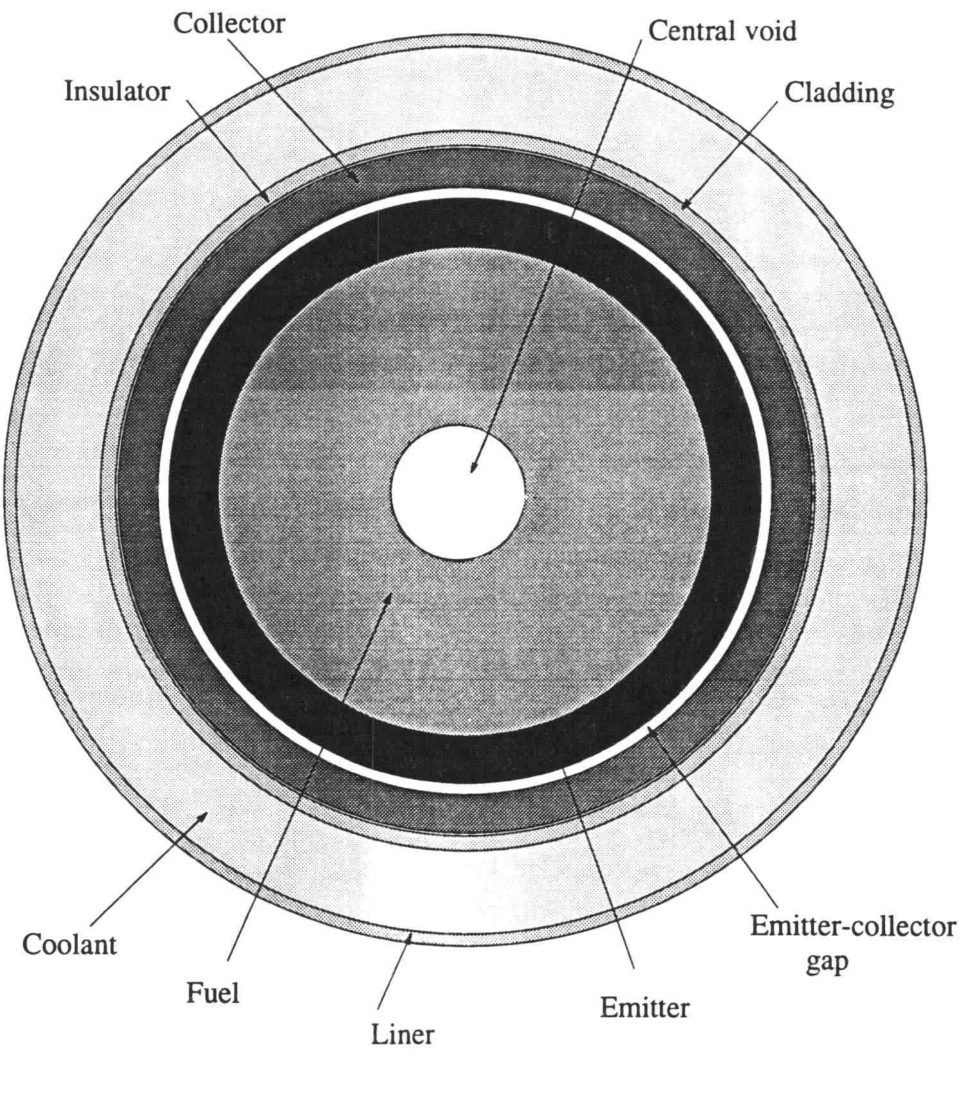

Figure 2.1 TFE Configuration

For the heat conduction flux through cesium vapor across the emitter/collector gap, a Kitrilakis and Meeker correlation [5] is used as follows:

$$Q_k^{Cond} = \frac{k_{Cs}(T_{e,k}-T_{c,k})}{d + 1.15x10^{-5}\dfrac{(T_{e,k}-T_{c,k})}{p_{Cs}}}\left[2\pi r_e \frac{(Z_{k+1}-Z_{k-1})}{2}\right] \qquad (2.2)$$

where

$T_{e,k}$ = Emitter temperature (°K).

$T_{c,k}$ = Collector temperature (°K).

k_{Cs} = Thermal conductivity of cesium vapor, W/cm.°K.

p_{Cs} = Pressure of cesium vapor at a cesium reservoir temperature (torr).

r_e = Emitter outside radius, cm.

d = Emitter/collector gap, cm.

The pressure p_{Cs} is given by the following correlation:

$$p_{Cs} = 2.45x10^{8}\frac{\exp\left(\dfrac{8910}{T_r}\right)}{\sqrt{T_r}} \qquad (2.3)$$

The thermal radiation term Q^{Rad} between the emitter and collector is given by the following equation:

$$Q_k^{Rad}=\sigma\epsilon_e F_{e\to c}(T_{e,k}^4-T_{c,k}^4)\,[2\pi r_e(\frac{Z_{k+1}-Z_{k-1}}{2})] \qquad (2.4)$$

where

σ = Stefan-Boltzman constant(5.67×10^{-12} Watts/cm^2°K^4)

ϵ_e = Thermal emissivity of the emitter surface.

$F_{e\to c}$ = View factor from the emitter surface to the collector surface ($F_{e\to c}$ = 1 for the emitter surface).

The electron cooling energy transfer term Q_k^{EEC} is computed using the TECMDL computer code [8] and can be given by:

$$Q^{EEC}=J_E(V_E+2\frac{kT_E}{e}) \qquad (2.5)$$

where

J_E = Current density of the emitter surface, amp/cm^2.

V_E = Voltage across the emitter surface, volt.

T_E = Emitter temperature, °K.

k = Boltzman constant = 8.62×10^{-5} ev/°K.

For the thermalhydraulic transient calculations of the TFE, the TFETC is modeled to accommodate three different situations namely:

1. Start up.
2. Loss of Flow Accident.
3. Shut down.

2.2.1 Start up

The reactor thermal power(P_{th}) is assumed to rise exponentially with time as follows:

$$P_{th}=P_{ss}.[1-e^{(-t/\tau)}] \tag{2.6}$$

where

P_{th}= Thermal power during start up, watt.

P_{ss}= Steady state power, watt

t = Transient time for start up, sec.

τ = Power rise coefficient, sec.

There are four different cases in which the thermal power rises until it reaches the steady state value. These values are:

1. τ = 100 sec.
2. τ = 300 sec.
3. τ = 600 sec.
4. τ = 1200 sec.

2.2.1.1 Helium Heating:

Thermal conductivity of helium is relatively high compared to cesium so it can be used as a heating element in the emitter/collector gap to speed up the heating process. At a certain temperature, around 900 °K, the helium heating is stopped and cesium vapor takes place. The temperature at which the helium heating is stopped is part of the input file of the TFETC code.

2.2.1.2 Electron cooling:

The electron cooling for the emitter surface is negligible at low temperatures at the beginning of the start up process. At certain temperatures (1500-2000 °K), the electron cooling is effective. The temperature at which the electron cooling starts is part of the input file of the TFETC code.

2.2.2 Loss of Flow Accident (LOFA)

In a loss of flow accident four cases of pump failure are discussed. These cases are listed below

1. Complete pump failure (1/1).
2. 50% pump failure (1/2).
3. 33% pump failure (1/3).
4. 25% pump failure (1/4).

The mass flow rate in LOFA behaves according to the following equation:

$$m(t)=m_o.[A+B.e^{-t/\tau}] \tag{2.7}$$

where

m_o = Mass flow rate before LOFA begins.

m(t) = Mass flow rate after LOFA begins.

t = Transient time, sec.

A = 0.0 for 1/1 pump failure.

= 0.50 for 1/2 pump failure.

= 0.67 for 1/3 pump failure.

= 0.75 for 1/4 pump failure.

B = 1.0 for 1/1 pump failure.

= 0.50 for 1/2 pump failure.

= 0.33 for 1/3 pump failure.

= 0.25 for 1/4 pump failure.

Different values of rising mass flowrate coefficients, τ are discussed for LOFA. The smaller the τ, the faster the loss of coolant and vice versa. Three values have been chosen for describing four different schemes of LOFA

1. τ = 30 sec.
2. τ = 120 sec.
3. τ = 600 sec.

2.2.3 Shut down

The only shut down technique that has been considered is prompt jump according to the following equation:

$$P_{th}=P_{ss}\left[\frac{1-\beta\rho^{\cdot}}{1-\rho^{\cdot}}\right]e^{(-t/T)} \tag{2.8}$$

where

P_{ss} = Thermal power before shut down.

β = Total delayed fraction.

$\rho^{\cdot}$ = Negative reactivity insertion,$

T = Reactor period, sec.

Different negative reactivity insertions have been used for different shut down schemes. The values of T are taken according to Figure 2.2.

$\rho^{\cdot}$ = -$0.1,

$\rho^{\cdot}$ = -$0.3,

$\rho^{\cdot}$ = -$0.9,

$\rho^{\cdot}$ = -$3.0

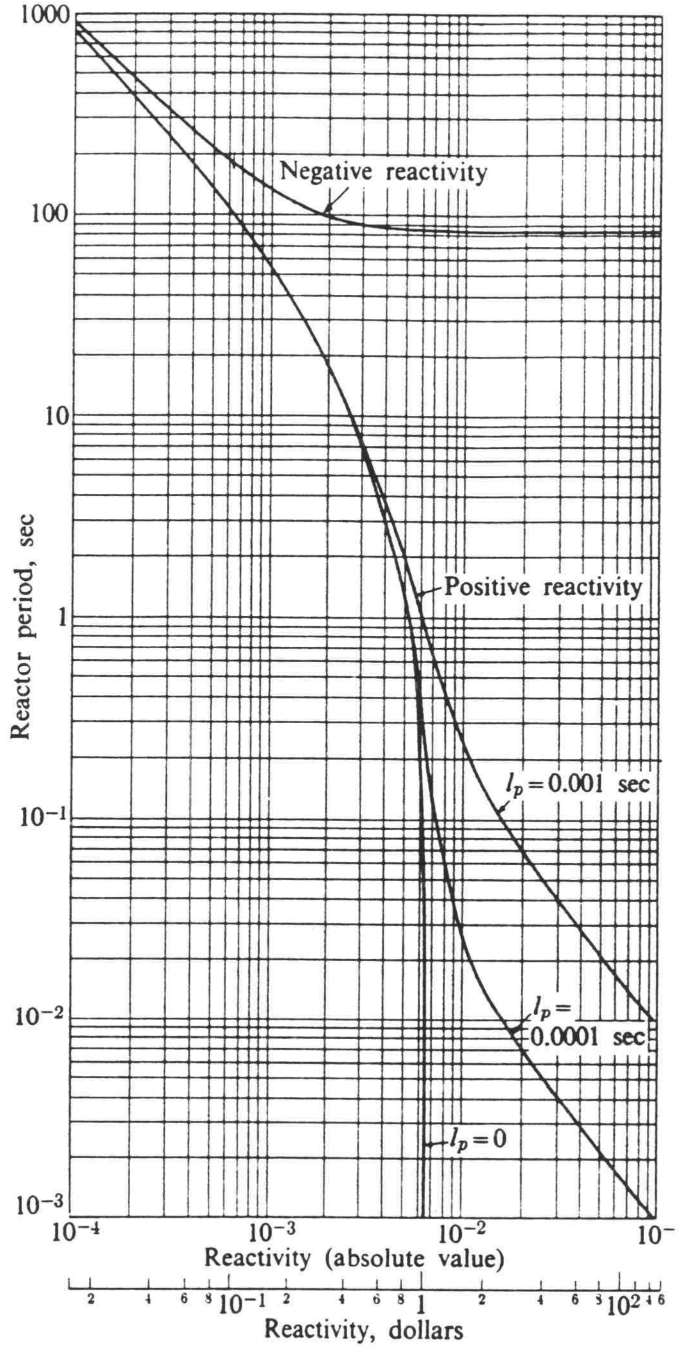

Figure 2.2 Reactor period as a function of positive and negative reactivity for a U-235 fueled reactor [1].

References:

1. J.R. Lamarsh, Introduction to Nuclear Reactor Theory, 2nd Edition, Addison-Wesley Inc., NY, 1972.

2. J.J. Duderstadt and L.J. Hamilton, Nuclear Reactor Analysis, 1st Edition, John Wiley and Sons, NY, 1976.

3. S. Glasstone and A. Sesonske, Nuclear Reactor Engineering, Van Nostrand Reinhold Company, NY, 1981.

4. A.C. Klein, H.H. Lee, B.R. Lewis, R.A. Pawlowski and Shahab Abdul-Hamid, "Advanced Single Cell Thermionic Reactor System Design Studies", Oregon State University, OSU-NE-9209, Corvallis, OR, Sept. 1992.

5. S.S. Kitrilakis and M. Meeker, "Experimental Determination of the Heat Conduction of Cesium Gas", Advanced Energy Conversion, Vol. **3**, pp. 59-68, 1963.

6. J. M. Houston, Theoretical Efficiency of the Thermionic Energy Converter, J. Appl. Phys., **30**, pp. 481-487, 1959.

7. J.B. McVey, G.L. Hatch, and K.J. Greek, Rasor Associates, Inc., and G.J. Parker, W.N.G. Hitchon, and J.E. Lawler, University of Wisconsin-Madison, "Comprehensive Time Dependent Semi-3D Modeling of Thermionic Converters In Core", NSR-53 / 92-1004, Sept. 30, 1992.

8. J.B. McVey, "TECMDL-Ignited Mode Planar Converter Model", E-563-004-C-082988, Rasor Associates, Inc., Sunnyvale, CA, Aug. 1990.

Chapter 3

Method of Analysis

3.1 Introduction:

This chapter is intended to describe the model used to calculate the temperature distribution throughout the radial and axial directions of the thermionic fuel element (TFE) as a function of time by using a finite difference method. Most of the heat is transferred, throughout the TFE's layers, by conduction except at the cladding/coolant interface. The heat through the emitter/collector gap is transferred by conduction, radiation, and electron cooling. The heat conduction equation in polar cylindrical coordinates can be written as :

$$k[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial T}{\partial r}) + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2}] + g = \rho C_p \frac{\partial T}{\partial t} \qquad (3.1)$$

because of the symmetry in the TFE, $\frac{\partial T}{\partial \theta}$ can be taken to be zero. Thus equation 3.1 can be written as :

$$k[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial T}{\partial r}) + \frac{\partial^2 T}{\partial z^2}] + g = \rho C_p \frac{\partial T}{\partial t}$$

Using the Laplacian operator in cylindrical coordinates yields

$$k(r,z,t)\nabla^2 T(r,z,t) + g(r,z,t) = \rho(r,z,t)C_p(r,z,t)\frac{\partial T(r,z,t)}{\partial t} \qquad (3.2)$$

The unsteady state heat conduction partial differential equation for the TFE can also be written as:

$$\nabla \bullet \{k(r,z,t)\nabla T(r,z,t)\} + g(r,z,t) = \rho(r,z,t)C_p(r,z,t)\frac{\partial T(r,z,t)}{\partial t} \qquad (3.3)$$

Integrating the above equation over an arbitrary volume V gives

$$\int_v \nabla \bullet \{k(r,z,t)\nabla T(r,z,t)\}dV + \int_v g(r,z,t)dV =$$

$$\int_v \rho(r,z,t)C_p(r,z,t)\frac{\partial T(r,z,t)}{\partial t}dV$$

Using the Divergence theorem gives

$$\int_A \{k(r,z,t)\nabla T(r,z,t)\} \bullet n dA + \int_v g(r,z,t)dV =$$

$$\int_v \rho(r,z,t)C_p(r,z,t)\frac{\partial T(r,z,t)}{\partial t}dV \qquad (3.4)$$

Now, the TFE is modeled at several discrete mesh points in the radial (r) and axial (z) directions at time t. Let V represents the volume of a ring element which is located at radial mesh point i and axial mesh point j at time k+1 as shown in Figure 3.1. This ring has a radial thickness Δr and axial length Δz. Let A_1, A_2, A_3, and A_4 be the areas of the four outside surfaces of the ring

Equation 3.4 can now be written as follows :

$$k_{i+1/2,j,k}(\frac{\partial T}{\partial r})_1 r.n_1 A_1 + k_{i,j+1/2,k}(\frac{\partial T}{\partial z})_2 z.n_2 A_2 +$$

$$k_{i-1/2,j,k}(\frac{\partial T}{\partial r})_3 r.n_3 A_3 + k_{i,j-1/2,k}(\frac{\partial T}{\partial r})_4 z.n_4 A_4 +$$

$$g_{i,j,k}.2\pi r \Delta r \Delta z = (\rho C_p)_{i,j,k} \frac{\partial T_{i,j,k}}{\partial t} 2\pi r_i \Delta r \Delta z \quad (3.5)$$

where n_1, n_2, n_3, and n_4 are the outward normal vectors to the surfaces 1,2,3, and 4, respectively. The energy balance on the volume about a mesh point(i,j,k+1) which is not located on an outer surface of the pin or on the emitter or collector surfaces is shown in Figure 3.2. The subscripts on the partial derivatives are computed at these surfaces.

The $k_{i+1/2,j,k}$ value is the average thermal conductivity along the surface A_1 . It is computed at a temperature which is the average of the two temperatures $T_{i,j,k}$ and $T_{i+1,j,k}$.

The other thermal conductivity values are calculated in a similar way. Density and specific heat values vary with respect to the temperature variations and the temperature varies in accordance with each time step.

$$\frac{\partial T(i,j,k)}{\partial t} = \frac{T_{i,j,k+1} - T_{i,j,k}}{\Delta t} \quad (3.6)$$

where

$T_{i,j,k+1}$ is the temperature at point (i,j,k+1)

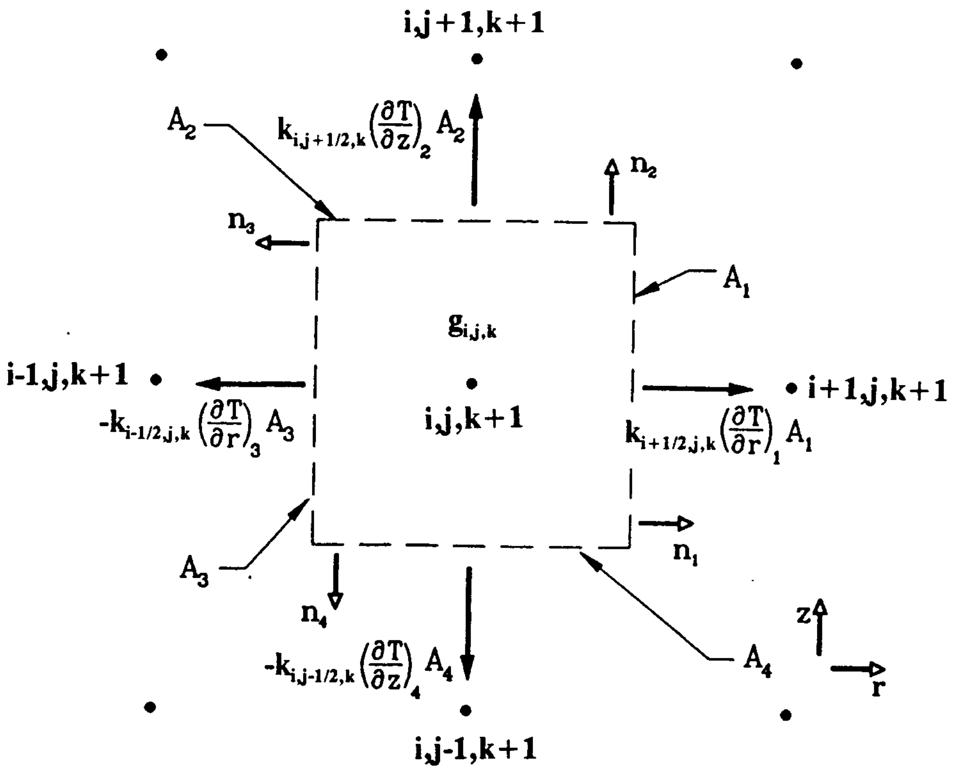

Figure 3.1 Energy balance on a ring volume about point (i,j,k+1).

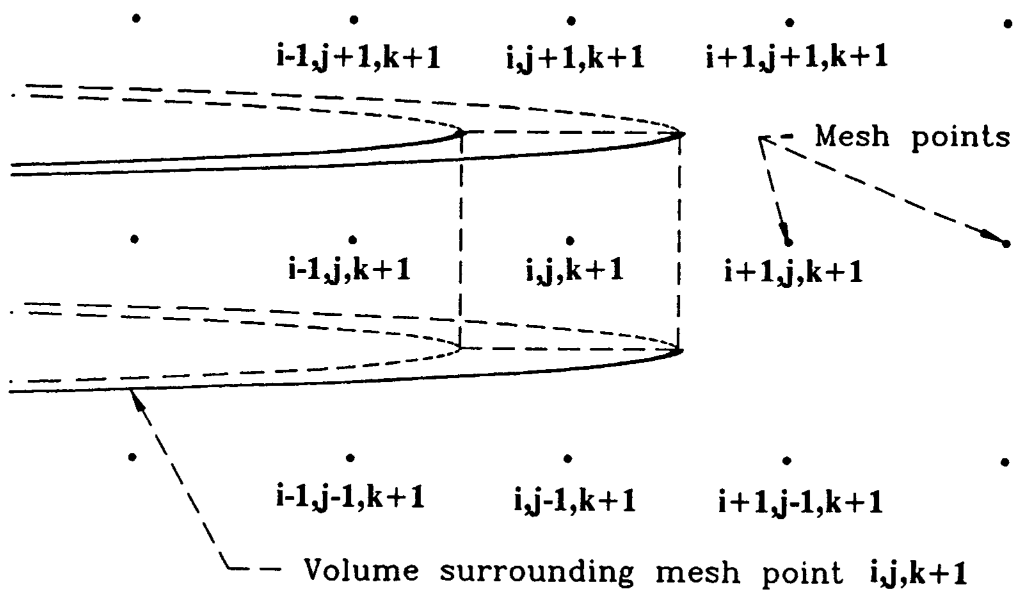

Figure 3.2 Cylindrical ring volume about the mesh point (i,j,k+1).

$T_{i,j,k}$ is the temperature at point (i,j,k)

Δt is the time increment in seconds

If the point (i,j,k+1) happens to lie on an interface between two materials as shown in Figure 3.3, then the thermal conductivities are computed as before for each of the two materials; then these two values are averaged to obtain an effective thermal conductivity for the surface.

From equation 3.5

$r.n_1 = 1$

$z.n_2 = 1$

$r.n_3 = -1$

$z.n_4 = -1$

3.1.1 Discretization Method

The implicit method [13] is used for solving the unsteady state heat conduction PDE. The main advantage of this method is its stability for any time increment. The temperature in the implicit method is advanced one time step and a system of linear simultaneous equations has to be solved at each time step. Thus as many as 100 temperatures should be determined at each time step in both r and z directions. In other words, each value of i=1 to i=10 has to be matched with all values of j (i.e., j=1 to j=10) so that a matrix of 100 x 100 is formed which has a banded structure of size 10.

Using this fact, and computing the partial derivatives in equation 3.5 as finite differences, leads to the following equation :

$$k_{i+1/2,j,k}\left(\frac{T_{i+1,j,k+1}-T_{i,j,k+1}}{r_{i+1}-r_i}\right)A_1 + k_{i,j+1/2,k}\left(\frac{T_{i,j+1,k+1}-T_{i,j,k+1}}{Z_{j+1}-Z_j}\right)A_2$$

$$k_{i-1/2,j,k}\left(\frac{T_{i-1,j,k+1}-T_{i,j,k+1}}{r_i-r_{i-1}}\right)A_3 + k_{i,j-1/2,k}\left(\frac{T_{i,j-1,k+1}-T_{i,j,k+1}}{Z_j-Z_{j-1}}\right)A_4$$

$$g_{i,j,k}2\pi r_i\Delta r\Delta z = (\rho C_p)_{i,j,k}\left(\frac{T_{i,j,k+1}-T_{i,j,k}}{t_{k+1}-t_k}\right)2\pi r_i\Delta r\Delta z \qquad (3.7)$$

3.1.2. Boundary Conditions

Adiabatic boundary conditions are assumed at the fuel pin extremities

$$\left(\frac{\partial T}{\partial Z}\right)_{Z=Z_{min.}} = 0, \qquad \left(\frac{\partial T}{\partial Z}\right)_{Z=Z_{max.}} = 0 \qquad (3.8)$$

$$\left(\frac{\partial T}{\partial r}\right)_{r=r_{min.}} = 0, \qquad \left(\frac{\partial T}{\partial r}\right)_{r=r_{max}} = h(T_{clad} - T_{cool}) \qquad (3.9)$$

3.1.3. Initial Conditions

$$T(r,z,0) = f(r,z) \qquad (3.10)$$

where f(r,z) is the given forcing function

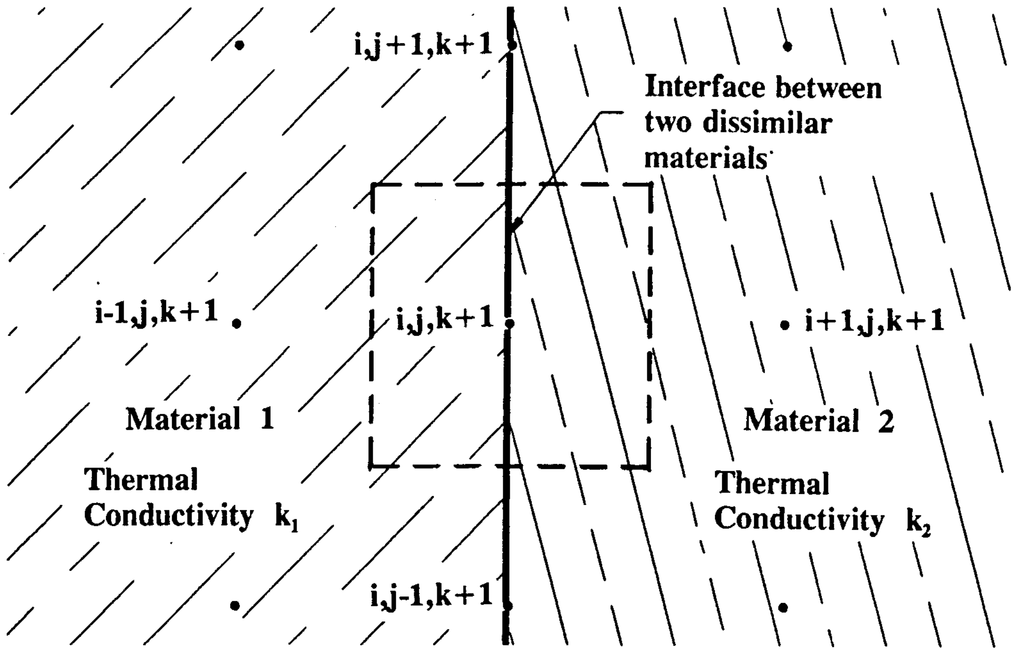

Figure 3.3 A mesh point (i,j,k+1), which lies on a material interface.

3.1.4. Stability Calculations:

The implicit method is stable for any Δt. It is considered to be more complicated than the explicit method because for each time step along the radial or axial directions of a cylinderical mesh point, it assumes that the $T_{i-1,j,k+1}$, $T_{i,j,k+1}$, $T_{i+1,j,k+1}$, $T_{i,j+1,k+1}$ and $T_{i,j-1,k+1}$ values are unknown, in other words, for each single time step there are five unknown values for temperature discretized in each equation.

Equation 3.7 can be rearranged to be used in the TFETC code as follows:

$$k_{i+1/2,j,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)T_{i+1,j,k+1} + k_{i,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right)T_{i,j+1,k+1} +$$

$$k_{i-1/2,j,k}\left(\frac{A_3}{r_i-r_{i-1}}\right)T_{i-1,j,k+1} + k_{i,j-1/2,k}\left(\frac{A_4}{z_j-z_{j-1}}\right)T_{i,j-1,k+1} -$$

$$[k_{i+1/2,j,k}\left(\frac{A_1}{r_{i+1}-r_i}\right) + k_{i,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right) + k_{i-1/2,j,k}\left(\frac{A_3}{r_{i-1}-r_i}\right)$$

$$+ \quad k_{i,j-1/2,k}\left(\frac{A_4}{z_{j-1}-z_j}\right) - (\rho C_p)_{i,j,k}\left(\frac{2\pi r_i \Delta r \Delta z}{t_{k+1}-t_k}\right)]T_{i,j,k+1}$$

$$= -g_{i,j,k} 2\pi r_i \Delta r \Delta z - (\rho C_p)_{i,j,k}\left(\frac{2\pi r_i \Delta r \Delta z}{t_{k+1}-t_k}\right)T_{i,j,k} \qquad (3.11)$$

Equation 3.11 is valid for all interior mesh points of the fuel emitter region. It is not valid for the following mesh points :

1. Those mesh points located at the upper and lower limits of the pin (i.e., the top and the bottom).
2. Those on the emitter and collector radial surfaces.
3. Those on the surface of the void region located at the center of the TFE.
4. Those on the outside radial surface of the cladding.

Equations for these points are derived in the following sections. The stability equation can be derived easily from equation 3.11. The stability equation enhances the fact that the implicit method is stable for any time step.

$$\frac{-(\rho C_p)_{i,j,k}\frac{2\pi\Delta r\Delta z}{\Delta t}}{-[k_{i+1/2,j,k}(\frac{A_1}{r_{i+1}-r_i})+k_{i,j+1/2,k}(\frac{A_2}{z_{j+1}-z_j})+k_{i-1/2,j,k}(\frac{A_3}{r_i-r_{i-1}})+k_{i,j-1/2,k}(\frac{A_4}{z_j-z_{j-1}})+(\rho C_p)_{i,j,k}(\frac{2\pi\Delta r\Delta z}{\Delta t})]} \rangle\ 0 \qquad (3.12)$$

All values in the stability equation above are positive. For the sake of accuracy, Δt has to be very small so that the truncation error $O[\Delta t+(\Delta r)^2+(\Delta z)^2]$ can be reduced.

3.2.1 Fuel Pellet/Central Void Interface-Top of the TFE

The energy balance for the mesh point at this location is shown in Figure 3.4. The top of the fuel and the central void surfaces, in the TFE, are assumed to be adiabatic (i.e., heat flow equals zero). Therefore, heat transfer does not occur in the positive r and negative z directions. When all these terms are removed from equation 3.10, the equation for the temperature at this mesh point results in:

$$k_{3/2,j\max,k}(\frac{A_1}{r_2-r_1})T_{2,j\max,k+1}+k_{1,j\max-1/2,k}(\frac{A_4}{z_{j\max}-z_{j\max-1}})T_{1,j\max-1,k+1}$$

$$-[k_{3/2,j\max,k}(\frac{A_1}{r_2-r_1})+k_{1,j\max-1/2,k}(\frac{A_4}{z_{j-1}-z_j})$$

$$-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_1(r_2-r_1)(z_{j\max}-z_{j\max-1})}{t_{k=1}-t_k}\right)]T_{i,j\max,k+1}$$

$$= -g_{i,j\max,k}2\pi r_1(r_2-r_1)(z_{j\max}-z_{j\max-1})$$

$$-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_1(r_2-r_1)(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k}\right)T_{i,j\max,k} \qquad (3.13)$$

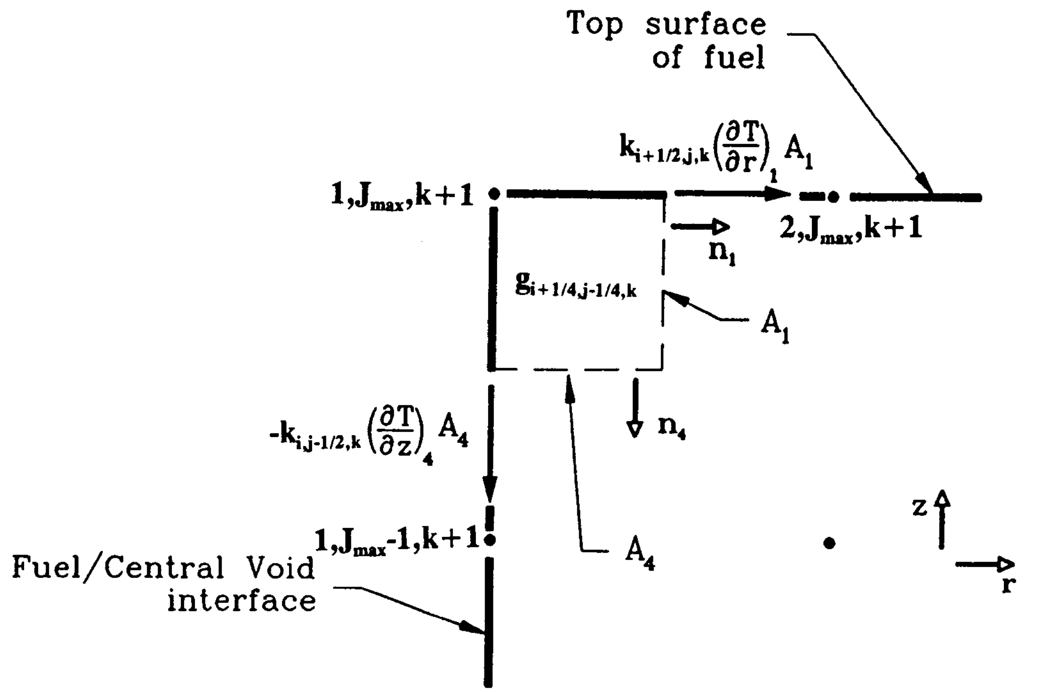

Figure 3.4 Energy balance for the mesh point $(1, J_{max}, k+1)$, which is located at the top of the fuel pin and at the surface of the central void.

3.2.2 Fuel Pellet/Central Void Interface-TFE Bottom

The energy balance for the mesh point is shown in Figure 3.5. The bottom of the TFE is assumed to be an adiabatic surface. Thus the temperature $T_{1,1,k+1}$ at this location is:

$$k_{3/2,1,k}\left(\frac{A_1}{r_2-r_1}\right)T_{2,1,k+1} + k_{1,3/2,k}\left(\frac{A_2}{z_2-z_1}\right)T_{1,2,k+1}$$

$$-[k_{3/2,1,k}\left(\frac{A_1}{r_2-r_1}\right) + k_{1,3/2,k}\left(\frac{A_2}{z_2-z_1}\right) - (\rho C_p)_{1,1,k}\left(\frac{2\pi r_1(r_2-r_1)(z_2-z_1)}{t_{k+1}-t_k}\right)]T_{1,1,k+1}$$

$$= -g_{1,1,k}2\pi r_1(r_2-r_1)(z_2-z_1) \quad -$$

$$(\rho C_p)_{1,1,k}\left(\frac{2\pi r_1(r_2-r_1)(z_2-z_1)}{t_{k+1}-t_k}\right)T_{1,1,k} \qquad (3.14)$$

3.2.3 Other Locations on the Fuel/Void Interface

The energy balance for these points is shown in Figure 3.6. The equations for the temperatures of these points are given as follows

$$k_{3/2,j,k}\left(\frac{A_1}{r_2-r_1}\right)T_{2,j,k+1} + k_{1,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right)T_{1,j+1,k+1}$$

$$+k_{1,j-1/2,k}\left(\frac{A_4}{z_j-z_{j-1}}\right)T_{1,j-1,k+1} \quad - [k_{3/2,j,k}\left(\frac{A_1}{r_2-r_1}\right)$$

$$+k_{1,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right) + k_{1,j-1/2,k}\left(\frac{A_4}{z_{j-1}-z_j}\right)$$

$$-(\rho C_p)_{1,j,k}\left(\frac{2\pi r_1(r_2-r_1)(z_{j+1}-z_j)}{t_{k+1}-t_k}\right)]T_{1,j,k+1}$$

$$= -g_{1,j,k}2\pi r_1(r_2-r_1)(z_{j+1}-z_j)$$

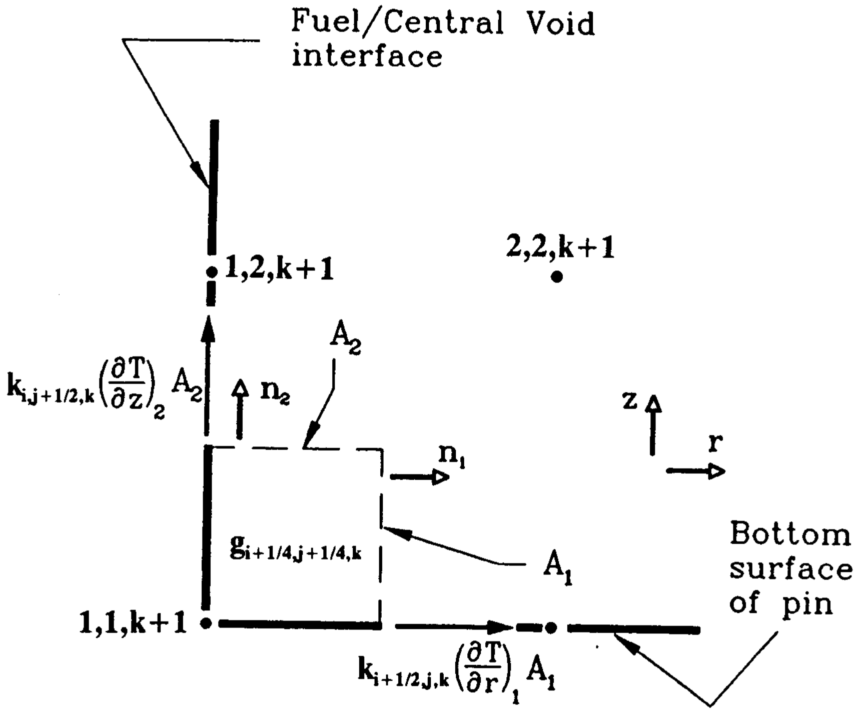

Figure 3.5 Energy balance for the mesh point (1,1,k+1), which is located at the bottom of the pin and at the surface of the central void.

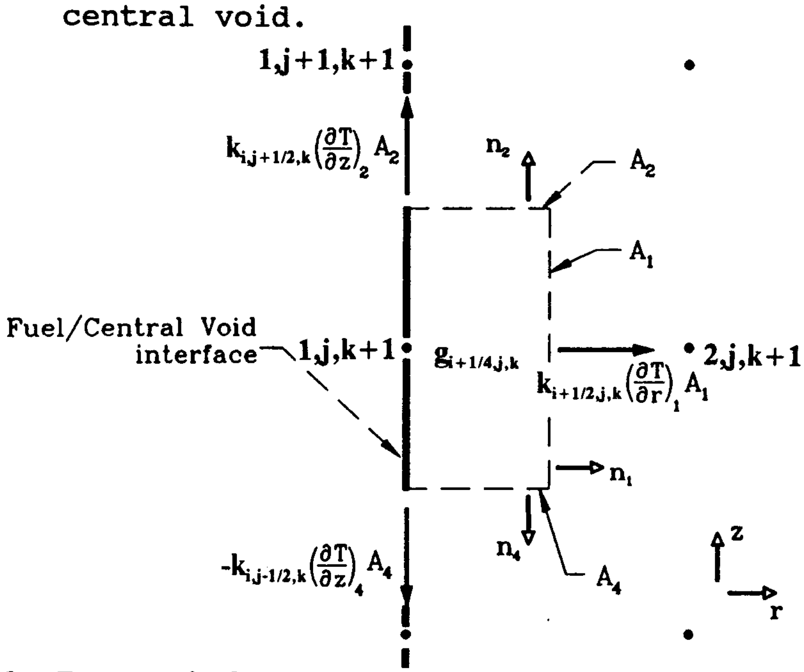

Figure 3.6 Energy balance for mesh points (1,j,k+1), which are on the fuel/void interface, but which are not at the top or bottom.

$$-(\rho C_p)_{1,j,k}\left(\frac{2\pi r_1(r_2-r_1)(z_{j+1}-z_j)}{t_{k+1}-t_k}\right)T_{1,j,k} \qquad (3.15)$$

3.2.4 Top Surface of the TFE:

The energy balance for these points is shown in Figure 3.7. The equations for the temperatures of these points are given in the following equation:

$$k_{i+1/2,j\max,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)T_{i+1,j\max,k+1} + k_{i-1/2,j\max,k}\left(\frac{A_3}{r_i-r_{i-1}}\right)T_{i-1,j\max,k+1}$$

$$+k_{i,j\max-1/2,k}\left(\frac{A_4}{z_{j\max}-z_{j\max-1}}\right)T_{i,j\max-1,k+1} - [k_{i+1/2,j\max,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)$$

$$+k_{i-1/2,j\max,k}\left(\frac{A_3}{r_{i-1}-r_i}\right) + k_{i,j\max-1/2,k}\left(\frac{A_4}{z_{j\max-1}-z_{j\max}}\right)$$

$$-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_i \Delta r(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k}\right)]T_{i,j\max,k+1}$$

$$= -g_{i,j\max,k}2\pi r_i \Delta r(z_{j\max} - z_{\max-1})$$

$$-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_i \Delta r(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k}\right)T_{i,j\max,k} \qquad (3.16)$$

3.2.5 Bottom Surface of the TFE:

The mesh points are located at the bottom surface of the TFE but are not located on any radial boundaries. The energy balance for these points is shown in Figure 3.8.

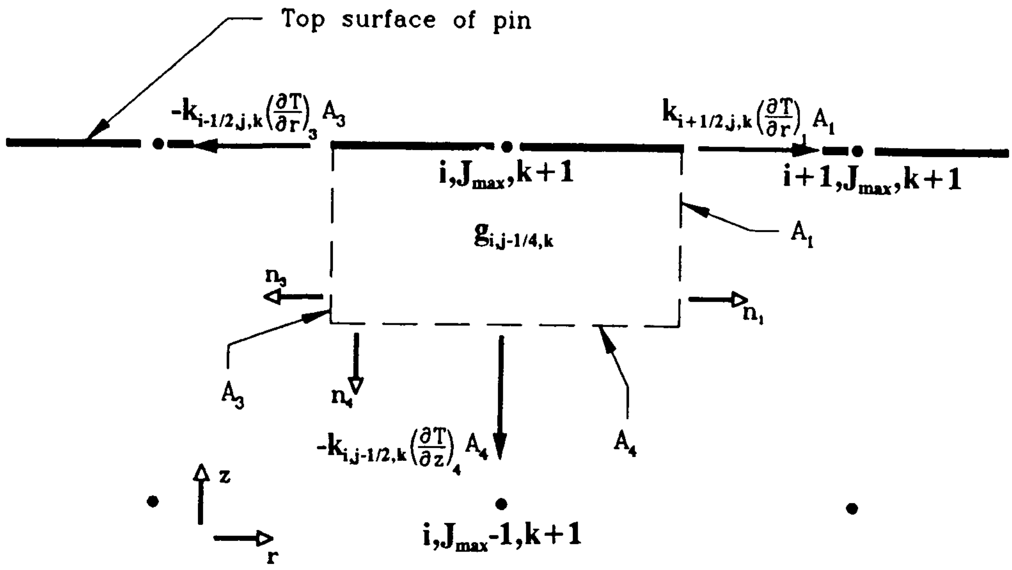

Figure 3.7 Energy balance on mesh points $(i, j_{max}, k+1)$, which are located on the top of the TFE pin, but not on any radial boundaries.

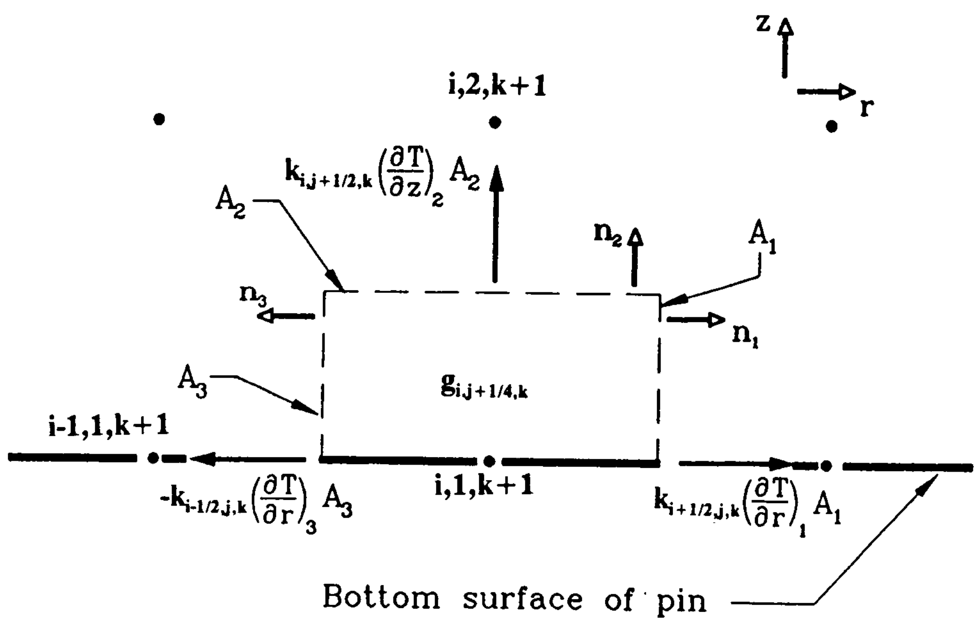

Figure 3.8 Energy balance for mesh points (i,1,k+1), which are located on the bottom of the TFE pin away from any radial boundaries.

The temperature equation is given as:

$$k_{i+1/2,1,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)T_{i+1,1,k+1} + k_{i,3/2,k}\left(\frac{A_2}{z_2-z_1}\right)T_{i,2,k+1}$$

$$+k_{i-1/2,1,k}\left(\frac{A_3}{r_i-r_{i-1}}\right)T_{i-1,1,k+1} - [k_{i+1/2,1,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)$$

$$+k_{i,3/2,k}\left(\frac{A_2}{z_2-z_1}\right) + k_{i-1/2,1,k}\left(\frac{A_3}{r_{i-1}-r_i}\right)$$

$$-(\rho C_p)_{i,1,k}\left(\frac{2\pi r_i \Delta r(z_2-z_1)}{t_{k+1}-t_k}\right)]T_{i,1,k+1}$$

$$= -g_{i,1,k} 2\pi r_i \Delta r(z_2 - z_1)$$

$$-(\rho C_p)_{i,1,k}\left(\frac{2\pi r_i \Delta r(z_2-z_1)}{t_{k+1}-t_k}\right)T_{i,1,k} \qquad (3.17)$$

3.2.6 Emitter Surface:

As shown in Figure 3.9, energy is transferred away from the surface of the emitter in the positive r-direction by different modes fully explained in Chapter 2. The equation for the temperatures at these points is:

$$k_{i,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right)T_{i,j+1,k+1} + k_{i-1/2,j,k}\left(\frac{A_3}{r_i-r_{i-1}}\right)T_{i-1,j,k+1}$$

$$+k_{i,j-1/2,k}\left(\frac{A_4}{z_j-z_{j-1}}\right)T_{i,j-1,k+1} - [k_{i,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right)$$

$$+k_{i-1/2,j,k}\left(\frac{A_3}{r_{i-1}-r_i}\right) + k_{i,j-1/2,k}\left(\frac{A_4}{z_{j-1}-z_j}\right)$$

$$-(\rho C_p)_{i,j,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_{j+1}-z_j)}{t_{k+1}-t_k})]T_{i,j,k+1}$$

$$= -g_{i,j,k}2\pi r_i(r_i - r_{i-1})(z_{j+1} - z_j)$$

$$-(\rho C_p)_{i,j,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_{j+1}-z_j)}{t_{k+1}-t_k})T_{i,j,k}$$

$$+Q_{j,k}^{Cond.}A_1 + Q_{j,k}^{Rad.}A_1 + Q_{ave}^{EEC}A_1 \qquad (3.18)$$

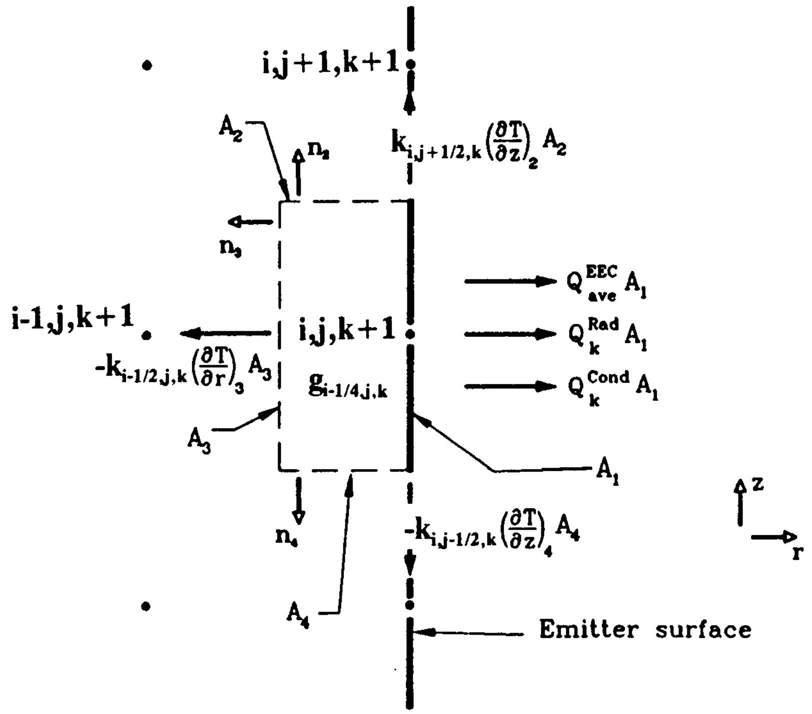

Figure 3.9 Energy balance for mesh points (i,j,k+1), which are located along the emitter surface.

3.2.7 Emitter Surface-Top of the TFE:

As shown in Figure 3.10, the equation for the temperature at the mesh point i, j_{max}, k+1 is

$$k_{i-1/2,j\max,k}(\frac{A_3}{r_i-r_{i-1}})T_{i-1,j\max,k+1} + k_{i,j\max-1/2,k}(\frac{A_4}{z_{j\max}-z_{j\max-1}})T_{i,j\max-1,k+1}$$

$$-[k_{i-1/2,j\max,k}(\frac{A_3}{r_{i-1}-r_i}) + k_{i,j\max-1/2,k}(\frac{A_4}{z_{j\max-1}-z_{j\max}})$$

$$-(\rho C_p)_{i,j\max,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k})]T_{i,j\max,k+1}$$

$$= -g_{i,j\max,k}2\pi r_i(r_i - r_{i-1})(z_{j\max} - z_{j\max-1})$$

$$-(\rho C_p)_{i,j\max,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k})T_{i,j\max,k}$$

$$+Q^{Cond.}_{j\max,k}A_1 + Q^{Rad.}_{j\max,k}A_1 + Q^{EEC}_{ave}A_1 \qquad (3.19)$$

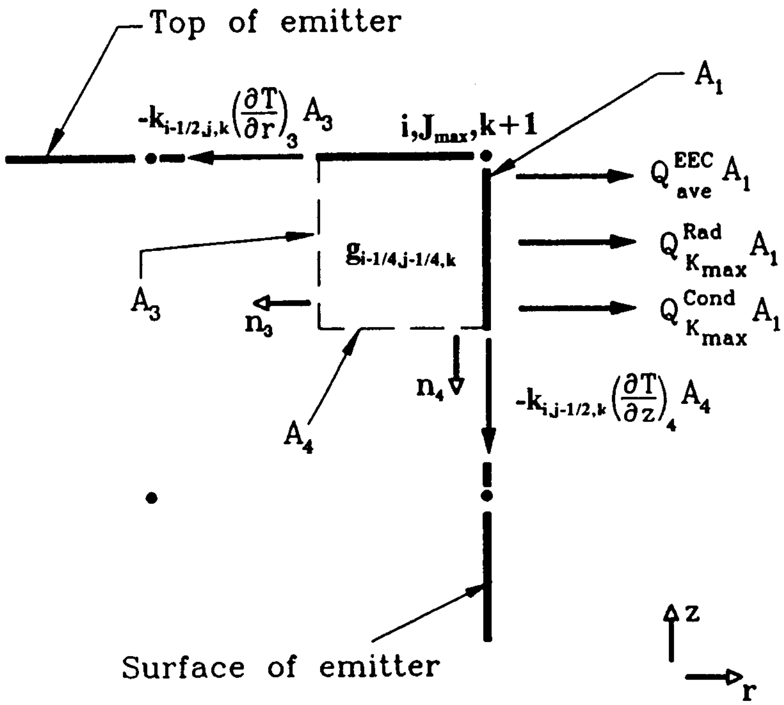

Figure 3.10 Energy balance for the mesh point $(i,j_{max},k+1)$, which is located at the top of the pin and on the emitter surface.

3.2.8 Emitter Surface-Bottom of the TFE:

As shown in Figure 3.11, the temperature equation at the mesh point i,1,k+1 is:

$$k_{i,3/2,k}(\frac{A_2}{z_2-z_1})T_{i,2,k+1}+k_{i-1/2,1,k}(\frac{A_3}{r_i-r_{i-1}})T_{i-1,1,k+1}$$

$$-[k_{i,3/2,k}(\frac{A_2}{z_2-z_1})+k_{i-1/2,1,k}(\frac{A_3}{r_{i-1}-r_i})$$

$$-(\rho C_p)_{i,1,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_2-z_1)}{t_{k+1}-t_k})]T_{i,1,k+1}$$

$$=-g_{i,1,k}2\pi r_i(r_i-r_{i-1})(z_2-z_{1)}$$

$$-(\rho C_p)_{i,1,k}(\frac{2\pi r_i(r_i-r_{i-1})(z_2-z_1)}{t_{k+1}-t_k})T_{i,1,k}$$

$$+Q_{1,k}^{Cond.}A_1+Q_{1,k}^{Rad.}A_1+Q_{ave}^{EEC}A_1 \qquad (3.20)$$

3.2.9 Collector Surface:

The energy balance for the collector surface is shown in Figure 3.12. The temperature equations for this surface were derived in a similar way to those for the emitter surface and are as follows:

$$k_{i,j+1/2,k}(\frac{A_2}{z_{j+1}-z_j})T_{i,j+1,k+1}+k_{i+1/2,j,k}(\frac{A_1}{r_{i+1}-r_i})T_{i+1,j,k+1}$$

$$+k_{i,j-1/2,k}\left(\frac{A_4}{z_j-z_{j-1}}\right)T_{i,j-1,k+1} - [k_{i+1/2,j,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)$$

$$+k_{i,j+1/2,k}\left(\frac{A_2}{z_{j+1}-z_j}\right) + k_{i,j-1/2,k}\left(\frac{A_4}{z_{j-1}-z_j}\right)$$

$$-(\rho C_p)_{i,j,k}\left(\frac{2\pi r_i(r_{i+1}-r_i)(z_{j+1}-z_j)}{t_{k+1}-t_k}\right)]T_{i,j,k+1}$$

$$= -g_{i,j,k}2\pi r_i \Delta r \Delta z$$

$$-(\rho C_p)_{i,j,k}\left(\frac{2\pi r_i \Delta r \Delta z}{t_{k+1}-t_k}\right)T_{i,j,k}$$

$$-Q_{j,k}^{Cond.}A_3 - Q_{j,k}^{Rad.}A_3 + Q_{ave}^{CEH}A_3 \qquad (3.21)$$

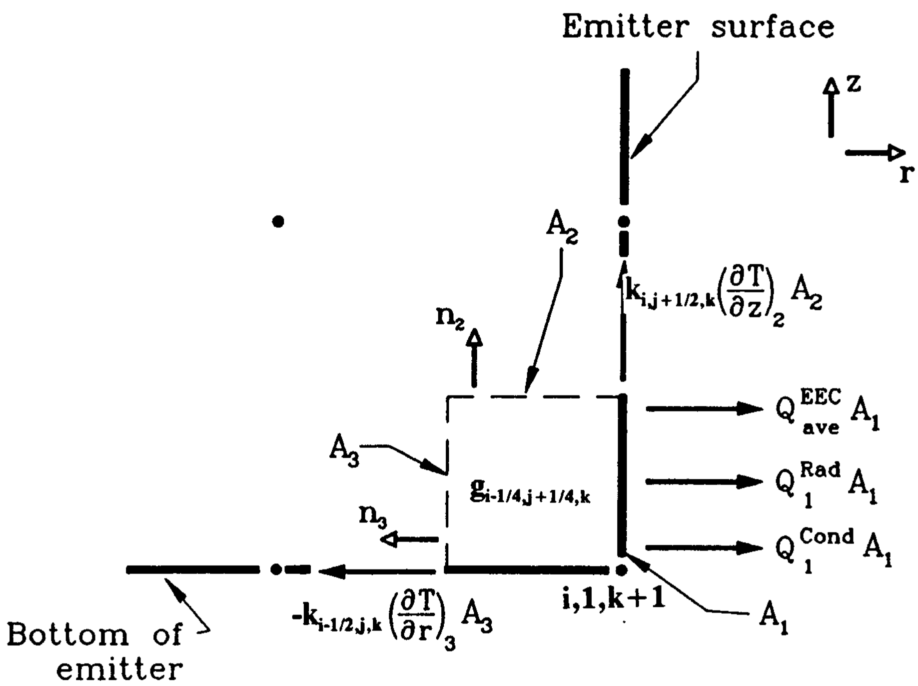

Figure 3.11 Energy balance for the mesh point (i,1,k+1), which is located on the emitter surface at the bottom of the pin.

The Q^{CEH}_{ave} term differs from the Q_{ave}^{EEC} term in that the effect of the plasma radiation is added to Q_{ave}^{CEH}, whereas it is subtracted from Q_{ave}^{EEC}; plasma radiation reduces the amount of heat removed from the emitter, but it increases the amount of heat added to the collector.

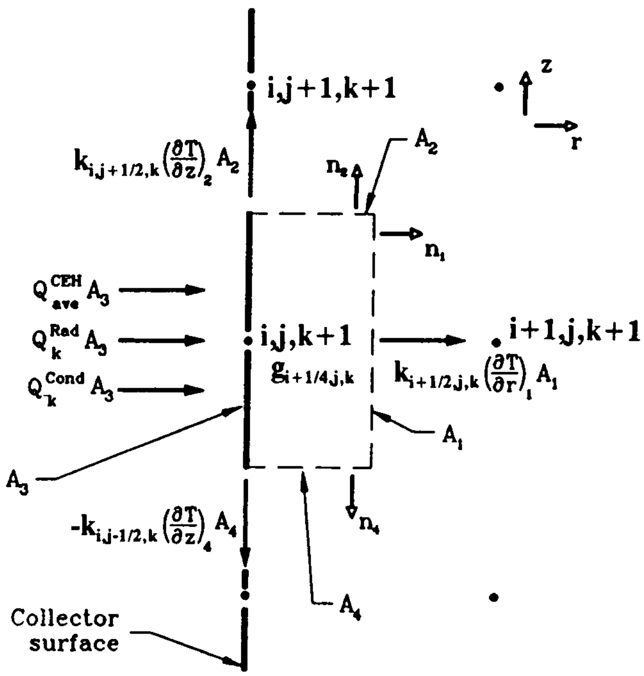

Figure 3.12 Energy balance for mesh points (i,j,k+1), which are located on the collector surface.

3.2.10 Collector Surface-Top of the TFE:

Figure 3.13 shows the heat balance for the mesh point at the top of the collector surface. The temperature equation is given as:

$$k_{i+1/2,j\max,k}\left(\frac{A_1}{r_{i+1}-r_i}\right)T_{i+1,j\max,k+1} + k_{i,j\max-1/2,k}\left(\frac{A_4}{z_{j\max}-z_{j\max-1}}\right)T_{i,j\max-1,k+1}$$

$$-\left[k_{i+1/2,j\max,k}\left(\frac{A_1}{r_{i+1}-r_i}\right) + k_{i,j\max-1/2,k}\left(\frac{A_4}{z_{j\max-1}-z_{j\max}}\right)\right.$$

$$\left.-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_i(r_i-r_{i-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k}\right)\right]T_{i,j\max,k+1}$$

$$= -g_{i,j\max,k}2\pi r_i(r_i - r_{i-1})(z_{j\max} - z_{j\max-1})$$

$$-(\rho C_p)_{i,j\max,k}\left(\frac{2\pi r_i(r_i-r_{i-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k}\right)T_{i,j\max,k}$$

$$-Q^{Cond.}_{j\max,k}A_3 - Q^{Rad.}_{j\max,k}A_3 + Q^{CEH}_{ave}A_3 \qquad (3.22)$$

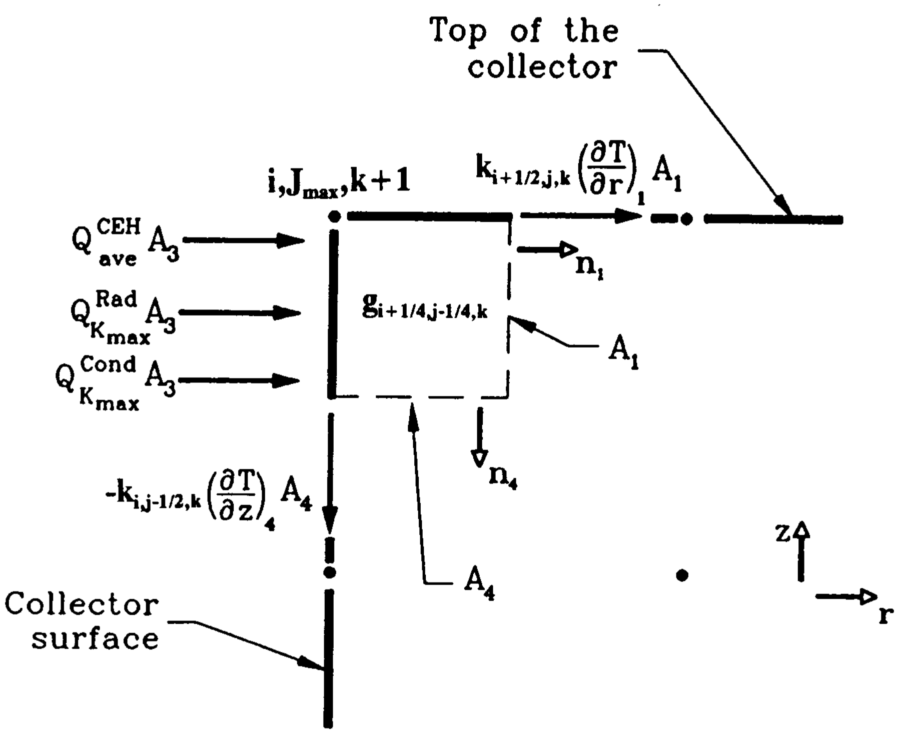

Figure 3.13 Energy balance for the mesh point $(i,j_{max},k+1)$, which is located at the top of the pin and on the collector surface.

3.2.11 Collector Surface-Bottom of the TFE:

Figure 3.14 shows the heat balance for the mesh point at the bottom of the collector surface. The temperature equation is:

$$k_{i+1/2,1,k}(\frac{A_1}{r_{i+1}-r_i})T_{i+1,1,k+1} + k_{i,3/2,k}(\frac{A_2}{z_2-z_1})T_{i,2,k+1}$$

$$-[k_{i+1/2,1,k}(\frac{A_1}{r_{i+1}-r_i}) + k_{i,3/2,k}(\frac{A_2}{z_2-z_1})$$

$$-(\rho C_p)_{i,1,k}(\frac{2\pi r_i(r_{i+1}-r_i)(z_2-z_1)}{t_{k+1}-t_k})]T_{i,1,k+1}$$

$$= -g_{i,1,k}2\pi r_i\Delta r(z_2 - z_1)$$

$$-(\rho C_p)_{i,1,k}(\frac{2\pi r_i\Delta r(z_2-z_1)}{t_{k+1}-t_k})T_{i,1,k}$$

$$-Q_{1,k}^{Cond.}A_3 - Q_{1,k}^{Rad.}A_3 + Q_{ave}^{CEH}A_3 \qquad (3.23)$$

3.2.12 Cladding/Coolant Interface:

The energy balance for the mesh points at the cladding/coolant interface is shown in Figure 3.15. The temperature equations are:

$$h_{j,k}(A_1)T_{j,k+1}^{Coolant} + k_{i,j+1/2,k}(\frac{A_2}{z_{j+1}-z_j})T_{i,j+1,k+1}$$

$$+k_{i-1/2,j,k}(\frac{A_3}{r_i-r_{i-1}})T_{i-1,j,k+1} + k_{i,j-1/2,k}(\frac{A_4}{z_j-z_{j-1}})T_{i,j-1,k+1}$$

$$-[h_{j,k}(A_1) + k_{i,j+1/2,k}(\frac{A_2}{z_{j+1}-z_j})$$

$$+k_{i-1/2,j,k}(\frac{A_3}{r_{i-1}-r_i}) + k_{i,j-1/2,k}(\frac{A_4}{z_{j-1}-z_j})$$

$$-(\rho C_p)_{i,j,k}(\frac{2\pi r_i \Delta r \Delta z}{t_{k+1}-t_k})]T_{i,j,k+1}$$

$$= -g_{i,j,k} 2\pi r_i \Delta r \Delta z - (\rho C_p)_{i,j,k}(\frac{2\pi r_i \Delta r \Delta z}{t_{k+1}-t_k})T_{i,j,k} \tag{3.24}$$

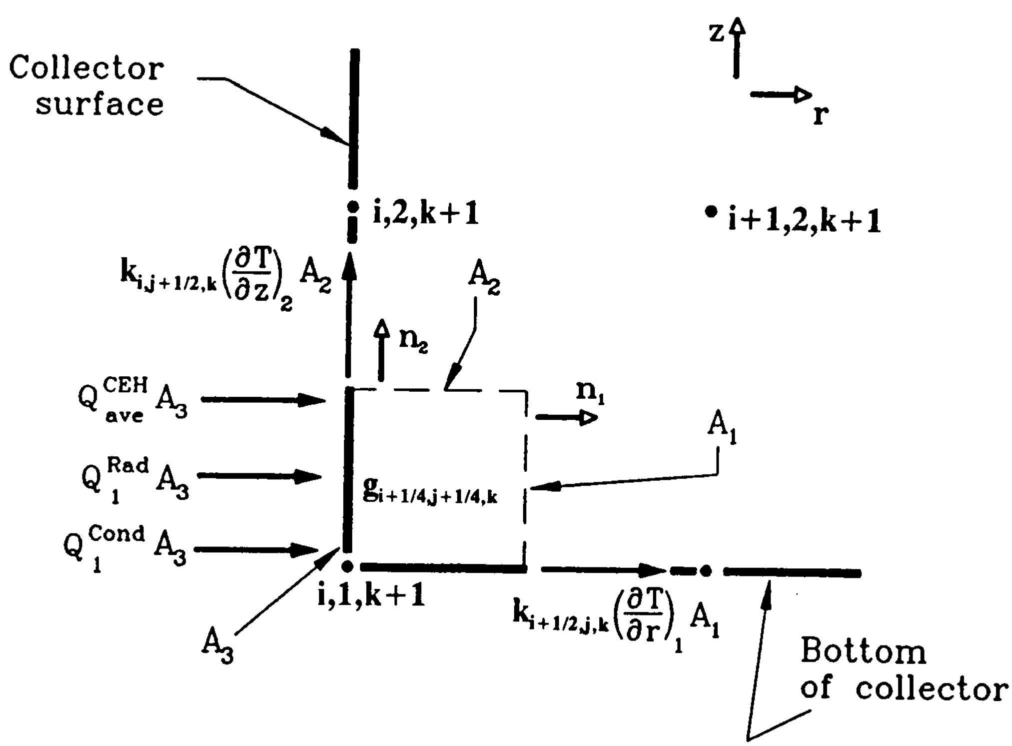

Figure 3.14 Energy balance for the mesh point (i,1,k+1), which is located at the bottom of the TFE and on the collector surface.

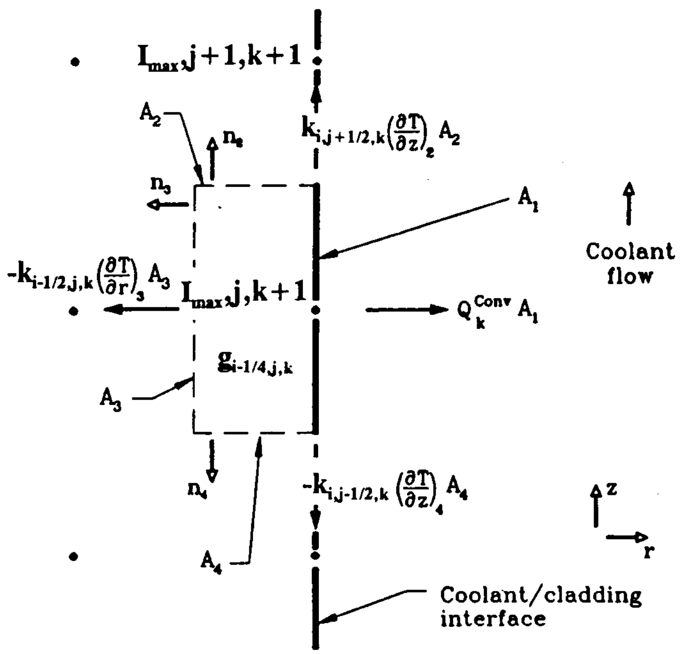

Figure 3.15 Energy balance for mesh points (I_{max},j,k+1), which are located at the cladding/coolant interface.

3.2.13 Cladding/Coolant Interface-Top of the TFE:

The energy balance on the cladding/coolant interface at the top of the TFE is shown in Figure 3.16. The temperature equation for this point is as follows:

$$h_{j\max,k}(A_1)T^{Coolant}_{j\max,k+1} + k_{i\max-1/2,j\max,k}\left(\frac{A_3}{r_{i\max}-r_{i\max-1}}\right)T_{i\max-1,j\max,k+1}$$

$$+k_{i\max,j\max-1/2,k}\left(\frac{A_4}{z_{j\max}-z_{j\max-1}}\right)T_{i\max,j\max-1,k+1} - [h_{j\max,k}(A_1)$$

$$+k_{i\max-1/2,j\max,k}\left(\frac{A_3}{r_{i\max-1}-r_{i\max}}\right) + k_{i\max,j\max-1/2,k}\left(\frac{A_4}{z_{j\max-1}-z_{j\max}}\right)$$

$$-(\rho C_p)_{i\max,j\max,k}(\frac{2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k})]T_{i\max,j\max,k+1}$$

$$= -g_{i\max,j\max,k}(2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_{j\max}-z_{j\max-1}))$$

$$-(\rho C_p)_{i\max,j\max,k}(\frac{2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_{j\max}-z_{j\max-1})}{t_{k+1}-t_k})T_{i\max,j\max,k} \qquad (3.25)$$

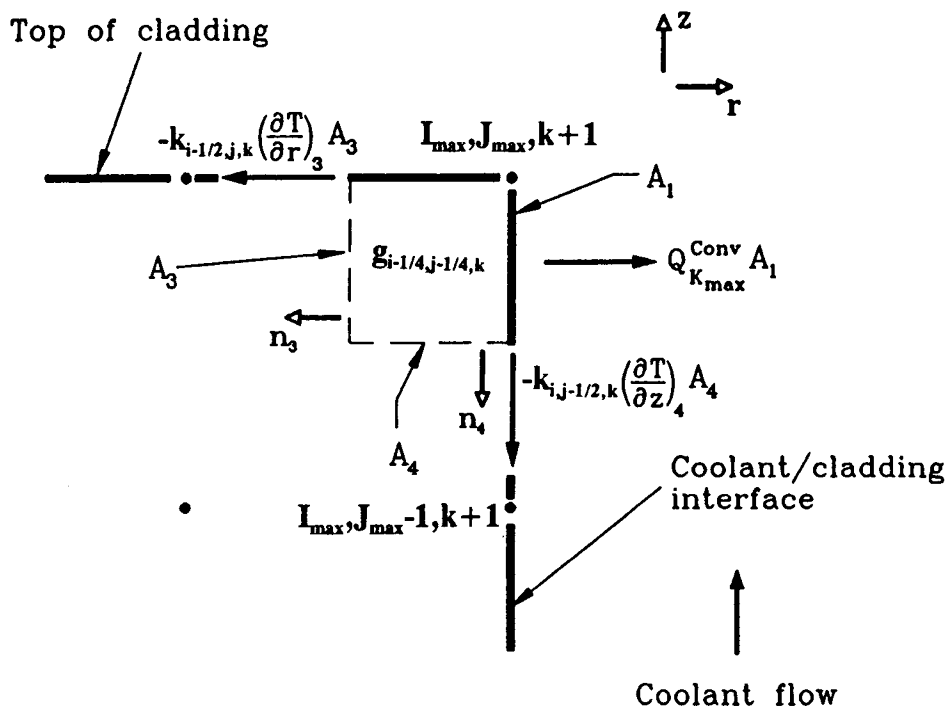

Figure 3.16 Energy balance at the mesh point (I_{max}, J_{max}, k+1), which is located at the cladding/coolant interface and at the top of the TFE.

3.2.14 Cladding/Coolant Interface-Bottom of the TFE:

Figure 3.17 shows the energy balance for the mesh point at the bottom of the cladding/coolant interface of the TFE. The temperature equation for this point is:

$$h_{1,k}(A_1)T^{Coolant}_{1,k+1} + k_{i\max,3/2,k}\left(\frac{A_2}{z_2-z_1}\right)T_{i\max,2,k+1}$$

$$+k_{i\max-1/2,1,k}\left(\frac{A_3}{r_{i\max}-r_{i\max-1}}\right)T_{i\max-1,1,k+1} - [h_{1,k}(A_1)$$

$$+k_{i\max,3/2,k}\left(\frac{A_2}{z_2-z_1}\right) + k_{i\max-1/2,1,k}\left(\frac{A_3}{r_{i\max}-r_{i\max-1}}\right)$$

$$-(\rho C_p)_{i\max,1,k}\left(\frac{2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_2-z_1)}{t_{k+1}-t_k}\right)]T_{i\max,1,k+1}$$

$$= -g_{i\max,1,k}(2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_2-z_1))$$

$$-(\rho C_p)_{i\max,1,k}\left(\frac{2\pi r_{i\max}(r_{i\max}-r_{i\max-1})(z_2-z_1)}{t_{k+1}-t_k}\right)T_{i\max,1,k} \qquad (3.26)$$

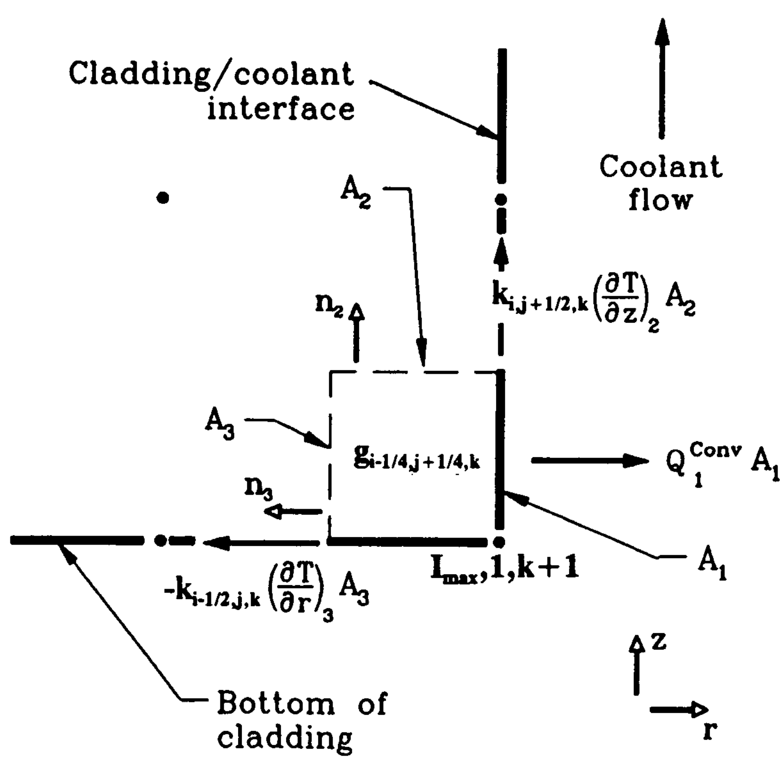

Figure 3.17 Energy balance at the mesh point (I_{max},1,k+1), which is located at the cladding/coolant interface and at the bottom of the TFE pin.

3.3 Coolant Transient Convection:

The heat convection subroutine TConvect is used to solve for the transient convection heat transfer in the coolant [12] by using the fully implicit scheme as follows.

$$\frac{Pq''}{A} = G\frac{\partial h}{\partial Z} + \rho\frac{\partial h}{\partial t} \tag{3.27}$$

Applying the discretization method yields

$$\frac{\partial h}{\partial z} = \frac{h_{j,k+1} - h_{j-1,k+1}}{\Delta z} \tag{3.28}$$

$$\frac{\partial h}{\partial t} = \frac{h_{j,k+1} - h_{j,k}}{\Delta t} \tag{3.29}$$

where

h = Enthalpy, J/Kg.

G = Mass flowrate (Kg/sec.)

P = Heated parameter of channel, cm^2.

A = Area of channel, cm^2.

q"= Heat flux at surface of channel, watt/cm^2.

ρ = Density of the coolant, Kg/cm^3.

Plugging equations 3.28 and 3.29 into equation (3.27) yields:

$$\frac{P}{A}q^{\approx}_{j,k} = G_k\frac{h_{j,k+1}-h_{j-1,k+1}}{\Delta z}+\rho_k\frac{h_{j,k+1}-h_{j,k}}{\Delta t} \tag{3.30}$$

$$\frac{\rho_k}{\Delta t}h_{j,k}+\frac{P}{A}q^{\approx}_{j,k} = \left(\frac{G_k}{\Delta z}+\frac{\rho_k}{\Delta t}\right)h_{j,k+1}-\frac{G_k}{\Delta z}h_{j-1,k+1}$$

but $dh = C_p dT$

$$\frac{\rho_k}{\Delta t}T_{j,k}+\frac{P}{AC_p}q^{\approx}_{j,k} = \left(\frac{G_k}{\Delta z}+\frac{\rho_k}{\Delta t}\right)T_{j,k+1}-\frac{G_k}{\Delta z}T_{j-1,k+1}$$

3.3.1 Stability

From equation 3.30

$$\frac{\frac{\rho_k}{\Delta t}}{\frac{G_k}{\Delta z}+\frac{\rho_k}{\Delta t}} \rangle\ 0 \tag{3.31}$$

Equation 3.31 is true for all Δt and Δz

3.3.2 Initial Conditions

$t=0,\ k=1$

$T_{j,1} = T_{coolant}$

$j = 1,N$

3.3.3 Boundary Conditions

$z=0,\ j=1$

$T_{1,k} = T_{inlet}$

For j=1

$$\frac{\rho_k}{\Delta t}T_{1,k} + \frac{P}{AC_p}\tilde{\tilde{q}}_{1,k} = \left(\frac{G_k}{\Delta z} + \frac{\rho_k}{\Delta t}\right)T_{1,k+1}$$

$$\Rightarrow \quad T_{1,k+1} \quad = \quad \left[\frac{G_k}{\Delta z} + \frac{\rho_k}{\Delta t}\right]^{-1}\cdot\left\{\frac{\rho_k}{\Delta t}T_{1,k} + \frac{\rho}{AC_p}\tilde{\tilde{q}}_{1,k}\right\}$$

The general equation for transient convection in the coolant can be written as:

$$T_{j,k+1} \quad = \quad \left[\frac{G_k}{\Delta z} + \frac{\rho_k}{\Delta t}\right]^{-1}\left\{\frac{\rho_k}{\Delta t}T_{j,k} + \frac{\rho}{AC_p}\tilde{\tilde{q}}_{j,k} + \frac{G_k}{\Delta z}T_{j-1,k+1}\right\} \tag{3.32}$$

3.4 Computer Code Implementation:

The spatial temperature distribution within the TFE is computed through finite volume analysis. The materials within the TFE [3] are subdivided into a series of small control volumes upon which an energy balance is performed. This results in a set of first-order nonhomogeneous partial differential equations which have variable coefficients, and which may be non-linear, depending on the location of the finite element within the TFE. Variable coefficients occur due to the temperature dependence of thermal conductivities, specific heats, and densities of the TFE materials. Nonlinear terms arise due to the heat transfer processes across the emitter/collector gap (i.e., thermionic emission, thermal radiation, and conduction through the vapor) and due to convection to the coolant.

The fully implicit method is used to solve the system of linear equations for temperatures at each time step. In the TFE, as many as 100 temperatures need to be solved at each time step. All temperature coefficients are cast into a matrix which has a banded structure of 10. The implicit method is efficient and reliable due to its stability at any given time step.

Except for the transient specific codes and subroutines, much of the thermionic-specific internals of the TFETC are based on TFEHX, a steady state code for thermionic fuel element which in turn uses CYLCON6 for modifying the TFE. CYLCON6 is called by TIMPLCIT, a transient subroutine in the TFETC which solves for the fully implicit scheme. For solving the linear equation resulting from discretization of the heat transfer equation, the user has the option of selecting either a Gaussian elimination package, which was inherited from TFEHX, or a sparse linear solver called Y12M. Y12M solves sparse systems of linear algebraic equations by Gaussian elimination. The subroutine is a "black box subroutine" designed to solve efficiently problems which contain only one system with a single right hand side. For details, see the documentation of Y12M is available on the internet by anonymous ftp from netlib at research.att.com in the directory Y12M; login as netlib[17].

References:

1. M. M. El-Wakil, Nuclear Heat Transport, 3rd Edition, The American Nuclear Society Press., 1981.

2. O. Faust,"Sodium-NaK Engineering Handbook", Vol.1, pp. 52-53, 1972.

3. A.C. Klein, H.H. Lee, B.R. Lewis, R.A. Pawlowski and Shahab Abdul-Hamid, "Advanced Single Cell Thermionic Reactor System Design Studies", Oregon State University, OSU-NE-9209, Corvallis, OR Sept. 1992.

4. B.R. Lewis, R.A. Pawlowski, K.J. Greek and A.C. Klein, "Advanced Thermionic Reactor System Design Code", Proceedings of 8th Symposium on Space Nuclear Power Systems, CONF-910116, Albuquerque, NM, 1991.

5. Borland, Inc., "QUATTRO-PRO 3.0 Superior Spreadsheet Power", 1991.

6. Lahey Computer System, Inc.,"F77L/EM-32 Version 3", NV, 1990.

7. J.K. Hohorst, "SCDAP/RELAP5/MOD2 Code Manual, vol.4:MATPRO-A Library of Materials Properties for Light-Water Reactor Accident Analysis", NUREG/CR-5273; EGG-2555; EG&G Idaho, Inc., February 1990.

8. J.B. McVey, "Preliminary Technical Report-CYLCON Semi-2D Cylindrical Converter Model", Rasor Associates, Inc., Sunnyvale, CA, 1990.

9. J.B. McVey, "Planar Converter Standard Model Documentation Supplementary Description of TECMDL Converter Physics", E-563-002-B-063087, Rasor Associates, Inc., Sunnyvale, CA, 1990.

10. J.B. McVey, "TECMDL-Ignited Mode Planar Converter Model", E-563-004-C-082988, Rasor Associates, Inc., Sunnyvale, CA, Aug. 1990.

11. J.B. McVey, G.L. Hatch, and K.J. Greek, Rasor Associates, Inc., and G.J. Parker, W.N.G. Hitchon, and J.E. Lawler, University of Wisconsin-Madison, "Comprehensive Time Dependent Semi-3D Modeling of Thermionic Converters In Core", NSR-53 / 92-1004, Sept. 30, 1992.

12. J. Weisman, Elements of Nuclear Reactor Design, Kreiger Publishing Company, 1983.

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

14. G.D. Smith, Numerical Solution of Partial Differential Equations, Oxford University Press, NY 1965.

15. D.R. Croft and D.G. Lilley, Heat Transfer Calculations Using Finite Difference Equations, Applied Science Publishers Ltd., London 1977.

16. M.L. James, G.M. Smith, and J.C. Wolford, Applied Numerical Methods for Digital Computation with FORTRAN and CSMP, Harper & Row, Publishers, Inc., 2nd Edition, 1977.

17. Z. Zlatev, J. Wasniewski, and k. Schaumburg, Y12M Solution of Large and Sparse Systems of Linear Algebraic Equations, Springer-Verlag Inc., 1981.

18. E.E. Lewis, Nuclear Power Reactor Safety, 1st Edition, John Wiley & Sons, Inc., 1977.

Chapter 4

Results and Analysis

A transient code (TFETC) for calculating the temperature distribution throughout the radial and axial positions of a thermionic fuel element (TFE) has been successfully developed. It accommodates the variations of temperatures, thermal power, electrical power, voltage, and current density throughout the TFE as a function of time as well as the variations of heat fluxes arising from radiation, conduction, electron cooling, and collector heating. The thermionic fuel element transient code (TFETC) is designed to calculate all the above variables for three different cases namely: 1) Start-up; 2) Loss of flow accident; and 3) Shut down. In this chapter, the results obtained from the code are presented and analyzed. The results for the start up case are shown on Figures 4.1 through 4.11, those for the loss of flow accident (LOFA) on Figures 4.12 through 4.25, and finally those for the shut down case are on Figures 4.26 through 4.45.

4.1 Start up:

In Figure 4.1, the fuel temperature increases with time until it reaches the steady state temperature (2496°K). Many factors affect the fuel temperature profile of the TFE. Among these are a) thermal power rise coefficient, τ, b) helium heating, c) heat removal, and d) onset of electron cooling (EC). Values of 100, 300, 600, and 1200 seconds were used for the power rise coefficient, τ. During start up, helium heating is used to raise the temperature of the collector surface of the TFE assembly. The emitter temperature Tstop, at which helium heating stops, is a user supplied input with a nominal value of 900 °K. Similarly, the emitter temperature, Tstart, at which electron cooling begins, is a user supplied input with a nominal value of 1900 °K. When the heat generation in the fuel is high, i.e. τ=100 sec., the net effect of electron cooling (EC)(which begins a little later) is small, hence the little dip in the fuel temperature. For an appreciably lower power rise (τ=1200 sec.), however, the effect of EC is rather pronounced. Regardless of these transient effects, the fuel temperature does reach the steady state value for all τ's.

The emitter temperature profiles follow a similar pattern to that of the fuel as shown in Figure 4.2, because the only heat transfer mode between the emitter and the fuel surfaces is conduction. Notice that all the temperature fluctuations in the emitter occur around 1900 °K, the EC temperature set-point.

The coolant temperatures, as shown on Figure 4.3, increase with time until the steady state temperature is reached for all power rise coefficients. The small dips are due to the nonlinear EC effects.

As expected, the thermal power (see Figure 4.4) increases exponentially until it attains the steady state value (3177 watts) according to the following equation:

$$P_{th}=P_{ss}\cdot[1-e^{(-t/\tau)}]$$

where

P_{ss} = steady state thermal power

= 3177 watts.

The thermal power may reach the steady state value either in fast mode or slow mode depending on the power rise coefficient, τ. The electric power profile is a function of emitter electron cooling according to the following equation [2] :

$$P_E=\frac{Q^{EC}}{(V_E+2\frac{kT_E}{e})}\cdot V \tag{4.1}$$

where

Q^{EC} = Electron cooling, watt/cm^2.

V = Output voltage, volts.

V_E = Emitter potential difference, volt.

K = Boltzman constant = 8.62 x 10^{-5} ev/°K.

e = Electron charge

It is obvious from equation 4.1 that the electric power, as shown in Figure 4.5, is directly proportional to the electron cooling which follows the profile illustrated in Figure 4.8. The electron cooling heat flux, Q^{EC}, increases with time without any fluctuations for fast start up but the situation is different for slow start up. Q^{EC} is computed directly by CYLCON6, while Q^{CH} is computed from

$$Q^{CH} = Q^{EC} - J.V$$

where

J = Current density, watt/cm^2.

V = Interelectrode voltage, volt.

Q^{CH} is shown on Figure 4.9, while the current density, J and interelectrode voltage V are shown on Figures 4.10 and 4.11 respectively. The voltage follows the pattern of electrical power according to the following equation

$$P_E = I.V$$

where I = total current input = 490 watts.

Radiation, Q^{Rad} and conduction, Q^{cscond} through the emitter collector gap are shown on Figures 4.6 and 4.7 respectively. The dip in the curve of radiation heat flux is due to the nonlinearity in the following equation:

$$Q_k^{Rad} = \sigma \epsilon_e F_{e\to c} (T_{e,k}^4 - T_{c,k}^4) \; [2\pi r_e (\frac{Z_{k+1} - Z_{k-1}}{2})]$$

where

σ = Stefan-Boltzman constant($5.67x10^{-12}$ Watts/cm^2k^4)

ϵ_e = Thermal emissivity of the emitter surface.

$F_{e\rightarrow c}$= View factor from the emitter surface to the collector surface ($F_{e\rightarrow c}$ = 1 for the emitter surface).

The cesium conductive heat flux, Q^{cscond}, profile is affected by the following equation:

$$Q_k^{Cond} = \frac{k_{Cs}(T_{e,k}-T_{c,k})}{d+1.15x10^{-5}\frac{(T_{e,k}-T_{c,k})}{p_{Cs}}}[2\pi r_e\frac{(Z_{k+1}-Z_{k-1})}{2}] \tag{4.3}$$

where

$T_{c,k}$ = Emitter temperature (°K).

$T_{c,k}$ = Collector temperature (°K).

k_{Cs} = Thermal conductivity of cesium vapor (W/cm.°K).

p_{Cs} = Pressure of cesium vapor at a cesium reservoir temperature (torr).

r_e = Emitter outside radius.

d = Emitter/collector gap (cm).

The TFETC code, for the start up case, works as expected without showing any deficiency. All temperatures, powers, fluxes, voltages, and currents behaved in a very consistent manner by increasing from zero power until reaching the steady state values. The running time of the code in the case of start up can be summarized as follows:

For a transient time of 1000 seconds and Δt of 0.5 second, on IBM PC486 machine of 33 Mhz with a math coprocessor, it takes about two hours and forty minutes . While for a transient time of 9000 seconds, the execution time is about 23 hours. For $\Delta t<0.5$ second, it takes a longer time for execution. Thermal power and coolant temperature for start up are in a good agreement with the results of the TOPAZ-II simulation [6].

4.2 Loss of Flow Accident (LOFA):

The second set of graphs, Figures 4.12 through 4.25 depict the results for the loss of flow accident (LOFA). The mass flow rate of the coolant for the LOFA case is modeled as

$$m^{\cdot}(t) = m^{\cdot}_o \; [\; A + B.e^{-t/\tau} \;] \qquad (4.4)$$

where

t = Transient time, sec.

$m^{\cdot}(t)$ = Mass flow rate as a function of time, Kg/sec.

$m^{\cdot}_o$ = Mass flow rate before LOFA begins, Kg/sec.

A,B = Pump failure coefficients. $(A+B)=1$, $B \neq 0$

τ = Mass loss coefficient, sec.

In Figure 4.12, the mass flow rate for complete pump failure (A=0.0, B=1.0) decreases quickly or slowly depending on the mass loss coefficient, τ. In the case of a LOF accident, four different pump failure cases are studied. After a period of time (4000 sec.) as shown on Figure 4.13, the mass flow rates attain the steady state values. The fuel temperature increases with time during the LOFA. The rate of temperature rise and final steady state values being governed by the fraction of pump failure. For complete pump failure (i.e 1/1 pump failure), the fuel temperature increases up to 2497 °K then stops because the code is halted upon reaching the boiling temperature of the coolant in 34 seconds. If the reactor is not shut down, the fuel temperature continues to increase until reaching the melting point of the fuel. Also, the fuel temperature increases about one degree in a complete pump failure after 34 seconds. The slow increase in fuel temperature is due to the existence of the emitter/collector gap that rejects heat to the coolant. On the other hand, the fuel temperature rise is directly proportional to the type of pump failure. For 1/2 pump failure as in Figure 4.14, the highest attainable steady state fuel temperature is about 2499 °K while for 1/4 pump failure, the highest attainable steady state fuel temperature is about 2498 °K. This means that when the heat removal of the coolant is small the fuel temperature, as in the case of 1/2 pump failure, has a high steady state value but when the heat removal is large, the fuel temperature, as in the case of 1/4 pump failure increases at a slower rate and attains a lower steady state value. In Figure 4.14, it should be noticed that the TFE design is reliable and efficient since the highest fuel temperature does not reach the melting point of the fuel. The heat removal in the coolant is a function of the mass flow rate. Figure 4.15 shows the emitter temperature profile for different pump failures. It has a similar profile as the fuel temperature because conduction is the only mode of heat transfer between the emitter and fuel surfaces.

In the case of a complete pump failure, as shown in Figure 4.16, the time needed for the coolant temperature to exceed the boiling point, which is 1057 °K, depends on the mass loss coefficient, τ. In the case with τ=30 seconds, it takes about 34 seconds to reach the NaK coolant boiling point. However, it takes about 120 seconds and 580 seconds to reach the NaK coolant boiling point for mass loss coefficients of 120 and 600 respectively. The maximum coolant temperatures, in the case of mass loss coefficient, τ, of 30 seconds, at different pump failures (see Figure 4.17) are listed below

$T_{max.}$ > 1057 °K for 1/1 pump failure.

$T_{max.}$ = 1017 °K for 1/2 pump failure.

$T_{max.}$ = 987 °K for 1/3 pump failure.

$T_{max.}$ = 977 °K for 1/4 pump failure.

The coolant temperature may exceed its boiling point if an appropriate action is not taken. The reactor would probably have a set point to trip the reactor when the coolant exit temperature got too high. Also, if the coolant temperature gets high then in a zirconium hydride moderated reactor, like the ATI or TOPAZ-II designs, the hydrogen will begin to disassociate from the ZrH, and this will add negative reactivity, thereby shutting down the reactor.

The thermal power in LOFA keeps constant before tripping the reactor as shown in Figure 4.18. It starts decreasing after the reactor is shut down. In the case of 1/1 pump failure, the mass flowrate decreases quickly and the heat is accumulated in the collector and emitter surfaces without being removed. This, in turn, increases the emission of electrons from the emitter surface that would be significant according to the following equation:

$$\eta = (J_E - J_C) \cdot \frac{V}{\Sigma Q} \tag{4.5}$$

where

η = Efficiency of thermionic fuel element.

J_E = Emitter current density, watt/cm^2.

J_C = Collector current density, watt/cm^2.

V = Output voltage, volt.

$$\Sigma Q = Q^{Rad}\ (\downarrow) + Q^{EC}\ (\uparrow) + Q^{k}\ (\downarrow)$$
$$= -\ 0.08 \ \ + 0.85 - 0.0036$$
$$= 0.7664 \text{ watt/cm}^2$$

The efficiency of the reactor will decrease according to the above equation. The electrical power drops off to its lowest value at a complete pump failure while it decreases a little until reaching the steady state value as in the cases of 1/2, 1/3, 1/4 pump failures, as shown in Figure 4.19. In the case of 1/2 pump failure, the maximum attainable power value is 315 watt. In the case of 1/3 pump failure, the maximum attainable power value is 314 watt. In the case of 1/4 pump failure, the maximum thermal power value is 307 watt.

The decrease of heat fluxes in Q^{Rad} and Q^{k}, as shown in Figures 4.20 and 4.21, is very small so their changes can be neglected. The decrease in Q^{Rad} is due to the back emission of the collector which affects the emissivity properties of the emitter surface. The electron cooling and collector heating terms are dependent on temperature variations. When the fuel temperature rises, the emitter temperature will rise too so the emission of electrons will increase. This increase in electron cooling is a function of heat removal which, in turn, is a function of the mass flowrate. In a complete loss of flow, the electron cooling increases by 0.8 watt/cm^2 in about 34 seconds (see Figure 4.22) while it does not increase by more than 0.1 watt/cm^2 in the case of loss of half pump failure. The electrical current density is a function of electron cooling as shown in Figure 4.24.

4.3 Shut down:

In the case of shut down four types of negative reactivity insertions are introduced. These reactivities are:

ρ = - \$0.10

ρ = - \$0.30

ρ = - \$0.90

ρ = - \$3.0

The larger the negative reactivity insertion the faster the shut down of the reactor. The fuel temperature shows the most decrease for the case of -\$3.0 reactivity insertion without showing any oscillations in the curve (see Figure 4.26), because the heat generation in the fuel drops off abruptly according to the prompt jump approximation. The sharp change shown for the -\$0.1 and -\$0.3 insertions are due to 1) the change of shut down mode of thermal power from prompt jump to exponential, and 2) the weak absorption of thermal neutrons in the case of low negative reactivity insertions which allows some thermal neutrons, not absorbed yet, to generate heat to the fuel and these neutrons will die out eventually. In large reactivity insertions, most of the thermal neutrons are absorbed so that the rest are not able to induce any significant change in the fuel temperature behavior.

The emitter temperature follows the same profile as the fuel temperature except that at low negative reactivity insertions (i.e., -$0.1), when the heat removal from the emitter surface will be less than the heat supply just right after the prompt jump. The coolant temperature drops off to the coolant inlet temperature of 895 °K. It takes about 700 seconds to reach the inlet temperature in the case of $3.0 negative reactivity insertion and 1500 seconds in the case of $0.1 insertion. The heat transfer coefficients (ρ, C_p) vary with time as a function of heat generation to the fuel. The enthalpy equation is:

$$dh = C_p \, Dt$$

or

$$\int dh = \int C_p dT$$

and the transient implicit finite difference equation [7] for the coolant is given by:

$$T_{j,k+1} = \left[\frac{G_k}{\Delta z} + \frac{\rho_k}{\Delta t}\right]^{-1} \left(\frac{\rho_k}{\Delta t} T_{j,k} + \frac{P}{AC_p} q''_{j,k} + \frac{G_k}{\Delta z} T_{j-1,k+1}\right) \qquad (4.6)$$

where

C_p = Specific heat of coolant, J/Kg.°K.

ρ = Coolant density, Kg/cm^3.

A = Flow area, cm^2.

P = Flow perimeter, cm.

G = Mass velocity, $Kg/sec.cm^2$.

q'' = Heat flux at coolant channel surface, $watt/cm^2$.

The coolant temperature behaves according to the above equation. It is noticed that the coolant temperature is a function of some fixed and some variable parameters as follows:

$T(\Delta z, \Delta t, P, A)$ Fixed Parameters

$T(\rho, C_p, G_k, q'')$ Variable Parameters

Equation 4.6 can be simplified by the following equation:

$$T_{j,k+1} = \frac{[\rho_k(\uparrow) T_{j,k} + \frac{q''_{j,k}(\downarrow)}{C_p(\uparrow)} + G_k(\downarrow) T_{j-1,k+1}]}{G_k(\downarrow) + \rho_k(\uparrow)} \quad (4.7)$$

The thermal power drops off sharply at large negative reactivity insertions and slower but with the same exponential pattern at low negative reactivity insertion as in Figure 4.29. The electrical power reaches zero at -\$0.1 insertion after 150 seconds and reaches zero at -\$3.0 insertion after 50 seconds as illustrated in Figure 4.30. The zero electrical power has to do with the emitter temperature set point Tstart. When the emitter temperature falls below this value, the CYLCON6 subroutine may not be able to converge and provide results for electrical power, hence, the call to CYLCON6 is stopped and the electrical power is set to zero. The radiation heat flux from the emitter surface follows the behavior of the emitter temperature. The higher the emitter temperature the higher the radiation heat flux and vice versa (see Figure 4.31). The conductive heat flux profile of cesium in the emitter/collector gap follows the same pattern as the emitter surface temperature as shown in Figure 4.32. The behavior of electron cooling and collector heating in Figures 4.33 and 4.34 is due to the behavior of the electrical current according to the following equation:

$$Q^{EEC} = J_E\left(V_E + 2\frac{kT_E}{e}\right) \tag{4.8}$$

where

J_E = Current density of the emitter surface, amp/cm^2.

V_E = Voltage across the emitter surface, volt.

T_E = Emitter temperature, °K.

K = Boltzman constant = 8.62 x 10^{-5} eV/°K.

The electron cooling (EC) is directly proportional to the current density as in the above equation. For the electron current density, it is stated that the current density increases at low output voltage values [3,4,5] and decreases at high output voltage according to the Richardson-Dushman equation. However, it is noticed in Figure 4.35 that the current density, for a large negative reactivity insertion (-\$3.0), increases to 8.5 watt/cm^2 in a transient time of about 50 seconds then decreases to its lowest value (zero) due to zero electrical power. For the lowest negative reactivity insertion (-\$0.1), the highest attainable value of current density is 6.29 watt/cm^2 after about 150 seconds. The voltage drop during shut down follows the electrical power profile as shown in Figure 4.36.

4.4 Start up at Different EC temperatures:

The start up of the TFE has been tested at different electron cooling temperatures of 1500, 1900, and 2000 °K respectively. The behavior of the temperatures, electrical power, current density, thermal power, voltage, and heat fluxes are nearly the same. The fuel, emitter, and coolant temperatures, in a small portion of the curves, are high at 2000 °K and low at 1500 °K as shown in Figures 4.37, 4.38, and 4.39 respectively. The reason behind this behavior arises from the fact that the heat removal by electron cooling at 2000 °K starts a little bit later than the heat removal by electron cooling at 1500 °K which allows a small increase in the temperatures before they attain the steady state temperatures. The electrical power at 2000 °K starts after 320 seconds of start up while it starts after 300 seconds in the case of 1900 °K and after 240 seconds in the case of 1500 °K. Of course, the electrical power depends on the current and in turn the current depends on the temperature difference between the emitter surface and the collector [1] surfaces. When the emitter and collector surface temperatures reach the steady state value then the electrical power will be stable and will attain the steady state value. The radiation heat flux (see Figure 4.41) is higher when the electron cooling starts late because the emissivity of the emitter surface increases with temperature increase and reaches its maximum value before the electron cooling starts. When the electron cooling starts earlier, however, the emissivity would not reach its maximum value because the heat removal starts earlier too. Figures 4.42 and 4.43 show that the electron cooling and collector heating heat fluxes are dependent on the emitter and collector surfaces respectively. The electron cooling starts earlier at 1500 °K and starts increasing until reaching the steady state value. The current density, as shown in Figure 4.44, is a function of the emitter and collector surface temperatures thus it follows the same profile. The voltage drop is a function of the electrical current and follows the same behavior.

4.5 The accuracy of the results:

There are two types of errors associated with the results. The first error is due to the discretization of the implicit method [9] and the second one is due to the round-off error of Gaussian elimination [8]. The error from the implicit method is considered to be as a function of 1) the time step; 2) Δr; and 3) Δz, where Δr and Δz are the radial thickness and axial length in respectively. The error arising from the Gaussian elimination depends somewhat on different parameters listed below:

1. Condition number of the matrix, K(A).
2. Size of the matrix.
3. Floating point precision.
4. The inaccuracy in the matrix element.

The exact temperature is given by:

$$T_1 = T_{exact} \pm M[\Delta t_1 + (\Delta r)^2 + (\Delta z)^2] \pm EGAUSS \tag{4.9}$$

$$T_2 = T_{exact} \pm M[\Delta t_2 + (\Delta r)^2 + (\Delta z)^2] \pm EGAUSS \tag{4.10}$$

where

Δr = Radial thickness of the TFE, cm.

Δz = Axial thickness of the TFE, cm.

Δt = Time increment, sec.

T_1 = Computed temperature for Δt_1 at a given point, °K.

T_2 = Computed temperature for Δt_2 at a given point, °K.

T_{exact} = Exact temperature at a given point, °K.

EGAUSS = Error arising from Gaussian.

M = Constant of the error arising from the implicit method.

M can be determined as follows:

$$M=\frac{T_2-T_1}{\Delta t_2-\Delta t_1} \tag{4.11}$$

where

Δt_2 = Time step at T_2, sec.

Δt_1 = Time step at T_1, sec.

$\Delta t_1 \ll \Delta t_2$

Substituting the value of M in equation 4.10 yields

$$T_2=T_{exact}\pm\frac{T_2-T_1}{\Delta t_2-\Delta t_1}\,[\Delta t_2+(\Delta r)^2+(\Delta z)^2]\pm EGAUSS \tag{4.12}$$

The relative bound error associated with Gaussian elimination is given by the following equation:

$$\frac{\|T_{exact}-T_1\|}{\|T_{exact}\|}\leq K(A)\,\frac{\|E\|/\|A\|}{1-K(A)\,\|E\|/\|A\|} \tag{4.13}$$

where

$K(A)$ = Condition of the matrix = $\|A\| . \|A^{-1}\|$.

A = Matrix.

E = Error associated with a matrix A.

$$\|E\| \leq (2n+1)g2^{-p} \tag{4.14}$$

where

g = Growth factor

n = Size of the matrix.

p = Binary floating point arithmetic.

If K(A) is close to unity, the matrix A is well-conditioned and if it is large, the matrix A is ill-conditioned. The size of E in equation 4.13 depends on the precision of the arithmetic used in the computation. The typical values used in the TFETC code were as follows:

Δt = 0.5 sec.

Δr = 0.125 cm.

Δz = 2.54 cm.

The error associated with the discretization of the implicit method depends mainly on the number of nodes in both axial and radial directions. It decreases with increasing of the node points and increases with decreasing of the node points. The largest error associated with the implicit method is the one arises from Δz, so it is advisable to reduce the value of Δz to be less than 1.

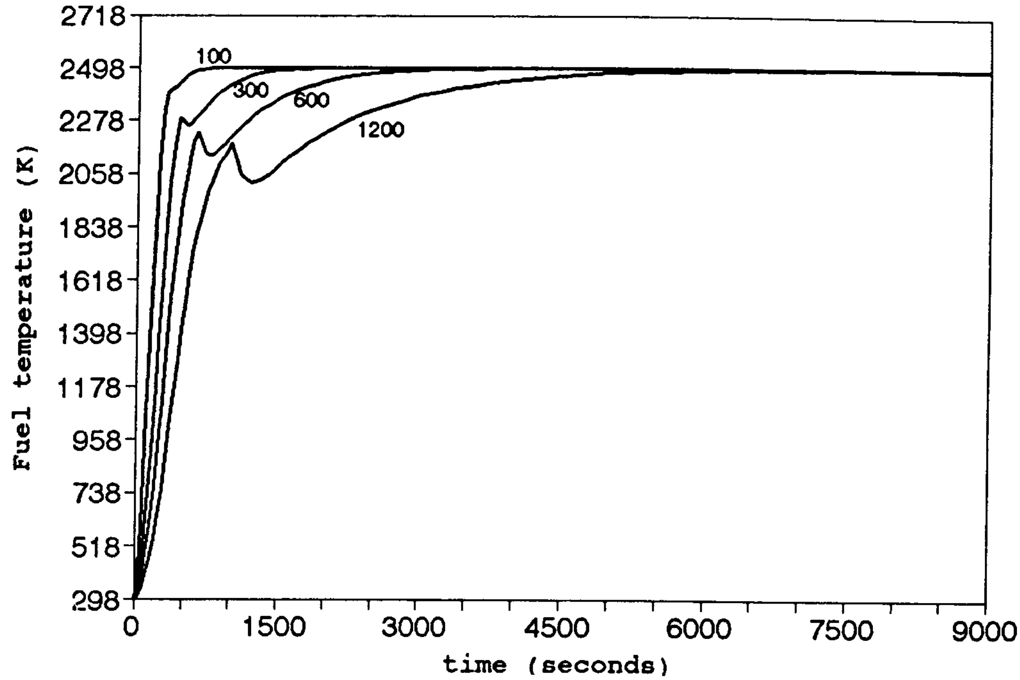

Figure 4.1 Fuel temperature profile for reactor start-up.

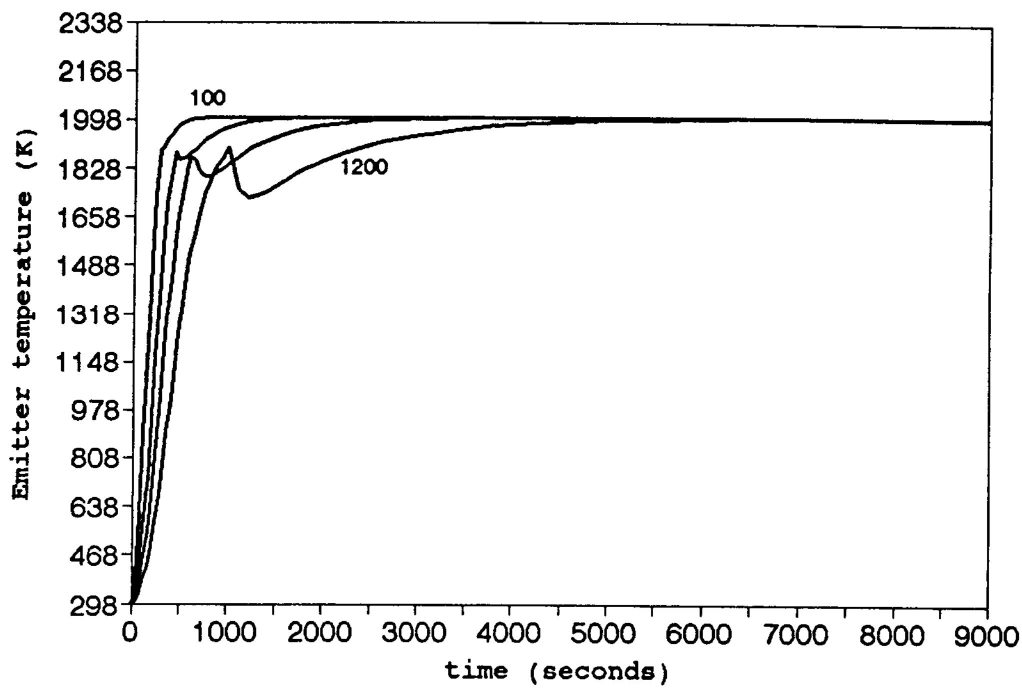

Figure 4.2 Emitter temperature profile for reactor start-up.

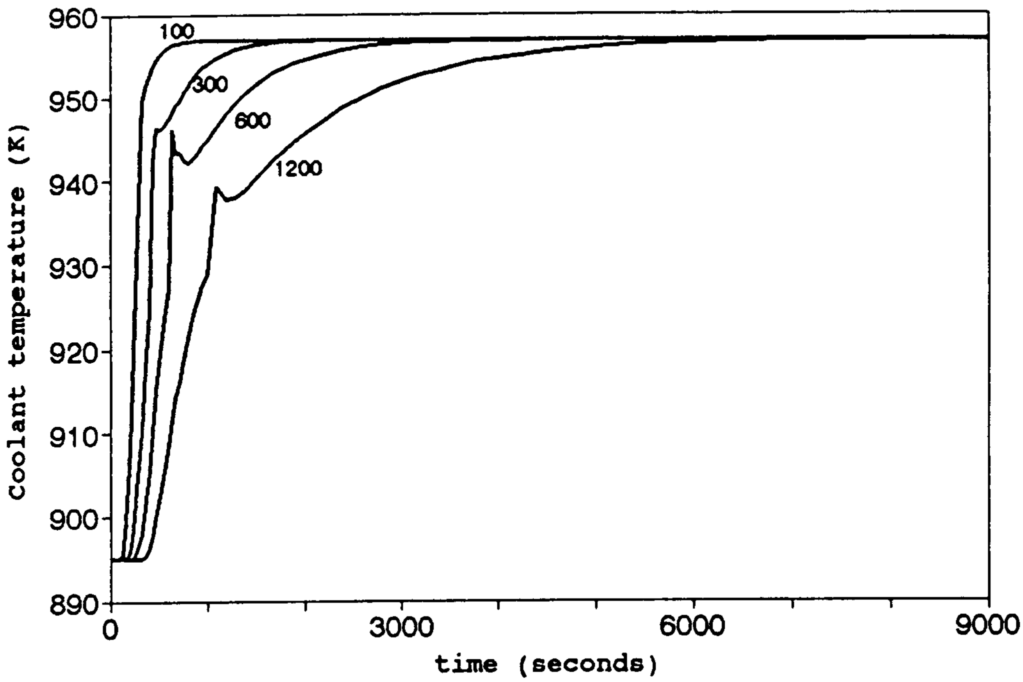

Figure 4.3 Coolant temperature profile for reactor start-up.

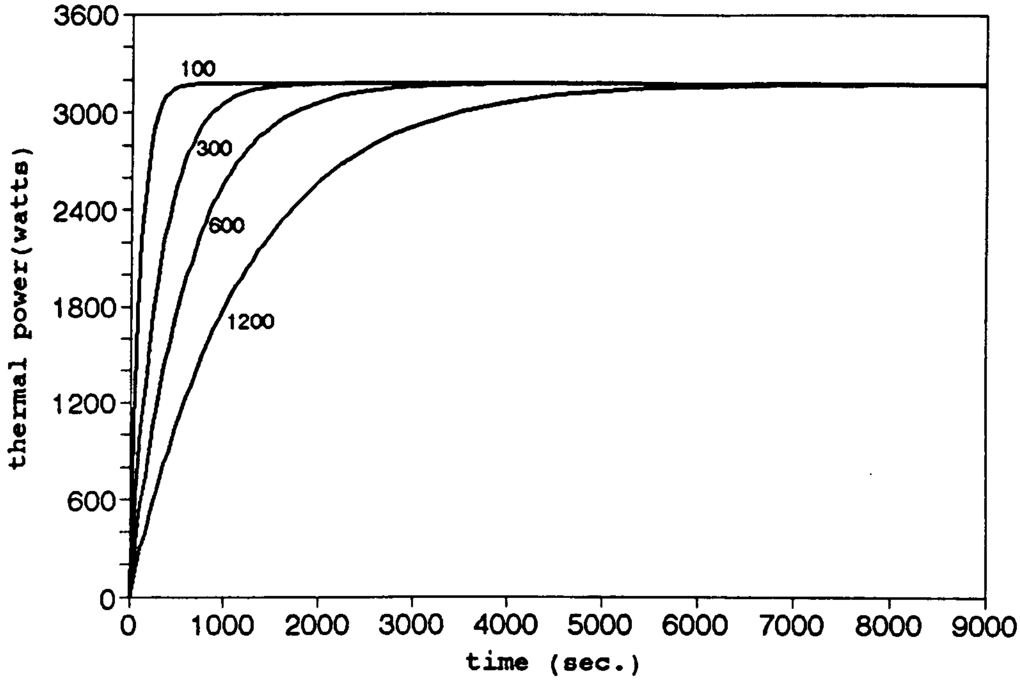

Figure 4.4 Thermal power profile for reactor start-up.

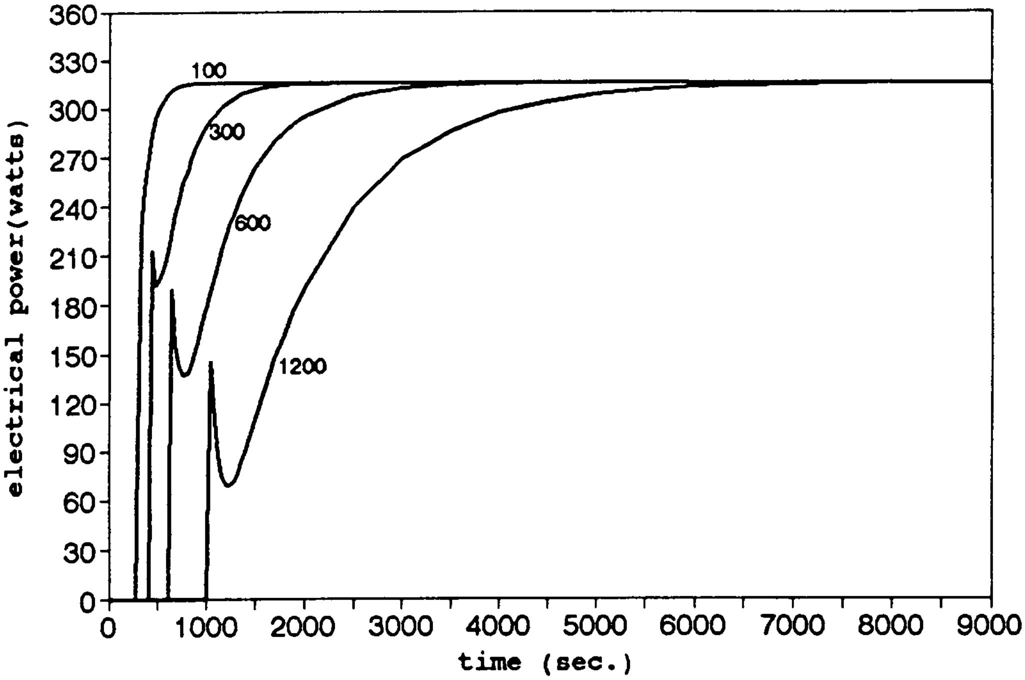

Figure 4.5 Electrical power profile for reactor start-up.

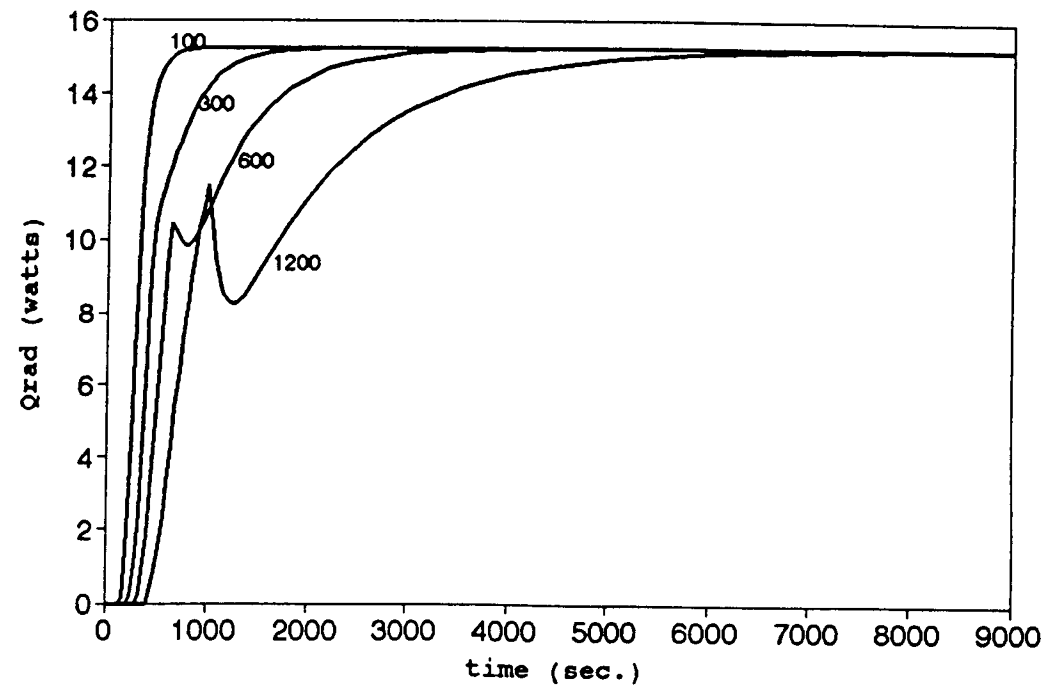

Figure 4.6 Radiation heat flux distribution for reactor start-up.

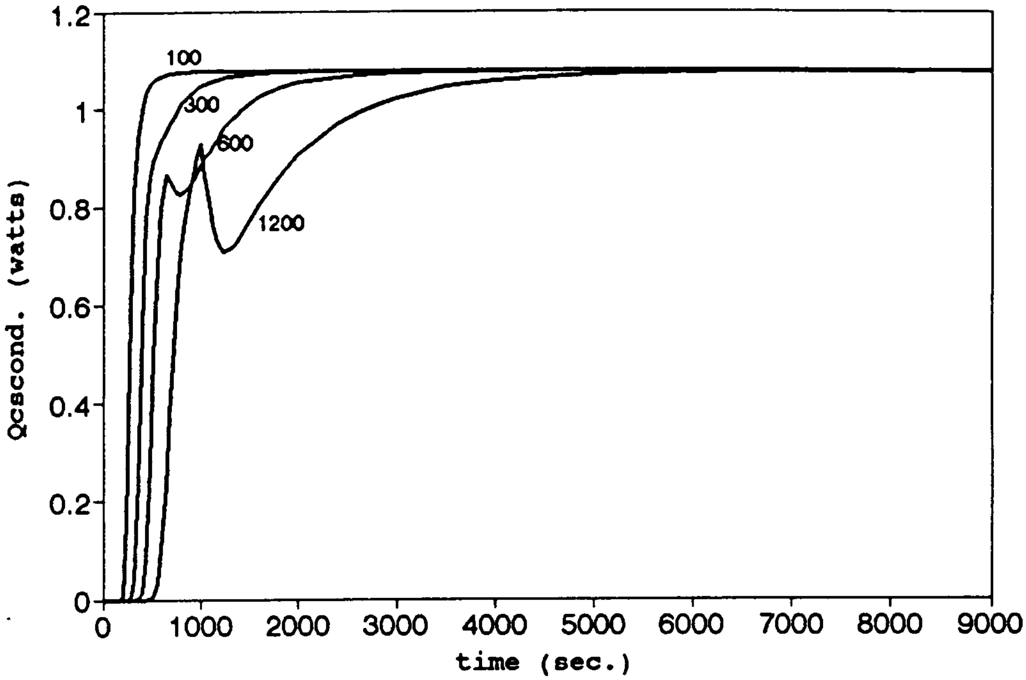

Figure 4.7 Conductive heat flux distribution of cesium for reactor start-up.

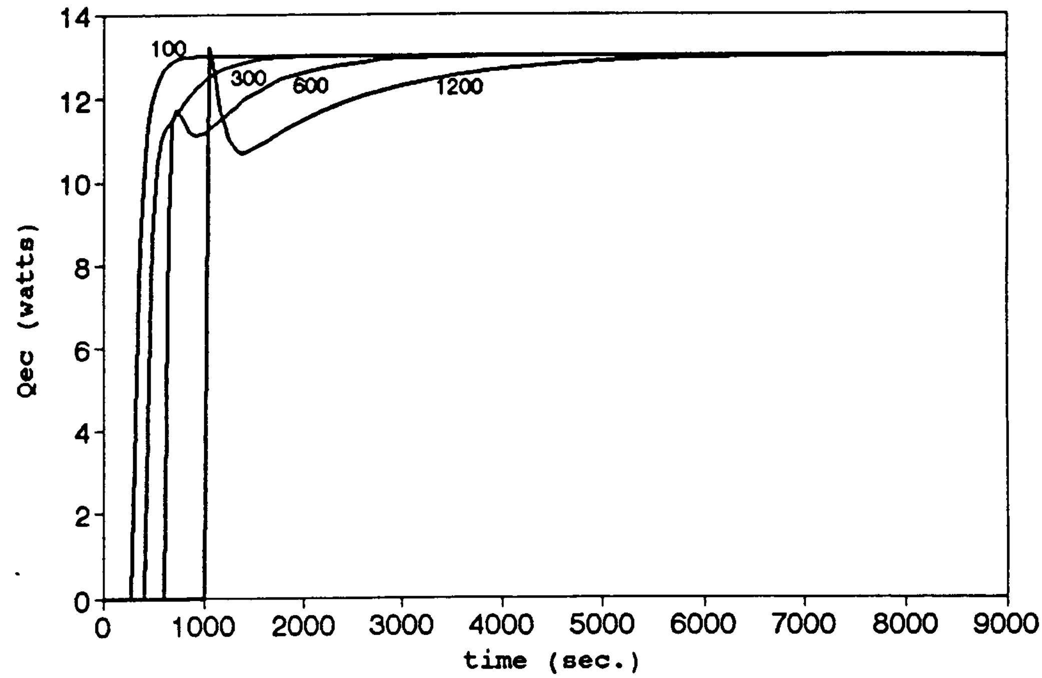

Figure 4.8 Heat flux distribution of emitter electron cooling for reactor start-up.

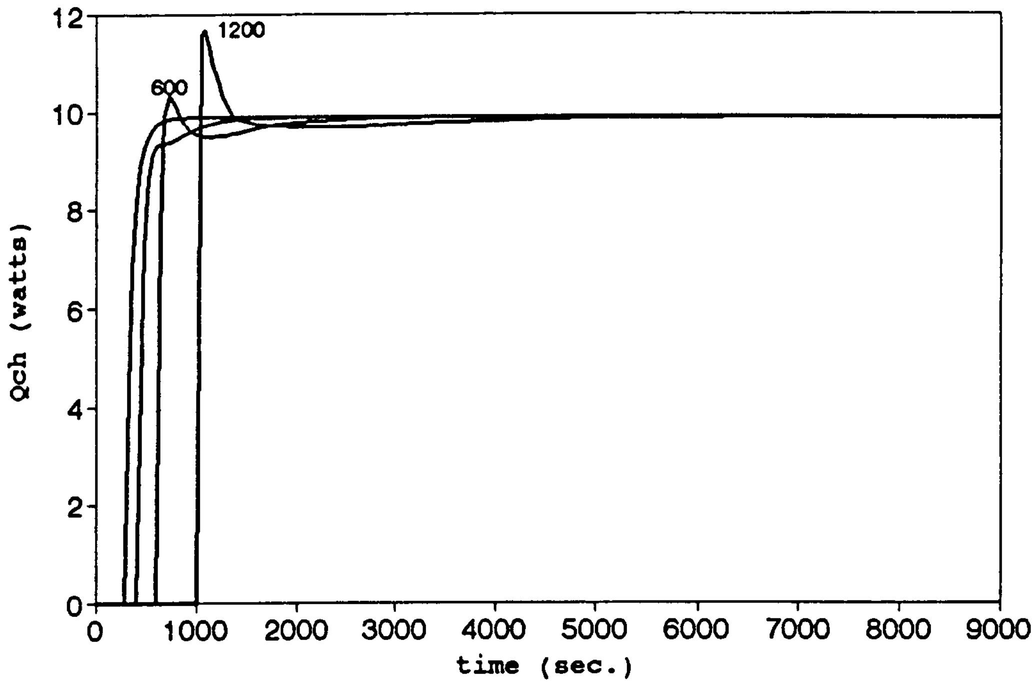

Figure 4.9 Heat flux distribution of collector electron heating for reactor start-up.

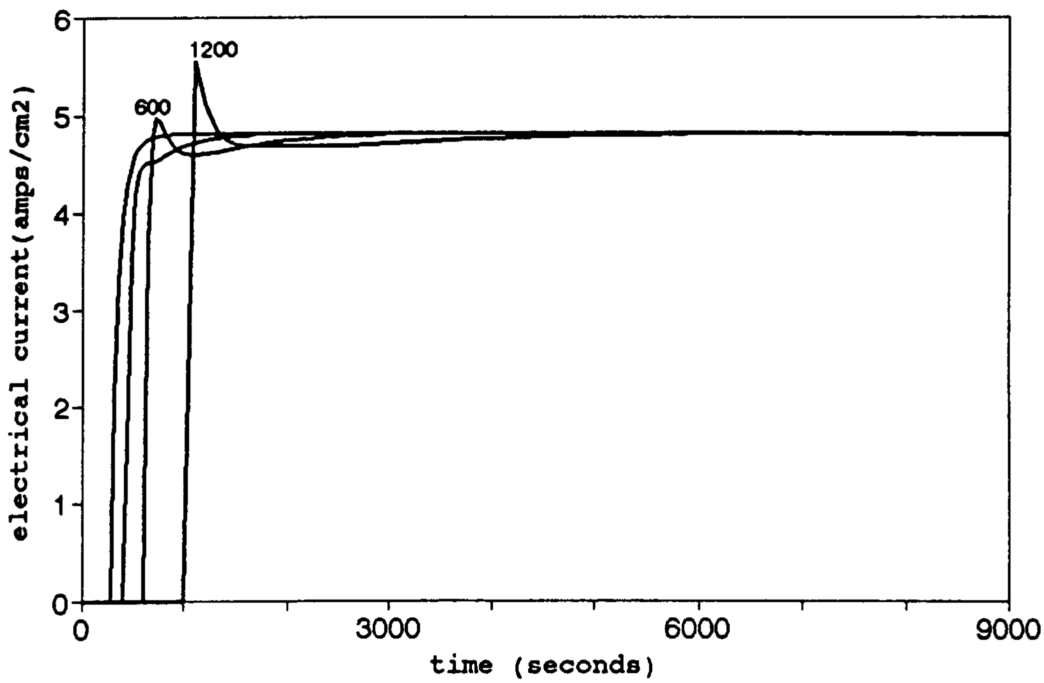

Figure 4.10 Electrical current profile for reactor start-up.

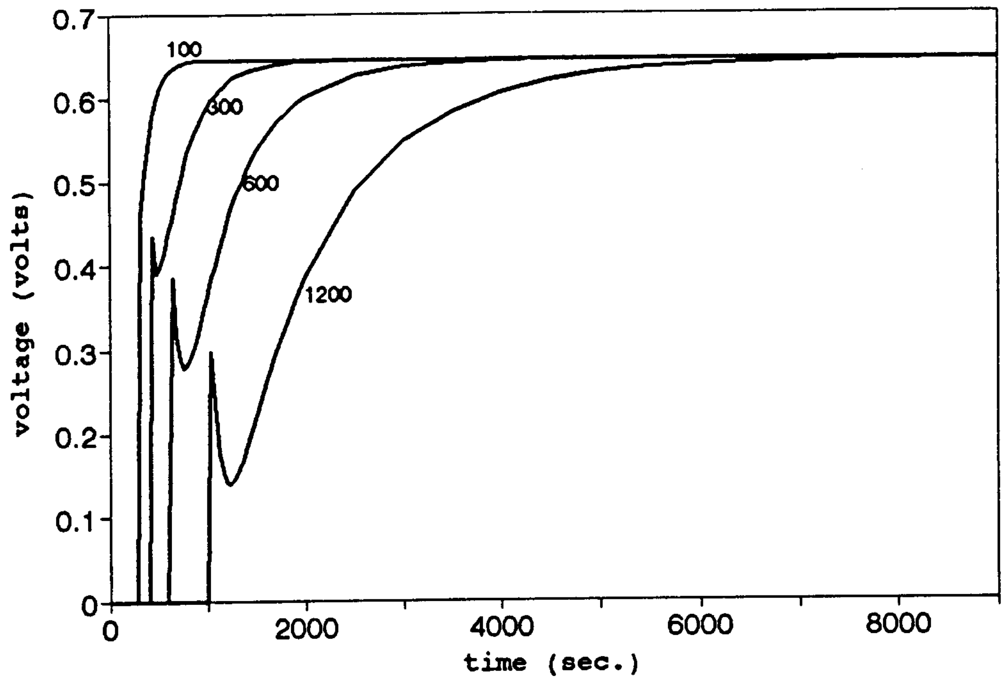

Figure 4.11 Electrical voltage profile for reactor start-up.

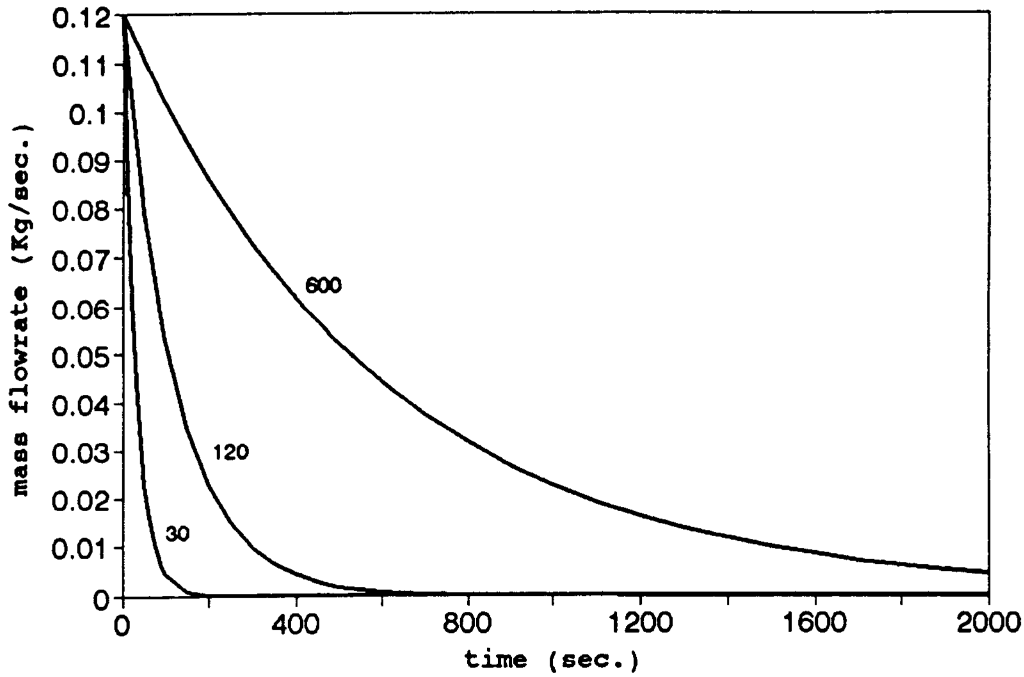

Figure 4.12 Mass flow rate for different decreasing exponential coefficients in LOF accident.

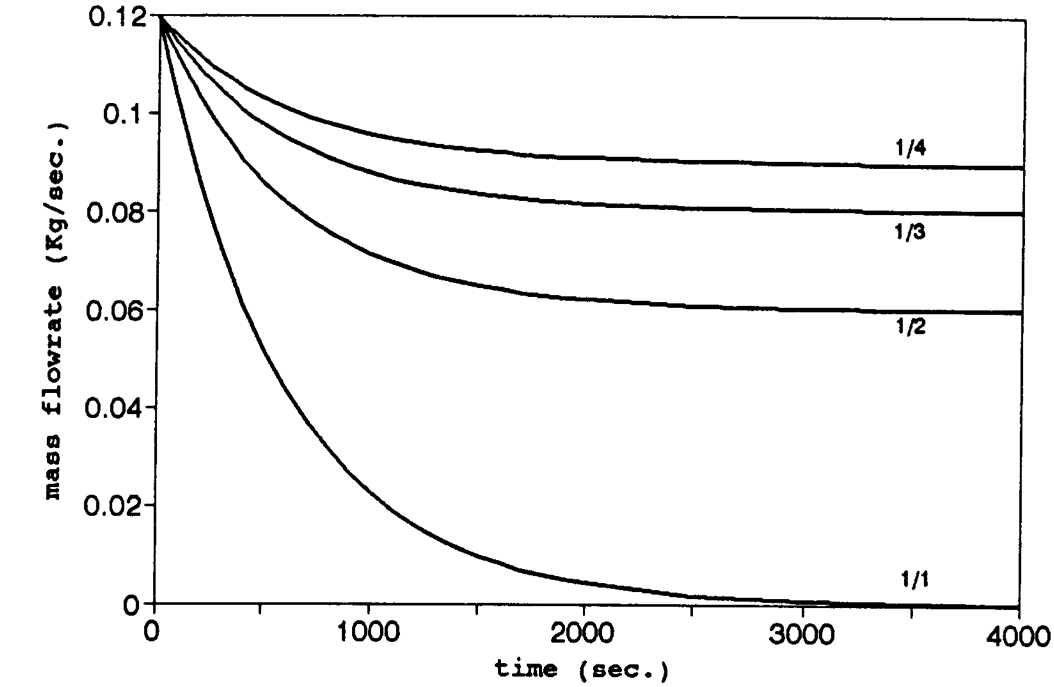

Figure 4.13 Mass flow rate distribution for different types of pump failures in LOF accident.

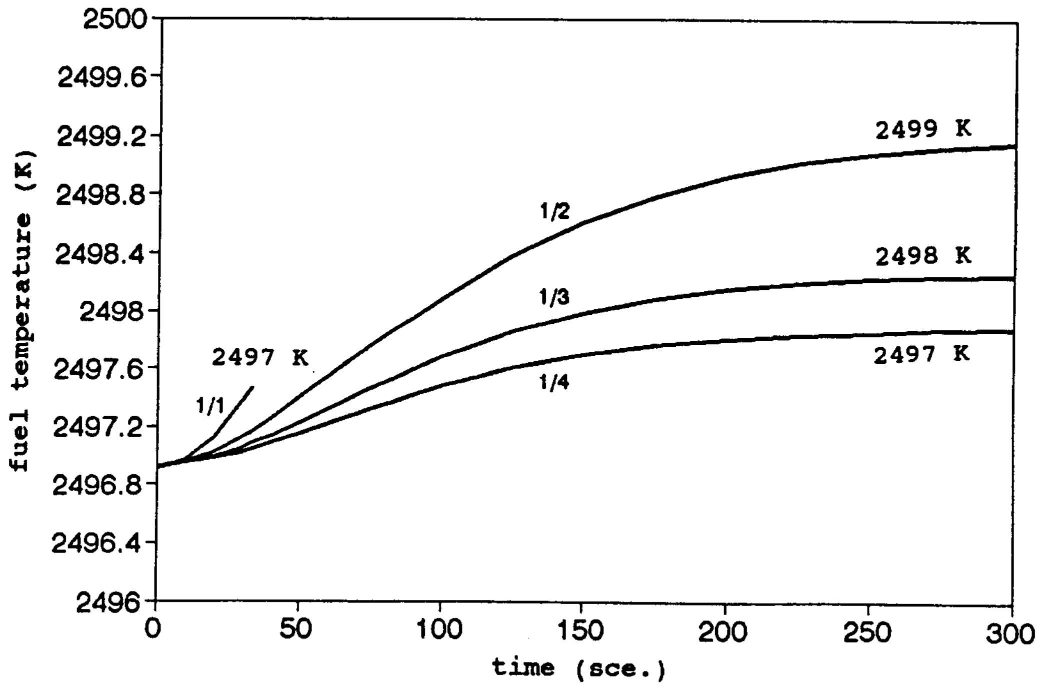

Figure 4.14 Fuel temperature profile for different types of pump failures at (τ = 30 seconds).

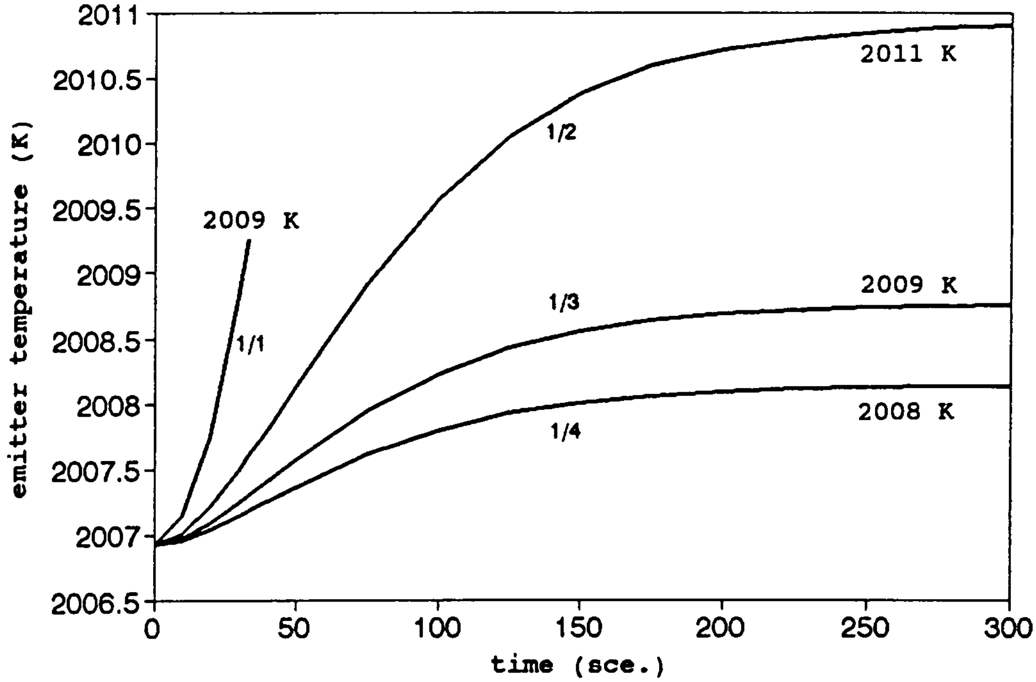

Figure 4.15 Emitter temperature profile for different types of pump failures at (τ = 30 seconds).

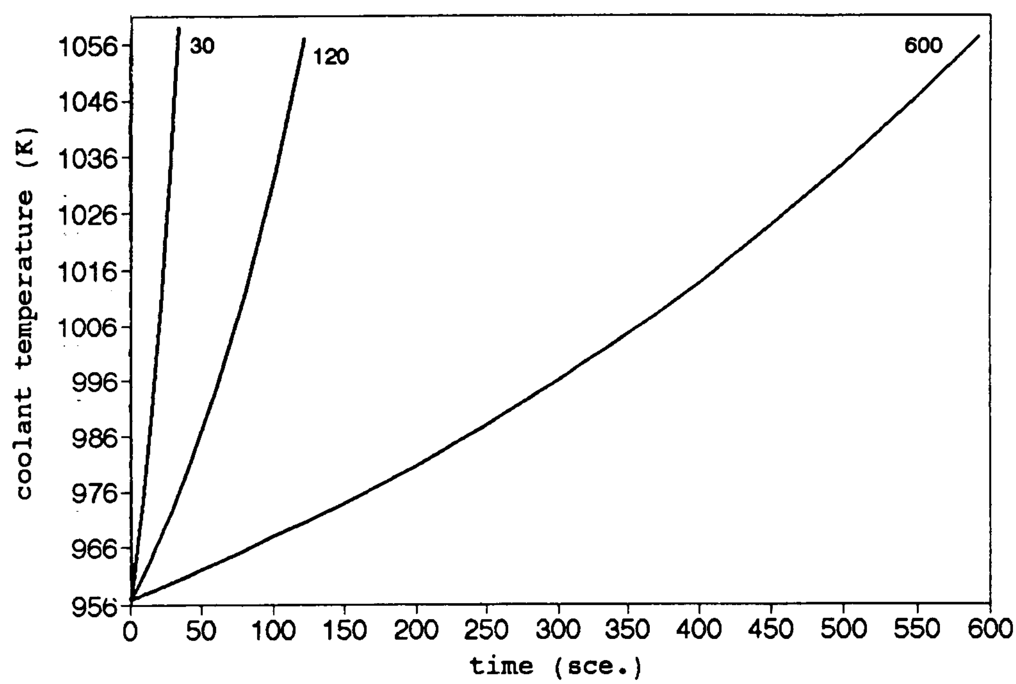

Figure 4.16 Coolant temperature profile for 1/1 pump failure at different decreasing exponential coeficients.

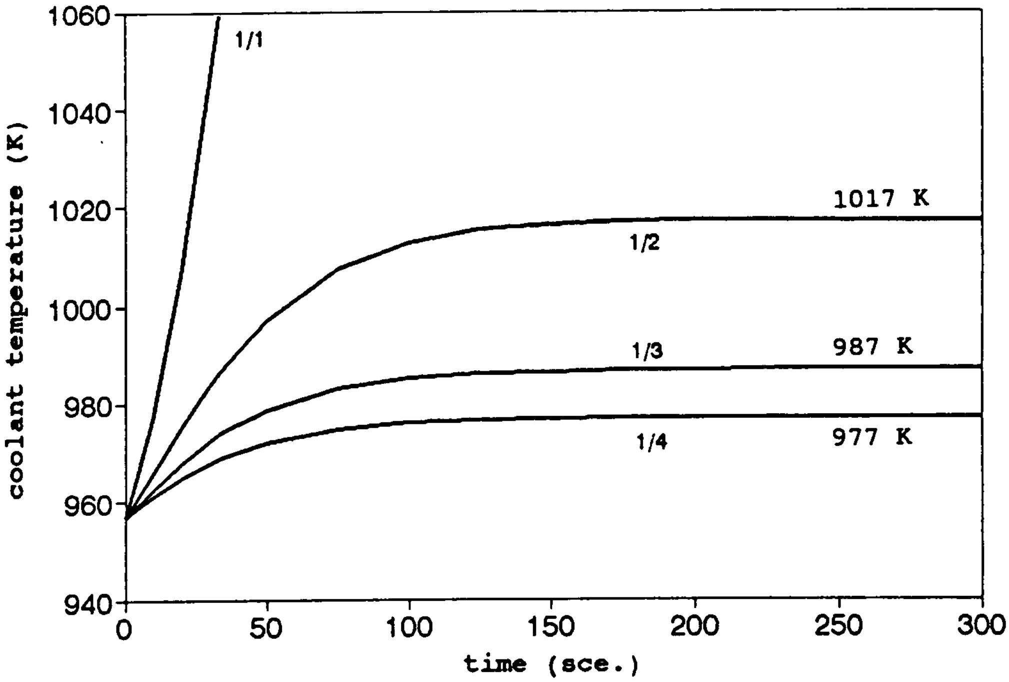

Figure 4.17 Coolant temperature profile for different types of pump failures at (τ = 30 seconds).

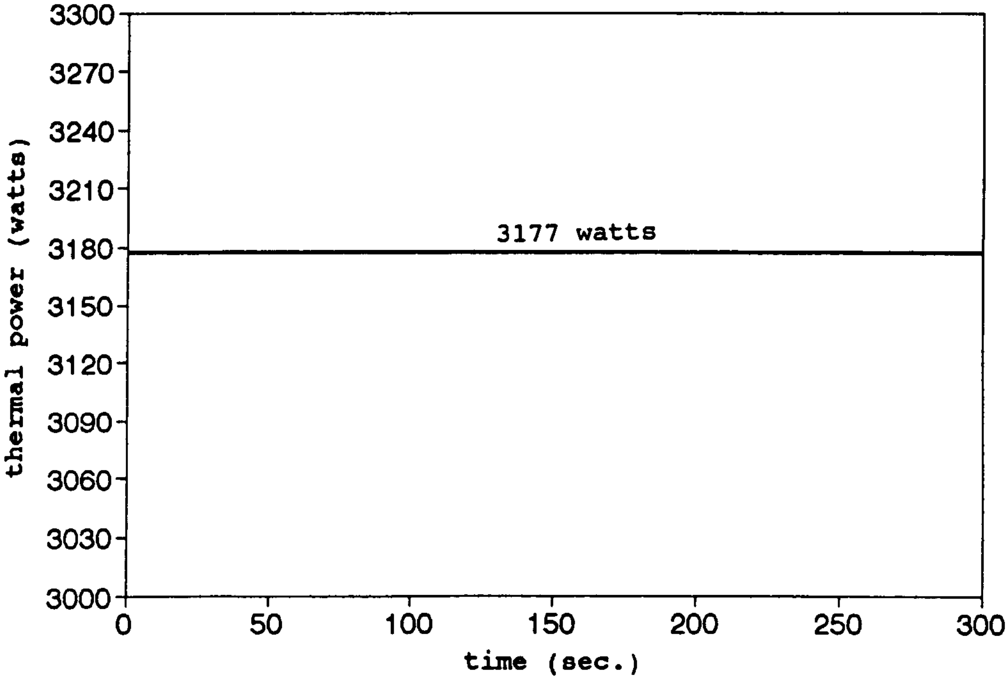

Figure 4.18 Thermal power distribution for LOF.

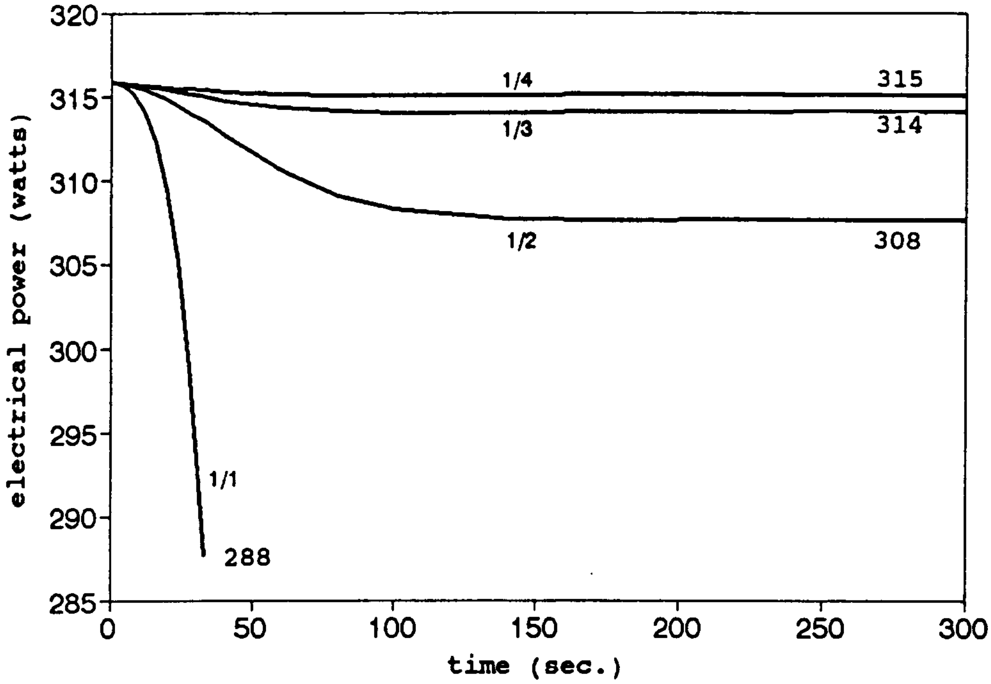

Figure 4.19 Electrical power profile for LOF accident at (τ = 30 seconds).

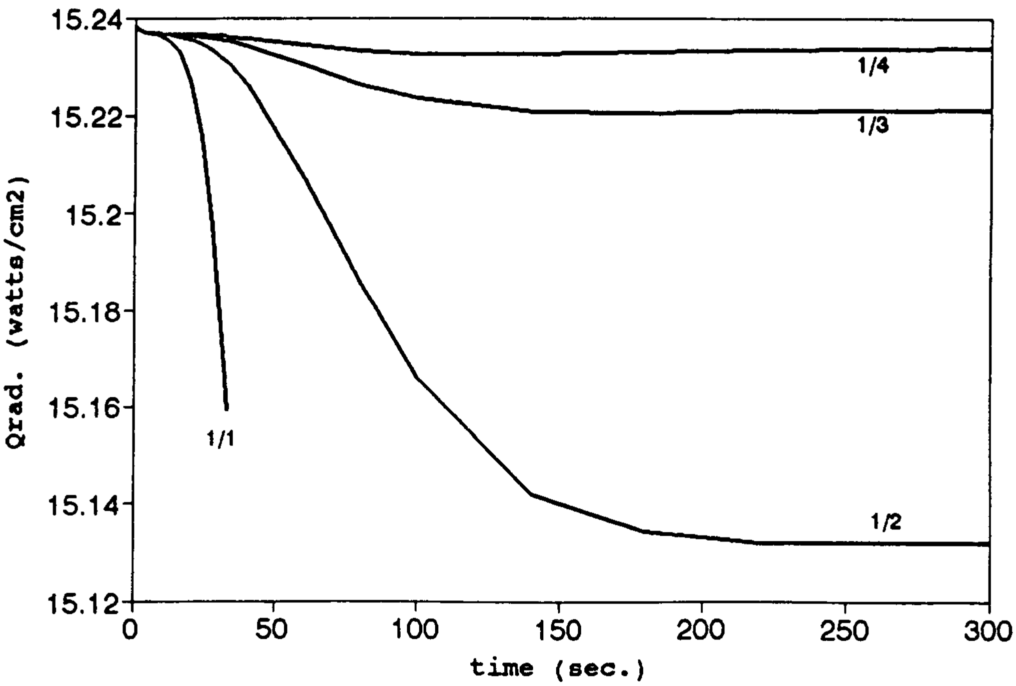

Figure 4.20 Radiation heat flux distribution for LOF accident at (τ = 30 seconds).

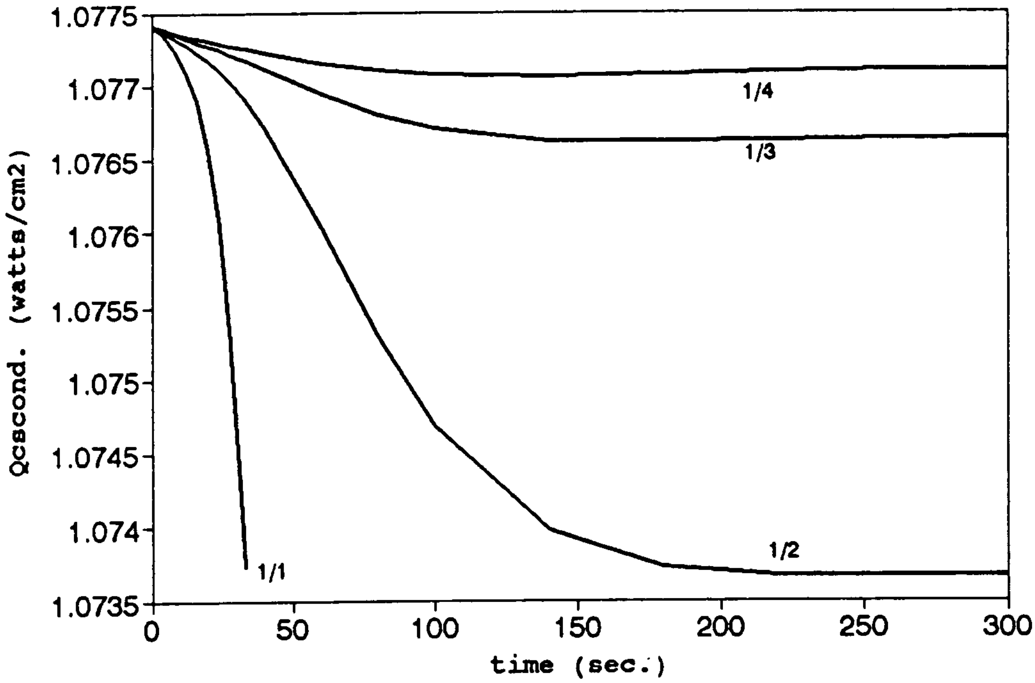

Figure 4.21 Conductive heat flux distribution of cesium for LOF accident at (τ = 30 seconds).

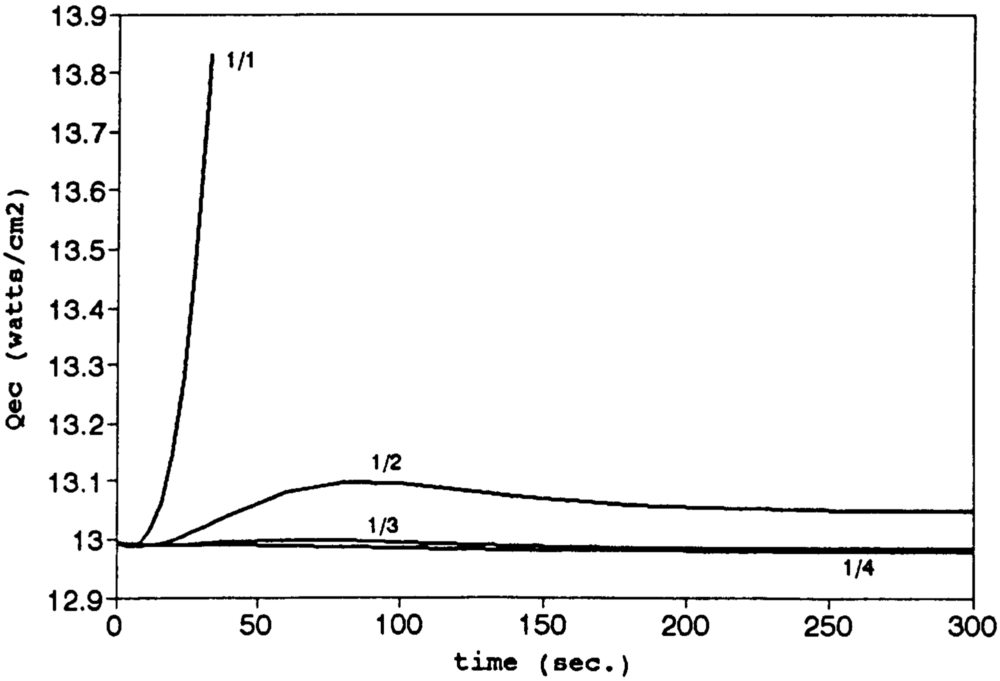

Figure 4.22 Heat flux distribution of emitter electron cooling for LOF accident at (τ = 30 seconds).

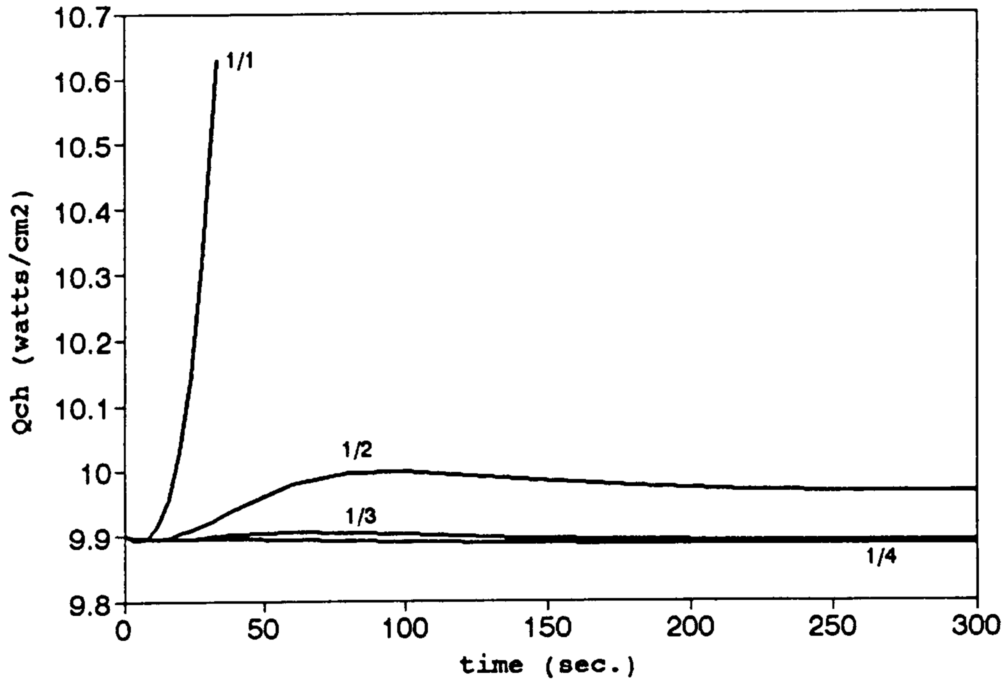

Figure 4.23 Heat flux distribution of collector electron heating for LOF accident at (τ = 30 seconds).

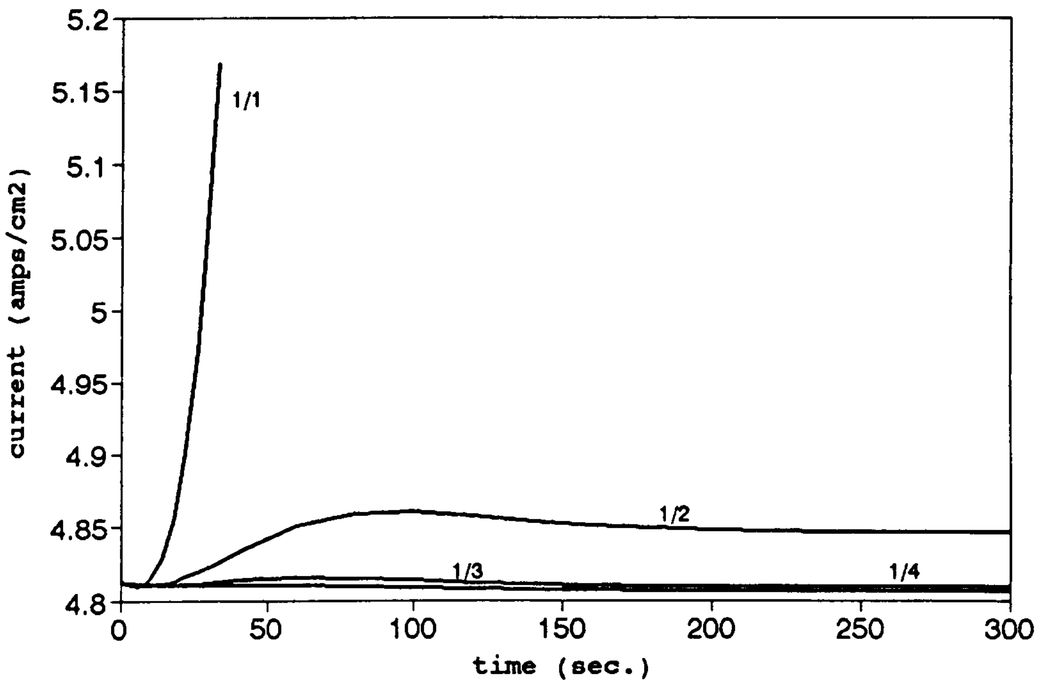

Figure 4.24 Electrical current profile for LOF accident at (τ = 30 seconds).

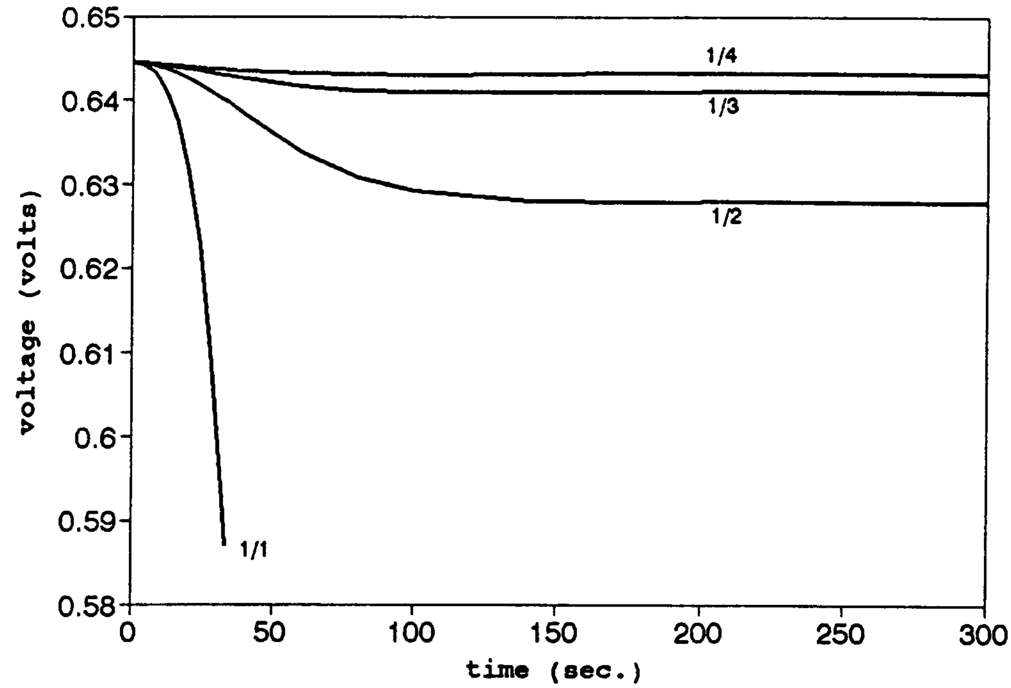

Figure 4.25 Electrical voltage profile for LOF accident at (τ = 30 seconds).

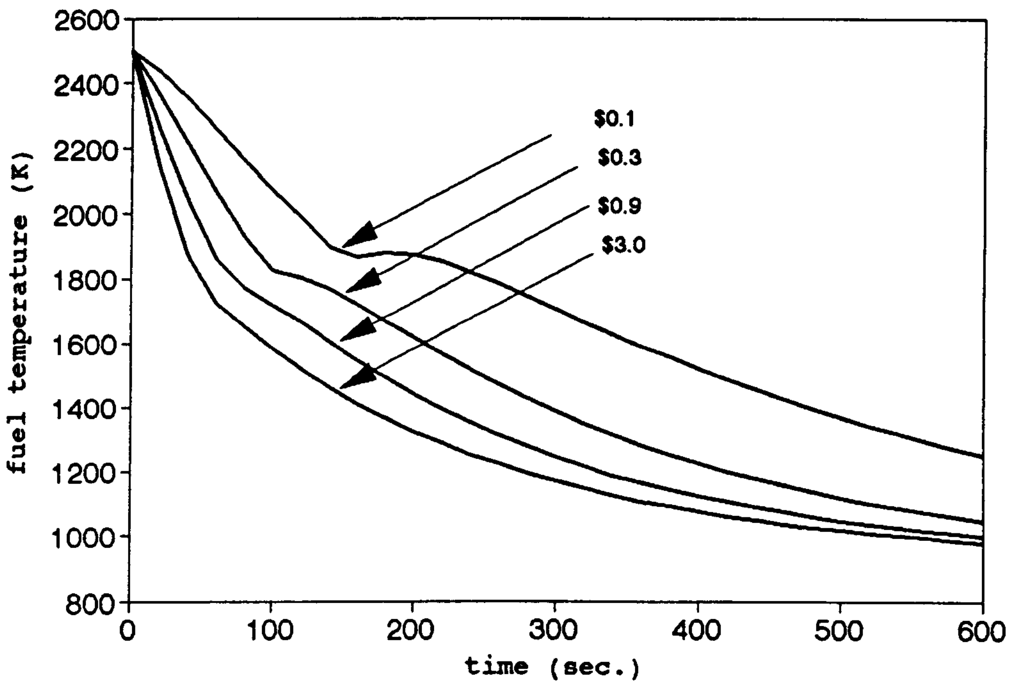

Figure 4.26 Fuel temperature profile for reactor shut down at different negative reactivity insertions.

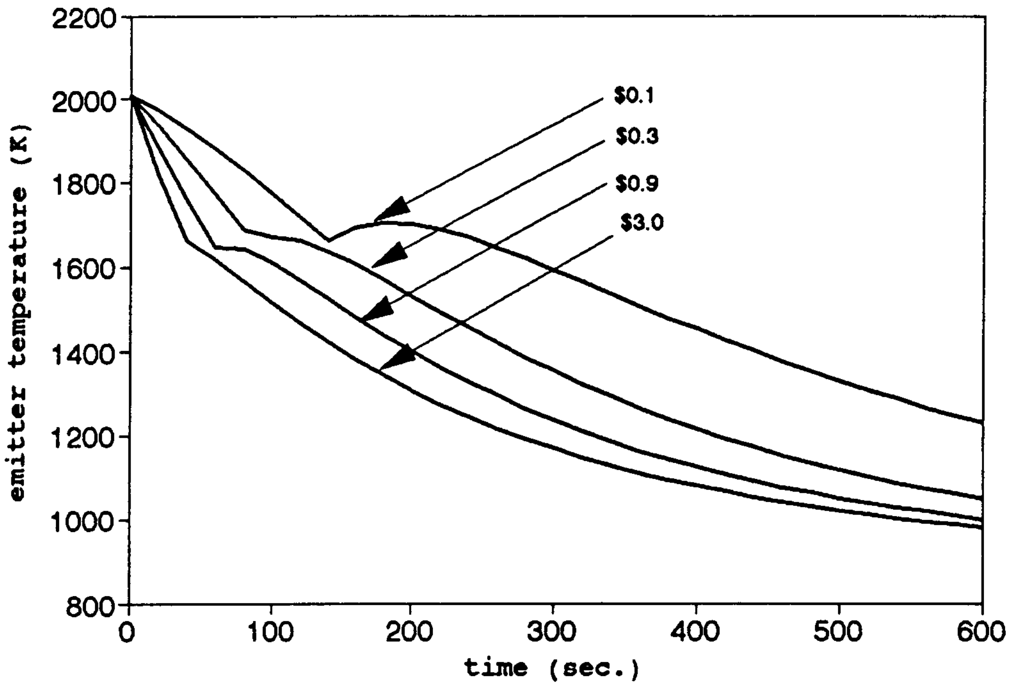

Figure 4.27 Emitter temperature profile for reactor shut down at different negative reactivity insertions.

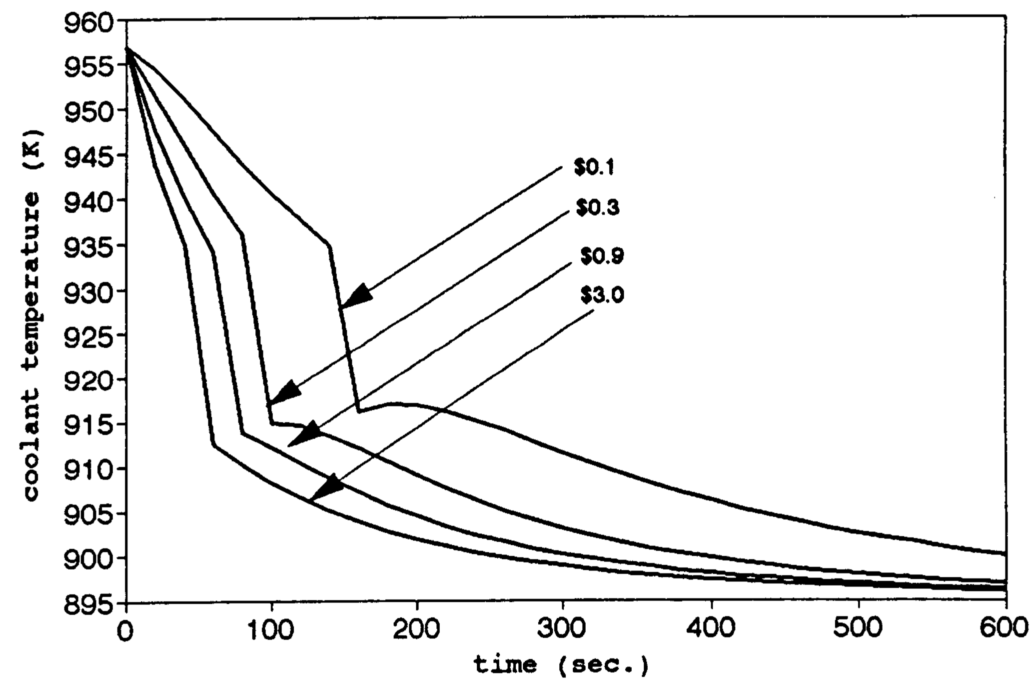

Figure 4.28 Coolant temperature profile for reactor shut down at different negative reactivity insertions.

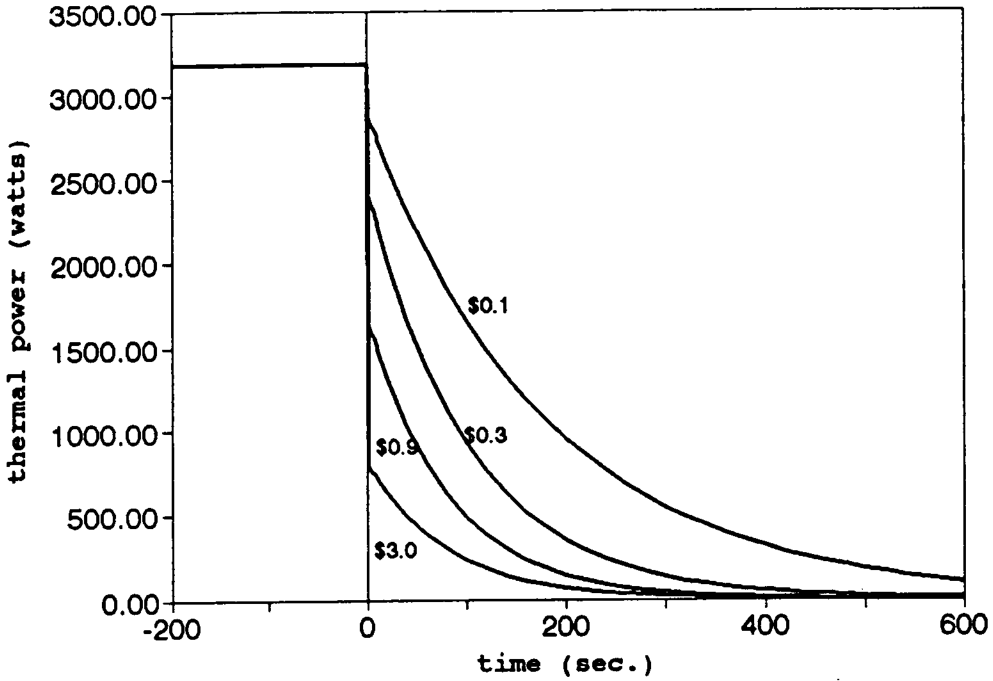

Figure 4.29 Thermal power profile for reactor shut down at different negative reactivity insertions.

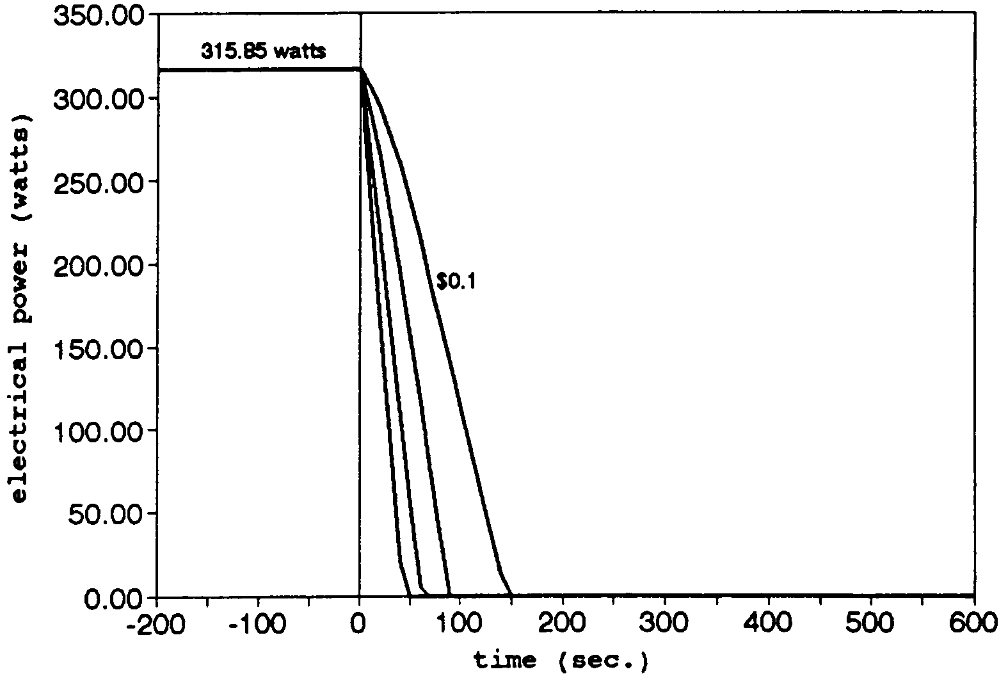

Figure 4.30 Electrical power profile for reactor shut down at different negative reactivity insertions.

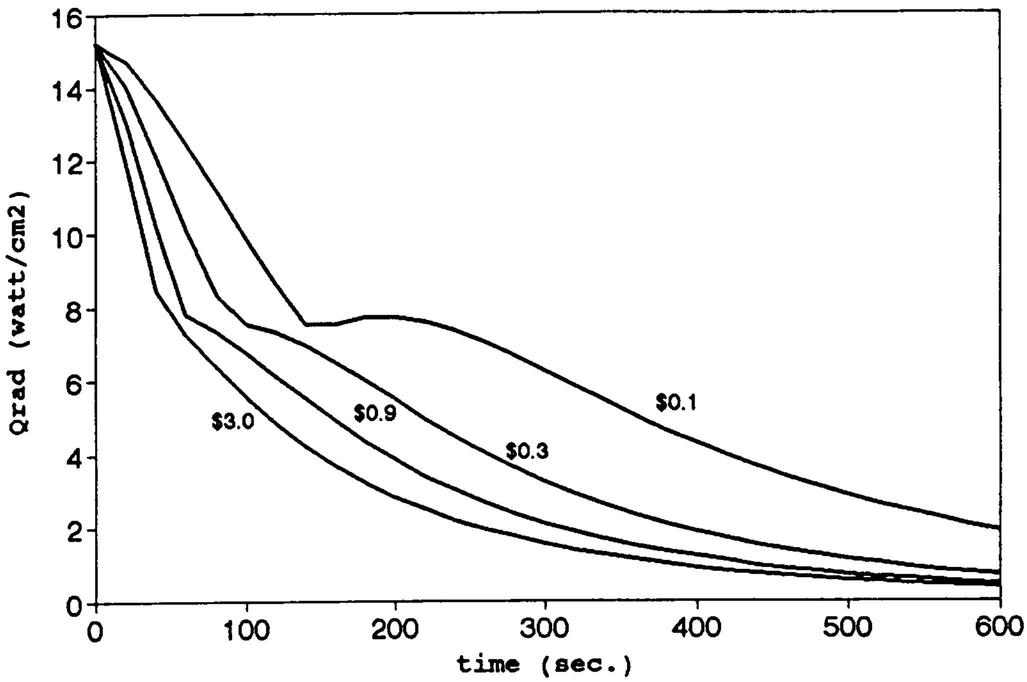

Figure 4.31 Radiation heat flux profile for reactor shut down at different negative reactivity insertions.

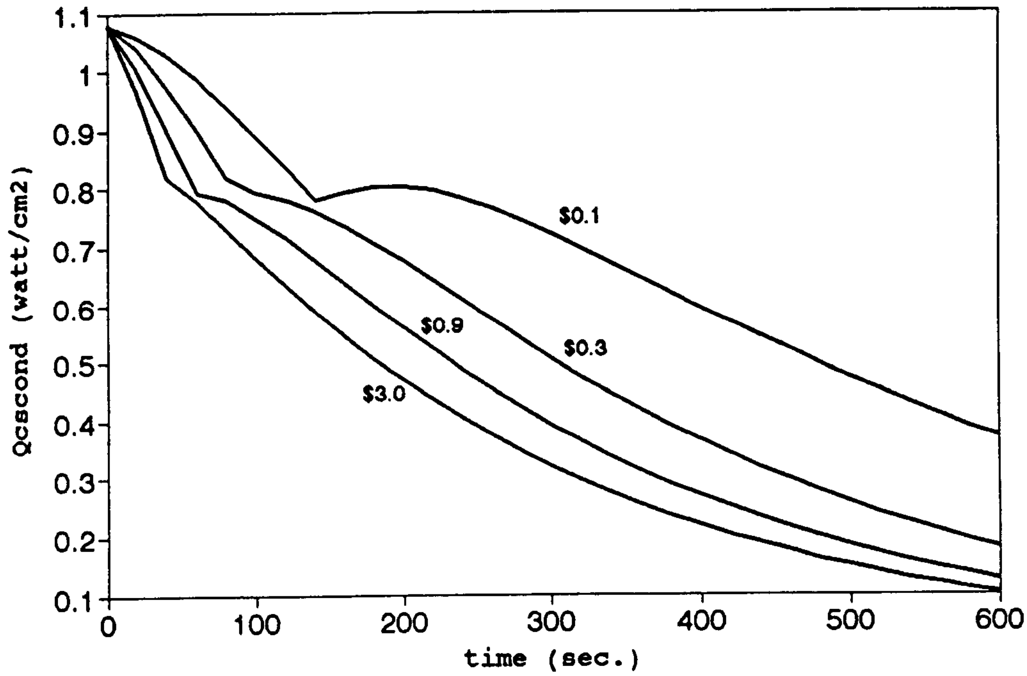

Figure 4.32 Conductive heat flux profile of cesium for reactor shut down at different negative reactivity insertions.

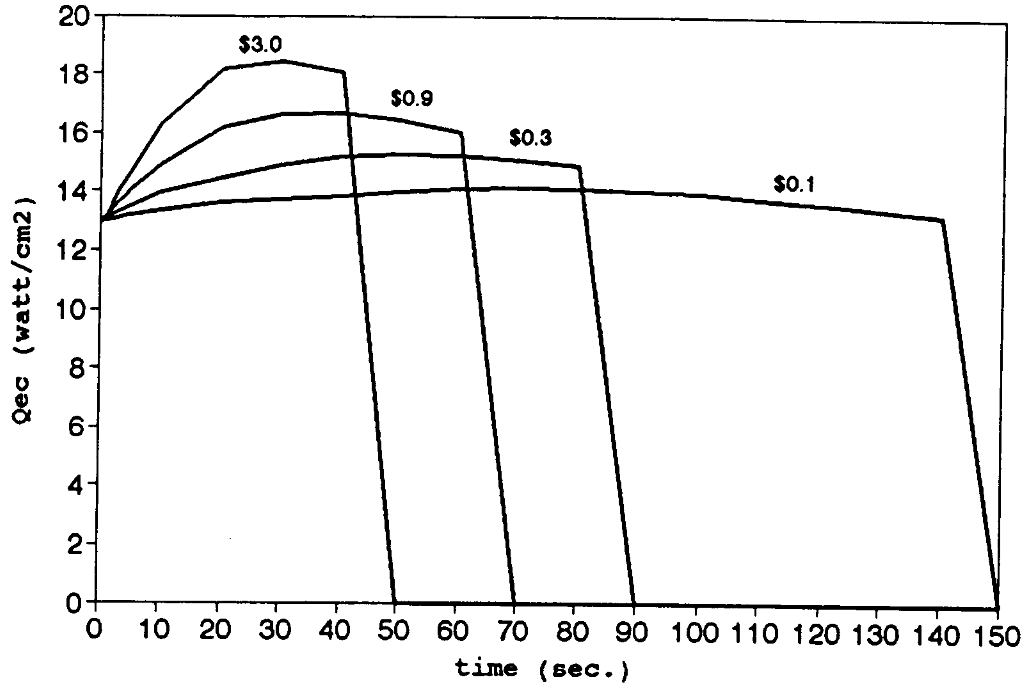

Figure 4.33 Heat flux profile of emitter electron cooling for reactor shut down at different negative reactivity insertions.

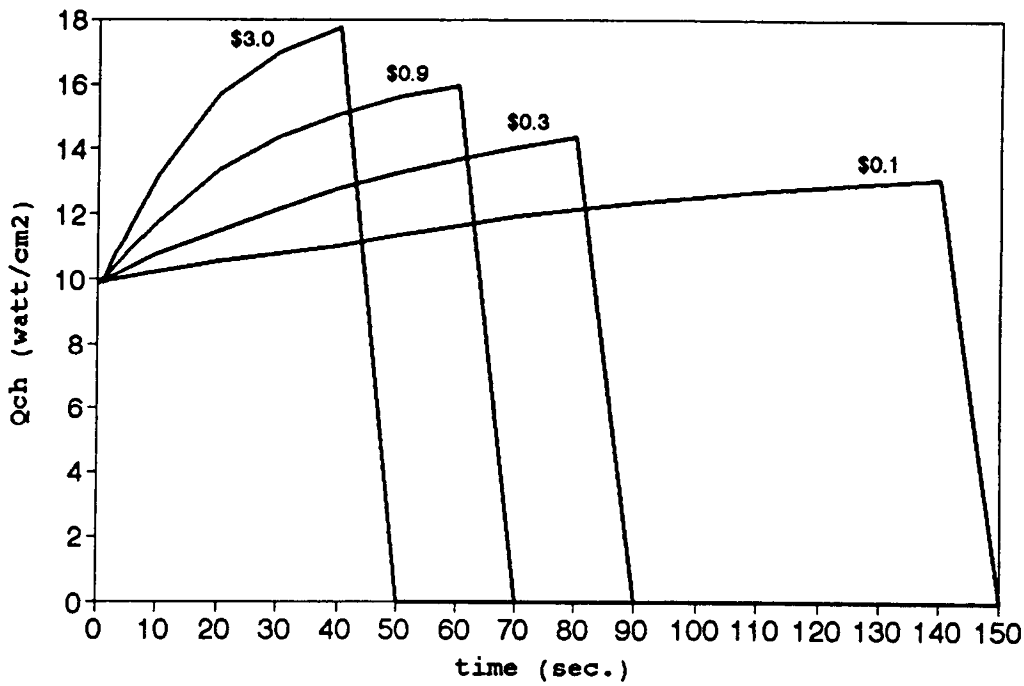

Figure 4.34 Heat flux profile of collector electron heating for reactor shut down at different negative reactivity insertions.

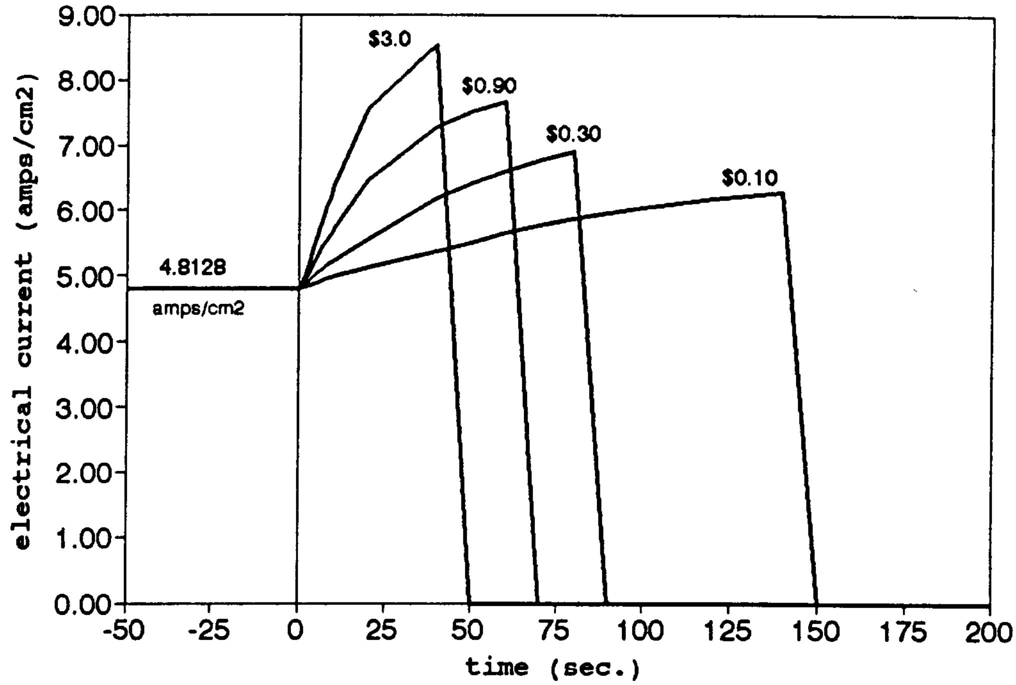

Figure 4.35 Electrical Current profile for reactor shut down at different negative reactivity insertions.

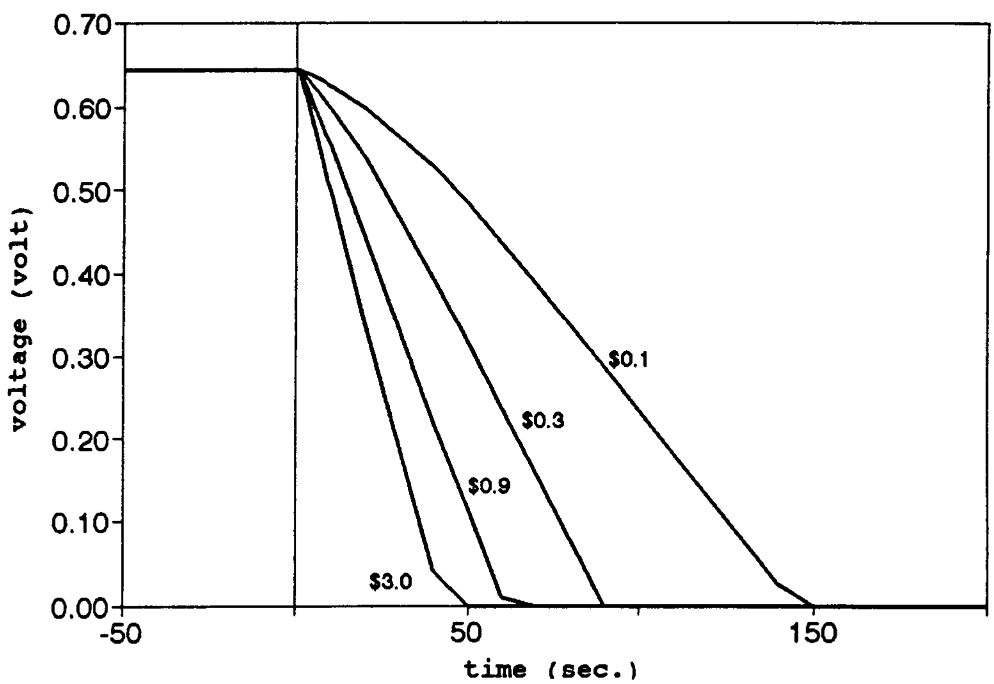

Figure 4.36 Electrical voltage profile for reactor shut down at different negative reactivity insertions.

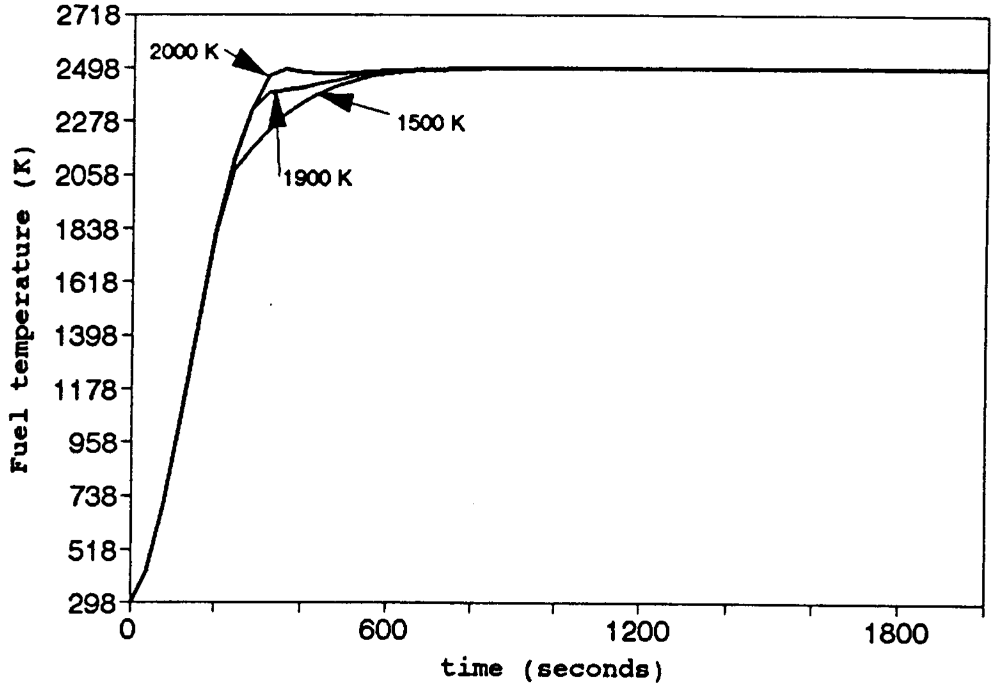

Figure 4.37 Fuel temperature profile for different electron cooling temperatures (Start-up).

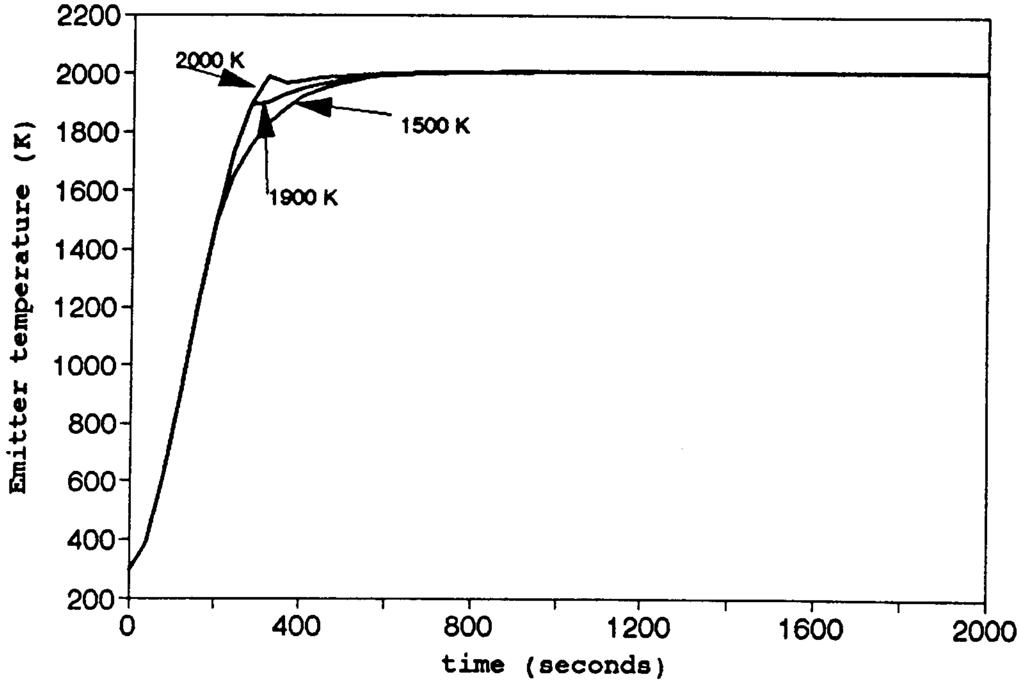

Figure 4.38 Emitter temperature profile for different electron cooling temperatures (Start-up).

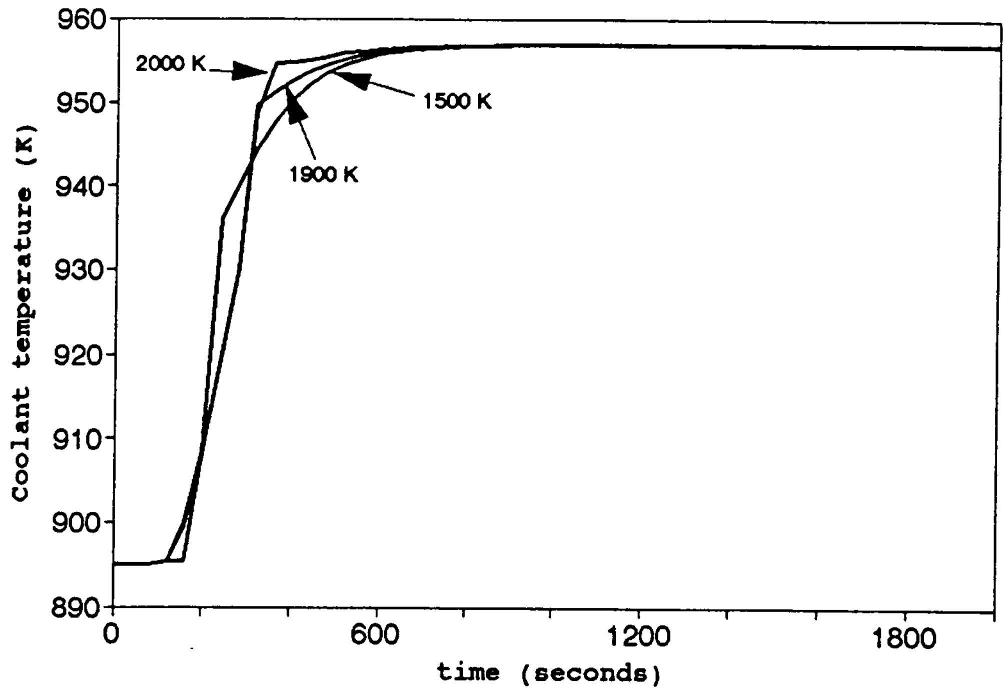

Figure 4.39 Coolant temperature profile for different electron cooling temperatures (Start-up).

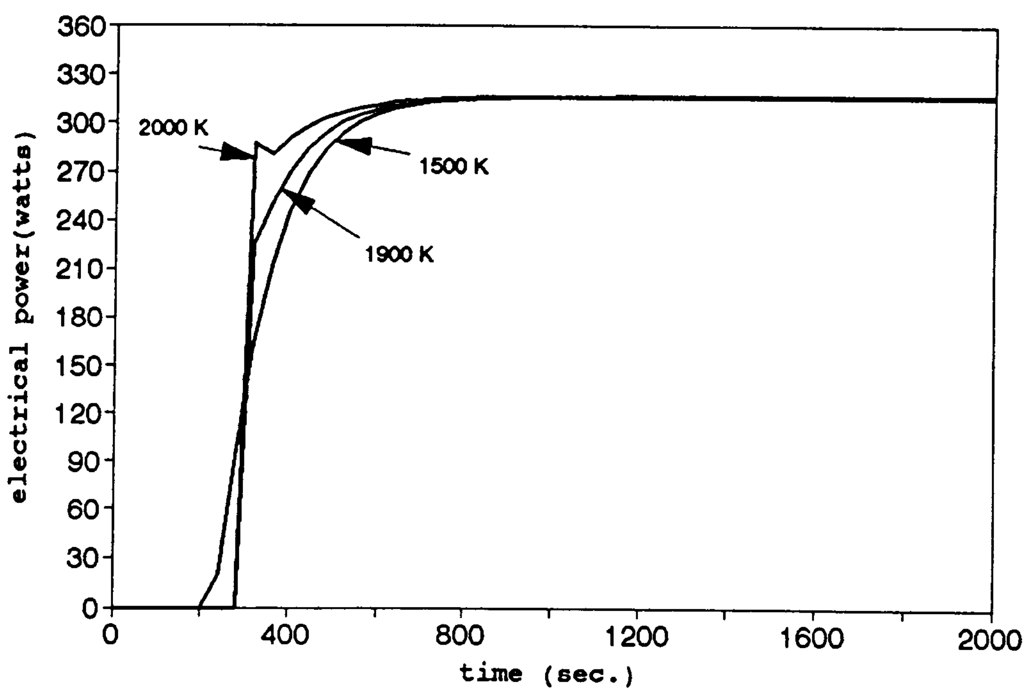

Figure 4.40 Electrical power profile for different electron cooling temperatures (Start-up).

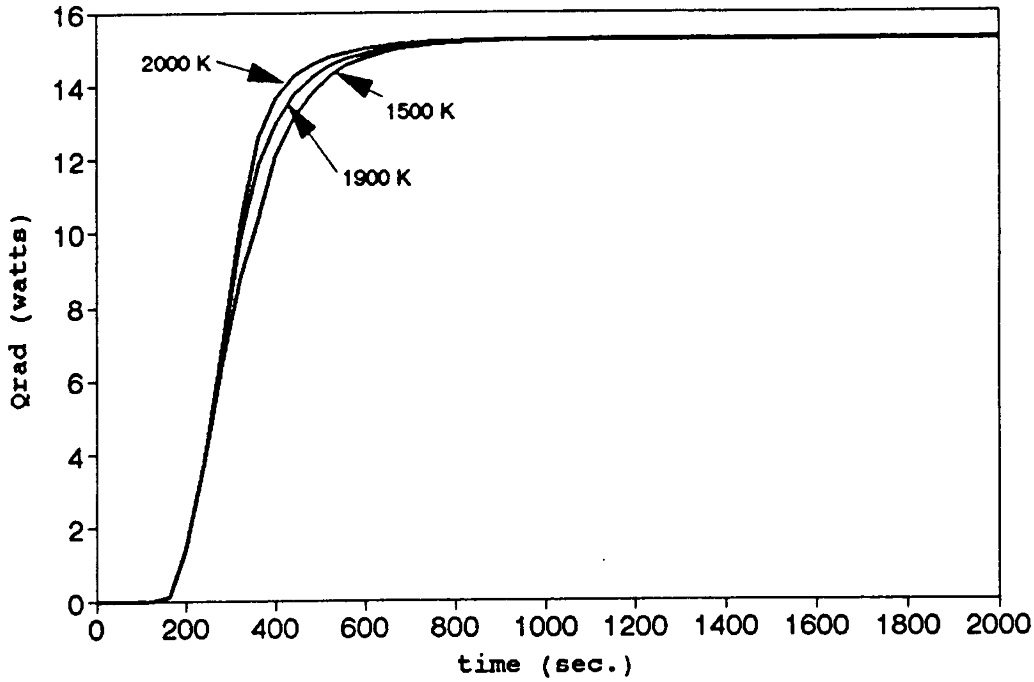

Figure 4.41 Radiation heat flux profile for different electron cooling temperatures (Start-up).

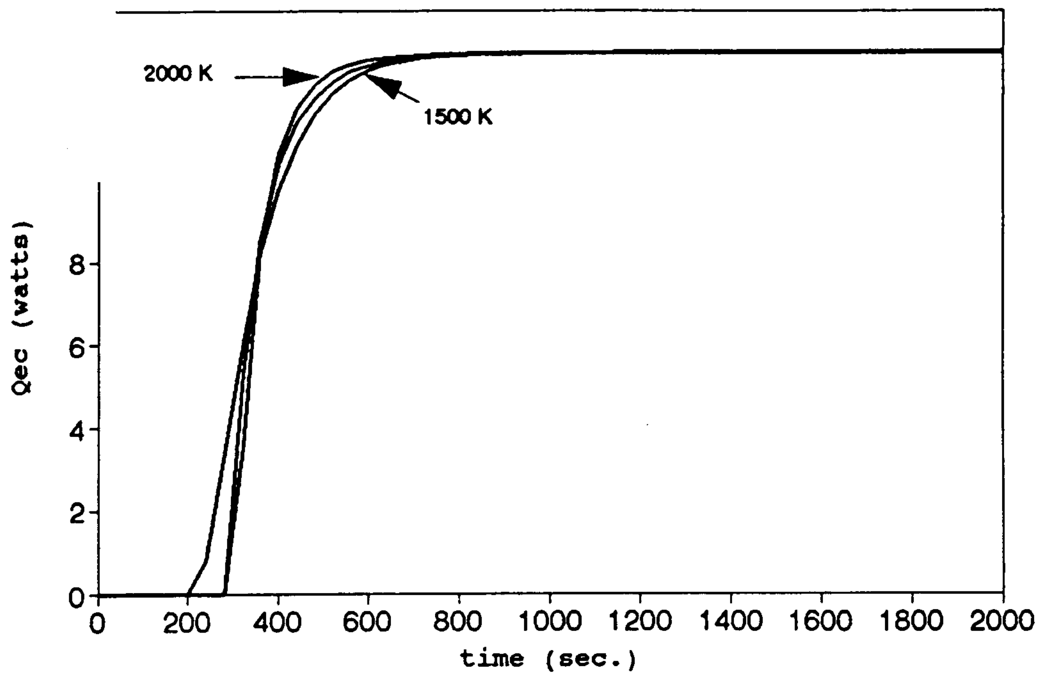

Figure 4.42 Heat flux profile of emitter electron cooling for different electron cooling temperatures (Start-up).

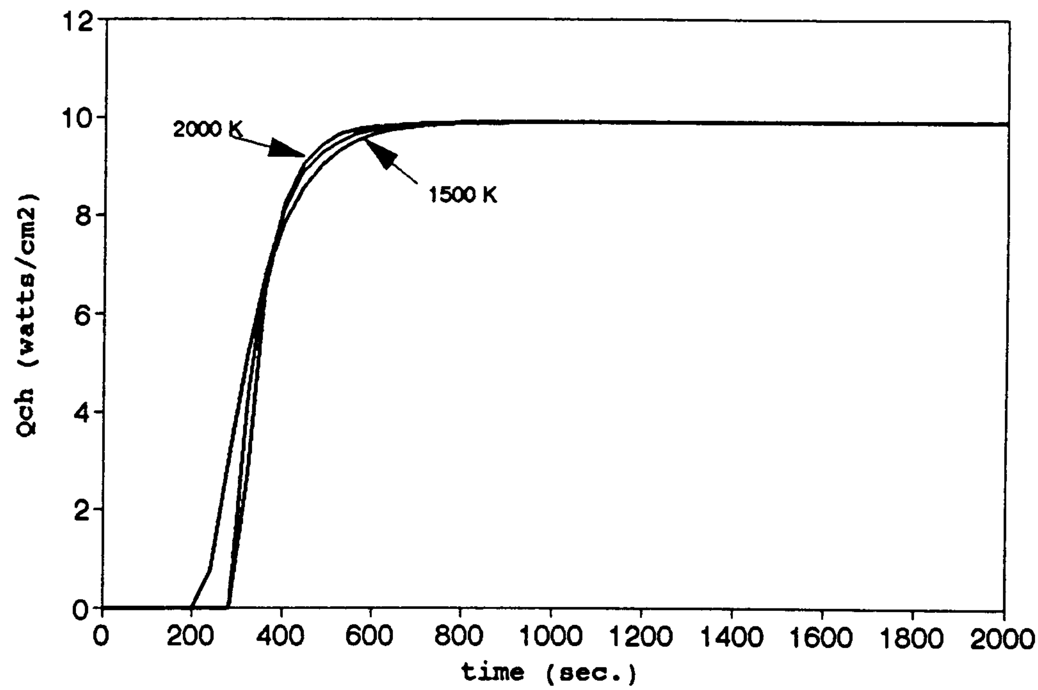

Figure 4.43 Heat flux profile of collector electron heating for different electron cooling temperatures (Start-up).

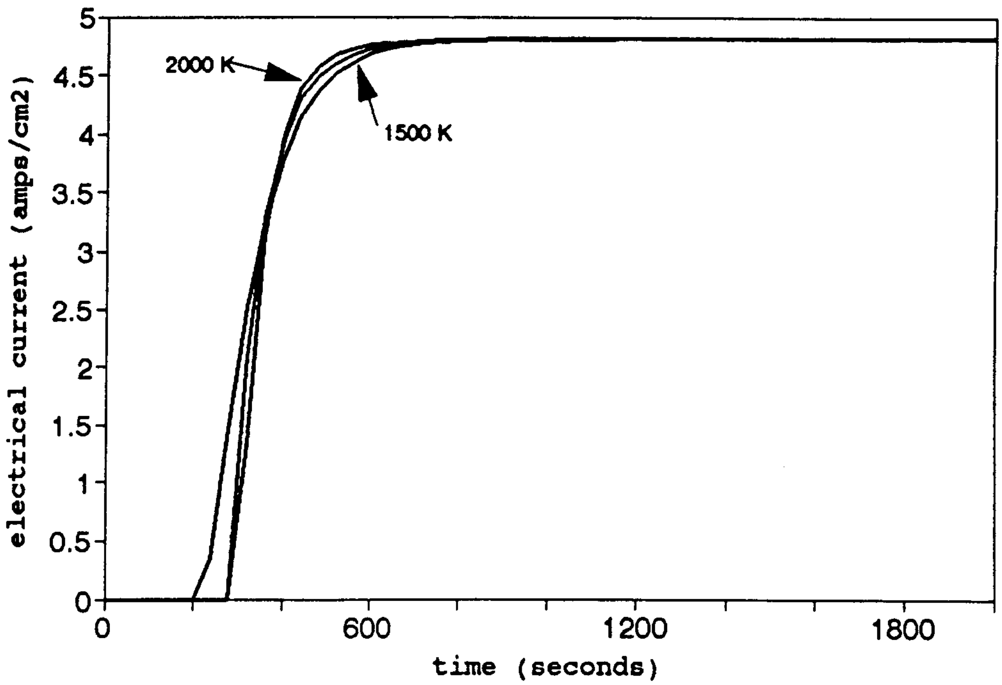

Figure 4.44 Electrical current profile for different electron cooling temperatures (Start-up).

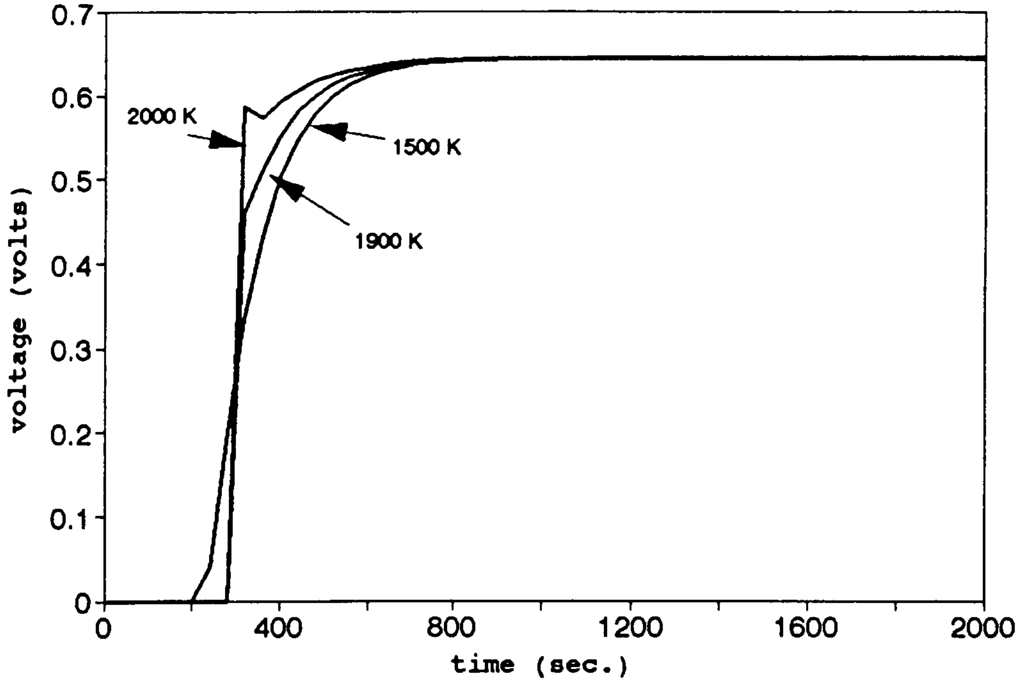

Figure 4.45 Electrical voltage profile for different electron cooling temperatures (Start-up).

References:

1. A.C. Klein, H.H. Lee, B.R. Lewis, R.A. Pawlowski and Shahab Abdul-Hamid, "Advanced Single Cell Thermionic Reactor System Design Studies", Oregon State University, OSU-NE-9209, Corvallis, OR Sept. 1992.

2. J. M. Houston, Theoretical Efficiency of the Thermionic Energy Converter, J. Appl. Phys. **30**, pp. 481-487, 1959.

3. G. N. Hatsopoulos and E. P. Gyftopoulos, Thermionic Energy Conversion (MIT, Camridge, MA, 1979), Vols. **I** and **II**.

4. M.M. El-Wakil, Nuclear Energy Conversion, The American Nuclear Society, Illinois, 1978.

5. N.S. Rasor,"Thermionic Energy Conversion," in Applied Atomic Collision Processes, Massey, H.S.W., McDaniel, E.W., and Bederson,B., eds. Vol. **5**, pp. 169-200, Academic Press, New York, 1982.

6. V.V. Skorlygin, T.O. Skorlygina, Y.A. Nechaev, and M.Y. Yermoshin, Kurchatov Institute, Moscow, Russia, Alexey N. Luppov, Central Design Bureau of Machine Building, St. Petersburg, Russia, and Norm Gunther, Space Power, Inc., San Jose, CA, "Simulation Behavior of TOPAZ-2 and Relation to the Operation of TSET System," Proceedings of 10^{th} Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1495-1498, Albuquerque, NM, Jan. 1993.

7. J. Weisman, Elements of Nuclear Reactor Design, Kreiger Publishing Company, 1983.

8. A. Ralston, A first Course in Numerical Analysis, 2^{nd} Edition, McGraw Hill Book Company, 1978.

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

Chapter 5

Conclusions and Recommendations

Conclusions:

A computer model has been developed to simulate the transient temperature distribution of the thermionic fuel element (TFE) for nuclear space reactors. The transient computer code TFETC is based on TFEHX, a steady state code for TFE's.

Presently built into the transient code is the ability to handle the following scenarios: a) start up for user prescribed power rise coefficients; b) loss of flow for different mass loss coefficients and pump failures; and c) shut down for different negative reactivity insertions and total delayed neutron fractions for the UO_2 fuel. The user has the ability to control the helium heating phase through a proper choice of the emitter temperature at which helium heating ends, TSTOP, and likewise for the electron cooling phase through TSTART, the emitter temperature at the onset of electron cooling.

Another feature, in the TFETC code, of great importance is the ability to use more than one solver to solve the linear systems of algebraic equations.

The TFETC has been tested in the case of accidents such as loss of flow and has shown good stability and integrity at the worst case in which the pump failures are complete. It allows some time, depending on the mass loss coefficient which is part of the input file, to scram the reactor safely. The TFETC model shows that the fuel temperature in a complete LOFA does reach 2497 °K, after the coolant attains its boiling point, in 34 seconds. The fuel temperature may exceed the fuel melting point after a period of time if no apropriate action to trip the reactor is taken. For 1/2, 1/3, and 1/4 pump failures, the fuel temperatures attain steady state values. They never exceed the fuel melting point even if the reactor is not shut down. Also, for the coolant temperature, the boiling point is exceeded in a complete LOFA while for 1/2, 1/3, and 1/4 pump failures, the maximum attainable coolant temperatures are: 1017, 987, 977 °K's respectively.

Recommendations:

The following recommendations for further studies are proposed:

1. The only gap considered in the thermionic fuel element is the emitter/collector gap, otherwise all the TFE regions are stacked next to each other. For further development of the TFETC code, it is recommended to induce some gaps between fuel and emitter or collector and insulator in order to increase the safety margins in the event of an accident.

2. The Gauss elimination methods are not the only technique to solve the unsteady state conduction equations but rather there are some methods that give accurate and reliable results. The round off error in the Gaussian elimination is large. Hence, it will be useful to try the code with other linear system solvers.

3. For more accurate calculations, it is recommended to choose very small time increments. The smaller the Δt, the smaller the truncation error of the implicit method.

4. For a large transient time as in the case of a TFE pin, it is not recommended to use the explicit method due to its deficiency that allows stability for only very small time steps and hence requires an immense period of computer time to study the transient. Other discretization methods can be studied which allow better control of the accuracy and faster execution times.

5. It would be desirable to improve the TFE model which is currently implemented in CYLCON6 to handle transient effects.

6. It is important to test the code results against experiments such as the Thermionic System Evaluation Test (TSET) of the TOPAZ-II system. Also, it is necessary to compare results with single cell test start experiments.

7. To analyze additional accident scenarios such as LOFA with reactor start up and reactor shut down.

8. The heat transfer mechanism is completely

different when the coolant reaches its boiling point, so it is desirable to develop a methodology to be able to handle cases in which boiling occurs in the NaK coolant.

9. It is recommended to decrease Δz, to be less than 1, to reduce the error arising from the implicit method. It is also advisable to replace the subroutine of the Gauss elimination with another subroutine which takes care of the bound error associated with the solution. For example, DGESVX subroutine.

BIBLIOGRAPHY

Abdul-Hamid, S., Klein, A.C., Lee, H.H., Lewis, B.R. and Pawlowski, R.A.,"Advanced Single Cell Thermionic Reactor System Design Studies", Oregon State University, OSU-NE-9209, Corvallis, OR Sept. 1992.

Angrist, S.W., Direct Energy Conversion, Allyn and Bacon, Inc., MT, 3rd Edition, 1976.

Auer, P.L., and Hurwitz, H., Space Charge Neutralization by Positive Ions in Diodes, J. Appl. Phys.**30**, pp. 161-165, 1959.

Beggs, J.E., and Webster, H.F., High Vacuum Thermionic Energy Converter, Bull. Am. Phys. Soc. **3**, pp. 266, 1958.

Bloch, E., Thermionic Phenomena, Methuen & co. ltd., London, 1927.

Borland, Inc., "QUATTRO-PRO 3.0 Superior Spreadsheet Power", 1991.

Buden, D., Nuclear Reactors For Space Power, Aerospace America, pp. 66-69, 1985.

Carabateas, E.N., Hatsopoulos, G.N., and Pezaris, S.D.,"Interpretation of Experimental Characteristics of Cesium Thermionic Converters", J. Appl. Phys. **32**, pp. 352, 1961.

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

Cayless, M.A.,"Thermionic Generation of Electricity", Brit. J. Appl. Phys. **12**, pp. 433-442, 1961.

Croft, D.R., and Lilley, D.G., Heat Transfer Calculations Using Finite Difference Equations, Applied Science Publishers Ltd., London 1967.

Deane, N.A., Marcille, T.F., Newkirk, D.W., Stewart, S.L."SP-100 Reactor and Shield Design Update,"Proceedings of 9th Symposium on Space Nuclear Power Systems, CONF-920104, Albuquerque, NM, Jan. 1992.

Dewitt, D.B., and Incropera, F.P., Introduction to Heat Transfer, 1st Edition, John Wiley & Sons, Inc., 1985.

Dieckamp, H.M., and Wilson, R.F.,"What Happened to SNAP10A," Astronautics and Aeronautics, pp. 60-65, Oct. 1965.

Dobrestov, L.N.,"Thermoelectronic Converters of Thermal Energy into Electric Energy", Soviet Phys.Tech. Phys.(English Transl.), **5**, pp. 343-368, 1960.

Duderstadt, J.J., and Hamilton, L.J., Nuclear Reactor Analysis, 1st Edition, John Wiley & Sons, NY, 1976.

Dugan, A.F.,"Contribution of Anode Emission to Space Charge in Thermionic Power Converters", J. Appl. Phys. **31**, pp. 1397, 1960.

El-Wakil, M.M., Nuclear Energy Conversion, 2nd Edition, The American Nuclear Society Press., Illinois, 1978.

El-Wakil, M.M., Nuclear Heat Transport, 3rd Edition, The American Nuclear Society Press., 1981.

Faust, O.,"Sodium-NaK Engineering Handbook", vol.**1**, pp. 52-53, 1972.

Foley, T.M.,"Soviet Reveal Testing in Space of Thermionic Nuclear Reactor (TOPAZ)", Aviation Week & Space Technology, Vol. **130**, pp 30, Jan. 16, 1989.

Glasstone, S., and Sesonske, A., Nuclear Reactor Engineering, Van Nostrand Reinhold Company, NY, 1981.

Greek, K.J., Hatch, G.L., and McVey, J.B., Rasor Associates, Inc., and G.J. Parker, W.N.G. Hitchon, and J.E. Lawler, University of Wisconsin-Madison, "Comprehensive Time Dependent Semi-3D Modeling of Thermionic Converters In Core", NSR-53 / 92-1004, Sept. 30, 1992.

Greek, K.J., Klein, A.C., Lewis, B.R., and Pawlowski, R.A., "Advanced Thermionic Reactor System Design Code", Proceedings of 8th Symposium on Space Nuclear Power Systems, CONF-910116, Albuquerque, NM, 1991.

Grover, G.M., Pidd, R.W., Roehling, D.J., and Salmi, E.W., " Properties of a Thermoelectric Cell", J. Appl. Phys. **29**, pp. 1611-1612, 1958.

Grover, G.M., Ranken, W.A., and Salmi, E.W., "Experimental Investigations of the Cesium Plasma Cell", J. Appl. Phys. **31**, pp. 2140, 1960.

Gubbels, G.H.M., and Metselaar, R., A Thermionic Energy Converter with an Electrolytically Etched Tungsten Emitter, J. Appl. Phys. **68**, pp. 1883-1888, Aug. 1990.

Gyftopoulos, E.P., and Hatsopoulos, G.N., Thermionic Energy Conversion (MIT, Camridge, MA, 1979), Vols. **I** and **II**.

Gyftopoulos, E.P., and Levine, J.D.,"Work Function Variation of Metals Coated by Metallic Films", J. Appl. Phys. **33**, pp. 67, 1962.

Hensley, E.B.,"Thermionic Emission Constants and Their Interpretation", J. Appl. Phys. **32**, pp. 301, 1961.

Hernqvist, K.G., Kanefsky, M., and Norman, F.H., "Thermionic Energy Converter", RCA Rev. **19**, pp. 244-258, 1959.

Herring, C., and Nichols, M.H., Thermionic Emission, Rev. Mod. Phys. **21**, pp. 185-270; April 1949.

Hohorst, J.K.,"SCDAP/RELAP5/MOD2 Code Manual, vol.**4**:MATPRO-A Library of Materials Properties for Light-Water Reactor Accident Analysis", NUREG/CR-5273; EGG-2555; EG&G Idaho, Inc., February 1990.

Houston, J.M.,"Theoretical Efficiency of the Thermionic Energy Converter", J. Appl. Phys. **30**, pp. 481-487, 1959.

Houston, J.M.,"Thermionic Emission of Refractory Metals in Cesium", Vol. **6**, pp. 358, 1961.

Houston, J.M., and Webster, H.F.,"Thermionic Energy Conversion", Electronics and Electron Physics, Vol. **17**, pp. 125-206, Academic Press Inc., New York, 1962.

Ingold, J.H.,"Calculation of the Maximum Efficiency of the Thermionic Converter", J. Appl. Phys. **32**, pp. 769, 1961.

James, M.L., Smith, G.M., and Wolford, J.C., Applied Numerical Methods for Digital Computation with FORTRAN and CSMP, Harper & Row, Publishers, Inc., 2nd edition, 1977.

Kaye, J., and Welsh, J.A., Direct Conversion of Heat To Electricity, John Wiley & Sons, Inc., NY, 1960.
Joint Soviet-American Venture, JV Inertek, Thermionic Fuel Element of Power Plant "TOPAZ-II", Report, Moscow, 1991.

Kitrilakis, S.S., and Meeker, M.,"Experimental Determination of the Heat Conduction of Cesium Gas", Advanced Energy Conversion, Vol. 3, pp. 59-68, 1963.

Klein, A.C., Lee, H.H., Lewis, B.R., and Pawlowski, R.A., "Design Analysis Code for Space Nuclear Reactor Using Single Cell Thermionic Fuel Element", at Nuclear Technologies for Space Exploration, p271, Jackson Hole, WY, 1992.

Klein, A.C., and Pawlowski, R.A.,"Analysis of TOPAZ-II Thermionic Fuel Element Performance Using TFEHX", Proceedings of 10th Symposium on Space Nuclear Power Systems, Albuquerque, NM, 1993.

Lahey Computer System, Inc.,"F77L/EM-32 Version 3", NV 1990.

Lamarsh, J.R., Introduction to Nuclear Reactor Theory, 2nd Edition, Addison-Wesley Inc., NY, 1972.

Lee, H.H.,"System Modeling and Reactor Design Study of an Advanced Incore Thermionic Space Reactor", M.S. Thesis, Oregon State University, Corvallis, OR, Sept. 1992.

Lewis, E.E., Nuclear Power Reactor Safety, 1st Edition, John Wiley & Sons, Inc., 1977.

Luppov, A.N., Nechaev, Y.A., Skorlygina, T.O., Skorlygin, V.V., and Yermoshin, M.Y., Kurchatov Institute, Moscow, Russia, Central Design Bureau of Machine Building, St. Petersburg, Russia, and Gunther, N., Space Power, Inc., San Jose, CA, "Simulation Behavior of TOPAZ-2 and Relation to the Operation of TSET System,"Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1495-1498, Albuquerque, NM, January 1993.

Luppov, A.N., Nickitin, V.P., Nicolaev, Y.V., Ogloblin, B.G., Ponomarev-Stepnoi, N.N., Usov,V.A., and Wetch, J.R.,"TOPAZ-2 Thermionic Space Nuclear Power System and Perspectives of its Development," 8th Symposium on Space Nuclear Power Systems Proceedings, CONF-910116, Albuquerque, NM, Jan. 1991.

McVey, J.B.,"Preliminary Technical Report-CYLCON Semi-2D Cylindrical Converter Model", Rasor Associates, Inc., Sunnyvale, CA, 1990.

McVey, J.B.,"Planar Converter Standard Model Documentation Supplementary Description of TECMDL Converter Physics", E-563-002-B-063087, Rasor Associates, Inc., Sunnyvale, CA, 1990.

McVey, J.B.,"TECMDL-Ignited Mode Planar Converter Model", E-563-004-C-082988, Rasor Associates, Inc., Sunnyvale, CA, Aug. 1990.

Morris, D.B.,"The Thermionic System Evaluation Test (TSET): Description, Limitations, and the Invovment of the Space Nuclear Power Community," Proceedings of 10th Symposium on Space Nuclear Power and Propulsion, Part 3, pp. 1251-1256, Albuquerque, NM, January 1993.

Moss, H.,"Thermionic Diodes as Energy Converters", Brit. J. Electron., **2**, pp. 305-322, 1957.

Ponomarev-Stepnoi, N.N., et al, "NPS "TOPAZ-II" Description,"JV INERTEK report, Moscow, Russia, 1991.

Ponomarev-Stepnoi, N.N., et al, "Thermionic Fuel Element of Power Plant TOPAZ-2, "JV INERTEK report, Moscow, Russia, 1991. N.N. Ponomarev-Stepnoi, et al, "NPS "TOPAZ-II" Trials Results, "JV INERTEK report, Moscow, Russia, 1991.

Ralston, A., A first Course in Numerical Analysis, 2nd Edition, McGraw Hill Book Company, 1978.

Rasor, N.S.,"Figure of Merit for Thermionic Energy Conversion, J. Appl. Phys. **31**, pp. 163-167, 1960.

Rasor, N.S.,"Thermionic Energy Conversion," in Applied Atomic Collision Processes, Massey, H.S.W., McDaniel, E.W., and Bederson,B., eds. Vol. **5**, pp. 169-200, Academic Press, New York, 1982.

Rasor, N.S.,"Thermionic Energy Conversion Plasmas", IEEE Transactions on Plasma Science, Vol. **19**, No. 6, Dec. 1991

Reimann, A.L., Thermionic Emission, J. Wiley & Sons, Inc., New York, 1934.

Rittner, E.S.,"On the Theory of the Close-Spaced Impregnated Cathode Thermionic Converter", J. Appl. Phys. **31**, pp. 1065, 1960.

Schaumburg, K., Wasniewski, J., Zlatev, Z., Y12M Solution of Large and Sparse Systems of Linear Algebraic Equations, Springer-Verlag Inc., 1981.

Schock, A.,"Optimization of Emission-Limited Thermionic Generators", J. Appl. Phys. **32**, pp. 1564, 1961.

Smith, G.D., Numerical Solution of Partial Differential Equations, Oxford University Press, NY 1965.

Soo, S.L., Direct Energy Conversion, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1968.

Spring, K.H., Direct Energy Conversion, Academic Press, London and New York, 1965.

Steinberg, R.K.,"Hot-Cathode Arcs in Cesium Vapor", J. Appl. Phys. **21**, pp. 1028, 1950.

Sutton, G.W., Direct Energy Conversion, Vol. **3**, McGraw Hill Book Company, N.Y., 1966.

Webster, H.F.,"Calculation of the Performance of a High-Vacuum Thermionic Energy Converter", J. Appl. Phys. **30**, pp. 488, 1959.

Weisman, J., Elements of Nuclear Reactor Design, Kreiger Publishing Company, 1983.

Wilson, V.C.,"Conversion of Heat to Electricity by Thermionic Emission", Bull. Am. Phys. Soc.**3**, pp. 266, 1958.

Wilson, V.C.,"Conversion of Heat to Electricity by Thermionic Emission", J. Appl. Phys. **30**, pp 475-481, 1959.

APPENDICES

Appendix A

Sample Input File

SAMPLE INPUT FILE FOR

```
***  S T E A D Y   S T A T E    P R O B L E M ***
4
1
895.0.12 0.78 620.0 245.0 245.0 3177.705
2
0.73 0.425
0.15 0.60 1 3         ! fuel
0.60 0.60 0          ! fuel emitter gap
0.60 0.75 2 0         ! emitter
0.75 0.80 7 0         ! emitter-collector gap
0.80 0.90 3 0         ! collector
0.90 0.90 0          ! collector -insulator gap
0.90 0.95 8 0         ! insulator
0.95 0.95 0          ! insulator-cladding gap
0.95 1.00 3 0         ! cladding
1.00 1.25             ! coolant channel
0.20 0.00 25.40 25.40
```

SAMPLE INPUT FILE FOR

```
***  S T A R T U P   P R O B L E M ***
1
1
1
2000.0
40.0
0.5
0
900.0
2000.0
100.0
895.0.12 0.78 620.0 245.0 245.0 3177.705
2
0.73 0.425
0.15 0.60 1 3       ! fuel
0.60 0.60 0       ! fuel emitter gap
0.60 0.75 2 0       ! emitter
0.75 0.80 7 0       ! emitter-collector gap
0.80 0.90 3 0       ! collector
0.90 0.90 0       ! collector -insulator gap
0.90 0.95 8 0       ! insulator
0.95 0.95 0       ! insulator-cladding gap
0.95 1.00 3 0       ! cladding
1.00 1.25           ! coolant channel
0.20 0.00 25.40 25.40
```

SAMPLE INPUT FILE FOR

```
*** L O S S   O F   F L O W   P R O B L E M   ***
3
1
1
500
10
0.5
0
0.5 0.5 30.0
895.0.12 0.78 620.0 245.0 245.0 3177.705
2
0.73 0.425
0.15 0.60 1 3       ! fuel
0.60 0.60 0       ! fuel emitter gap
0.60 0.75 2 0       ! emitter
0.75 0.80 7 0       ! emitter-collector gap
0.80 0.90 3 0       ! collector
0.90 0.90 0       ! collector -insulator gap
0.90 0.95 8 0       ! insulator
0.95 0.95 0       ! insulator-cladding gap
0.95 1.00 3 0       ! cladding
1.00 1.250          ! coolant channel
0.20 0.00 25.40 25.40
2207.611 2186.197 2131.858 2049.918 1943.400 1939.525  919.364  915.027  902.619  900.716
2261.743 2237.103 2174.669 2080.671 1958.691 1954.265  923.913  919.955  908.548  906.816
2362.666 2333.887 2261.168 2152.020 2010.708 2005.680  932.373  928.077  915.561  913.681
2449.023 2416.885 2335.950 2214.947 2058.707 2053.209  941.053  936.471  922.963  920.958
2496.643 2462.591 2377.041 2249.513 2085.177 2079.421  949.393  944.654  930.523  928.448
2496.923 2462.867 2377.311 2249.774 2085.429 2079.673  956.919  952.191  937.950  935.880
2449.686 2417.542 2336.592 2215.569 2059.308 2053.810  963.355  958.808  944.983  942.992
2363.191 2334.407 2261.677 2152.512 2011.179 2006.151  968.705  964.462  951.458  949.601
2261.853 2237.203 2174.740 2080.698 1958.669 1954.236  973.350  969.444  957.386  955.675
2222.000 2200.541 2144.165 2058.677 1947.356 1943.316  982.179  977.829  964.311  962.414
895.000  901.026  907.582  914.603  921.953  929.398   936.677  943.567  949.961  955.821
```

SAMPLE INPUT FILE FOR

```
***  S H U T   D O W N   P R O B L E M ***
2
1
1
5.0
1.0
0.5
0
0.0065
-3.0
895.0.12 0.78 620.0 245.0 245.0 3177.705
2
0.73 0.425
0.15 0.60 1 3         ! fuel
0.60 0.60 0          ! fuel emitter gap
0.60 0.75 2 0         ! emitter
0.75 0.80 7 0         ! emitter-collector gap
0.80 0.90 3 0         ! collector
0.90 0.90 0          ! collector -insulator gap
0.90 0.95 8 0         ! insulator
0.95 0.95 0          ! insulator-cladding gap
0.95 1.00 3 0         ! cladding
1.00 1.25             ! coolant channel
0.20 0.00 25.40 25.40
2207.611 2186.197 2131.858 2049.918 1943.400 1939.525  919.364  915.027  902.619  900.716
2261.743 2237.103 2174.669 2080.671 1958.691 1954.265  923.913  919.955  908.548  906.816
2362.666 2333.887 2261.168 2152.020 2010.708 2005.680  932.373  928.077  915.561  913.681
2449.023 2416.885 2335.950 2214.947 2058.707 2053.209  941.053  936.471  922.963  920.958
2496.643 2462.591 2377.041 2249.513 2085.177 2079.421  949.393  944.654  930.523  928.448
2496.923 2462.867 2377.311 2249.774 2085.429 2079.673  956.919  952.191  937.950  935.880
2449.686 2417.542 2336.592 2215.569 2059.308 2053.810  963.355  958.808  944.983  942.992
2363.191 2334.407 2261.677 2152.512 2011.179 2006.151  968.705  964.462  951.458  949.601
2261.853 2237.203 2174.740 2080.698 1958.669 1954.236  973.350  969.444  957.386  955.675
2222.000 2200.541 2144.165 2058.677 1947.356 1943.316  982.179  977.829  964.311  962.414
895.000  901.026  907.582  914.603  921.953  929.398   936.677  943.567  949.961  955.821
```

Appendix B

Code User Manual

TFETC INPUT DESCRIPTION (CODE MANUAL)

********** S T E A D Y   S T A T E **********

Line 1 A 80   Title (Steady state)

Line 2 I1   must be 4 for steady state

Line 3 I1   Type of solver (must be 1 or 2)

1. Gaussian Elimination Solver
2. Y12M Solver

Line 4 7F10.0   Tinlet, Mdot, W, Tr, Itop, Ibottom, PowerTh

- Tinlet : Temperature of the NaK coolant as it enters the coolant channel of the TFE (K).
- Mdot : Mass Flowrate of the NaK coolant within the TFE coolant channel (Kg/s).
- W : Weight Fraction of Potassium within the NaK coolant. Range: $0 \leq W \leq 1.0$. Standard value: 0.78 (Eutectic NaK)
- Tr : Cesium reservoir temperature (K), $Tr \geq 0$
- Itop : Current flow at the top connection of the TFE is defined as the end at which the coolant exits the TFE coolant channel (Amperes). Range: ≥ 0.0
- Ibottom: Current flow at the bottom connection of the TFE is defined as the end at which the coolant enters the TFE coolant channel (Amperes). Range: $\geq$ 0.0
- PowerTh: For PowerTh $\geq$ 0: Total power produced in the fuel of the TFE (Watts).

  For PowerTh $<$ 0: ABS(PowerTh) =

Average Volumetric heat generation rate within the fuel of the TFE (Watts/cm^3).

Line 5: I2 If TabFlag = 1, a table of axial power peaking factors versus axial position follows.

If TabFlag = 2, the following line contains coefficients for a correlation for the axial power peaking factors versus axial position.

Other values of TabFlag generate an error condition.

Line 6:2F10.0 Z,G if TabFlag = 2

- If TabFlag = 1, then Z and G are entries in a table of axial power peaking factors.

Z = Axial position of the table entry (cm). Measured from the bottom of the TFE.

G = Ratio of the linear heat generation rate at position Z to the average linear heat generation rate in the TFE fuel.

This line is repeated until G < 0.
OR
A,B if TabFlag = 2.

- If TabFlag = 2, then A and B are coefficients in the following correlation:

G = A + B * SIN ((Z - Zmin) / (Zmax - Zmin) * Pi)

where G is the ratio of the linear heat generation rate in the TFE fuel at axial position Z to the average linear heat generation rate.

Zmin is the axial position of the

bottom of the fueled region (cm).

Zmax is the axial position of the top of the fueled region (cm).

Pi = 3.1415926

The following line is repeated for each of the nine internal regions of the TFE excluding the coolant channel.

2F10.0, 2I5 IR(I), OR(I), MatNum(I), Rmest(I).I = 1 to 9

IR(I): Inside radius of region I(cm). IR(I) must be greater than or equal to zero; IR(I) must equal OR(I=1) for I = 1 through 9.

OR(I): Outside radius of region I(cm). OR(I) must equal IR(I+1) for I = 1 through 9.

MatNum(I): Identification number for the material in region I. The numbers are currently defined as follows:

MatNum	Material
1	UO_2
2	Tungsten
3	Niobium
4	Nb1Zr
5	Molybdenum
6	Rhenium
7	Cesium
8	Al_2O_3

Rmesh(I): Rmesh(I) equals the number of radial mesh intervals within region I minus 1. Mesh points automatically exist at the interior and exterior surfaces of each region.

(If IR(1) = 0, then the inner most mesh point occurs at the fuel centerline). Specifying Rmest(I) 〉0 generates additional mesh points within region I. For instance, consider Rmesh(1) = 3. This results in 5 mesh points (4 mesh intervals) within the fueled region of the TFE: one on the interior surface of the fuel, and three equally spaced radially within the fuel. The interior and exterior mesh points are shared with the adjacent regions. For example, if Rmest(7) = 0, the insulator region contains only 2 radial mesh points (1 radial mesh interval covering the entire insulator region) and the interior mesh point is also the exterior mesh point of the collector/insulator gap region, while the exterior mesh point is also the interior mesh point of the insulator/cladding gap region.

Notes:

- The TFE regions (I = 1 to 9) are as follows:

I	Region
1	Fuel
2	Fuel/Emitter Gap
3	Emitter
4	Emitter/Collector Gap
5	Collector
6	Collector/Insulator Gap
7	Insulator
8	Insulator/Cladding Gap
9	Cladding
10	Coolant Channel

- The region 2, 6 and 8 are included to allow

for small gaps to be modeled between the solid regions of the TFE.
However, it is recommended that the TFE be modeled with all of the solid regions in close contact, i.e. with the following specifications:

OR(1) = IR(2) = OR(2) = IR(3)

OR(5) = IR(6) = OR(6) = IR(7)

OR(7) = IR(8) = OR(8) = IR(9)

<u>2F10.0</u> IR(10), OR(10)

IR(10): Inside radius of the coolant channel (cm). Must equal OR(9).

OR(10): Outside radius of the coolant channel (cm)

<u>4F10.0</u> Ems, Zmin, Zmax, L

Ems: Effective radiative emissivity between the emitter and collector surfaces (across the emitter/collector gap). The heat transfer Q due to radiation across the emitter/collector gap is given by:

Q(Z) = Ems * (Temitter(Z)** 4 - Tcollector(Z)** 4)

where Temitter(Z) is the emitter surface temperature at position Z.

Tcollector(Z) is the collector surface temperature at position Z.

Note: An effective emissivity --- "less than 0.1 is not maintainable, and one greater than 0.2 is undesirable." Hatsopoulos and Gyftopoulos, Thermionic Energy Conversion, Vol. 1, 1975, p.83.

Zmin: The axial position of the bottom of the fueled region of the TFE (cm).

Zmax: The axial position of the top of the fueled region of the TFE (cm).

L: Total length of the TFE, including electrode leads.

Note: TFETC does not model heat conduction away from the TFE via the electrode leads. Therefore, the value of L must equal Zmax - Zmin; otherwise an error condition results.

TFETC INPUT DESCRIPTION (CODE MANUAL)

************* S T A R T U P *************

Line 1 A 80 — Title (Start up)

Line 2 I1 — must be 1 for start up

Line 3 I1 — Type of solver (must be 1 or 2)

1. Gaussian Elimination Solver
2. Y12M Solver

Line 4 I1 — Options: (must be 1 or 2)

1 use default ambient initial temperature (i.e. T = 298 K)

2 a table of T(r,z)

Other values of options generate an error.

Line 5: F10.0 — Transient Test Time, sec. (Transient Time must be greater than 0 and greater than Print Time Step)

Line 6: F10.0 — Print Time Step, sec.

Line 7: F10.0 — delta t, sec. (must be less than Print Time Step)

Line 8: F10.0 — printout options (must be 0 or 1)

0 prints everything except the temperature profile

throughout the TFE.

1 prints everything

Line 9: F10.0 temperature at which heating by helium is stopped, K.

(recommended between 700 and 900)

Line 10: F10.0 temperature at which electron cooling starts, K.

(recommended between 1800 and 1950)

Line 11: F10.0 power rise coefficient, tau, sec.

(must be greater than 0)

Line 12: 7F10.0 will be the same as Line 4 in the steady state input file

After Line 11, the startup input file follows the same description of steady state input file starting from Line 4.

TFETC INPUT DESCRIPTION (CODE MANUAL)

********** L O S S   O F   F L O W **********

<u>Line 1 A 80</u> Title (Loss of Flow)

<u>Line 2 I1</u> must be 3 for loss of flow

<u>Line 3 I1</u> Type of solver
(must be 1 or 2)

1. Gaussian Elimination Solver
2. Y12M Solver

<u>Line 4 I1</u> Options:
(must be 1 or 2)

1. The steady state temperature distribution is generated by the TFEHX code.
2. a table of T(r,z) can be used by the user as a forcing function.

Other values of options generate an error.

<u>Line 5: F10.0</u> Transient Test Time, sec.
(Transient Time must be greater than 0 and greater than Print Time Step)

<u>Line 6: F10.0</u> Print Time Step, sec.

<u>Line 7: F10.0</u> delta t, sec.
(must be less than Print Time Step)

<u>Line 8: F10.0</u> printout options
(must be 0 or 1)

0 prints everything except the temperature profile

throughout the TFE.

1 prints everything

Line 9: 3F10.0 Loss of mass flowrate coefficients: A, B, τ (τ should be in seconds)

((A + B) must equal 1)

$$m = m \, (A + B * \exp(-t/\tau))$$

1) 1/1 Pump Failure
A = 0.0
B = 1.0
$\tau > 0.0$

2) 1/2 Pump Failure
A = 0.5
B = 0.5
$\tau > 0.0$

3) 1/3 Pump Failure
A = 0.67
B = 0.33
$\tau > 0.0$

4) 1/4 Pump Failure
A = 0.75
B = 0.25
$\tau > 0.0$

After Line 9, the loss of flow input file follows the same description of steady state input file starting from Line 4.

TFETC INPUT DESCRIPTION (CODE MANUAL)

************* S H U T   D O W N *************

Line 1 A 80   Title (Shut down)

Line 2 I1   must be 2 for shut down

Line 3 I1   Type of solver
(must be 1 or 2)

1. Gaussian Elimination Solver
2. Y12M Solver

Line 4 I1   Options:
(must be 1 or 2)

1. The steady state temperature distribution is generated by the TFEHX code.
2. a table of T(r,z) can be used by the user as a forcing function.

Other values of options generate an error.

Line 5: F10.0 Transient Test Time, sec.
(Transient Time must be greater than 0 and greater than Print Time Step)

Line 6: F10.0 Print Time Step, sec.

Line 7: F10.0 delta t, sec.
(must be less than Print Time Step)

Line 8: F10.0 printout options
(must be 0 or 1)

0 prints everything except the temperature profile throughout the TFE.

1 prints everything

Line 9: F10.0 Total delayed fraction, β

β = 0.0065 for ^{235}U
β = 0.0026 for ^{233}U
β = 0.0021 for ^{239}Pu
β = 0.0203 for ^{232}Th

Line 10:F10.0 Negative Reactivity Coefficient must be less than 0.0, $ unit.

After Line 10, the loss of flow input file follows the same description of steady state input file starting from Line 4.

Appendix C

Sample Output File

```
1******************* TFETC *******************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
***  S T A R T  U P   P R O B L E M ***
 *****                 *****
 *****      Start Up Problem *****
            Linear Equations solved using Gaussian elimination
            Simulation Period, TIME =   20.00 Secs.
            Print Time Step, TPRINT =   10.00 Secs.
            Time Step Increment, delta t =     0.5000 Secs.
            Print Option, ipout = 1.
            Stop helium heating at the emitter temperature, Tstop =  900.00.
            Start Electron cooling at the emitter temperature, Tstart = 2000.00.
            Power-rise coefficient, tau = 0.100E+03,
            P(t) = P(0) * [ 1 - Exp ( - t / tau ) ] .

 COOLANT TYPE: Molten Sodium-Potassium Alloy (NaK)
               Potassium composition = 78%
 COOLANT MASS FLOW RATE:   0.12 kilograms per second.
 TEMPERATURE OF COOLANT AT CHANNEL INLET:  895.0 K.

 TEMPERATURE OF CESIUM RESERVOIR:  620.0 K.
 PRESSURE OF CESIUM VAPOR:    5.6 Torr.

 EFFECTIVE EMISSIVITY FOR RADIANT HEAT TRANSFER FROM
          THE EMITTER SURFACE TO THE COLLECTOR SURFACE:  0.200000

 OUTPUT CURRENT FROM THE TOP OF THE TFE:    245.0 Amperes.
 OUTPUT CURRENT FROM THE BOTTOM OF THE TFE:  245.0 Amperes.

 TOTAL THERMAL POWER PRODUCED IN THE TFE FUEL: 3177.7 Watts.
 AVERAGE VOLUMETRIC HEAT GENERATION RATE FOR THE TFE FUEL:  118.0 Watts.
 CORRELATION FOR THE RATIO OF THE HEAT GENERATION RATE
          AT POSITION Z TO THE AVERAGE HEAT GENERATION RATE IN
          THE TFE FUEL:

         F =     0.7300+    0.4250 * SIN((Z-Zmin)/(Zmax-Zmin)*3.14159)

 AXIAL PEAK-TO-AVERAGE RATIO FOR HEAT GENERATION IS: 1.1543

1******************* TFETC *******************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
***  S T A R T U P   R O B L E M ***

 ***** GEOMETRY DATA EDIT *****
  ***** RADIAL GEOMETRY *****

                               Inside        Outside                 Number of
        Region                 Radius             Material      Interior
                                (cm)          (cm)                   Mesh Points
 ----------------------     ----------    ----------    --------    -----------
  fuel                       0.150000      0.600000      uo2          3
  emitter                    0.600000      0.750000      w            0
  emitter-collector gap      0.750000      0.800000      cs           0
  collector                  0.800000      0.900000      nb           0
  insulator                  0.900000      0.950000      al2o3        0
  cladding                   0.950000      1.000000      nb           0
  coolant channel            1.000000      1.250000

  ***** AXIAL GEOMETRY *****

 AXIAL POSITION OF THE UPPER LIMIT FOR
         THE FUELED REGION OF THE TFE:  0.000000 (cm)
 AXIAL POSITION OF THE LOWER LIMIT FOR
         THE FUELED REGION OF THE TFE: 25.400000 (cm)
 AXIAL EXTENT OF THE FUELED REGION OF THE TFE: 25.400000 (cm)
 TOTAL LENGTH OF THE TFE, INCLUDING ELECTRODE LEADS: 25.400000 (cm)
1******************* TFETC *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
***  S T A R T U P   R O B L E M ***
 TIME =      0.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,   1) =   298.0000000
 t(  1,   2) =   298.0000000
 t(  1,   3) =   298.0000000
 t(  1,   4) =   298.0000000
 t(  1,   5) =   298.0000000
 t(  1,   6) =   298.0000000
 t(  1,   7) =   298.0000000
 t(  1,   8) =   298.0000000
 t(  1,   9) =   298.0000000
 t(  1,  10) =   298.0000000
 t(  2,   1) =   298.0000000
 t(  2,   2) =   298.0000000
 t(  2,   3) =   298.0000000
```

```
t(  2,   4) =    298.0000000
t(  2,   5) =    298.0000000
t(  2,   6) =    298.0000000
t(  2,   7) =    298.0000000
t(  2,   8) =    298.0000000
t(  2,   9) =    298.0000000
t(  2,  10) =    298.0000000
t(  3,   1) =    298.0000000
t(  3,   2) =    298.0000000
t(  3,   3) =    298.0000000
t(  3,   4) =    298.0000000
t(  3,   5) =    298.0000000
t(  3,   6) =    298.0000000
t(  3,   7) =    298.0000000
t(  3,   8) =    298.0000000
t(  3,   9) =    298.0000000
t(  3,  10) =    298.0000000
t(  4,   1) =    298.0000000
t(  4,   2) =    298.0000000
t(  4,   3) =    298.0000000
t(  4,   4) =    298.0000000
t(  4,   5) =    298.0000000
t(  4,   6) =    298.0000000
t(  4,   7) =    298.0000000
t(  4,   8) =    298.0000000
t(  4,   9) =    298.0000000
t(  4,  10) =    298.0000000
t(  5,   1) =    298.0000000
t(  5,   2) =    298.0000000
t(  5,   3) =    298.0000000
t(  5,   4) =    298.0000000
t(  5,   5) =    298.0000000
t(  5,   6) =    298.0000000
t(  5,   7) =    298.0000000
t(  5,   8) =    298.0000000
t(  5,   9) =    298.0000000
t(  5,  10) =    298.0000000
t(  6,   1) =    298.0000000
t(  6,   2) =    298.0000000
t(  6,   3) =    298.0000000
t(  6,   4) =    298.0000000
t(  6,   5) =    298.0000000
t(  6,   6) =    298.0000000
t(  6,   7) =    298.0000000
t(  6,   8) =    298.0000000
t(  6,   9) =    298.0000000
t(  6,  10) =    298.0000000
t(  7,   1) =    298.0000000
t(  7,   2) =    298.0000000
t(  7,   3) =    298.0000000
t(  7,   4) =    298.0000000
t(  7,   5) =    298.0000000
t(  7,   6) =    298.0000000
t(  7,   7) =    298.0000000
t(  7,   8) =    298.0000000
t(  7,   9) =    298.0000000
t(  7,  10) =    298.0000000
t(  8,   1) =    298.0000000
t(  8,   2) =    298.0000000
t(  8,   3) =    298.0000000
t(  8,   4) =    298.0000000
t(  8,   5) =    298.0000000
t(  8,   6) =    298.0000000
t(  8,   7) =    298.0000000
t(  8,   8) =    298.0000000
t(  8,   9) =    298.0000000
t(  8,  10) =    298.0000000
t(  9,   1) =    298.0000000
t(  9,   2) =    298.0000000
t(  9,   3) =    298.0000000
t(  9,   4) =    298.0000000
t(  9,   5) =    298.0000000
t(  9,   6) =    298.0000000
t(  9,   7) =    298.0000000
t(  9,   8) =    298.0000000
t(  9,   9) =    298.0000000
t(  9,  10) =    298.0000000
t( 10,   1) =    298.0000000
t( 10,   2) =    298.0000000
t( 10,   3) =    298.0000000
t( 10,   4) =    298.0000000
t( 10,   5) =    298.0000000
t( 10,   6) =    298.0000000
t( 10,   7) =    298.0000000
t( 10,   8) =    298.0000000
t( 10,   9) =    298.0000000
```

```
t( 10, 10) =   298.0000000
tcool(  1) =   895.0000000
tcool(  2) =   895.0000000
tcool(  3) =   895.0000000
tcool(  4) =   895.0000000
tcool(  5) =   895.0000000
tcool(  6) =   895.0000000
tcool(  7) =   895.0000000
tcool(  8) =   895.0000000
tcool(  9) =   895.0000000
tcool( 10) =   895.0000000

Temperature of coolant at core exit:    895.000 degrees K.
Voltage across bottom of cell:     0.0000000
Voltage across top of cell:     0.0000000
Output current =     0.0000000
Output electrical power =     0.0000000
Total Thermal power =     0.0000000

         Z                    V                   Qec                  Jdens
---------------      ---------------      ---------------      ---------------

     0.00000000           0.00000000           0.00000000           0.00000000
     2.82222222           0.00000000           0.00000000           0.00000000
     5.64444444           0.00000000           0.00000000           0.00000000
     8.46666667           0.00000000           0.00000000           0.00000000
    11.28888889           0.00000000           0.00000000           0.00000000
    14.11111111           0.00000000           0.00000000           0.00000000
    16.93333333           0.00000000           0.00000000           0.00000000
    19.75555556           0.00000000           0.00000000           0.00000000
    22.57777778           0.00000000           0.00000000           0.00000000
    25.40000000           0.00000000           0.00000000           0.00000000

         Z                 EmHeat               ColHeat
---------------      ---------------      ---------------

     0.00000000           0.00000000           0.00000000
     2.82222222           0.00000000           0.00000000
     5.64444444           0.00000000           0.00000000
     8.46666667           0.00000000           0.00000000
    11.28888889           0.00000000           0.00000000
    14.11111111           0.00000000           0.00000000
    16.93333333           0.00000000           0.00000000
    19.75555556           0.00000000           0.00000000
    22.57777778           0.00000000           0.00000000
    25.40000000           0.00000000           0.00000000

         Z                   Qch                  Qrad               QCsCond
---------------      ---------------      ---------------      ---------------

     0.00000000           0.00000000           0.00000000           0.00000000
     2.82222222           0.00000000           0.00000000           0.00000000
     5.64444444           0.00000000           0.00000000           0.00000000
     8.46666667           0.00000000           0.00000000           0.00000000
    11.28888889           0.00000000           0.00000000           0.00000000
    14.11111111           0.00000000           0.00000000           0.00000000
    16.93333333           0.00000000           0.00000000           0.00000000
    19.75555556           0.00000000           0.00000000           0.00000000
    22.57777778           0.00000000           0.00000000           0.00000000
    25.40000000           0.00000000           0.00000000           0.00000000
1******************** TFETC ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
***  S T A R T U P   P R O B L E M ***
 TIME =      10.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,  1) =   302.8718008
 t(  1,  2) =   304.9487739
 t(  1,  3) =   306.0289744
 t(  1,  4) =   306.7853917
 t(  1,  5) =   307.1854458
 t(  1,  6) =   307.1843456
 t(  1,  7) =   306.7823554
 t(  1,  8) =   306.0245954
 t(  1,  9) =   304.9437831
 t(  1, 10) =   303.0465792
 t(  2,  1) =   302.6851613
 t(  2,  2) =   304.8625006
 t(  2,  3) =   305.9377254
 t(  2,  4) =   306.6858891
 t(  2,  5) =   307.0813961
 t(  2,  6) =   307.0803280
```

```
t(  2,   7) =    306.6829409
t(  2,   8) =    305.9334694
t(  2,   9) =    304.8577151
t(  2,  10) =    302.8713899
t(  3,   1) =    302.1843453
t(  3,   2) =    304.6326810
t(  3,   3) =    305.6944395
t(  3,   4) =    306.4205951
t(  3,   5) =    306.8039771
t(  3,   6) =    306.8029909
t(  3,   7) =    306.4178717
t(  3,   8) =    305.6905006
t(  3,   9) =    304.6283846
t(  3,  10) =    302.3549294
t(  4,   1) =    301.3532343
t(  4,   2) =    304.2518257
t(  4,   3) =    305.2917614
t(  4,   4) =    305.9814804
t(  4,   5) =    306.3447454
t(  4,   6) =    306.3438835
t(  4,   7) =    305.9790991
t(  4,   8) =    305.2883162
t(  4,   9) =    304.2485143
t(  4,  10) =    301.4856964
t(  5,   1) =    300.1233274
t(  5,   2) =    303.6884295
t(  5,   3) =    304.6972746
t(  5,   4) =    305.3331979
t(  5,   5) =    305.6666669
t(  5,   6) =    305.6659676
t(  5,   7) =    305.3312647
t(  5,   8) =    304.6945012
t(  5,   9) =    303.6870040
t(  5,  10) =    300.1934887
t(  6,   1) =    300.0115130
t(  6,   2) =    303.6476278
t(  6,   3) =    304.6519975
t(  6,   4) =    305.2837096
t(  6,   5) =    305.6149047
t(  6,   6) =    305.6142110
t(  6,   7) =    305.2817920
t(  6,   8) =    304.6492483
t(  6,   9) =    303.6462754
t(  6,  10) =    300.0763484
t(  7,   1) =    887.0273947
t(  7,   2) =    893.8633088
t(  7,   3) =    893.9128826
t(  7,   4) =    893.9132047
t(  7,   5) =    893.9132066
t(  7,   6) =    893.9132066
t(  7,   7) =    893.9132025
t(  7,   8) =    893.9126478
t(  7,   9) =    893.8454120
t(  7,  10) =    887.0264901
t(  8,   1) =    887.5363084
t(  8,   2) =    893.9263743
t(  8,   3) =    893.9726393
t(  8,   4) =    893.9729389
t(  8,   5) =    893.9729408
t(  8,   6) =    893.9729407
t(  8,   7) =    893.9729369
t(  8,   8) =    893.9724191
t(  8,   9) =    893.9094610
t(  8,  10) =    887.5354542
t(  9,   1) =    890.5895308
t(  9,   2) =    894.5193399
t(  9,   3) =    894.5403220
t(  9,   4) =    894.5404529
t(  9,   5) =    894.5404537
t(  9,   6) =    894.5404537
t(  9,   7) =    894.5404520
t(  9,   8) =    894.5402253
t(  9,   9) =    894.5117879
t(  9,  10) =    890.5889927
t( 10,   1) =    891.2092255
t( 10,   2) =    894.6369246
t( 10,   3) =    894.6530197
t( 10,   4) =    894.6531194
t( 10,   5) =    894.6531200
t( 10,   6) =    894.6531200
t( 10,   7) =    894.6531187
t( 10,   8) =    894.6529464
t( 10,   9) =    894.6312302
t( 10,  10) =    891.2087526
tcool(   1) =    895.0000000
tcool(   2) =    895.0000000
```

```
tcool(  3) =   895.0000000
tcool(  4) =   895.0000000
tcool(  5) =   895.0000000
tcool(  6) =   895.0000000
tcool(  7) =   895.0000000
tcool(  8) =   895.0000000
tcool(  9) =   895.0000000
tcool( 10) =   895.0000000

Temperature of coolant at core exit:    895.000 degrees K.
Voltage across bottom of cell:     0.0000000
Voltage across top of cell:     0.0000000
Output current =     0.0000000
Output electrical power =     0.0000000
Total Thermal power =   302.3986125
```

Z	V	Qec	Jdens
0.00000000	0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000	0.00000000

Z	EmHeat	ColHeat
0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000

Z	Qch	Qrad	QCsCond
0.00000000	0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000	0.00000000

```
1******************** TFETC ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S T A R T U P   P R O B L E M ***
 TIME =     20.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,  1) =   314.3188972
 t(  1,  2) =   323.8055061
 t(  1,  3) =   328.0695175
 t(  1,  4) =   330.8827027
 t(  1,  5) =   332.3510992
 t(  1,  6) =   332.3372837
 t(  1,  7) =   330.8439274
 t(  1,  8) =   328.0099727
 t(  1,  9) =   323.7156140
 t(  1, 10) =   314.8662271
 t(  2,  1) =   313.8481635
 t(  2,  2) =   323.5737879
 t(  2,  3) =   327.8365972
 t(  2,  4) =   330.6305580
 t(  2,  5) =   332.0876954
 t(  2,  6) =   332.0741236
 t(  2,  7) =   330.5924504
 t(  2,  8) =   327.7779808
 t(  2,  9) =   323.4857544
```

```
t(  2, 10) =   314.4124792
t(  3,  1) =   312.6150764
t(  3,  2) =   322.9732739
t(  3,  3) =   327.2312886
t(  3,  4) =   329.9751740
t(  3,  5) =   331.4030291
t(  3,  6) =   331.3900775
t(  3,  7) =   329.9387688
t(  3,  8) =   327.1750775
t(  3,  9) =   322.8900245
t(  3, 10) =   313.1308168
t(  4,  1) =   310.6686031
t(  4,  2) =   322.0224680
t(  4,  3) =   326.2736928
t(  4,  4) =   328.9383730
t(  4,  5) =   330.3197435
t(  4,  6) =   330.3077369
t(  4,  7) =   328.9045654
t(  4,  8) =   326.2212440
t(  4,  9) =   321.9478092
t(  4, 10) =   311.0827745
t(  5,  1) =   307.9729533
t(  5,  2) =   320.6966308
t(  5,  3) =   324.9407257
t(  5,  4) =   327.4952554
t(  5,  5) =   328.8116071
t(  5,  6) =   328.8008465
t(  5,  7) =   327.4648822
t(  5,  8) =   324.8934389
t(  5,  9) =   320.6361492
t(  5, 10) =   308.2334946
t(  6,  1) =   307.7350411
t(  6,  2) =   320.6104272
t(  6,  3) =   324.8470828
t(  6,  4) =   327.3932048
t(  6,  5) =   328.7049117
t(  6,  6) =   328.6941956
t(  6,  7) =   327.3629555
t(  6,  8) =   324.7999906
t(  6,  9) =   320.5504811
t(  6, 10) =   307.9838412
t(  7,  1) =   894.9136469
t(  7,  2) =   894.9967241
t(  7,  3) =   894.9977590
t(  7,  4) =   894.9977702
t(  7,  5) =   894.9977703
t(  7,  6) =   894.9977703
t(  7,  7) =   894.9977702
t(  7,  8) =   894.9977578
t(  7,  9) =   894.9966875
t(  7, 10) =   894.9136327
t(  8,  1) =   894.9192991
t(  8,  2) =   894.9969118
t(  8,  3) =   894.9978826
t(  8,  4) =   894.9978931
t(  8,  5) =   894.9978932
t(  8,  6) =   894.9978932
t(  8,  7) =   894.9978931
t(  8,  8) =   894.9978815
t(  8,  9) =   894.9968773
t(  8, 10) =   894.9192858
t(  9,  1) =   894.9538973
t(  9,  2) =   894.9986310
t(  9,  3) =   894.9990637
t(  9,  4) =   894.9990683
t(  9,  5) =   894.9990684
t(  9,  6) =   894.9990684
t(  9,  7) =   894.9990683
t(  9,  8) =   894.9990632
t(  9,  9) =   894.9986157
t(  9, 10) =   894.9538897
t( 10,  1) =   894.9606406
t( 10,  2) =   894.9989646
t( 10,  3) =   894.9992939
t( 10,  4) =   894.9992973
t( 10,  5) =   894.9992974
t( 10,  6) =   894.9992974
t( 10,  7) =   894.9992973
t( 10,  8) =   894.9992935
t( 10,  9) =   894.9989531
t( 10, 10) =   894.9606341
tcool(  1) =   895.0000000
tcool(  2) =   895.0000000
tcool(  3) =   895.0000000
tcool(  4) =   895.0000000
tcool(  5) =   895.0000000
```

```
tcool(  6) =   895.0000000
tcool(  7) =   895.0000000
tcool(  8) =   895.0000000
tcool(  9) =   895.0000000
tcool( 10) =   895.0000000

Temperature of coolant at core exit:    895.000 degrees K.
Voltage across bottom of cell:     0.0000000
Voltage across top of cell:     0.0000000
Output current =     0.0000000
Output electrical power =     0.0000000
Total Thermal power =   576.0201923
```

Z	V	Qec	Jdens
0.00000000	0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000	0.00000000

Z	EmHeat	ColHeat
0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000

Z	Qch	Qrad	QCsCond
0.00000000	0.00000000	0.00000000	0.00000000
2.82222222	0.00000000	0.00000000	0.00000000
5.64444444	0.00000000	0.00000000	0.00000000
8.46666667	0.00000000	0.00000000	0.00000000
11.28888889	0.00000000	0.00000000	0.00000000
14.11111111	0.00000000	0.00000000	0.00000000
16.93333333	0.00000000	0.00000000	0.00000000
19.75555556	0.00000000	0.00000000	0.00000000
22.57777778	0.00000000	0.00000000	0.00000000
25.40000000	0.00000000	0.00000000	0.00000000

```
1******************** TFETC ********************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 *****                   *****
 ***** Loss of Flow Problem *****
          Linear Equations solved using Gaussian elimination
          Simulation Period, TIME =   20.00 Secs.
          Print Time Step, TPRINT =   10.00 Secs.
          Time Step Increment, delta t =     0.5000 Secs.
          Print Option, ipout = 1.
          Mass-loss coefficients, A =  0.5000, B =  0.5000, tau = 0.300E+02.
          Mdot(t) = Mdot(0) * [  A +  B * Exp(- t / tau ) ]

 COOLANT TYPE: Molten Sodium-Potassium Alloy (NaK)
              Potassium composition = 78%
 COOLANT MASS FLOW RATE:   0.12 kilograms per second.
 TEMPERATURE OF COOLANT AT CHANNEL INLET:  895.0 K.

 TEMPERATURE OF CESIUM RESERVOIR:  620.0 K.
 PRESSURE OF CESIUM VAPOR:    5.6 Torr.

 EFFECTIVE EMISSIVITY FOR RADIANT HEAT TRANSFER FROM
          THE EMITTER SURFACE TO THE COLLECTOR SURFACE:  0.200000

 OUTPUT CURRENT FROM THE TOP OF THE TFE:     245.0 Amperes.
 OUTPUT CURRENT FROM THE BOTTOM OF THE TFE:  245.0 Amperes.

 TOTAL THERMAL POWER PRODUCED IN THE TFE FUEL: 3177.7 Watts.
 AVERAGE VOLUMETRIC HEAT GENERATION RATE FOR THE TFE FUEL:  118.0 Watts.
 CORRELATION FOR THE RATIO OF THE HEAT GENERATION RATE
          AT POSITION Z TO THE AVERAGE HEAT GENERATION RATE IN
          THE TFE FUEL:

         F =     0.7300+    0.4250 * SIN((Z-Zmin)/(Zmax-Zmin)*3.14159)

 AXIAL PEAK-TO-AVERAGE RATIO FOR HEAT GENERATION IS: 1.1543

1******************** TFETC ********************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***

 ***** GEOMETRY DATA EDIT *****
  ***** RADIAL GEOMETRY *****

                          Inside      Outside                Number of
      Region              Radius      Radius     Material     Interior
                           (cm)        (cm)                  Mesh Points
 ----------------------  ----------  ----------  ---------  -----------
  fuel                   0.150000    0.600000      uo2           3
  emitter                0.600000    0.750000      w             0
  emitter-collector gap  0.750000    0.800000      cs            0
  collector              0.800000    0.900000      nb            0
  insulator              0.900000    0.950000      al2o3         0
  cladding               0.950000    1.000000      nb            0
  coolant channel        1.000000    1.250000

  ***** AXIAL GEOMETRY *****

 AXIAL POSITION OF THE UPPER LIMIT FOR
         THE FUELED REGION OF THE TFE:  0.000000 (cm)
 AXIAL POSITION OF THE LOWER LIMIT FOR
         THE FUELED REGION OF THE TFE: 25.400000 (cm)
 AXIAL EXTENT OF THE FUELED REGION OF THE TFE: 25.400000 (cm)
 TOTAL LENGTH OF THE TFE, INCLUDING ELECTRODE LEADS: 25.400000 (cm)

 Temperature of coolant at core exit:      911.2 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  1  RMS error =   352.3744542  Ave Diff. =   244.0701024
                    Max. Error =  836.2653934

 Temperature of coolant at core exit:      935.1 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  2  RMS error =   120.6911636  Ave Diff. =   82.8249886
                    Max. Error =  262.7163942

 Temperature of coolant at core exit:      946.3 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
```

```
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  3  RMS error =    48.0790538  Ave Diff. =    28.9756862
                     Max. Error =  124.2231601

 Temperature of coolant at core exit:      951.7 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  4  RMS error =    22.1990022  Ave Diff. =    15.0890168
                     Max. Error =   47.7020788

 Temperature of coolant at core exit:      954.4 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  5  RMS error =    10.5703466  Ave Diff. =     6.1852238
                     Max. Error =   24.5401210

 Temperature of coolant at core exit:      955.7 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  6  RMS error =     5.3631696  Ave Diff. =     2.8889611
                     Max. Error =   10.0539319

 Temperature of coolant at core exit:      956.3 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  7  RMS error =     2.7477195  Ave Diff. =     1.1581796
                     Max. Error =    5.1710185

 Temperature of coolant at core exit:      956.6 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  8  RMS error =     1.4849013  Ave Diff. =     0.4761796
                     Max. Error =    2.9321308

 Temperature of coolant at core exit:      956.8 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration :  9  RMS error =     0.8003439  Ave Diff. =     0.1667677
                     Max. Error =    1.6387800

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration : 10  RMS error =     0.4468618  Ave Diff. =     0.0509478
                     Max. Error =    0.9027483

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration : 11  RMS error =     0.2453117  Ave Diff. =     0.0066867
                     Max. Error =    0.4927094

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                    Converging the RMS error to less than 0.1 K.
 Iteration : 12  RMS error =     0.1374013  Ave Diff. =    -0.0057550
                     Max. Error =    0.2662746
```

```
 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration : 13  RMS error =     0.0754500  Ave Diff. =    -0.0080428
                    Max. Error =    0.1428042
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 T(  1,  1) =  2207.6099799
 T(  1,  2) =  2261.7498155
 T(  1,  3) =  2362.6692432
 T(  1,  4) =  2449.0244839
 T(  1,  5) =  2496.6438504
 T(  1,  6) =  2496.9186250
 T(  1,  7) =  2449.6735003
 T(  1,  8) =  2363.1763020
 T(  1,  9) =  2261.8518754
 T(  1, 10) =  2221.9650827
 T(  2,  1) =  2186.1929268
 T(  2,  2) =  2237.1053504
 T(  2,  3) =  2333.8847952
 T(  2,  4) =  2416.8802095
 T(  2,  5) =  2462.5836797
 T(  2,  6) =  2462.8553788
 T(  2,  7) =  2417.5228984
 T(  2,  8) =  2334.3872511
 T(  2,  9) =  2237.1977085
 T(  2, 10) =  2200.5034192
 T(  3,  1) =  2131.8455701
 T(  3,  2) =  2174.6616482
 T(  3,  3) =  2261.1536691
 T(  3,  4) =  2335.9307086
 T(  3,  5) =  2377.0188307
 T(  3,  6) =  2377.2839338
 T(  3,  7) =  2336.5595171
 T(  3,  8) =  2261.6454175
 T(  3,  9) =  2174.7248924
 T(  3, 10) =  2144.1203825
 T(  4,  1) =  2049.8943113
 T(  4,  2) =  2080.6498301
 T(  4,  3) =  2151.9899401
 T(  4,  4) =  2214.9105351
 T(  4,  5) =  2249.4727497
 T(  4,  6) =  2249.7299490
 T(  4,  7) =  2215.5219128
 T(  4,  8) =  2152.4669546
 T(  4,  9) =  2080.6711635
 T(  4, 10) =  2058.6227204
 T(  5,  1) =  1943.3652400
 T(  5,  2) =  1958.6571780
 T(  5,  3) =  2010.6630404
 T(  5,  4) =  2058.6554004
 T(  5,  5) =  2085.1206973
 T(  5,  6) =  2085.3690260
 T(  5,  7) =  2059.2461008
 T(  5,  8) =  2011.1211889
 T(  5,  9) =  1958.6304420
 T(  5, 10) =  1947.2927154
 T(  6,  1) =  1939.4898721
 T(  6,  2) =  1954.2307309
 T(  6,  3) =  2005.6346468
 T(  6,  4) =  2053.1568933
 T(  6,  5) =  2079.3647793
 T(  6,  6) =  2079.6131762
 T(  6,  7) =  2053.7478765
 T(  6,  8) =  2006.0928726
 T(  6,  9) =  1954.1970618
 T(  6, 10) =  1943.2526925
 T(  7,  1) =   919.3693636
 T(  7,  2) =   923.9247743
 T(  7,  3) =   932.3917029
 T(  7,  4) =   941.0796696
 T(  7,  5) =   949.4278157
 T(  7,  6) =   956.9612147
 T(  7,  7) =   963.4052389
 T(  7,  8) =   968.7628206
 T(  7,  9) =   973.4131859
 T(  7, 10) =   982.2504182
 T(  8,  1) =   915.0295911
 T(  8,  2) =   919.9643110
 T(  8,  3) =   928.0941405
```

```
T(  8,  4) =   936.4957394
T(  8,  5) =   944.6864194
T(  8,  6) =   952.2318028
T(  8,  7) =   958.8557014
T(  8,  8) =   964.5168861
T(  8,  9) =   969.5054473
T(  8, 10) =   977.8976175
T(  9,  1) =   902.6200325
T(  9,  2) =   908.5560904
T(  9,  3) =   915.5751364
T(  9,  4) =   922.9840447
T(  9,  5) =   930.5510512
T(  9,  6) =   937.9848177
T(  9,  7) =   945.0246739
T(  9,  8) =   951.5067375
T(  9,  9) =   957.4406258
T(  9, 10) =   964.3720872
T( 10,  1) =   900.7165965
T( 10,  2) =   906.8230273
T( 10,  3) =   913.6950848
T( 10,  4) =   920.9783348
T( 10,  5) =   928.4762371
T( 10,  6) =   935.9152912
T( 10,  7) =   943.0341280
T( 10,  8) =   949.6493791
T( 10,  9) =   955.7301540
T( 10, 10) =   962.4749495
Voltage across bottom of cell:     0.6427667
Voltage across top of cell:     0.6465651
Output current =   490.0000000
Output electrical power =   315.8862832
```

Z	V	Qec	Jdens
0.00000000	0.64276668	12.99430163	4.81276691
2.82222222	0.73959046	10.19453024	3.65189375
5.64444444	0.81543279	11.26422623	3.93880029
8.46666667	0.86809799	11.79150175	4.05750063
11.28888889	0.89565457	12.06399812	4.11770336
14.11111111	0.89685880	12.08239618	4.12389963
16.93333333	0.87147959	11.85322300	4.07907171
19.75555556	0.82030557	11.39452654	3.98658183
22.57777778	0.74481487	10.47748564	3.75821204
25.40000000	0.64656509	13.89597476	5.16933972

Z	EmHeat	ColHeat
0.00000000	14.49810519	12.78456733
2.82222222	8.68565016	7.62157940
5.64444444	4.81674372	4.12394584
8.46666667	1.87878625	1.57487539
11.28888889	0.24869437	0.20682254
14.11111111	0.15331479	0.12842295
16.93333333	1.60678297	1.37606431
19.75555556	4.41224414	3.91338449
22.57777778	8.23791408	7.58845246
25.40000000	14.53486721	13.59574866

Z	Qch	Qrad	QCsCond
0.00000000	9.90081543	15.23808053	1.07744326
2.82222222	7.49362446	15.71537048	1.08777951
5.64444444	8.05239930	17.49574903	1.13124448
8.46666667	8.26919362	19.26610657	1.17042804
11.28888889	8.37595830	20.28345587	1.18841945
14.11111111	8.38384050	20.26435874	1.18111695
16.93333333	8.29839525	19.20286184	1.14859681
19.75555556	8.12431126	17.37203651	1.09503360
22.57777778	7.67831344	15.52319973	1.03762210
25.40000000	10.55366016	15.11997007	1.01755208

```
Total computational time required:  1 min.,  0.37 sec.
Time spent in Convect/CoolantTemp:  0 min.,  0.00 sec.  ( 0.0%)
Time spent in CYLCON6:  1 min., 27.10 sec. (84.3%)
Time spent in Gauss:  0 min., 15.33 sec. (14.8%)
1******************* TFETC *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 TIME =      0.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---
```

```
t(  1,  1) =  2207.6106582
t(  1,  2) =  2261.7426054
t(  1,  3) =  2362.6664057
t(  1,  4) =  2449.0229475
t(  1,  5) =  2496.6433830
t(  1,  6) =  2496.9234677
t(  1,  7) =  2449.6863479
t(  1,  8) =  2363.1914258
t(  1,  9) =  2261.8530039
t(  1, 10) =  2221.9998042
t(  2,  1) =  2186.1971281
t(  2,  2) =  2237.1025284
t(  2,  3) =  2333.8873935
t(  2,  4) =  2416.8852771
t(  2,  5) =  2462.5906059
t(  2,  6) =  2462.8674079
t(  2,  7) =  2417.5418559
t(  2,  8) =  2334.4072952
t(  2,  9) =  2237.2029112
t(  2, 10) =  2200.5411066
t(  3,  1) =  2131.8581104
t(  3,  2) =  2174.6690550
t(  3,  3) =  2261.1684885
t(  3,  4) =  2335.9499908
t(  3,  5) =  2377.0411968
t(  3,  6) =  2377.3109644
t(  3,  7) =  2336.5916177
t(  3,  8) =  2261.6765124
t(  3,  9) =  2174.7395541
t(  3, 10) =  2144.1652664
t(  4,  1) =  2049.9178867
t(  4,  2) =  2080.6705441
t(  4,  3) =  2152.0199439
t(  4,  4) =  2214.9465505
t(  4,  5) =  2249.5126048
t(  4,  6) =  2249.7739613
t(  4,  7) =  2215.5694463
t(  4,  8) =  2152.5116847
t(  4,  9) =  2080.6980165
t(  4, 10) =  2058.6768568
t(  5,  1) =  1943.4002334
t(  5,  2) =  1958.6914577
t(  5,  3) =  2010.7078927
t(  5,  4) =  2058.7072181
t(  5,  5) =  2085.1766716
t(  5,  6) =  2085.4286152
t(  5,  7) =  2059.3079608
t(  5,  8) =  2011.1788302
t(  5,  9) =  1958.6693771
t(  5, 10) =  1947.3555271
t(  6,  1) =  1939.5252885
t(  6,  2) =  1954.2654800
t(  6,  3) =  2005.6800497
t(  6,  4) =  2053.2093102
t(  6,  5) =  2079.4213757
t(  6,  6) =  2079.6733825
t(  6,  7) =  2053.8103232
t(  6,  8) =  2006.1510542
t(  6,  9) =  1954.2364049
t(  6, 10) =  1943.3159623
t(  7,  1) =   919.3642134
t(  7,  2) =   923.9127148
t(  7,  3) =   932.3725376
t(  7,  4) =   941.0528808
t(  7,  5) =   949.3930788
t(  7,  6) =   956.9185405
t(  7,  7) =   963.3549215
t(  7,  8) =   968.7053012
t(  7,  9) =   973.3498345
t(  7, 10) =   982.1790161
t(  8,  1) =   915.0268697
t(  8,  2) =   919.9545935
t(  8,  3) =   928.0772959
t(  8,  4) =   936.4712201
t(  8,  5) =   944.6539122
t(  8,  6) =   952.1913380
t(  8,  7) =   958.8076000
t(  8,  8) =   964.4616232
t(  8,  9) =   969.4443033
t(  8, 10) =   977.8286091
t(  9,  1) =   902.6190560
t(  9,  2) =   908.5484778
t(  9,  3) =   915.5611828
t(  9,  4) =   922.9632337
t(  9,  5) =   930.5231087
t(  9,  6) =   937.9496898
```

```
t(  9,  7) =   944.9825023
t(  9,  8) =   951.4578048
t(  9,  9) =   957.3857064
t(  9, 10) =   964.3110205
t( 10,  1) =   900.7157871
t( 10,  2) =   906.8155358
t( 10,  3) =   913.6812466
t( 10,  4) =   920.9576199
t( 10,  5) =   928.4483757
t( 10,  6) =   935.8802369
t( 10,  7) =   942.9920328
t( 10,  8) =   949.6005390
t( 10,  9) =   955.6753043
t( 10, 10) =   962.4140332
tcool(  1) =   895.0000000
tcool(  2) =   901.5697693
tcool(  3) =   908.1042149
tcool(  4) =   915.1116661
tcool(  5) =   922.4532633
tcool(  6) =   929.8935392
tcool(  7) =   937.1691484
tcool(  8) =   944.0572319
tcool(  9) =   950.4384966
tcool( 10) =   956.8961135

Temperature of coolant at core exit:    956.896 degrees K.

Mass flow rate =      0.120
Voltage across bottom of cell:     0.6427047
Voltage across top of cell:     0.6464798
Output current =   490.0000000
Output electrical power =   315.8501975
Total Thermal power =  3177.7050000
```

Z	V	Qec	Jdens
0.00000000	0.64270471	12.99420332	4.81284388
2.82222222	0.73952756	10.19447559	3.65196351
5.64444444	0.81536882	11.26434223	3.93893884
8.46666667	0.86803225	11.79162961	4.05764643
11.28888889	0.89558633	12.06404159	4.11782102
14.11111111	0.89678741	12.08230457	4.12396904
16.93333333	0.87140461	11.85295220	4.07907416
19.75555556	0.82022683	11.39399910	3.98648550
22.57777778	0.74473245	10.47667932	3.75799840
25.40000000	0.64647977	13.89467367	5.16896030

Z	EmHeat	ColHeat
0.00000000	14.49777300	12.78461813
2.82222222	8.68536675	7.62158438
5.64444444	4.81644251	4.12387927
8.46666667	1.87853834	1.57475799
11.28888889	0.24858863	0.20674847
14.11111111	0.15340144	0.12850539
16.93333333	1.60703735	1.37639999
19.75555556	4.41255720	3.91401618
22.57777778	8.23815769	7.58931886
25.40000000	14.53427214	13.59664908

Z	Qch	Qrad	QCsCond
0.00000000	9.90096591	15.23808053	1.07744326
2.82222222	7.49374791	15.71537048	1.08777951
5.64444444	8.05265430	17.49574903	1.13124448
8.46666667	8.26946163	19.26610657	1.17042804
11.28888889	8.37617736	20.28345587	1.18841945
14.11111111	8.38398105	20.26435874	1.18111695
16.93333333	8.29842819	19.20286184	1.14859681
19.75555556	8.12417673	17.37203651	1.09503360
22.57777778	7.67797598	15.52319973	1.03762210
25.40000000	10.55304538	15.11997007	1.01755208

```
1******************** TFETC ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 TIME =     10.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,  1) =  2207.6297829
```

```
t( 1,  2) =  2261.7641721
t( 1,  3) =  2362.6880617
t( 1,  4) =  2449.0488481
t( 1,  5) =  2496.6740644
t( 1,  6) =  2496.9572843
t( 1,  7) =  2449.7220171
t( 1,  8) =  2363.2304574
t( 1,  9) =  2261.9027013
t( 1, 10) =  2222.0225448
t( 2,  1) =  2186.2143660
t( 2,  2) =  2237.1206139
t( 2,  3) =  2333.9039784
t( 2,  4) =  2416.9060295
t( 2,  5) =  2462.6173644
t( 2,  6) =  2462.8999423
t( 2,  7) =  2417.5798142
t( 2,  8) =  2334.4526150
t( 2,  9) =  2237.2644789
t( 2, 10) =  2200.5666713
t( 3,  1) =  2131.8710108
t( 3,  2) =  2174.6781620
t( 3,  3) =  2261.1724539
t( 3,  4) =  2335.9592115
t( 3,  5) =  2377.0612400
t( 3,  6) =  2377.3448834
t( 3,  7) =  2336.6409831
t( 3,  8) =  2261.7445697
t( 3,  9) =  2174.8402423
t( 3, 10) =  2144.2013789
t( 4,  1) =  2049.9254129
t( 4,  2) =  2080.6649284
t( 4,  3) =  2152.0040485
t( 4,  4) =  2214.9398358
t( 4,  5) =  2249.5276965
t( 4,  6) =  2249.8194352
t( 4,  7) =  2215.6499137
t( 4,  8) =  2152.6331210
t( 4,  9) =  2080.8843339
t( 4, 10) =  2058.7403573
t( 5,  1) =  1943.4031565
t( 5,  2) =  1958.6637339
t( 5,  3) =  2010.6631867
t( 5,  4) =  2058.6793966
t( 5,  5) =  2085.1903829
t( 5,  6) =  2085.5023466
t( 5,  7) =  2059.4522923
t( 5,  8) =  2011.4049191
t( 5,  9) =  1959.0177440
t( 5, 10) =  1947.4791499
t( 6,  1) =  1939.5280559
t( 6,  2) =  1954.2365168
t( 6,  3) =  2005.6337608
t( 6,  4) =  2053.1802724
t( 6,  5) =  2079.4348273
t( 6,  6) =  2079.7483491
t( 6,  7) =  2053.9577586
t( 6,  8) =  2006.3824265
t( 6,  9) =  1954.5936576
t( 6, 10) =  1943.4423966
t( 7,  1) =   919.4722115
t( 7,  2) =   924.8298080
t( 7,  3) =   934.3335273
t( 7,  4) =   944.1024127
t( 7,  5) =   953.4961673
t( 7,  6) =   961.9822995
t( 7,  7) =   969.2421922
t( 7,  8) =   975.2606102
t( 7,  9) =   980.4302251
t( 7, 10) =   989.5514930
t( 8,  1) =   915.1332143
t( 8,  2) =   920.8723737
t( 8,  3) =   930.0429403
t( 8,  4) =   939.5302137
t( 8,  5) =   948.7715572
t( 8,  6) =   957.2746355
t( 8,  7) =   964.7189533
t( 8,  8) =   971.0451129
t( 8,  9) =   976.5581917
t( 8, 10) =   985.2621260
t( 9,  1) =   902.7227612
t( 9,  2) =   909.4779268
t( 9,  3) =   917.5496337
t( 9,  4) =   926.0581396
t( 9,  5) =   934.6935765
t( 9,  6) =   943.1080631
t( 9,  7) =   950.9967511
```

```
t(  9,  8) =   958.1765717
t(  9,  9) =   964.6739118
t(  9, 10) =   972.0118532
t( 10,  1) =   900.8194385
t( 10,  2) =   907.7517064
t( 10,  3) =   915.6834329
t( 10,  4) =   924.0739195
t( 10,  5) =   932.6482095
t( 10,  6) =   941.0758875
t( 10,  7) =   949.0510420
t( 10,  8) =   956.3709359
t( 10,  9) =   963.0220593
t( 10, 10) =   970.1901425
tcool(  1) =   895.0000000
tcool(  2) =   902.6347593
tcool(  3) =   910.1883199
tcool(  4) =   918.2672600
tcool(  5) =   926.7017600
tcool(  6) =   935.2141737
tcool(  7) =   943.4954344
tcool(  8) =   951.2829759
tcool(  9) =   958.4410535
tcool( 10) =   965.6293706

Temperature of coolant at core exit:    965.629 degrees K.

Mass flow rate =       0.103
Voltage across bottom of cell:     0.6429185
Voltage across top of cell:     0.6449749
Output current =   490.0000000
Output electrical power =   315.5338830
Total Thermal power =  3177.7050000
```

Z	V	Qec	Jdens
0.00000000	0.64291849	12.98885191	4.81062783
2.82222222	0.73977079	10.20737706	3.65700034
5.64444444	0.81563836	11.28544589	3.94727532
8.46666667	0.86828479	11.81524599	4.06689152
11.28888889	0.89574570	12.08521675	4.12589394
14.11111111	0.89675829	12.09609826	4.12886904
16.93333333	0.87108999	11.85549180	4.07924725
19.75555556	0.81954415	11.38162494	3.98043695
22.57777778	0.74362737	10.43377433	3.73960942
25.40000000	0.64497491	13.85338112	5.14996506

Z	EmHeat	ColHeat
0.00000000	14.49779980	12.78601378
2.82222222	8.68348115	7.62714921
5.64444444	4.80880364	4.12555855
8.46666667	1.86859828	1.57119688
11.28888889	0.24311790	0.20301259
14.11111111	0.15887757	0.13374339
16.93333333	1.62602413	1.40046217
19.75555556	4.44130449	3.96374542
22.57777778	8.26320266	7.66211558
25.40000000	14.53545285	13.69156969

Z	Qch	Qrad	QCsCond
0.00000000	9.89601033	15.23661496	1.07730135
2.82222222	7.50203502	15.71022427	1.08683681
5.64444444	8.06589674	17.48567689	1.12927056
8.46666667	8.28402594	19.25199444	1.16742285
11.28888889	8.38946499	20.26641525	1.18444564
14.11111111	8.39350072	20.24542832	1.17627858
16.93333333	8.30210035	19.18281780	1.14302871
19.75555556	8.11948112	17.35171400	1.08889198
22.57777778	7.65289841	15.50449963	1.03109522
25.40000000	10.53178287	15.09135130	1.01051432

```
1******************** TFETC *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** L O S S   O F   F L O W   P R O B L E M   ***
 TIME =      20.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,  1) =  2207.6383214
 t(  1,  2) =  2261.7421265
```

```
t(  1,   3) =  2362.6588720
t(  1,   4) =  2449.0365761
t(  1,   5) =  2496.6933684
t(  1,   6) =  2497.0195192
t(  1,   7) =  2449.8388375
t(  1,   8) =  2363.4213240
t(  1,   9) =  2262.2112551
t(  1,  10) =  2222.1673669
t(  2,   1) =  2186.2213254
t(  2,   2) =  2237.0905596
t(  2,   3) =  2333.8657641
t(  2,   4) =  2416.8866620
t(  2,   5) =  2462.6346318
t(  2,   6) =  2462.9684996
t(  2,   7) =  2417.7140760
t(  2,   8) =  2334.6743344
t(  2,   9) =  2237.6209227
t(  2,  10) =  2200.7382163
t(  3,   1) =  2131.8735725
t(  3,   2) =  2174.6269389
t(  3,   3) =  2261.1107292
t(  3,   4) =  2335.9213815
t(  3,   5) =  2377.0729640
t(  3,   6) =  2377.4292534
t(  3,   7) =  2336.8197844
t(  3,   8) =  2262.0456400
t(  3,   9) =  2175.3198330
t(  3,  10) =  2144.4446130
t(  4,   1) =  2049.9199520
t(  4,   2) =  2080.5796496
t(  4,   3) =  2151.9052960
t(  4,   4) =  2214.8728090
t(  4,   5) =  2249.5297206
t(  4,   6) =  2249.9264560
t(  4,   7) =  2215.8956128
t(  4,   8) =  2153.0549764
t(  4,   9) =  2081.5510899
t(  4,  10) =  2059.1017282
t(  5,   1) =  1943.3842160
t(  5,   2) =  1958.5313956
t(  5,   3) =  2010.5144355
t(  5,   4) =  2058.5724956
t(  5,   5) =  2085.1771698
t(  5,   6) =  2085.6352703
t(  5,   7) =  2059.7811099
t(  5,   8) =  2011.9798525
t(  5,   9) =  1959.9215982
t(  5,  10) =  1948.0077430
t(  6,   1) =  1939.5087211
t(  6,   2) =  1954.1021747
t(  6,   3) =  2005.4829891
t(  6,   4) =  2053.0717478
t(  6,   5) =  2079.4208803
t(  6,   6) =  2079.8820188
t(  6,   7) =  2054.2894057
t(  6,   8) =  2006.9627433
t(  6,   9) =  1955.5069482
t(  6,  10) =  1943.9769048
t(  7,   1) =   919.5871032
t(  7,   2) =   925.9585206
t(  7,   3) =   936.5041239
t(  7,   4) =   947.3822243
t(  7,   5) =   957.9259656
t(  7,   6) =   967.5616307
t(  7,   7) =   975.9265021
t(  7,   8) =   982.9699304
t(  7,   9) =   989.0588224
t(  7,  10) =   999.0991141
t(  8,   1) =   915.2463400
t(  8,   2) =   922.0004452
t(  8,   3) =   932.2143452
t(  8,   4) =   942.8125886
t(  8,   5) =   953.2057394
t(  8,   6) =   962.8600849
t(  8,   7) =   971.4109915
t(  8,   8) =   978.7637224
t(  8,   9) =   985.1993488
t(  8,  10) =   994.8282434
t(  9,   1) =   902.8243304
t(  9,   2) =   910.5826400
t(  9,   3) =   919.6798213
t(  9,   4) =   929.2800708
t(  9,   5) =   939.0487149
t(  9,   6) =   948.5994633
t(  9,   7) =   957.5858610
t(  9,   8) =   965.7904454
```

```
t(  9,  9) =   973.2189328
t(  9, 10) =   981.4816996
t( 10,  1) =   900.9191634
t( 10,  2) =   908.8556674
t( 10,  3) =   917.8130995
t( 10,  4) =   927.2957086
t( 10,  5) =   937.0036164
t( 10,  6) =   946.5679489
t( 10,  7) =   955.6411876
t( 10,  8) =   963.9862458
t( 10,  9) =   971.5696672
t( 10, 10) =   979.6660120
tcool(  1) =   895.0000000
tcool(  2) =   903.6649141
tcool(  3) =   912.2331384
tcool(  4) =   921.3938151
tcool(  5) =   930.9535666
tcool(  6) =   940.5974077
tcool(  7) =   949.9752122
tcool(  8) =   958.7897911
tcool(  9) =   966.8859962
tcool( 10) =   975.0037525

Temperature of coolant at core exit:    975.004 degrees K.

Mass flow rate =      0.091
Voltage across bottom of cell:     0.6427620
Voltage across top of cell:    0.6422057
Output current =   490.0000000
Output electrical power =   314.8171011
Total Thermal power =  3177.7050000
```

Z	V	Qec	Jdens
0.00000000	0.64276201	12.99809452	4.81446777
2.82222222	0.73963081	10.23212465	3.66692499
5.64444444	0.81548281	11.31532518	3.95931168
8.46666667	0.86804489	11.84626533	4.07923207
11.28888889	0.89532345	12.11261159	4.13647415
14.11111111	0.89603864	12.11412424	4.13529262
16.93333333	0.86995761	11.85830471	4.07908370
19.75555556	0.81790157	11.36436099	3.97157466
22.57777778	0.74140948	10.37823873	3.71464933
25.40000000	0.64220575	13.77511295	5.11377653

Z	EmHeat	ColHeat
0.00000000	14.49761917	12.78749844
2.82222222	8.67431180	7.62844859
5.64444444	4.79182512	4.12030308
8.46666667	1.85091109	1.56147227
11.28888889	0.23435304	0.19654217
14.11111111	0.16762173	0.14185401
16.93333333	1.65554048	1.43472696
19.75555556	4.48632671	4.03182268
22.57777778	8.30599548	7.76005390
25.40000000	14.54047118	13.81384544

Z	Qch	Qrad	QCsCond
0.00000000	9.90353754	15.23565582	1.07716729
2.82222222	7.51995396	15.70179082	1.08555797
5.64444444	8.08657457	17.47227288	1.12692305
8.46666667	8.30530880	19.23527965	1.16400256
11.28888889	8.40912926	20.24834296	1.17996203
14.11111111	8.40874226	20.22804197	1.17077627
16.93333333	8.30967482	19.16764175	1.13658820
19.75555556	8.11600383	17.33960623	1.08162785
22.57777778	7.62416252	15.49730457	1.02319164
25.40000000	10.49101626	15.06581421	1.00127424

```
1******************** TFETC ********************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
*** S H U T  D O W N  P R O B L E M ***
 *****                  *****
 *****      Shutdown Problem *****
            Linear Equations solved using Gaussian elimination
            Simulation Period, TIME =    6.00 Secs.
            Print Time Step, TPRINT =    3.00 Secs.
            Time Step Increment, delta t =     0.5000 Secs.
            Print Option, ipout = 1.
            Total delayed neutron fraction, BETA =   0.00650 .
            Negative Reactivity insertion in $, dollar =   -3.00.
            Reactor period, T =   80.00 Secs.
            P(t) = P(0) [(1 - BETA * RHO)/(1 - RHO)] * Exp(- t / Period)

 COOLANT TYPE: Molten Sodium-Potassium Alloy (NaK)
               Potassium composition = 78%
 COOLANT MASS FLOW RATE:   0.12 kilograms per second.
 TEMPERATURE OF COOLANT AT CHANNEL INLET:  895.0 K.

 TEMPERATURE OF CESIUM RESERVOIR:  620.0 K.
 PRESSURE OF CESIUM VAPOR:    5.6 Torr.

 EFFECTIVE EMISSIVITY FOR RADIANT HEAT TRANSFER FROM
            THE EMITTER SURFACE TO THE COLLECTOR SURFACE:  0.200000

 OUTPUT CURRENT FROM THE TOP OF THE TFE:    245.0 Amperes.
 OUTPUT CURRENT FROM THE BOTTOM OF THE TFE:  245.0 Amperes.

 TOTAL THERMAL POWER PRODUCED IN THE TFE FUEL: 3177.7 Watts.
 AVERAGE VOLUMETRIC HEAT GENERATION RATE FOR THE TFE FUEL:  118.0 Watts.
 CORRELATION FOR THE RATIO OF THE HEAT GENERATION RATE
            AT POSITION Z TO THE AVERAGE HEAT GENERATION RATE IN
            THE TFE FUEL:

        F =     0.7300+    0.4250 * SIN((Z-Zmin)/(Zmax-Zmin)*3.14159)

 AXIAL PEAK-TO-AVERAGE RATIO FOR HEAT GENERATION IS: 1.1543

1******************** TFETC ********************
 ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***

 ***** GEOMETRY DATA EDIT *****
  ***** RADIAL GEOMETRY *****
```

Region	Inside Radius (cm)	Outside Radius (cm)	Material	Number of Interior Mesh Points
fuel	0.150000	0.600000	uo2	3
emitter	0.600000	0.750000	w	0
emitter-collector gap	0.750000	0.800000	cs	0
collector	0.800000	0.900000	nb	0
insulator	0.900000	0.950000	al2o3	0
cladding	0.950000	1.000000	nb	0
coolant channel	1.000000	1.250000		

```
  ***** AXIAL GEOMETRY *****

 AXIAL POSITION OF THE UPPER LIMIT FOR
          THE FUELED REGION OF THE TFE:  0.000000 (cm)
 AXIAL POSITION OF THE LOWER LIMIT FOR
          THE FUELED REGION OF THE TFE: 25.400000 (cm)
 AXIAL EXTENT OF THE FUELED REGION OF THE TFE: 25.400000 (cm)
 TOTAL LENGTH OF THE TFE, INCLUDING ELECTRODE LEADS: 25.400000 (cm)

 Temperature of coolant at core exit:      911.2 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                     Converging the RMS error to less than 0.1 K.
 Iteration :  1  RMS error =   352.3744542  Ave Diff. =   244.0701024
                      Max. Error =  836.2653934

 Temperature of coolant at core exit:      935.1 degrees K.
1******************** TFEHX ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                     Converging the RMS error to less than 0.1 K.
 Iteration :  2  RMS error =   120.6911636  Ave Diff. =    82.8249886
                      Max. Error =  262.7163942

 Temperature of coolant at core exit:      946.3 degrees K.
```

```
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  3  RMS error =   48.0790538  Ave Diff. =    28.9756862
                    Max. Error =  124.2231601

 Temperature of coolant at core exit:      951.7 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  4  RMS error =   22.1990022  Ave Diff. =    15.0890168
                    Max. Error =   47.7020788

 Temperature of coolant at core exit:      954.4 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  5  RMS error =   10.5703466  Ave Diff. =     6.1852238
                    Max. Error =   24.5401210

 Temperature of coolant at core exit:      955.7 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  6  RMS error =    5.3631696  Ave Diff. =     2.8889611
                    Max. Error =   10.0539319

 Temperature of coolant at core exit:      956.3 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  7  RMS error =    2.7477195  Ave Diff. =     1.1581796
                    Max. Error =    5.1710185

 Temperature of coolant at core exit:      956.6 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  8  RMS error =    1.4849013  Ave Diff. =     0.4761796
                    Max. Error =    2.9321308

 Temperature of coolant at core exit:      956.8 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration :  9  RMS error =    0.8003439  Ave Diff. =     0.1667677
                    Max. Error =    1.6387800

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration : 10  RMS error =    0.4468618  Ave Diff. =     0.0509478
                    Max. Error =    0.9027483

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
 Iteration : 11  RMS error =    0.2453117  Ave Diff. =     0.0066867
                    Max. Error =    0.4927094

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N  P R O B L E M ***
 ITERATION HISTORY --
                   Converging the RMS error to less than 0.1 K.
```

```
 Iteration : 12  RMS error =     0.1374013  Ave Diff. =    -0.0057550
                       Max. Error =    0.2662746

 Temperature of coolant at core exit:      956.9 degrees K.
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
***  S H U T D O W N   P R O B L E M ***
 ITERATION HISTORY --
                       Converging the RMS error to less than 0.1 K.
 Iteration : 13  RMS error =     0.0754500  Ave Diff. =    -0.0080428
                       Max. Error =    0.1428042
1******************** TFEHX *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
***  S H U T D O W N   P R O B L E M ***
 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 T(  1,  1) =  2207.6099799
 T(  1,  2) =  2261.7498155
 T(  1,  3) =  2362.6692432
 T(  1,  4) =  2449.0244839
 T(  1,  5) =  2496.6438504
 T(  1,  6) =  2496.9186250
 T(  1,  7) =  2449.6735003
 T(  1,  8) =  2363.1763020
 T(  1,  9) =  2261.8518754
 T(  1, 10) =  2221.9650827
 T(  2,  1) =  2186.1929268
 T(  2,  2) =  2237.1053504
 T(  2,  3) =  2333.8847952
 T(  2,  4) =  2416.8802095
 T(  2,  5) =  2462.5836797
 T(  2,  6) =  2462.8553788
 T(  2,  7) =  2417.5228984
 T(  2,  8) =  2334.3872511
 T(  2,  9) =  2237.1977085
 T(  2, 10) =  2200.5034192
 T(  3,  1) =  2131.8455701
 T(  3,  2) =  2174.6616482
 T(  3,  3) =  2261.1536691
 T(  3,  4) =  2335.9307086
 T(  3,  5) =  2377.0188307
 T(  3,  6) =  2377.2839338
 T(  3,  7) =  2336.5595171
 T(  3,  8) =  2261.6454175
 T(  3,  9) =  2174.7248924
 T(  3, 10) =  2144.1203825
 T(  4,  1) =  2049.8943113
 T(  4,  2) =  2080.6498301
 T(  4,  3) =  2151.9899401
 T(  4,  4) =  2214.9105351
 T(  4,  5) =  2249.4727497
 T(  4,  6) =  2249.7299490
 T(  4,  7) =  2215.5219128
 T(  4,  8) =  2152.4669546
 T(  4,  9) =  2080.6711635
 T(  4, 10) =  2058.6227204
 T(  5,  1) =  1943.3652400
 T(  5,  2) =  1958.6571780
 T(  5,  3) =  2010.6630404
 T(  5,  4) =  2058.6554004
 T(  5,  5) =  2085.1206973
 T(  5,  6) =  2085.3690260
 T(  5,  7) =  2059.2461008
 T(  5,  8) =  2011.1211889
 T(  5,  9) =  1958.6304420
 T(  5, 10) =  1947.2927154
 T(  6,  1) =  1939.4898721
 T(  6,  2) =  1954.2307309
 T(  6,  3) =  2005.6346468
 T(  6,  4) =  2053.1568933
 T(  6,  5) =  2079.3647793
 T(  6,  6) =  2079.6131762
 T(  6,  7) =  2053.7478765
 T(  6,  8) =  2006.0928726
 T(  6,  9) =  1954.1970618
 T(  6, 10) =  1943.2526925
 T(  7,  1) =   919.3693636
 T(  7,  2) =   923.9247743
 T(  7,  3) =   932.3917029
 T(  7,  4) =   941.0796696
 T(  7,  5) =   949.4278157
 T(  7,  6) =   956.9612147
 T(  7,  7) =   963.4052389
 T(  7,  8) =   968.7628206
 T(  7,  9) =   973.4131859
 T(  7, 10) =   982.2504182
```

```
T(  8,  1) =   915.0295911
T(  8,  2) =   919.9643110
T(  8,  3) =   928.0941405
T(  8,  4) =   936.4957394
T(  8,  5) =   944.6864194
T(  8,  6) =   952.2318028
T(  8,  7) =   958.8557014
T(  8,  8) =   964.5168861
T(  8,  9) =   969.5054473
T(  8, 10) =   977.8976175
T(  9,  1) =   902.6200325
T(  9,  2) =   908.5560904
T(  9,  3) =   915.5751364
T(  9,  4) =   922.9840447
T(  9,  5) =   930.5510512
T(  9,  6) =   937.9848177
T(  9,  7) =   945.0246739
T(  9,  8) =   951.5067375
T(  9,  9) =   957.4406258
T(  9, 10) =   964.3720872
T( 10,  1) =   900.7165965
T( 10,  2) =   906.8230273
T( 10,  3) =   913.6950848
T( 10,  4) =   920.9783348
T( 10,  5) =   928.4762371
T( 10,  6) =   935.9152912
T( 10,  7) =   943.0341280
T( 10,  8) =   949.6493791
T( 10,  9) =   955.7301540
T( 10, 10) =   962.4749495
Voltage across bottom of cell:     0.6427667
Voltage across top of cell:     0.6465651
Output current =   490.0000000
Output electrical power =   315.8862832
```

Z	V	Qec	Jdens
0.00000000	0.64276668	12.99430163	4.81276691
2.82222222	0.73959046	10.19453024	3.65189375
5.64444444	0.81543279	11.26422623	3.93880029
8.46666667	0.86809799	11.79150175	4.05750063
11.28888889	0.89565457	12.06399812	4.11770336
14.11111111	0.89685880	12.08239618	4.12389963
16.93333333	0.87147959	11.85322300	4.07907171
19.75555556	0.82030557	11.39452654	3.98658183
22.57777778	0.74481487	10.47748564	3.75821204
25.40000000	0.64656509	13.89597476	5.16933972

Z	EmHeat	ColHeat
0.00000000	14.49810519	12.78456733
2.82222222	8.68565016	7.62157940
5.64444444	4.81674372	4.12394584
8.46666667	1.87878625	1.57487539
11.28888889	0.24869437	0.20682254
14.11111111	0.15331479	0.12842295
16.93333333	1.60678297	1.37606431
19.75555556	4.41224414	3.91338449
22.57777778	8.23791408	7.58845246
25.40000000	14.53486721	13.59574866

Z	Qch	Qrad	QCsCond
0.00000000	9.90081543	15.23808053	1.07744326
2.82222222	7.49362446	15.71537048	1.08777951
5.64444444	8.05239930	17.49574903	1.13124448
8.46666667	8.26919362	19.26610657	1.17042804
11.28888889	8.37595830	20.28345587	1.18841945
14.11111111	8.38384050	20.26435874	1.18111695
16.93333333	8.29839525	19.20286184	1.14859681
19.75555556	8.12431126	17.37203651	1.09503360
22.57777778	7.67831344	15.52319973	1.03762210
25.40000000	10.55366016	15.11997007	1.01755208

```
Total computational time required:  1 min.,  0.47 sec.
Time spent in Convect/CoolantTemp:  0 min.,  0.00 sec. ( 0.0%)
Time spent in CYLCON6:  1 min., 27.17 sec. (84.2%)
Time spent in Gauss:  0 min., 15.20 sec. (14.7%)
1******************* TFETC *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N   P R O B L E M ***
```

```
TIME =      0.00000000

TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

t(  1,   1) =  2207.6106582
t(  1,   2) =  2261.7426054
t(  1,   3) =  2362.6664057
t(  1,   4) =  2449.0229475
t(  1,   5) =  2496.6433830
t(  1,   6) =  2496.9234677
t(  1,   7) =  2449.6863479
t(  1,   8) =  2363.1914258
t(  1,   9) =  2261.8530039
t(  1,  10) =  2221.9998042
t(  2,   1) =  2186.1971281
t(  2,   2) =  2237.1025284
t(  2,   3) =  2333.8873935
t(  2,   4) =  2416.8852771
t(  2,   5) =  2462.5906059
t(  2,   6) =  2462.8674079
t(  2,   7) =  2417.5418559
t(  2,   8) =  2334.4072952
t(  2,   9) =  2237.2029112
t(  2,  10) =  2200.5411066
t(  3,   1) =  2131.8581104
t(  3,   2) =  2174.6690550
t(  3,   3) =  2261.1684885
t(  3,   4) =  2335.9499908
t(  3,   5) =  2377.0411968
t(  3,   6) =  2377.3109644
t(  3,   7) =  2336.5916177
t(  3,   8) =  2261.6765124
t(  3,   9) =  2174.7395541
t(  3,  10) =  2144.1652664
t(  4,   1) =  2049.9178867
t(  4,   2) =  2080.6705441
t(  4,   3) =  2152.0199439
t(  4,   4) =  2214.9465505
t(  4,   5) =  2249.5126048
t(  4,   6) =  2249.7739613
t(  4,   7) =  2215.5694463
t(  4,   8) =  2152.5116847
t(  4,   9) =  2080.6980165
t(  4,  10) =  2058.6768568
t(  5,   1) =  1943.4002334
t(  5,   2) =  1958.6914577
t(  5,   3) =  2010.7078927
t(  5,   4) =  2058.7072181
t(  5,   5) =  2085.1766716
t(  5,   6) =  2085.4286152
t(  5,   7) =  2059.3079608
t(  5,   8) =  2011.1788302
t(  5,   9) =  1958.6693771
t(  5,  10) =  1947.3555271
t(  6,   1) =  1939.5252885
t(  6,   2) =  1954.2654800
t(  6,   3) =  2005.6800497
t(  6,   4) =  2053.2093102
t(  6,   5) =  2079.4213757
t(  6,   6) =  2079.6733825
t(  6,   7) =  2053.8103232
t(  6,   8) =  2006.1510542
t(  6,   9) =  1954.2364049
t(  6,  10) =  1943.3159623
t(  7,   1) =   919.3642134
t(  7,   2) =   923.9127148
t(  7,   3) =   932.3725376
t(  7,   4) =   941.0528808
t(  7,   5) =   949.3930788
t(  7,   6) =   956.9185405
t(  7,   7) =   963.3549215
t(  7,   8) =   968.7053012
t(  7,   9) =   973.3498345
t(  7,  10) =   982.1790161
t(  8,   1) =   915.0268697
t(  8,   2) =   919.9545935
t(  8,   3) =   928.0772959
t(  8,   4) =   936.4712201
t(  8,   5) =   944.6539122
t(  8,   6) =   952.1913380
t(  8,   7) =   958.8076000
t(  8,   8) =   964.4616232
t(  8,   9) =   969.4443033
t(  8,  10) =   977.8286091
t(  9,   1) =   902.6190560
t(  9,   2) =   908.5484778
```

```
t(  9,  3) =   915.5611828
t(  9,  4) =   922.9632337
t(  9,  5) =   930.5231087
t(  9,  6) =   937.9496898
t(  9,  7) =   944.9825023
t(  9,  8) =   951.4578048
t(  9,  9) =   957.3857064
t(  9, 10) =   964.3110205
t( 10,  1) =   900.7157871
t( 10,  2) =   906.8155358
t( 10,  3) =   913.6812466
t( 10,  4) =   920.9576199
t( 10,  5) =   928.4483757
t( 10,  6) =   935.8802369
t( 10,  7) =   942.9920328
t( 10,  8) =   949.6005390
t( 10,  9) =   955.6753043
t( 10, 10) =   962.4140332
tcool(  1) =   895.0000000
tcool(  2) =   901.5697693
tcool(  3) =   908.1042149
tcool(  4) =   915.1116661
tcool(  5) =   922.4532633
tcool(  6) =   929.8935392
tcool(  7) =   937.1691484
tcool(  8) =   944.0572319
tcool(  9) =   950.4384966
tcool( 10) =   956.8961135

Temperature of coolant at core exit:    956.896 degrees K.
Voltage across bottom of cell:     0.6427047
Voltage across top of cell:     0.6464798
Output current =   490.0000000
Output electrical power =   315.8501975
Total Thermal power =  3177.7050000
```

Z	V	Qec	Jdens
0.00000000	0.64270471	12.99420332	4.81284388
2.82222222	0.73952756	10.19447559	3.65196351
5.64444444	0.81536882	11.26434223	3.93893884
8.46666667	0.86803225	11.79162961	4.05764643
11.28888889	0.89558633	12.06404159	4.11782102
14.11111111	0.89678741	12.08230457	4.12396904
16.93333333	0.87140461	11.85295220	4.07907416
19.75555556	0.82022683	11.39399910	3.98648550
22.57777778	0.74473245	10.47667932	3.75799840
25.40000000	0.64647977	13.89467367	5.16896030

Z	EmHeat	ColHeat
0.00000000	14.49777300	12.78461813
2.82222222	8.68536675	7.62158438
5.64444444	4.81644251	4.12387927
8.46666667	1.87853834	1.57475799
11.28888889	0.24858863	0.20674847
14.11111111	0.15340144	0.12850539
16.93333333	1.60703735	1.37639999
19.75555556	4.41255720	3.91401618
22.57777778	8.23815769	7.58931886
25.40000000	14.53427214	13.59664908

Z	Qch	Qrad	QCsCond
0.00000000	9.90096591	15.23808053	1.07744326
2.82222222	7.49374791	15.71537048	1.08777951
5.64444444	8.05265430	17.49574903	1.13124448
8.46666667	8.26946163	19.26610657	1.17042804
11.28888889	8.37617736	20.28345587	1.18841945
14.11111111	8.38398105	20.26435874	1.18111695
16.93333333	8.29842819	19.20286184	1.14859681
19.75555556	8.12417673	17.37203651	1.09503360
22.57777778	7.67797598	15.52319973	1.03762210
25.40000000	10.55304538	15.11997007	1.01755208

```
1******************* TFETC *******************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N   P R O B L E M ***
 TIME =      3.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---
```

```
t(  1,   1) =  2175.2091168
t(  1,   2) =  2224.4941554
t(  1,   3) =  2321.5539056
t(  1,   4) =  2406.2573295
t(  1,   5) =  2453.6435868
t(  1,   6) =  2454.5622991
t(  1,   7) =  2408.6250986
t(  1,   8) =  2324.1860239
t(  1,   9) =  2226.0284061
t(  1,  10) =  2188.8984176
t(  2,   1) =  2153.9197566
t(  2,   2) =  2199.6888875
t(  2,   3) =  2292.4566360
t(  2,   4) =  2373.6281585
t(  2,   5) =  2418.9759244
t(  2,   6) =  2419.8541150
t(  2,   7) =  2375.8863134
t(  2,   8) =  2294.9454717
t(  2,   9) =  2201.1036342
t(  2,  10) =  2167.0138313
t(  3,   1) =  2100.8278294
t(  3,   2) =  2137.3601775
t(  3,   3) =  2219.5099990
t(  3,   4) =  2292.0662889
t(  3,   5) =  2332.5036905
t(  3,   6) =  2333.2754037
t(  3,   7) =  2294.0380708
t(  3,   8) =  2221.6295496
t(  3,   9) =  2138.4733762
t(  3,  10) =  2111.7750324
t(  4,   1) =  2024.4381794
t(  4,   2) =  2045.6193771
t(  4,   3) =  2112.3681245
t(  4,   4) =  2172.6792992
t(  4,   5) =  2206.2624914
t(  4,   6) =  2206.8712683
t(  4,   7) =  2174.2175042
t(  4,   8) =  2113.9377039
t(  4,   9) =  2046.3165153
t(  4,  10) =  2032.0477657
t(  5,   1) =  1934.5739920
t(  5,   2) =  1932.1721604
t(  5,   3) =  1979.9686793
t(  5,   4) =  2025.4903420
t(  5,   5) =  2050.9792358
t(  5,   6) =  2051.3987602
t(  5,   7) =  2026.5290091
t(  5,   8) =  1980.8912098
t(  5,   9) =  1932.4543528
t(  5,  10) =  1938.2045205
t(  6,   1) =  1931.9105523
t(  6,   2) =  1928.3942892
t(  6,   3) =  1975.6567174
t(  6,   4) =  2020.7490227
t(  6,   5) =  2046.0120942
t(  6,   6) =  2046.4283126
t(  6,   7) =  2021.7787669
t(  6,   8) =  1976.5621809
t(  6,   9) =  1928.6648864
t(  6,  10) =  1935.4357084
t(  7,   1) =   919.6834321
t(  7,   2) =   922.9062992
t(  7,   3) =   931.3488388
t(  7,   4) =   940.0559262
t(  7,   5) =   948.3270923
t(  7,   6) =   955.7868239
t(  7,   7) =   962.1640739
t(  7,   8) =   967.3831763
t(  7,   9) =   971.8382477
t(  7,  10) =   982.1357663
t(  8,   1) =   915.2734764
t(  8,   2) =   919.1136182
t(  8,   3) =   927.2221537
t(  8,   4) =   935.6381489
t(  8,   5) =   943.7602506
t(  8,   6) =   951.2288734
t(  8,   7) =   957.7717363
t(  8,   8) =   963.2889378
t(  8,   9) =   968.0846745
t(  8,  10) =   977.7028604
t(  9,   1) =   902.6938868
t(  9,   2) =   908.1276375
t(  9,   3) =   915.1181665
t(  9,   4) =   922.5250416
t(  9,   5) =   930.0375575
t(  9,   6) =   937.3797742
```

```
t(  9,  7) =   944.2951952
t(  9,  8) =   950.6094306
t(  9,  9) =   956.3519517
t(  9, 10) =   963.8994377
t( 10,  1) =   900.7660353
t( 10,  2) =   906.4540152
t( 10,  3) =   913.2935346
t( 10,  4) =   920.5702153
t( 10,  5) =   928.0133501
t( 10,  6) =   935.3566712
t( 10,  7) =   942.3432381
t( 10,  8) =   948.7856105
t( 10,  9) =   954.6735845
t( 10, 10) =   961.9558884
tcool(  1) =   895.0000000
tcool(  2) =   901.5016237
tcool(  3) =   907.8243883
tcool(  4) =   914.6487377
tcool(  5) =   921.8235578
tcool(  6) =   929.1103884
tcool(  7) =   936.2558890
tcool(  8) =   943.0329490
tcool(  9) =   949.3114083
tcool( 10) =   955.8092844

Temperature of coolant at core exit:    955.809 degrees K.
Voltage across bottom of cell:     0.6054920
Voltage across top of cell:     0.6098648
Output current =   490.0000000
Output electrical power =   297.7624247
Total Thermal power =   780.1080744

          Z                     V                  Qec                 Jdens
---------------      ---------------      ---------------      ---------------

     0.00000000           0.60549203          13.97631552           5.24725793
     2.82222222           0.70021916           9.58456395           3.47982406
     5.64444444           0.77484743          10.77465962           3.81891060
     8.46666667           0.82722199          11.66113025           4.06688624
    11.28888889           0.85470084          11.94602880           4.13241495
    14.11111111           0.85603961          11.97170440           4.14127668
    16.93333333           0.83098788          11.74628298           4.09712193
    19.75555556           0.78029598          11.01423062           3.90594247
    22.57777778           0.70598699           9.86219958           3.58501651
    25.40000000           0.60986480          14.83566770           5.59654031

          Z                EmHeat              ColHeat
---------------      ---------------      ---------------

     0.00000000          14.42058898          12.78828925
     2.82222222           8.38173326           7.47437845
     5.64444444           4.73827708           4.13186324
     8.46666667           1.88148833           1.60812993
    11.28888889           0.25599615           0.21713951
    14.11111111           0.14332497           0.12243004
    16.93333333           1.56198780           1.36341936
    19.75555556           4.28108126           3.86541502
    22.57777778           7.88786912           7.37976079
    25.40000000          14.45428255          13.59554922

          Z                   Qch                 Qrad              QCsCond
---------------      ---------------      ---------------      ---------------

     0.00000000          10.79914265          15.04593000           1.07132425
     2.82222222           7.14792448          15.02346659           1.06741982
     5.64444444           7.81558657          16.63043851           1.10755816
     8.46666667           8.29691251          18.26547716           1.14485593
    11.28888889           8.41405029          19.21336909           1.16211976
    14.11111111           8.42660754          19.19909208           1.15496288
    16.93333333           8.34162432          18.21457766           1.12341254
    19.75555556           7.96643941          16.51971006           1.07178535
    22.57777778           7.33122457          14.84062511           1.01767785
    25.40000000          11.42253475          14.91919756           1.01129477
1******************** TFETC ********************
 ***** RESULTS FOR THE FOLLOWING CASE:
*** S H U T D O W N   P R O B L E M ***
 TIME =      6.00000000

 TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---

 t(  1,  1) =  2137.1537602
 t(  1,  2) =  2178.7763202
 t(  1,  3) =  2270.3424920
```

```
t(  1,   4) =  2352.2058158
t(  1,   5) =  2398.8105113
t(  1,   6) =  2400.4207526
t(  1,   7) =  2356.3709744
t(  1,   8) =  2275.0686959
t(  1,   9) =  2181.5979032
t(  1,  10) =  2149.4854712
t(  2,   1) =  2117.4182109
t(  2,   2) =  2154.7121441
t(  2,   3) =  2241.9155529
t(  2,   4) =  2320.0969459
t(  2,   5) =  2364.5500674
t(  2,   6) =  2366.0542532
t(  2,   7) =  2323.9814644
t(  2,   8) =  2246.2862579
t(  2,   9) =  2157.2737901
t(  2,  10) =  2129.0996137
t(  3,   1) =  2069.0376110
t(  3,   2) =  2094.8966270
t(  3,   3) =  2171.4240386
t(  3,   4) =  2240.6737363
t(  3,   5) =  2280.0103436
t(  3,   6) =  2281.2625374
t(  3,   7) =  2243.8909742
t(  3,   8) =  2174.9459411
t(  3,   9) =  2096.8532271
t(  3,  10) =  2078.6792651
t(  4,   1) =  2000.9933116
t(  4,   2) =  2008.3847474
t(  4,   3) =  2069.7767331
t(  4,   4) =  2126.4120062
t(  4,   5) =  2158.7897204
t(  4,   6) =  2159.7142171
t(  4,   7) =  2128.7577585
t(  4,   8) =  2072.1758999
t(  4,   9) =  2009.5969578
t(  4,  10) =  2007.6816471
t(  5,   1) =  1921.0854419
t(  5,   2) =  1902.9221561
t(  5,   3) =  1946.3163071
t(  5,   4) =  1987.5966110
t(  5,   5) =  2011.9647916
t(  5,   6) =  2012.5569247
t(  5,   7) =  1989.0427574
t(  5,   8) =  1947.5315793
t(  5,   9) =  1903.4461349
t(  5,  10) =  1924.3583482
t(  6,   1) =  1918.6167527
t(  6,   2) =  1899.3746759
t(  6,   3) =  1942.3047530
t(  6,   4) =  1983.1087120
t(  6,   5) =  2007.2524160
t(  6,   6) =  2007.8376664
t(  6,   7) =  1984.5345570
t(  6,   8) =  1943.4912502
t(  6,   9) =  1899.8799782
t(  6,  10) =  1921.7948770
t(  7,   1) =   920.2385783
t(  7,   2) =   921.7163918
t(  7,   3) =   929.4294905
t(  7,   4) =   938.0418324
t(  7,   5) =   945.9787956
t(  7,   6) =   953.1670869
t(  7,   7) =   959.3564472
t(  7,   8) =   964.0893700
t(  7,   9) =   968.6705203
t(  7,  10) =   980.8985011
t(  8,   1) =   915.7287224
t(  8,   2) =   918.1148562
t(  8,   3) =   925.5615558
t(  8,   4) =   933.8362900
t(  8,   5) =   941.6269111
t(  8,   6) =   948.8196982
t(  8,   7) =   955.1623723
t(  8,   8) =   960.2351176
t(  8,   9) =   965.0904933
t(  8,  10) =   976.3451849
t(  9,   1) =   902.8580353
t(  9,   2) =   907.7025904
t(  9,   3) =   914.2278019
t(  9,   4) =   921.3640554
t(  9,   5) =   928.5614262
t(  9,   6) =   935.6157394
t(  9,   7) =   942.2880032
t(  9,   8) =   948.2706591
t(  9,   9) =   953.8761746
```

```
t(  9, 10) =   962.1157834
t( 10,  1) =   900.8866062
t( 10,  2) =   906.1150878
t( 10,  3) =   912.5154417
t( 10,  4) =   919.4995424
t( 10,  5) =   926.6273554
t( 10,  6) =   933.6785544
t( 10,  7) =   940.4136359
t( 10,  8) =   946.5380125
t( 10,  9) =   952.2604084
t( 10, 10) =   960.1015218
tcool(  1) =   895.0000000
tcool(  2) =   901.4203347
tcool(  3) =   907.3837013
tcool(  4) =   913.8438917
tcool(  5) =   920.6952541
tcool(  6) =   927.6682153
tcool(  7) =   934.5241628
tcool(  8) =   941.0033013
tcool(  9) =   947.0140176
tcool( 10) =   953.5389272

Temperature of coolant at core exit:    953.539 degrees K.
Voltage across bottom of cell:     0.5582231
Voltage across top of cell:     0.5635919
Output current =   490.0000000
Output electrical power =   274.8446806
Total Thermal power =   751.3957425
```

Z	V	Qec	Jdens
0.00000000	0.55822307	14.99565931	5.72925528
2.82222222	0.65062244	9.13030960	3.37249491
5.64444444	0.72343508	9.97983511	3.59903814
8.46666667	0.77550410	11.44195623	4.05955636
11.28888889	0.80303253	11.85489062	4.17155476
14.11111111	0.80453951	11.89502216	4.18592555
16.93333333	0.77974177	11.59297690	4.11436851
19.75555556	0.72952052	10.23862522	3.69467618
22.57777778	0.65720277	9.42946520	3.48799570
25.40000000	0.56359195	15.87532742	6.09624958

Z	EmHeat	ColHeat
0.00000000	14.29456808	12.79484983
2.82222222	8.00458752	7.26845157
5.64444444	4.61603461	4.10566820
8.46666667	1.88868670	1.65001363
11.28888889	0.26574308	0.23041701
14.11111111	0.13489302	0.11774609
16.93333333	1.52509930	1.35933496
19.75555556	4.10780373	3.77755644
22.57777778	7.45534498	7.08832947
25.40000000	14.32488978	13.57887468

Z	Qch	Qrad	QCsCond
0.00000000	11.79745682	14.63163961	1.05785902
2.82222222	6.93608875	14.08628468	1.03882279
5.64444444	7.37616467	15.47006236	1.07511735
8.46666667	8.29375361	16.87667920	1.10846876
11.28888889	8.50499643	17.73199865	1.12501096
14.11111111	8.52727965	17.72460964	1.11826101
16.93333333	8.38483192	16.84259733	1.08813083
19.75555556	7.54328311	15.37355212	1.04080057
22.57777778	7.13714477	13.91962742	0.99108494
25.40000000	12.43953024	14.49800445	0.99904635

Appendix D

Code Listing

```
      program TFETC
      implicit double precision (a-h,o-z)

*********************************************************************
*                                                                   *
*            Thermionic Fuel Element Transient Code (TFETC)         *
*            written by : Abdullah S. Al-Kheliewi                   *
*                                                                   *
*********************************************************************

      Parameter (Imax = 10, Jmax = 10)
      Integer Prob, Options
      double precision Time, Tprint
      Integer J, TabFlag, ipout, Isolver
      double precision T(Imax,Jmax), msave,Current
      double precision Tcoolant(Jmax),Zmin,Qec(jmax),Jdens(jmax)
      double precision Tinlet,dt,Tstop,Tstart,Qch(jmax),QcsCond(jmax)
      double precision Rbound(10),Ems,Tr,EmHeat(jmax),ColHeat(jmax)
      double precision QTable(Jmax),De,G1,W,PhiE,Qrad(jmax)
      double precision Zmax,Itop,Ibottom,powerTh
      double precision Dout,Din,Q3ave,PowerTabl(2,100)
      double precision mdot,A,B,tau
      Integer I, Rmesh(9), K2, Mat(5)
      Character*80 Title
      Common /Rdata/ Rbound,Rmesh,Mat
      Common /Zdata/ Zmin,Zmax,K2
      Common /QTAB/ QTable
      Common /Input/ Tr, Ems, PhiE, Itop, Ibottom, Title
      Common /CoolProp/  Tinlet, De, G1, W, Dout, Din, mdot
      Common /Steady/ Emheat,ColHeat,Qch,Qrad,QcsCond,Qec,Jdens,
     &                Current
      Common /PowerData/ Q3ave, PowerTabl, TabFlag
*     Data Pi/3.1415926D0/

c
c ... read input data for TFETC
      call input(Prob,Options,Time,Tprint,T,dt,Tstop,Tstart,
     &           tcoolant,powerTh,A,B,tau,ipout,Isolver)

c
c ... ************  run steady state problem  ************
      if ( Prob .eq. 4 ) then

          print*,' Calculating steady state temperature profile ... '
          print*

         call tfehx(tcoolant,t,Isolver)
         stop
      end if
c
c ... ************    run start up problem    ************
      if ( Prob .eq. 1 ) then
         if ( Options .eq. 1 ) then
c use default ambient conditions for start-up
            do 10 j=1,jmax
            do 10 i=1,imax
                T(i,j) = 298.0d0
                Tcoolant(j) = Tinlet
10          continue
         end if

         print*,' Calculating transient temperature profile ... '
         print*

         call timplcit(T,Time,Tprint,dt,Prob,tcoolant,msave,
     &                 Tstop,Tstart,options,powerTh,A,B,tau,
     &                 ipout,Isolver)
      end if
c
c ... **  run shutown problem or loss of flow problem  ***
      if ( Prob .eq. 2 .or. Prob .eq. 3) then
         if ( Options .eq. 1 ) then
c use steady state solution as forcing function

          print*,' Calculating steady state temperature profile ... '
          print*

            call tfehx(tcoolant,t,Isolver)
         end if

         if ( prob .eq. 3 ) then
c
c ... if loss of flow, set flow-rate at t=0 to msave
                msave = mdot
         end if
```

```
         print*,' Calculating transient temperature profile ... '
         print*

         call timplcit(T,Time,Tprint,dt,Prob,tcoolant,msave,
     &                 Tstop,Tstart,options,powerTh,A,B,tau,
     &                 ipout,Isolver)
      end if

      stop
      end

      subroutine gdot(tnow,msave,A,B,tau)
      double precision Tinlet, De, G1, W, Dout, Din, mdot, tnow, msave
      double precision ir(10), or(10), A, B, tau, Pi
      Common /CoolProp/  Tinlet, De, G1, W, Dout, Din, mdot
      Common /ggdot/ ir, or
      Data Pi/3.1415926D0/
c
c
      mdot = msave * (A + B*dexp(-tnow/tau) )
      g1 = mdot/(or(10)**2.0d0-ir(10)**2.0d0)/pi

      return
      end
      subroutine timplcit(t,time,tprint,dt,prob,tcoolant,msave,
     &                    Tstop,Tecool,option,powerTh,Aa,B,tau,
     &                    ipout,Isolver)
      implicit double precision (a-h,o-z)
*********************************************************************
*                                                                   *
*       This subroutine does transient calculations.                *
*       It is part of the TFETC code.                               *
*       Written by : Abdullah S. Al-Kheliewi (June 1993)
                 *
*********************************************************************
      parameter ( imax = 10, jmax = 10 )
      integer prob, N, icmax, itnow, Isolver
      double precision t(imax,jmax),time,tprint,mdot
      integer j, j2, j3, ioff, option, istart, ipout, iprint
      double precision A(Imax*jmax+1,Imax*jmax), X(Imax*jmax)
      double precision c1, kcond, r, z, r1, z1, msave,Aa,B
      double precision r2,z2,r3,z3,temp3,h,t1,temm(jmax),tau
      double precision deltaz,c3,t2,tcoolant(jmax),v(jmax),zmin
      double precision tinlet,v0,c4,cp1,dt,rho1,tnow,dmax1,minv
      double precision rbound(10),sig,ems,gapcond,tr,qch(jmax)
      double precision heffe(jmax),heffc(jmax), current,qec(jmax)
      double precision jdens(jmax),gtran,ccp,rho,pi,teaav,tcol(jmax)
      double precision qtable(jmax),de,g1,w,phie,emheat(jmax)
      double precision colheat(jmax),zmax,itop,ibottom,cden_av1,cden_av2
      double precision vguess(jmax),teav(jmax),tcav(jmax),Tstop,Tecool
      double precision dout,din,ThPower,Power
      double precision qrad(jmax),qcscond(jmax),tstart, powerTh
      integer i, k, i1, i2, i3, i4, rmesh(9), k2, i9, mat(5)
      character*80 title
      Common /GaussMAIN/ A, X, N
      common /rdata/ rbound,rmesh,mat
      common /zdata/ zmin,zmax,k2
      common /qtab/ qtable
      common /input/ tr, ems, phie, itop, ibottom, title
      common /coolprop/  tinlet, de, g1, w, dout, din, mdot
      Common /Steady/ Emheat,ColHeat,Qch,Qrad,QcsCond,Qec,Jdens,
     &                Current
      Data Pi/3.1415926D0/
c
c ... set the stefan-boltzman constant (watts/cm^2 k^4) sig
      sig = 5.67d-12
      N = Imax*jmax
      ioff = 0
      istart = 0
c
c ... set the initial guess value of the interelectrode voltage.
      v0 = 0.60d0
      do k=1, jmax
         vguess(k) = v0
      end do

c
c ... initialize loop parameters
      tstart = 0.0d0
      icmax = idint( time/dt ) + 1
c
c ... start major loop for transient calculations
      do itnow = 1,icmax
         tnow = dble( float(itnow-1) ) * dt
         if (itnow .eq. icmax ) tnow = time
c
```

```
c ... update mass-flow-rate and mass-velocity
        if ( prob .eq. 3 ) then
            call gdot(tnow,msave,Aa,B,tau)
        end if
c
c ... form a table for calculating the heatflux
        do k=1,jmax
           qtable(k) = kcond(imax,r(imax),t(imax,k))
     a               *(t(imax-1,k) - t(imax,k))/(r(imax) - r(imax-1))
           if (qtable(k) .lt. 0.d0 ) qtable(k) = 0.d0

        end do

c
c ... solve for the axial coolant temperature distribution
        call tconvect(tnow,dt,tcoolant)
        call coolanttemp(tcoolant)
c
c ... compute average emmiter and collector temperatures (axial)
        i1 = 1
        do i9=1,2
           i1 = i1 + rmesh(i9)
        end do
        i2 = 1
        do i9=1,3
           i2 = i2 + rmesh(i9)
        end do
        i3 = i2+1
        i4 = 1
        do i9=1,5
           i4 = i4 + rmesh(i9)
        end do
        teaav = 0.d0
        do k=1,jmax
           temm(k) = t(i2,k)
           teaav = teaav + temm(k)
           tcol(k) = t(i2+1,k)
           teav(k) = t(i1,k)*(r(i1+1)**2+2*r(i1+1)*r(i1)-3*r(i1)**2)/4
           tcav(k) = t(i3,k)*(r(i3+1)**2+2*r(i3+1)*r(i3)-3*r(i3)**2)/4
           do i=i1+1,i2-1
              teav(k) = teav(k) + t(i,k)*(r(i+1)**2+2*r(i)*
     a                       (r(i+1)-r(i-1))-r(i-1)**2)/4
           end do
           do i=i3+1,i4-1
              tcav(k) = tcav(k) + t(i,k)*(r(i+1)**2+2*r(i)*
     a                       (r(i+1)-r(i-1))-r(i-1)**2)/4
           end do
           teav(k) = teav(k) + t(i2,k)*(3*r(i2)**2-2*r(i2)*r(i2-1)
     a                    -r(i2-1)**2)/4
           tcav(k) = tcav(k) + t(i4,k)*(3*r(i4)**2-2*r(i4)*r(i4-1)
     a                    -r(i4-1)**2)/4
           teav(k) = teav(k)/(r(i2)**2-r(i1)**2)
           tcav(k) = tcav(k)/(r(i4)**2-r(i3)**2)
        end do

c       compute the axially-averaged emitter temperature at the outer
c ...   surface
        teaav = teaav/dble(float(jmax))

c
c     compute voltage and current density distribution along the length
c ... of the thermionic converter
        cden_av1 = ibottom/(pi*r(i2)*2*(zmax-zmin)/2)
        cden_av2 = itop/(pi*r(i2)*2*(zmax-zmin)/2)
c
c ... calculate the transient cesium reservoir temperature

        if ( (Prob .eq. 1 .and. Teaav .lt. Tecool .and. istart .eq. 0)
     &         .or. (Prob .eq. 2 .and. ioff .eq. 1) ) then
*    &         .or. itnow .lt. 2 ) then

              call helium(v,qec,qch,jdens,emheat,colheat)

        else
*        print*,' calling cylcon '
        call cylcon6(temm,teav,tcol,tcav,tr,phie,r(i3)-r(i2),cden_av1,
     1  cden_av2,zmax-zmin,2*r(i2),r(i2)-r(i1),r(i4)-r(i3),jmax,
     2  vguess, v, qec, jdens, emheat, colheat)

        current = itop+ibottom

        do k=1,jmax
           vguess(k) = v(k)
          qch(k) = qec(k)-jdens(k)*v(k)
        end do
```

```
         if ( prob .eq. 1 ) istart = 1
         if ( minv(v) .lt. 0.0d0 ) then
              call helium(v,qec,qch,jdens,emheat,colheat)
              ioff = 1
         end if

         end if
         if ( prob .ne. 1 .and. option .eq. 2) then
              call initial(prob,Tstop,Tecool,tr,sig,ems,
     &                     Qrad,Qcscond,t)
         end if
         Power = ThPower(prob,tau,tnow,powerTh)
c
c ... write output

         if (iprint(tnow,tprint,dt,time,tstart) .eq. 1) then
*            print *, ' Time = ', tnow
             j2 = jmax/2
             j3 = j2 + 1
             print 100, tnow,dmax1(t(1,j2),t(1,j3)),teaav,tcoolant(jmax)
  100        format(' Time = ',f9.3,' Tfuel = ',f9.3,' Teav = ',f9.3,
     &              ' Tcoolant = ',f9.3)
             call output(tnow,imax,jmax, v, qec, jdens, emheat,
     1                   tcoolant,colheat, qch, qrad, qcscond,
     2                   t, current, prob, Power, ipout)
         end if

         do 2000 j=1, jmax
         do 2000 i=1, imax
            i2 = (j-1)*imax + i
            do k=1,N+1
               A(k,i2) = 0.0d0
            end do
**          i = 1, j = 2, jmax -1  ****************************    (Figure 3.6)
            if ((i.eq.1).and.((j.ne.1).and.(j.ne.jmax))) then
              r1 = r(i)
              r3 = r(i+1)
              r2 = (r3 + r1)/2
              z1 = z(j)
              deltaz = (z(j+1)-z(j-1))/2
              t1 = (t(i+1,j) + t(i,j))/2
              c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
              c1 = c3

             j2 = (j-1)*imax + i + 1
             A(J2,I2) = C3

              temp3 = (r3**2 + 2*r1*r3 - 3*r1**2)/4
              z3 = z(j+1)
              z2 = (z3 + z1)/2
              t1 = (t(i,j+1) + t(i,j))/2
              c3 = kcond(i,r1,t1)/(z3-z1)*temp3
              c1 = c1 + c3
             J2 = j*Imax + I
             A(J2,I2) = C3

              z3 = z(j-1)
              z2 = (z3 + z1)/2
              t1 = (t(i,j-1) + t(i,j))/2
              c3 = kcond(i,r1,t1)/(z1-z3)*temp3
              c1 = c1 + c3
              J2 = (j-2)*Imax + I
              A(J2,I2) = C3
             rho1 = rho(i,(r1+r2)/2)
             cp1 - ccp(i,(r1+r2)/2,t1)
             c4 = rho1 * cp1 * deltaz * temp3/dt
             c1 = c1 + c4
             A(N+1,I2) = - gtran((r1+r2)/2,z1,prob,tau,tnow)
     &                    * deltaz * temp3 - c4 * t(i,j)

             A(i2,i2) = - c1

             go to 2000

            endif

           if ((i.eq.1).and.(j.eq.1)) then
**         i = 1, j = 1           ****************************    (Figure 3.5)
             r1 = r(i)
             r3 = r(i+1)
             r2 = (r3 + r1)/2
             z1 = z(j)
             z3 = z(j+1)
             deltaz = (z3-z1)/2
             t1 = (t(i+1,j) + t(i,j))/2
             c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
```

```
      c1 = c3
      J2 = (j-1)*Imax + I+1
      A(J2,I2) = C3

      temp3 = (r3**2 - 3*r1**2 + 2*r1*r3)/4
      z2 = (z3 + z1)/2
      t1 = (t(i,j+1) + t(i,j))/2
      c3 = kcond(i,r1,t1)/(z3-z1)*temp3
      c1 = c1 + c3
      J2 = j*Imax + I
      A(J2,I2) = C3

      rho1 = rho(i,(r1+r2)/2)
      cp1 = ccp(i,(r1+r2)/2,t1)
      c4 = rho1 * cp1 * deltaz * temp3/dt
      c1 = c1 + c4
      A(N+1,I2) = - gtran((r1+r2)/2,(z1+z2)/2,prob,tau,tnow)
     &                * deltaz * temp3 - c4 * t(i,j)

      A(i2,i2) = - c1
      go to 2000

    end if

    if ((i.eq.1).and.(j.eq.jmax)) then
**  i = 1 and j = jmax     ****************************     (Figure 3.4)
      r1 = r(i)
      r3 = r(i+1)
      r2 = (r3 + r1)/2
      z1 = z(j)
      z3 = z(j-1)
      deltaz = (z1-z3)/2
      t1 = (t(i+1,j) + t(i,j))/2
      c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
      c1 = c3
      J2 = (j-1)*Imax + I+1
      A(J2,I2) = C3

      temp3 = (r3**2 - 3*r1**2 + 2*r1*r3)/4
      z2 = (z3 + z1)/2
      t1 = (t(i,j-1) + t(i,j))/2
      c3 = kcond(i,r1,t1)/(z1-z3)*temp3
      c1 = c1 + c3
      J2 = (j-2)*Imax + I
      A(J2,I2) = C3

      rho1 = rho(i,(r1+r2)/2)
      cp1 = ccp(i,(r1+r2)/2,t1)
      c4 = rho1 * cp1 * deltaz * temp3/dt
      c1 = c1 + c4
      A(N+1,I2) = - gtran((r1+r2)/2,(z1+z2)/2,prob,tau,tnow)
     &                * deltaz * temp3 - c4 * t(i,j)

      A(i2,i2) = - c1
      go to 2000

    end if

**  Collector Surface, j = 2, jmax -1  ****************** (Figure 3.12)
    if ((r(i).eq.rbound(5)).and.((j.ne.1).and.(j.ne.jmax))) then
      r1 = r(i)
      r3 = r(i+1)
      r2 = (r3 + r1)/2
      z1 = z(j)
      deltaz = (z(j+1)-z(j-1))/2
      t1 = (t(i+1,j) + t(i,j))/2
      c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
      c1 = c3
      J2 = (j-1)*Imax + I+1
      A(J2,I2) = C3

      temp3 = (r3**2 + 2*r1*r3 - 3*r1**2)/4
      z3 = z(j+1)
      z2 = (z3 + z1)/2
      t1 = (t(i,j+1) + t(i,j))/2
      c3 = kcond(i,r1,t1)/(z3-z1)*temp3
      c1 = c1 + c3
      J2 = j*Imax + I
      A(J2,I2) = C3

      z3 = z(j-1)
      z2 = (z3 + z1)/2
      t1 = (t(i,j-1) + t(i,j))/2
      c3 = kcond(i,r1,t1)/(z1-z3)*temp3
      c1 = c1 + c3
      J2 = (j-2)*Imax + I
```

```
          A(J2,I2) = C3

          t1 = t(i-1,j)-t(i,j)
          t2 = dmax1( t1, 1.0d0 )
          if ( t1 .le. 0.0d0 ) then
               heffc(j) = qch(j)*2*r(i-1)*deltaz
          else
               heffc(j) = (qch(j)
b              + sig*ems*((t(i-1,j))**4-(t(i,j))**4)*r(i-1)/r(i)
c              + gapcond(t(i-1,j),t(i,j),Tstop,Tecool,Prob,tr,
d              r(i)-r(i-1)))*2*r(i-1)*deltaz/t2
          end if
          C1 = C1 + HeffC(j)
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = HeffC(j)

          rho1 = rho(i,(r1+r2)/2)
          t1 = (t(i,j-1) + t(i,j))/2
          cp1 = ccp(i,(r1+r2)/2,t1)
          c4 = rho1 * cp1 * deltaz * temp3/dt
          c1 = c1 + c4
          A(N+1,I2) = - (gtran((r1+r2)/2,z1,Prob,tau,tnow) +
a                        colheat(j))*deltaz * temp3 - c4 * t(i,j)

          A(i2,i2) = - c1
          go to 2000

        end if

**      Collector Surface, j = 1  **************************** (Figure 3.14)
        if ((r(i).eq.rbound(5)).and.(j.eq.1)) then
          r1 = r(i)
          r3 = r(i+1)
          r2 = (r3 + r1)/2
          z1 = z(j)
          z3 = z(j+1)
          deltaz = (z3-z1)/2
          t1 = (t(i+1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
          c1 = c3
          J2 = (j-1)*Imax + I+1
          A(J2,I2) = C3

          temp3 = (r3**2 - 3*r1**2 + 2*r1*r3)/4
          z2 = (z3 + z1)/2
          t1 = (t(i,j+1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z3-z1)*temp3
          c1 = c1 + c3
          J2 = j*Imax + I
          A(J2,I2) = C3

          t1 = t(i-1,j)-t(i,j)
          t2 = dmax1( t1, 1.0d0 )
          if ( t1 .le. 0.0d0 ) then
               heffc(j) = qch(j)*2*r(i-1)*deltaz
          else
               heffc(j) = (qch(j)
b              + sig*ems*((t(i-1,j))**4-(t(i,j))**4)*r(i-1)/r(i)
c              + gapcond(t(i-1,j),t(i,j),Tstop,Tecool,Prob,tr,
d              r(i)-r(i-1)))*2*r(i-1)*deltaz/t2
          end if
          C1 = C1 + HeffC(j)
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = HeffC(j)

          rho1 = rho(i,(r1+r2)/2)
          t1 = (t(i,j+1) + t(i,j))/2
          cp1 = ccp(i,(r1+r2)/2,t1)
          c4 = rho1 * cp1 * 2 * deltaz * temp3/dt
          c1 = c1 + c4
          A(N+1,I2) = - (gtran((r1+r2)/2,z1,Prob,tau,tnow) +
a                        colheat(j))*deltaz * temp3 - c4 * t(i,j)

          A(i2,i2) = - c1
          go to 2000

        end if

**      Collector Surface and j = jmax *********************  (Figure 3.13)
        if ((r(i).eq.rbound(5)).and.(j.eq.jmax)) then
          r1 = r(i)
          r3 = r(i+1)
          r2 = (r3 + r1)/2
          z1 = z(j)
          z3 = z(j-1)
          deltaz = (z1-z3)/2
```

```
          t1 = (t(i+1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
          c1 = c3
          J2 = (j-1)*Imax + I+1
          A(J2,I2) = C3

          temp3 = (r3**2 - 3*r1**2 + 2*r1*r3)/4
          z2 = (z3 + z1)/2
          t1 =(t(i,j-1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z1-z3)*temp3
          c1 = c1 + c3
          J2 = (j-2)*Imax + I
          A(J2,I2) = C3

          t1 = t(i-1,j)-t(i,j)
          t2 = dmax1( t1, 1.0d0 )
          if ( t1 .le. 0.0d0 ) then
              heffc(j) = qch(j)*2*r(i-1)*deltaz
          else
            heffc(j) = (qch(j)
     b      + sig*ems*((t(i-1,j))**4-(t(i,j))**4)*r(i-1)/r(i)
     c      + gapcond(t(i-1,j),t(i,j),Tstop,Tecool,Prob,tr,
     d      r(i)-r(i-1)))*2*r(i-1)*deltaz/t2
          end if
          C1 = C1 + HeffC(j)
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = HeffC(j)

          rho1 = rho(i,(r1+r2)/2)
          t1 =(t(i,j-1) + t(i,j))/2
          cp1 = ccp(i,(r1+r2)/2,t1)
          c4 = rho1 * cp1 * 2 * deltaz * temp3/dt
          c1 = c1 + c4
          A(N+1,I2) = - (gtran((r1+r2)/2,z1,Prob,tau,tnow) +
     a                 colheat(j))*deltaz * temp3 - c4 * t(i,j)

          A(i2,i2) = - c1
          go to 2000

        end if

**      Emitter Surface, j = 2, jmax -1  *******************  (Figure 3.9)
        if ((r(i).eq.rbound(4)).and.((j.ne.1).and.(j.ne.jmax))) then
          r1 = r(i)
          z1 = z(j)
          deltaz = (z(j+1)-z(j-1))/2
          r3 = r(i-1)
          r2 = (r3 + r1)/2
          t1 = (t(i-1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
          c1 = c3
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = C3

          temp3 = (3*r1**2 - r3**2 - 2*r1*r3)/4
          z3 = z(j+1)
          z2 = (z3 + z1)/2
          t1 = (t(i,j+1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z3-z1)*temp3
          c1 = c1 + c3
          J2 = j*Imax + I
          A(J2,I2) = C3

          z3 = z(j-1)
          z2 = (z3 + z1)/2
          t1 = (t(i,j-1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z1-z3)*temp3
          c1 = c1 + c3
          J2 = (j-2)*Imax + I
          A(J2,I2) = C3

          t1 = t(i,j)-t(i+1,j)
          t2 = dmax1( t1, 1.0d0 )
          if ( t1 .le. 0.0d0 ) then
              heffe(j) = qec(j)*2*r(i)*deltaz
              qrad(j) = 0.0d0
              qcscond(j) = 0.0d0
          else
            heffe(j) = (qec(j)
     b      + sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
     c      + gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,tr,
     d      r(i+1)-r(i)))*2*r(i)*deltaz/t2
            qrad(j)    = sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
            qcscond(j) = gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,
     a                          tr,r(i+1)-r(i))
          end if
```

```
               C1 = C1 + HeffE(j)
               J2 = (j-1)*Imax + I+1
               A(J2,I2) = HeffE(j)

               rho1 = rho(i,(r1+r2)/2)
               t1 = (t(i,j-1) + t(i,j))/2
               cp1 = ccp(i,(r1+r2)/2,t1)
               c4 = rho1 * cp1 * deltaz * temp3/dt
               c1 = c1 + c4
               A(N+1,I2) = - (gtran((r1+r2)/2,z1,prob,tau,tnow)
     b                       +emheat(j))* deltaz * temp3 - c4 * t(i,j)

               A(i2,i2) = - c1
               go to 2000

             end if

**           Emitter Surface, j = 1  ****************************** (Figure 3.11)
             if ((r(i).eq.rbound(4)).and.(j.eq.1)) then
               r1 = r(i)
               z1 = z(j)
               z3 = z(j+1)
               deltaz = (z3-z1)/2
               r3 = r(i-1)
               r2 = (r3 + r1)/2
               t1 = (t(i-1,j) + t(i,j))/2
               c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
               c1 = c3
               J2 = (j-1)*Imax + I-1
               A(J2,I2) = C3

               temp3 = (3*r1**2 - r3**2 + 2*r1*r3)/4
               z2 = (z3 + z1)/2
               t1 = (t(i,j+1) + t(i,j))/2
               c3 = kcond(i,r1,t1)/(z3-z1)*temp3
               c1 = c1 + c3
               J2 = j*Imax + I
               A(J2,I2) = C3

               t1 = t(i,j)-t(i+1,j)
               t2 = dmax1( t1, 1.0d0 )
               if ( t1 .le. 0.0d0 ) then
                    heffe(j) = qec(j)*2*r(i)*deltaz
                    qrad(j) = 0.0d0
                    qcscond(j) = 0.0d0
               else
                  heffe(j) = (qec(j)
     b            + sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
     c            + gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,tr,
     d            r(i+1)-r(i)))*2*r(i)*deltaz/t2
                  qrad(j)     = sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
                  qcscond(j) = gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,
     a                             tr,r(i+1)-r(i))
               end if
               C1 = C1 + HeffE(j)
               J2 = (j-1)*Imax + I+1
               A(J2,I2) = HeffE(j)

               rho1 = rho(i,(r1+r2)/2)
               t1 = (t(i,j+1) + t(i,j))/2
               cp1 = ccp(i,(r1+r2)/2,t1)
               c4 = rho1 * cp1 * 2 * deltaz * temp3/dt
               c1 = c1 + c4
               A(N+1,I2) = - (gtran((r1+r2)/2,(z1+z2)/2,prob,tau,tnow)
     a                       +emheat(j))*deltaz * temp3 - c4 * t(i,j)

               A(i2,i2) = - c1
               go to 2000

             end if

**           Emitter Surface, j = jmax  ************************** (Figure 3.10)
             if ((r(i).eq.rbound(4)).and.(j.eq.jmax)) then
               r1 = r(i)
               z1 = z(j)
               z3 = z(j-1)
               deltaz = (z1-z3)/2
               r3 = r(i-1)
               r2 = (r3 + r1)/2
               t1 = (t(i-1,j) + t(i,j))/2
               c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
               c1 = c3
               J2 = (j-1)*Imax + I-1
               A(J2,I2) = C3

               temp3 = (3*r1**2 - r3**2 + 2*r1*r3)/4
```

```
            z2 = (z3 + z1)/2
            t1 = (t(i,j-1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z1-z3)*temp3
            c1 = c1 + c3
            J2 = (j-2)*Imax + I
            A(J2,I2) = C3

            t1 = t(i,j)-t(i+1,j)
            t2 = dmax1( t1, 1.0d0 )
            if ( t1 .le. 0.0d0 ) then
                 heffe(j) = qec(j)*2*r(i)*deltaz
                 qrad(j) = 0.0d0
                 qcscond(j) = 0.0d0
            else
               heffe(j) = (qec(j)
b              + sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
c              + gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,tr,
d              r(i+1)-r(i)))*2*r(i)*deltaz/t2
               qrad(j)    = sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
               qcscond(j) = gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,
a                                 tr,r(i+1)-r(i))
            end if
            C1 = C1 + HeffE(j)
            J2 = (j-1)*Imax + I+1
            A(J2,I2) = HeffE(j)

            rho1 = rho(i,(r1+r2)/2)
            t1 = (t(i,j-1) + t(i,j))/2
            cp1 = ccp(i,(r1+r2)/2,t1)
            c4 = rho1 * cp1 * 2 * deltaz * temp3/dt
            c1 = c1 + c4
            A(N+1,I2) = - (gtran((r1+r2)/2,(z1+z2)/2,prob,tau,tnow)
a                         +emheat(j))*deltaz * temp3 - c4 * t(i,j)

            A(i2,i2) = - c1
            go to 2000

          end if

          if (((i.ne.1).and.(i.ne.imax)).and.(j.eq.1)) then
**        j = 1 and i = 2, imax -1 ****************************** (Figure 3.8)
            r1 = r(i)
            r3 = r(i+1)
            r2 = (r3 + r1)/2
            z1 = z(j)
            z3 = z(j+1)
            deltaz = (z3-z1)/2
            t1 = (t(i+1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
            c1 = c3
            J2 = (j-1)*Imax + I+1
            A(J2,I2) = C3

            r3 = r(i-1)
            r2 = (r3 + r1)/2
            t1 = (t(i-1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
            c1 = c1 + c3
            J2 = (j-1)*Imax + I-1
            A(J2,I2) = C3

            r2 = r(i+1)
            temp3 =  (r2**2 - r3**2 + 2*r1*(r2-r3))/4
            z2 = (z3 + z1)/2
            t1 = (t(i,j+1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z3-z1)*temp3
            c1 = c1 + c3
            J2 = j*Imax + I
            A(J2,I2) = C3

            rho1 = rho(i,r1)
            cp1 = ccp(i,r1,t1)
            c4 = rho1 * cp1 * deltaz * temp3/dt
            c1 = c1 + c4
            A(N+1,I2) = - gtran(r1,(z1+z2)/2,prob,tau,tnow)
a                      * deltaz * temp3 - c4 * t(i,j)

            A(i2,i2) = - c1
            go to 2000

          end if

          if (((i.ne.1).and.(i.ne.imax)).and.(j.eq.jmax)) then
**        j = jmax and i = 2, imax -1 *************************  (Figure 3.7)
            r1 = r(i)
            r3 = r(i+1)
```

```
          r2 = (r3 + r1)/2
          z1 = z(j)
          z3 = z(j-1)
          deltaz = (z1-z3)/2
          t1 = (t(i+1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
          c1 = c3
          J2 = (j-1)*Imax + I+1
          A(J2,I2) = C3

          r3 = r(i-1)
          r2 = (r3 + r1)/2
          t1 = (t(i-1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
          c1 = c1 + c3
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = C3

          r2 = r(i+1)
          temp3 = (r2**2 - r3**2 + 2*r1*(r2-r3))/4
          z2 = (z3 + z1)/2
          t1 = (t(i,j-1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z1-z3)*temp3
          c1 = c1 + c3
          J2 = (j-2)*Imax + I
          A(J2,I2) = C3

          rho1 = rho(i,r1)
          cp1 = ccp(i,r1,t1)
          c4 = rho1 * cp1 * deltaz * temp3/dt
          c1 = c1 + c4
          A(N+1,I2) = - gtran(r1,z2,prob,tau,tnow)*deltaz * temp3
     a                - c4 * t(i,j)

          A(i2,i2) = - c1
          go to 2000

        endif

        if ((i.eq.imax).and.((j.ne.1).and.(j.ne.jmax))) then
**      i = imax, j = 2, jmax -1  **************************  (Figure 3.15)
          r1 = r(i)
          z1 = z(j)
          deltaz = (z(j+1)-z(j-1))/2
          t2 = tcoolant(j)
          c3 = h(t2)*r1*deltaz*2
          c1 = c3
          A(N+1,I2) = -C3*T2

          r3 = r(i-1)
          r2 = (r3 + r1)/2
          t1 = (t(i-1,j) + t(i,j))/2
          c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
          c1 = c1 + c3
          J2 = (j-1)*Imax + I-1
          A(J2,I2) = C3

          temp3 = (3*r1**2 - r3**2 - 2*r1*r3)/4
          z3 = z(j+1)
          z2 = (z3 + z1)/2
          t1 = (t(i,j+1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z3-z1)*temp3
          c1 = c1 + c3
          J2 = j*Imax + I
          A(J2,I2) = C3

          z3 = z(j-1)
          z2 = (z3 + z1)/2
          t1 = (t(i,j-1) + t(i,j))/2
          c3 = kcond(i,r1,t1)/(z1-z3)*temp3
          c1 = c1 + c3
          J2 = (j-2)*Imax + I
          A(J2,I2) = C3

          rho1 = rho(i,r2)
          cp1 = ccp(i,r2,t1)
          c4 = rho1 * cp1 * deltaz * temp3/dt
          c1 = c1 + c4
          A(N+1,I2) = A(N+1,I2) - gtran(r2,z1,prob,tau,tnow)
     a                          * deltaz * temp3 - c4 * t(i,j)

          A(i2,i2) = - c1
          go to 2000

        endif
```

```
          if ((i.eq.imax).and.(j.eq.1)) then
**        i = imax, j = 1  **********************************   (Figure 3.17)
            r1 = r(i)
            z1 = z(j)
            z3 = z(j+1)
            deltaz = (z3-z1)/2
            t1 = t(i,j)
            t2 = tcoolant(j)
            c3 = h(t2)*r1*deltaz*2
            c1 = c3
           A(N+1,I2) = - c3 * t2

            r3 = r(i-1)
            r2 = (r3 + r1)/2
            t1 = (t(i-1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
            c1 = c1 + c3
            J2 = (j-1)*Imax + I-1
            A(J2,I2) = C3

            temp3 = (3*r1**2 - r3**2 + 2*r1*r3)/4
            z2 = (z3 + z1)/2
            t1 = (t(i,j+1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z3-z1)*temp3
            c1 = c1 + c3
            J2 = j*Imax + I
            A(J2,I2) = C3

            rho1 = rho(i,r1)
            cp1 = ccp(i,r1,t1)
            c4 = rho1 * cp1 * deltaz * temp3/dt
            c1 = c1 + c4
            A(N+1,I2) = A(N+1,I2) - gtran(r2,z2,prob,tau,tnow)
     a                              * deltaz * temp3 - c4 * t(i,j)

            A(i2,i2) = - c1
            go to 2000

          endif

          if ((i.eq.imax).and.(j.eq.jmax)) then
**        i = imax, j = jmax  ********************************* (Figure 3.16)
            r1 = r(i)
            z1 = z(j)
            z3 = z(j-1)
            deltaz = (z1-z3)/2
            t1 = t(i,j)
            t2 = tcoolant(j)
            c3 = h(t2)*r1*deltaz*2
            c1 = c3
            A(N+1,I2) = - c3 * t2

            r3 = r(i-1)
            r2 = (r3 + r1)/2
            t1 = (t(i-1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
            c1 = c1 + c3
            J2 = (j-1)*Imax + I-1
            A(J2,I2) = C3

            temp3 = (3*r1**2 - r3**2 + 2*r1*r3)/4
            z2 = (z3 + z1)/2
            t1 = (t(i,j-1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z1-z3)*temp3
            c1 = c1 + c3
            J2 = (j-2)*Imax + I
            A(J2,I2) = C3

            rho1 = rho(i,r2)
            cp1 = ccp(i,r2,t1)
            c4 = rho1 * cp1 * deltaz * temp3/dt
            c1 = c1 + c4
            A(N+1,I2) = A(N+1,I2) - gtran(r2,z2,prob,tau,tnow)
     a                            * deltaz * temp3 - c4 * t(i,j)

            A(i2,i2) = - c1
            go to 2000

          endif

**      all other points ****************************************  (Figure 3.1)
            r1 = r(i)
            r3 = r(i+1)
            r2 = (r3 + r1)/2
            z1 = z(j)
            deltaz = (z(j+1)-z(j-1))/2
```

```
            t1 = (t(i+1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r3-r1)*deltaz
            c1 = c3
            J2 = (j-1)*Imax + I+1
            A(J2,I2) = C3

            r3 = r(i-1)
            r2 = (r3 + r1)/2
            t1 = (t(i-1,j) + t(i,j))/2
            c3 = kcond(i,r2,t1)*(r3+r1)/(r1-r3)*deltaz
            c1 = c1 + c3
            J2 = (j-1)*Imax + I-1
            A(J2,I2) = C3

            r2 = r(i+1)
            temp3 = (r2**2 - r3**2 + 2*r1*(r2-r3))/4
            z3 = z(j+1)
            z2 = (z3 + z1)/2
            t1 = (t(i,j+1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z3-z1)*temp3
            c1 = c1 + c3
            J2 = j*Imax + I
            A(J2,I2) = C3

            z3 = z(j-1)
            z2 = (z3 + z1)/2
            t1 = (t(i,j-1) + t(i,j))/2
            c3 = kcond(i,r1,t1)/(z1-z3)*temp3
            c1 = c1 + c3
            J2 = (j-2)*Imax + I
            A(J2,I2) =  C3

            rho1 = rho(i,r1)
            cp1 = ccp(i,r1,t1)
            c4 = rho1 * cp1 * deltaz * temp3/dt
            c1 = c1 + c4
            A(N+1,I2) = - gtran(r1,z1,prob,tau,tnow)*deltaz * temp3
     a                  - c4 * t(i,j)

            A(i2,i2) = - c1
 2000    continue

         if ( Isolver .eq. 1 ) then
            Call Gauss
         else
            Call SGauss
         end if

        do j=1, jmax
           do i=1, imax
              I2 = (j-1)*Imax + I
              if ( prob .eq. 1 ) then
                 T(I,j) = (T(I,J)+X(I2))/2.0
              else
                 T(I,j) = X(I2)
              end if
           end do
        end do
        do i=1,N
           X(i) = 0.0d0
           do j=1,N
              A(i,j) = 0.0d0
           end do
           A(N+1,i) = 0.0d0
        end do

      end do

      return
      end

      double precision function gtran(r,z,prob,tau,tnow)
      parameter ( eps = 1.1d-16 )
      double precision r, z, g, tau, tnow, beta, period, dollar, de
      integer prob
      common /prompt/ beta, period, dollar

      if ( tau .le. 1.1d-16 ) then
         de = 0.0d0
      else
         de = dexp(-tnow/tau)
      end if
c
c ... if startup problem, increase the heat generation exponentially
      if ( prob .eq. 1 ) then
         gtran = g(r,z) * ( 1.0d0 - de )
```

```
c
c ... if shutdown problem, decrease the heat generation to zero
      elseif ( prob .eq. 2) then
         if ( tau .ne. 0.0d0 ) then
*               ... exponetial shutdown
                gtran = g(r,z) * de
         else
*               ... prompt jump shutdown
                if ( tnow .gt. 0.0d0 ) then
                   gtran = g(r,z) * (1.0-dollar*beta)/(1.0-dollar)
     &                            * dexp(- tnow/period)
                else
                   gtran = g(r,z)
                end if
         end if
      else
         gtran = g(r,z)
      end if
      return
      end

      double precision function ThPower(prob,tau,tnow,powerTh)
      parameter ( eps = 1.1d-16 )
      double precision  tau, tnow, beta, period, dollar, powerTh, de
      integer prob
      common /prompt/ beta, period, dollar

      if ( tau .le. 1.1d-16 ) then
         de = 0.0d0
      else
         de = dexp(-tnow/tau)
      end if

c
c ... if startup problem, increase the thermal power exponentially
      if ( prob .eq. 1 ) then
         Thpower = powerTh * ( 1.0d0 - de )
c
c ... if shutdown problem, decrease the thermal power to zero
      elseif ( prob .eq. 2) then
         if ( tau .ne. 0.0d0 ) then
*               ... exponetial shutdown
                Thpower = powerTh * de
         else
*               ... prompt jump shutdown
                if ( tnow .gt. 0.0d0 ) then
                   Thpower = powerTh * (1.0-dollar*beta)/(1.0-dollar)
     &                            * dexp(- tnow/period)
                else
                   Thpower = powerTh
                end if
         end if
      else
         Thpower = powerTh
      end if
      return
      end

      integer function iprint(tnow,tprint,dt,time,tstart)
      double precision tnow, tprint, dt, time, tol, tstart, dmin1
      double precision a, b

      iprint = 0
      if (tnow .eq. tstart .or. tnow .eq. time) then
          iprint = 1
                                                            return
      end if
      tol = dmin1(dt,1.0d-5)
      a = dmod(tnow,tprint)
      b = tnow/tprint
      if(a .le. tol .and. idint(b) .ge. 1) then
          iprint = 1
                                                            return
      end if

      end

      double precision function minv(v)
      parameter ( jmax = 10 )
      double precision v(jmax)

      minv = v(1)
      do j=2,jmax
         minv = dmin1(minv,v(j))
      end do
```

```
      return
      end
      subroutine input(prob,options,time,tprint,ffun,dt,
     &                 Tstop,Tstart,tcoolant,powerTh,
     &                 A, B, tau,ipout,Isolver)

      integer numofmats, j
      parameter (numofmats = 8, imax = 10, jmax = 10)                  KHEL 4/25/93
      double precision tinlet, mdot, w, tr, itop, ibottom, powerth
      double precision pwrtabl(2,100),ir(10),or(10),ems,phi0(numofmats)
      double precision zmin,zmax,l,phie,z,g,trl,pcs,rbound(10),pi,Tstop
      double precision de, g1, d2, d1, fuelvol, q3ave, totmesh, paratio
      integer rmesh(9),mat(5),k2
      integer prob, options, ipout,Isolver                             KHEL 4/11/93
      double precision time, tprint, ffun(imax,jmax),dt,Tstart         KHEL 4/11/93
      double precision beta, period, dollar, Tcoolant(jmax)            KHEL 6/28/93
      double precision tau, A, B, rperiod                              KHEL 7/6/93
      integer tabflag, matnum(9), meshpt(9), i, tablen
      character*80 title
      character*21 regname(10)
      character*5 matname(numofmats)
      logical deckerror
      common /input/ tr, ems, phie, itop, ibottom, title
      common /rdata/ rbound,rmesh,mat
      common /zdata/ zmin,zmax,k2
      common /powerdata/ q3ave, pwrtabl, tabflag
      Common /ggdot/ ir, or                                            KHEL 6/12/93
      common /prompt/ beta, period, dollar                             KHEL 6/29/93
      common /coolprop/  tinlet, de, g1, w, d2, d1, mdot
      data matname/'uo2  ','w    ','nb   ','nb1zr','mo   ','re   ',
     a           'cs   ','al2o3'/
      data phi0/0.0d0,4.9d0,6*0.0d0/
      data regname/'fuel                 ','fuel-emitter gap      ',
     a           'emitter              ','emitter-collector gap ',
     a           'collector            ','collector-insulator gap',
     c           'insulator            ','insulator-cladding gap ',
     b           'cladding             ','coolant channel       '/
      data pi/3.1415926d0/

      open (7, file='tfetc.inp', status='old')
      open (8, file='tfetc.out')

      deckerror=.false.
      Read (7,10) Title
      Read (7,*) Prob                                                  KHEL 4/11/93
      if (Prob .lt. 1 .or. Prob .gt. 4 .or. mod(Prob,1) .ne. 0) then   KHEL 4/11/93
          Write(8,9010) 'Invalid problem specification.'               KHEL 4/11/93
          Write(8,9120) 'Prob',1,2,3,4                                 KHEL 4/11/93
          DeckError = .True.                                           KHEL 4/11/93
      end if                                                           KHEL 4/11/93
      Read (7,*) Isolver                                               KHEL 4/11/93
      if (Isolver .ne. 1 .and. Isolver .ne. 2) then                    KHEL 4/11/93
         Write(8,9010) 'Invalid Isolver specification.'                KHEL 4/11/93
         Write(8,9020) 'Isolver',1,2                                   KHEL 4/11/93
         DeckError = .True.                                            KHEL 4/11/93
      end if                                                           KHEL 4/25/93
      if (Prob .ne. 4) then                                            KHEL 4/11/93
          Read (7,*) Options                                           KHEL 4/11/93
          if (Options .ne. 1 .and. Options .ne. 2) then                KHEL 4/11/93
             Write(8,9010) 'Invalid Options specification.'            KHEL 4/11/93
             Write(8,9020) 'Options',1,2                               KHEL 4/11/93
             DeckError = .True.                                        KHEL 4/11/93
          end if                                                       KHEL 4/25/93
          Read (7,*) Time                                              KHEL 4/11/93
          Read (7,*) Tprint                                            KHEL 5/25/93
          Read (7,*) dt                                                KHEL 6/20/93
          Read (7,*) ipout                                             KHEL 6/20/93
          if (dt .gt. Tprint .or. Tprint .gt. Time                     KHEL 7/7/93
     &        .or. Time .le. 0.0 .or. dt .gt. 1.0d0) then              KHEL 7/7/93
             Write(8,9010) ' Invalid Timing Specification.'            KHEL 7/7/93
             Write(8,9290) 'DT',Tprint,1.0d0,'Tprint',Time,'Time',0.0d0 KHEL 7/7/93
             DeckError = .True.                                        KHEL 4/11/93
          end if                                                       KHEL 7/7/93
          if ( ipout .ne. 0 .and. ipout .ne. 1 ) then                  KHEL 7/7/93
             Write(8,9010) ' Invalid Print Option.'                    KHEL 7/7/93
             Write(8,9020) 'IPOUT',0,1                                 KHEL 7/7/93
             DeckError = .True.                                        KHEL 4/11/93
          end if
      end if                                                           KHEL 4/11/93
      if ( prob .eq. 3 ) then                                          KHEL 6/26/93
           Read(7,*) A, B, tau                                         KHEL 6/26/93
           if (A .lt. 0.0d0 .or. B .le. 0.0d0 .or. A+B .ne. 1.0d0) then KHEL 6/26/93
              Write(8,9010) ' Invalid mass-loss coefficients '         KHEL 6/26/93
              Write(8,9280) ' A ',0.0d0,' B ',0.0d0,' A+B ',1.0d0      KHEL 6/26/93
             DeckError = .True.                                        KHEL 6/26/93
           end if                                                      KHEL 6/26/93
```

```
      end if                                                      KHEL 6/26/93
      if ( prob .eq. 2 ) then                                     KHEL 6/26/93
*        ... prompt jump case                                     KHEL 6/26/93
*        ... read the total delayed neutron fraction              KHEL 6/26/93
         Read(7,*) beta                                           KHEL 6/26/93
*        ... read the reactivity insertion in dollars             KHEL 6/26/93
         Read(7,*) dollar                                         KHEL 6/26/93
         if ( dollar .ge. 0.0d0 ) then                            KHEL 6/26/93
            Write(8,9010) ' Invalid reactivity insertion '        KHEL 6/26/93
            Write(8,9220) 'Rho ($)',0.0d0                         KHEL 6/26/93
            DeckError = .True.                                    KHEL 6/26/93
         end if                                                   KHEL 6/26/93
*        ... read the reactor period in seconds                   KHEL 6/26/93
*        Read(7,*) period                                         KHEL 6/26/93
*        if (period .le. 8.0d1) then                              KHEL 6/26/93
*           Write(8,9010) ' Invalid reactor period '              KHEL 6/26/93
*           Write(8,9040) 'Period',8.0d1                          KHEL 6/26/93
*           DeckError = .True.                                    KHEL 6/26/93
*        end if                                                   KHEL 6/26/93
*        ... calculate reactor period                             KHEL 6/26/93
         period = rperiod(dollar)                                 KHEL 6/26/93
*        print*,' period = ', period
      end if                                                      KHEL 6/26/93
      if ( prob .eq. 1 ) then                                     KHEL 6/21/93
          Read (7,*) Tstop                                        KHEL 6/21/93
          Read (7,*) Tstart                                       KHEL 6/25/93
          if ( Tstop .ge. Tstart ) then                           KHEL 6/26/93
             Write(8,9010)                                        KHEL 6/26/93
     &             'Invalid temperatures to stop helium heating'  KHEL 6/26/93
             Write(8,9010) ' and begin electron cooling. '        KHEL 6/26/93
             Write(8,9130) Tstop, Tstart                          KHEL 6/26/93
             DeckError = .True.                                   KHEL 6/26/93
         end if                                                   KHEL 6/26/93
         Read(7,*) tau                                            KHEL 6/26/93
         if (tau .le. 0.0d0) then                                 KHEL 6/26/93
             Write(8,9010) ' Invalid power-rise coefficient '     KHEL 6/26/93
             Write(8,9040) 'tau',0.0d0                            KHEL 6/26/93
            DeckError = .True.                                    KHEL 6/26/93
         end if                                                   KHEL 6/26/93
      end if
      Read (7,*) Tinlet, Mdot, W, Tr, Itop, Ibottom, PowerTh
      Read (7,*) TabFlag
      I=0
      if (TabFlag.EQ.1) then
    5     I=I+1
          Read (7,*) Z, G
          if (G.GE.0.0D0) then
              PwrTabl(1,I) = Z
              PwrTabl(2,I) = G
              Goto 5
            else
              TabLen = I
            end if
        elseif (TabFlag.EQ.2) then
          Read (7,*) PwrTabl(1,1), PwrTabl(2,1)
        else
          Write(8,9010) 'Invalid heat generation table flag.'
          Write(8,9020) 'TabFlag',1,2
          DeckError = .True.
        end if
      Do 7 I=1, 9
    7     Read (7,*) IR(I), OR(I), MatNum(I), MeshPt(I)
      Read (7,*) IR(10), OR(10)
      Read (7,*) Ems, Zmin, Zmax, L
C
C ... read forcing function
      if (options .eq. 2) then                                    KHEL 4/11/93
         Do 6 j=1,Jmax                                            KHEL 4/11/93
            Read (7,*) (Ffun(i,j), i=1,Imax)                      KHEL 4/11/93
    6    Continue                                                 KHEL 4/11/93
      end if                                                      KHEL 4/11/93
c
c ... read initial coolant temperature profile                    KHEL 7/5/93
      if ( (prob .eq. 2 .or. prob .eq. 3 ) .and. options .eq. 2 ) then  KHEL 7/5/93
         Read(7,*) (Tcoolant(j), j=1, jmax)                       KHEL 7/5/93
      end if

C***  Echo input data to output file  ****

      Write(8,100)
      Write(8,10) Title
      Write(8,9260) ' '                                           KHEL 4/11/93
      if (Prob .eq. 1) then                                       KHEL 4/11/93
         Write(8,9260) 'Start Up Problem'                         KHEL 4/11/93
      elseif (Prob .eq. 2) then                                   KHEL 4/11/93
         Write(8,9260) 'Shutdown Problem'                         KHEL 4/11/93
```

```
      elseif (Prob .eq. 3) then                                   KHEL 4/11/93
         Write(8,9260) 'Loss of Flow Problem'                     KHEL 4/11/93
      elseif (Prob .eq. 4) then                                   KHEL 4/11/93
         Write(8,9260) 'Steady State Problem'                     KHEL 4/11/93
      end if                                                      KHEL 6/12/93
      if (Isolver .eq. 1 ) then                                   KHEL 6/12/93
         Write(8,9310)                                            KHEL 7/18/93
      else                                                        KHEL 7/18/93
         Write(8,9320)                                            KHEL 7/18/93
      end if                                                      KHEL 7/18/93
      if (Prob .ne. 4) then                                       KHEL 6/26/93
         Write(8,9180) Time                                       KHEL 6/27/93
         Write(8,9140) Tprint                                     KHEL 6/26/93
         Write(8,9150) dt                                         KHEL 6/26/93
         Write(8,9300) ipout
      end if                                                      KHEL 6/26/93
      if (Prob .eq. 1) then                                       KHEL 6/26/93
         Write(8,9160) Tstop                                      KHEL 6/26/93
         Write(8,9170) Tstart                                     KHEL 6/26/93
         Write(8,9190) tau                                        KHEL 6/26/93
      end if                                                      KHEL 6/26/93
      if (Prob .eq. 2) then                                       KHEL 6/26/93
         Write(8,9230) beta                                       KHEL 6/26/93
         Write(8,9240) dollar                                     KHEL 6/26/93
         Write(8,9250) period                                     KHEL 6/26/93
      end if                                                      KHEL 6/26/93
      if (Prob .eq. 3) then                                       KHEL 6/26/93
         Write(8,9210) A,B,tau                                    KHEL 6/26/93
      end if                                                      KHEL 6/26/93
      if ((W.GT.0.0D0).AND.(W.LT.1.0D0)) then
          Write(8,120) INT(W*100.0D0)
        elseif (W.EQ.0.0D0) then
          Write(8,130)
        elseif (W.EQ.1.0D0) then
          Write(8,140)
        else
          Write(8,9010)
     a      'Invalid potassium weight fraction in NaK coolant.'
          Write(8,9030) 'W', 0.0, 1.0
          DeckError=.True.
        end if
      if (Mdot.GT.0.0D0) then
          Write(8,210) Mdot
        else
          Write(8,9010) 'Invalid coolant flow rate.'
          Write(8,9040) 'Mdot', 0.0D0
          DeckError=.True.
        end if
      if (Tinlet.GE.0.0D0) then
          Write(8,310) Tinlet
        else
          Write(8,9010) 'Invalid coolant inlet temperature.'
          Write(8,9040) 'Tinlet', 0.0D0
          DeckError=.True.
        end if
      if (Tr.GE.0.0D0) then
          Write(8,410) Tr
          Pcs = 2.45D8/SQRT(Tr)*EXP(-8910.0D0/Tr)
          Write(8,420) Pcs
        else
          Pcs = -Tr
          Tr1 = 600.0D0
    8     Tr = -8910D0/LOG(Pcs*SQRT(Tr1)/2.45D8)
          if (ABS(Tr-Tr1).GE.1.0D-5) then
              Tr1 = Tr
              Goto 8
            end if
          Write(8,420) Pcs
        end if
      if (Ems.GE.0.0D0) then
          Write(8,1010) Ems
        else
          Write(8,9010) ' Invalid effective emitter to collector',
     a      ' emissivity'
          Write(8,9050) 'Ems', 0.0D0
          DeckError = .True.
        end if
      if (Itop.GE.0.0D0) then
          Write(8,510) Itop
        else
          Write(8,9010) 'Invalid output current at the top of the pin.'
          Write(8,9040) 'Itop', 0.0D0
          DeckError=.True.
        end if
      if (Ibottom.GE.0.0D0) then
          Write(8,610) Ibottom
```

```
        else
          Write(8,9010) 'Invalid output current at the bottom of the',
     a                  ' pin.'
          Write(8,9040) 'Ibottom', 0.0D0
          DeckError=.True.
        end if
      if (PowerTh.GE.0.0D0) then
          Write(8,710) PowerTh
        else
          PowerTh = -PowerTh*(Zmax-Zmin)*Pi*(OR(1)**2-IR(1)**2)
          Write(8,710) PowerTh
        end if
      FuelVol = Pi*(OR(1)**2.0D0-IR(1)**2.0D0)*(Zmax-Zmin)
      Q3ave = Powerth/FuelVol
      Write(8,720) Q3ave
      if (TabFlag.EQ.1) then
          Write(8,810)
          Do 9 I=1, TabLen
    9        Write(8,820) PwrTabl(1,I), PwrTabl(2,I)
        else
          Write(8,830) PwrTabl(1,1), PwrTabl(2,1)
          PAratio = (PwrTabl(1,1)+PwrTabl(2,1))/(PwrTabl(1,1) +
     a               2.0D0*PwrTabl(2,1)/Pi)
          PwrTabl(1,1) = PwrTabl(1,1)/PAratio
          PwrTabl(2,1) = PwrTabl(2,1)/PAratio
          Write(8,835) PAratio
        end if
      Write(8,100)
      Write(8,10) Title
      Write(8,910)
      TotMesh = 1
      Do 70 I=1,9
          if (MatNum(I).NE.0) then
              Write(8,920) RegName(I), IR(I), OR(I),
     a        MatName(MatNum(I)), MeshPt(I)
            end if
          if (IR(I).GT.OR(I)) then
              Write(8,9010) 'Invalid geometry data.'
              Write(8,9060) RegName(I)
              DeckError = .True.
            end if
          if (I.GT.1) then
              if (IR(I).NE.OR(I-1)) then
                   Write(8,9010) 'Invalid geometry data.'
                   Write(8,9070) RegName(I), RegName(I-1)
                   DeckError = .True.
                 end if
            end if
          if (I.EQ.3) then
              PhiE = Phi0(MatNum(I))
              if (PhiE.EQ.0.0D0) then
                   Write(8,9010) 'Invalid emitter material specification'
                   Write(8,9080) MatName(MatNum(I))
                   DeckError = .True.
                 end if
            end if
          if (IR(I).NE.OR(I)) MeshPt(I) = MeshPt(I) + 1
          TotMesh = TotMesh + MeshPt(I)
   70   Continue
      if (TotMesh.NE.Imax) then
          Write(8,9010) 'Invalid mesh point specification.'
          Write(8,9090) Imax
          DeckError = .True.
        end if
      Write(8,930) RegName(10), IR(10), OR(10)
      Write(8,1110) Zmin, Zmax, Zmax-Zmin, L
      if (ABS(Zmax-Zmin).GT.L) then
          Write(8,9010) 'Invalid geometry data.'
          Write(8,9100)
          DeckError = .True.
        end if
      if ((Zmax-Zmin).LE.0.0D0) then
          Write(8,9010) 'Invalid geometry data.'
          Write(8,9105)
          DeckError = .True.
        end if

      if (Prob .lt. 4 .and. Options .eq. 2) then                       KHEL 4/11/93
         Write(8,9010) 'Forcing Function: Temperature Distribution for  KEHL 4/11/92
     &the Transient Case'                                               KHEL 4/11/92
         Do 11 j=1,Jmax                                                 KHEL 4/11/93
            Write(8,80) (Ffun(i,j), i=1,Imax)                           KHEL 4/11/93
   11    Continue                                                       KHEL 4/11/93
      end if                                                            KHEL 4/11/93
c
c ... write initial coolant temperature profile                         KHEL 7/5/93
```

```
      if ( (prob .eq. 2 .or. prob .eq. 3 ) .and. options .eq. 2 ) then   KHEL 7/5/93
         Write(8,9010) ' Initial coolant temperature profile '           KHEL 7/5/93
         Write(8,80) (Tcoolant(j), j=1, jmax)                            KHEL 7/5/93
      end if

      if (DeckError) then
          Write(8,9110)
          Stop
        end if

   10 Format (A80)
   20 Format (7F10.0)
   30 Format (I2)
   40 Format (2F10.0)
   50 Format (2F10.0,2I5)
   55 Format (2F10.0)
   60 Format (4F10.0)
   80 format (10F10.0)

  100 Format ('1',20('*'),' TFETC ',20('*')/
     a        ' ***** INPUT DATA SUMMARY FOR THE FOLLOWING CASE:')
  110 Format (/)
  120 Format (/' COOLANT TYPE: Molten Sodium-Potassium Alloy (NaK)'/,
     a        15X,'Potassium composition =',I3,'%')
  130 Format (/' COOLANT TYPE: Molten Potassium')
  140 Format (/' COOLANT TYPE: Molten Sodium')
  210 Format (' COOLANT MASS FLOW RATE:',F7.2,' kilograms per second.')
  310 Format (' TEMPERATURE OF COOLANT AT CHANNEL INLET:',F7.1,' K.')
  410 Format (/' TEMPERATURE OF CESIUM RESERVOIR:',F7.1,' K.')
  420 Format (' PRESSURE OF CESIUM VAPOR:',F7.1,' Torr.')
  510 Format (/' OUTPUT CURRENT FROM THE TOP OF THE TFE:   ',F7.1,
     a        ' Amperes.')
  610 Format (' OUTPUT CURRENT FROM THE BOTTOM OF THE TFE:',F7.1,
     a        ' Amperes.')
  710 Format (/' TOTAL THERMAL POWER PRODUCED IN THE TFE FUEL:',F7.1,
     a        ' Watts.')
  720 Format (' AVERAGE VOLUMETRIC HEAT GENERATION RATE FOR THE TFE',
     a        ' FUEL:',F7.1,' Watts.')
  810 Format (//' AXIAL POWER PROFILE TABLE ---------------------------'
     a //5X,'R is the ratio of the volumetric heat generation rate'/
     b 8X,'at point Z to the average volumetric heat generation'/
     c 8X,'rate for the TFE fuel.'//
     d 8X,'Z is measured from the ends of the emitter and collector'
     e 8X,'leads at the end of the TFE where the coolant enters.'//
     g 6X,'Z (cm)',8X,'R'/2(5X,8('-'))/)
  820 Format (6X,F5.1,7X,F6.3)
  830 Format (' CORRELATION FOR THE RATIO OF THE HEAT GENERATION RATE'/
     a    10X,' AT POSITION Z TO THE AVERAGE HEAT GENERATION RATE IN '/
     b    10X,' THE TFE FUEL:'//10X,'F = ',F10.4,'+',F10.4,
     c    ' * SIN((Z-Zmin)/(Zmax-Zmin)*3.14159)'/)
  835 Format (' AXIAL PEAK-TO-AVERAGE RATIO FOR HEAT GENERATION IS:',
     a        F7.4/)
  910 Format (/' ***** GEOMETRY DATA EDIT *****'/
     a  '  ***** RADIAL GEOMETRY *****'//
     a  30X,'Inside',9X,'Outside',17X,'Number of'/
     b  9X,'Region',15X,'Radius',9X,'Radius',6X,'Material',5X,
     c  'Interior'/31X,'(cm)',11X,'(cm)',18X,'Mesh Points'/1X,
     d  23('-'),4X,10('-'),5X,10('-'),4X,8('-'),3X,11('-'))
  920 Format (A23,4X,F10.6,5X,F10.6,8X,A5,5X,I3)
  930 Format (A23,4X,F10.6,5X,F10.6)
 1010 Format (/' EFFECTIVE EMISSIVITY FOR RADIANT HEAT TRANSFER FROM'/
     a    10X,' THE EMITTER SURFACE TO THE COLLECTOR SURFACE:',F10.6)
 1110 Format (/'  ***** AXIAL GEOMETRY *****'//
     a        ' AXIAL POSITION OF THE UPPER LIMIT FOR'/10X'THE FUELED'
     b        ' REGION OF THE TFE:',F10.6,' (cm)'/' AXIAL POSITION OF'
     c         ' THE LOWER LIMIT FOR'/10X'THE FUELED REGION OF THE TFE:'
     d         F10.6,' (cm)'/' AXIAL EXTENT OF THE FUELED REGION OF THE'
     e         ' TFE:',F10.6,' (cm) '/' TOTAL LENGTH OF THE TFE,'
     f         ' INCLUDING ELECTRODE LEADS:',F10.6,' (cm)')

 9010 Format (' ***** ',A66,' ***** ')
 9020 Format (10X,A23,' must be either',I2,' or',I2,'.')
 9030 Format (10X,A23,' should be between',F5.2,' and',F5.2,
     a        ', inclusive.')
 9040 Format (10X,A23,' should be greater than',F5.2,'.')
 9050 Format (10X,A23,' should be greater than or equal to',F5.2,'.')
 9060 Format (10X,'The inside radius for the ',A23,' region must be',
     a          ' less',10X,' than the outside radius.')
 9070 Format (10X,'The inside radius for the ',A23,' region must be',
     a   ' equal'/10X,' to the outside radius of the ',A23,' region.')
 9080 Format (10X,'No bare work function data is available for the '
     a   'material'/10X,A23,'.  Please add the necessary data to the '
     b   'Phi0 DATA'/10X,' statement in the Input subroutine.')
 9090 Format (10X,'There must be a total of',I3,' interior and'
     a   ' interfacial mesh points.')
 9100 Format (10X,'L should be greater than or equal to Zmax - Zmin.')
```

```
 9105 Format (10X,'Zmax should be greater than Zmin.')
 9110 Format (//' *****   EXECUTION HALTED DUE TO ERRORS IN THE',
     a         ' INPUT DECK *****'//)
 9120 Format (10X,A23,' must be one of ',I2,1X,I2,1X,I2,1X,I2,'.')     KHEL 4/11/93
 9130 Format (10X,'Tstop = ',F7.2,' must be less than Tstart = ',F7.2)  KHEL 6/26/93
 9140 Format (10X,' Print Time Step, TPRINT = ',F7.2,' Secs.')          KHEL 6/26/93
 9150 Format (10X,' Time Step Increament, delta t = ',F10.4,' Secs.')   KHEL 6/26/93
 9160 Format (10X,' Stop helium heating at the emitter temperature, Tst'KHEL 6/26/93
     &            ,'op = ',F7.2,'.')                                    KHEL 6/26/93
 9170 Format (10X,' Start Electron cooling at the emitter temperature, 'KHEL 6/26/93
     &            ,'Tstart = ',F7.2,'.')                                KHEL 6/26/93
 9180 Format (10X,' Simulation Period, TIME = ',F7.2,' Secs.')          KHEL 6/26/93
 9190 Format (10X,' Power-rise coefficient, tau = ',D9.3,','/,10X,      KHEL 6/26/93
     &            ' P(t) = P(0) * [ 1 - Exp ( - t / tau ) ] ','.')      KHEL 7/6/93
*9200 Format (10X,' Power-loss coefficient, PLOSS = ',F7.4,'.')         KHEL 6/26/93
 9210 Format (10X,' Mass-loss coefficients, A = ',F7.4,',',' B = ',F7.4,KHEL 6/26/93
     &            ',',' tau = ',D9.3,'.'/,10X, ' Mdot(t) = Mdot(0) * [ 'KHEL 7/6/93
     &            ' A +  B * Exp(- t / tau ) ] ' )                      KHEL 7/6/93
 9220 Format (10X,A23,' should be less than ',F5.2,'.')                 KHEL 6/26/93
 9230 Format (10X,' Total delayed neutron fraction, BETA = ',F9.5,' .') KHEL 6/26/93
 9240 Format (10X,' Negative Reactivity insertion in $, dollar = ',F7.2,KHEL 6/26/93
     &            '.')                                                  KHEL 6/26/93
 9250 Format (10X,' Reactor peroid, T = ',F7.2,' Secs.',/,10X,          KHEL 6/26/93
     & ' P(t) = P(0) [(1 - BETA * RHO)/(1 - RHO)] * Exp(- t / Period) ')KHEL 7/7/93
 9260 Format (' ***** ',A20,' ***** ')                                  KHEL 6/26/93
 9280 Format (10X,A5,' should be greater than or equal to',F7.2,',',/   KHEL 7/7/93
     &           ,A5,' should be greater than',F7.2,',',/               KHEL 7/7/93
     &           ,A5,' should be equal to',F7.2,'.')                    KHEL 7/7/93
 9290 Format (10X,A6,' should be less than or equal to',F7.2,',',F7.2,/ KHEL 7/7/93
     &           ,A6,' should be less than or equal to',F7.2,',',/      KHEL 7/7/93
     &           ,A6,' should be greater than',F7.2,'.')                KHEL 7/7/93
 9300 Format (10X,' Print Option, ipout = ',I1,'.')                     KHEL 7/7/93
 9310 Format (10X,' Linear Equations solved using Gaussian elimination')KHEL 7/19/93
 9320 Format (10X,' Linear Equations solved using Y12M Sparse solver')  KHEL 7/19/93
      D1 = 2.0D0*IR(10)
      D2 = 2.0D0*OR(10)
      De = D2-D1
      G1 = Mdot/(OR(10)**2.0D0-IR(10)**2.0D0)/Pi
      Do 2010 I = 1,10
 2010     Rbound(I)   = IR(I)
      Do 2020 I=1,9
 2020     Rmesh(I) = MeshPt(I)
      Do 2030 I=1,9,2
 2030     Mat(INT((I+1)/2)) = MatNum(I)

      Return
      end

      double precision function rperiod(dollar)
      implicit double precision (a-h,o-z)
      double precision dollar, result, period(14), rho(14), x
      data period /700.0, 400.0, 280.0, 200.0, 180.0, 125.0,  93.0,
     &              90.0,  80.0,  80.0,  80.0,  80.0,  80.0,  80.0 /
      data rho   / 0.02,  0.04,  0.06,  0.08,  0.10,  0.20,  0.40,
     &             0.60,  1.00, 10.0, 100.0, 200.0, 300.0, 600.0 /

      x = - dollar
      call intrpl(8,14,rho,period,1,x,result)
      rperiod = result

      return
      end
      double precision Function ccp(I,R2,T)
***       Units = J/Kg.K  ***

      Integer I,Rmesh(9),M(5)
      double precision R, Rbound(10), Temp, R2
      double precision T, ccp1
      Common /Rdata/ Rbound,Rmesh,M

      if (R2.LT.Rbound(1)) then
*              *** Void ( assumed to be air )
          Temp = 1.00488d3
          Goto 100
        endif
      if (R2.EQ.Rbound(1)) then
*              *** Void/Pellet boundary
          Temp = (1.00488d3 + ccp1(T,M(1)))/2
          Goto 100
        endif
      if ((R2.GT.Rbound(1)).AND.(R2.LT.Rbound(2))) then
          Temp = ccp1(T,M(1))
*              *** Fuel pellet
          Goto 100
        endif
      if (R2.EQ.Rbound(2)) then
```

```
*               *** Fuel Pellet outer surface
***        Does a gap exist between the fuel and the emmiter?  ***
           if (Rbound(2).NE.Rbound(3)) then
*               *** Gap
              Temp = (1.00488d3*(R(I+1)-R(I)) +
     a               ccp1(T,M(1))*(R(I)-R(I-1)))
     b                  /(R(I+1)-R(I-1))
            else
*               *** No Gap
              Temp = (ccp1(T,M(2))*(R(I+1)-R(I)) +
     a                ccp1(T,M(1))*(R(I)-R(I-1)))
     b                    /(R(I+1)-R(I-1))
            endif
           Goto 100
         endif
       if ((Rbound(2).NE.Rbound(3)).AND.((R2.GT.Rbound(2))
     a           .AND.(R2.LT.Rbound(3)))) then
*                *** Gap
           Temp = 1.00488d3
           Goto 100
         endif
       if ((R2.EQ.Rbound(3)).AND.(Rbound(2).NE.Rbound(3))) then
           Temp = (ccp1(T,M(2))*(R(I+1)-R(I)) + 1.00488d3*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
*                *** Emitter inner surface
           Goto 100
         endif
       if ((R2.GT.Rbound(3)).AND.(R2.LT.Rbound(4))) then
           Temp = ccp1(T,M(2))
*                *** Emitter
           Goto 100
         endif
       if (R2.EQ.Rbound(4)) then
*                *** Emitter outer surface (Cs vapor used in gap)
           Temp = (1.9D-3*(R(I+1)-R(I)) + ccp1(T,M(2))*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
           Goto 100
         endif
       if ((R2.GT.Rbound(4)).AND.(R2.LT.Rbound(5))) then
*                 *** Gap (Cesium vapor)
           Temp = 1.9D-3
           Goto 100
         endif
       if (R2.EQ.Rbound(5)) then
*                 *** Collector inner surface
           Temp = (ccp1(T,M(3))*(R(I+1)-R(I)) + 1.9D-3*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
           Goto 100
         endif
       if ((R2.GT.Rbound(5)).AND.(R2.LT.Rbound(6))) then
*                 *** Collector
           Temp = ccp1(T,M(3))
           Goto 100
         endif
       if (R2.EQ.Rbound(6)) then
*                 *** Collector outer surface
***        Does a gap exist between the collector and the insulator?  ***
           if (Rbound(6).NE.Rbound(7)) then
*                 *** Gap
              Temp = (1.00488d3*(R(I+1)-R(I)) +
     a                ccp1(T,M(3))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
            else
*                 *** No Gap
              Temp = (ccp1(T,M(4))*(R(I+1)-R(I)) +
     a                ccp1(T,M(3))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
            endif
           Goto 100
         endif
       if ((Rbound(6).NE.Rbound(7)).AND.((R2.GT.Rbound(6))
     a           .AND.(R2.LT.Rbound(7)))) then
*                 *** Gap
           Temp = 1.00488d3
           Goto 100
         endif
       if ((R2.EQ.Rbound(7)).AND.(Rbound(6).NE.Rbound(7))) then
*                 *** Insulator inner surface
           Temp = (ccp1(T,M(4))*(R(I+1)-R(I)) + 1.0D-2*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
           Goto 100
         endif
       if ((R2.GT.Rbound(7)).AND.(R2.LT.Rbound(8))) then
*                 *** Insulator
           Temp = ccp1(T,M(4))
           Goto 100
```

```
          endif
        if (R2.EQ.Rbound(8)) then
*                 *** Insulator outer surface
***         Does a gap exist between the insulator and the clad?  ***
            if (Rbound(8).NE.Rbound(9)) then
*                 *** Gap
              Temp = (1.00488d3*(R(I+1)-R(I)) +
     a                ccp1(T,M(4))*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
             else
*                 *** No Gap
              Temp = (ccp1(T,M(5))*(R(I+1)-R(I)) +
     a                ccp1(T,M(4))*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
             endif
            Goto 100
          endif
        if ((Rbound(8).NE.Rbound(9)).AND.((R2.GT.Rbound(8))
     a            .AND.(R2.LT.Rbound(9)))) then
*                 *** Gap
            Temp = 1.00488d3
            Goto 100
          endif
        if ((R2.EQ.Rbound(9)).AND.(Rbound(8).NE.Rbound(9))) then
*                 *** Clad inner surface
            Temp = (ccp1(T,M(5))*(R(I+1)-R(I)) + 1.00488d3*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
            Goto 100
          endif
        if ((R2.GT.Rbound(9)).AND.(R2.LT.Rbound(10))) then
*                 *** Clad
            Temp = ccp1(T,M(5))
            Goto 100
          endif
        if (R2.EQ.Rbound(10)) then
*                 *** Clad outer surface
*                 *** Clad
            Temp = ccp1(T,M(5))
            Goto 100
          endif

  100 ccp = Temp


        return
        end

        double precision Function ccp1(T,M1)
***         Units = J/Kg.K  ***

        Integer M1
        double precision Cpdata(8), MOcp(11), MOtm(11), Recp(10)
        double precision Retm(10), Alcp(7), Altm(7), wcp, ncp
        double precision k1, k2, k3, theta, ed, y, Rc, fcp, T1
        double precision T
*                                          Specific Heat J/Kg.K
*        Material #1 = UO2                 0.23d3
*        Material #2 = W                   1.33984d2
*        Material #3 = Nb                  2.72155d2
*        Material #4 = Nb1Zr
*        Material #5 = Mo                  2.55407d2
*        Material #6 = Re                  1.38171d2
*        Material #7 = Cs                  2.17724d2
*        Material #8 = Al2O3               8.374d2
*
         Data Cpdata/
     1            2.3d2     ,   1.33984d2,   2.72155d2,
     2            0         ,   2.55407d2,   1.38171d2,
     3            2.17724d2, 8.374d2/

        Data k1,   k2, k3/296.7,   2.43d-2, 8.745d7/

        Data theta,ed, Rc,y/535.285, 1.577d5, 8.3143, 2.0/

        Data MOcp/141.0, 224.0, 251.0, 261.0, 275.0, 285.0, 295.0,
     &          308.0, 330.0, 380.0, 459.0/

        Data MOtm/100.0, 200.0, 300.0, 400.0, 600.0, 800.0, 1000.0,
     &          1200.0, 1500.0, 2000.0, 2500.0/

        Data Recp/97.0, 127.0, 136.0, 139.0, 145.0, 151.0, 156.0, 162.0,
     &          171.0, 186.0/

        Data Retm/100.0, 200.0, 300.0, 400.0, 600.0, 800.0, 1000.0,
     &          1200.0, 1500.0, 2000.0/
```

```
      Data Alcp/778.782, 929.514, 1025.815, 1109.555, 1172.36,
     &          1256.100, 1306.344/

      Data Altm/293.15, 373.15, 473.15, 573.15, 773.15,
     &          1173.15, 1973.15/

      fcp(T1) = k1 * theta * theta * dexp(theta/T1) /
     &       ( T1*T1 * ( dexp(theta/T1) - 1.0 )**2.0 ) + k2 * T1 +
     &       ( y * k3 * ed / (2.0 * Rc * T1 * T1) ) * dexp(-ed/Rc/T1)

      wcp(T1) = 135.76 * (1.0d0 - 4805.0d0/T1/T1) + 9.1159d-3 * T1 +
     &          2.3134d-9 * (T1**3.0)

      ncp(T1) = ( 6.43d-2 + 0.772766d-5 * T1 + 0.234774d-8 * T1 * T1 )
     &          * 4187.0d0


      if ( M1 .eq. 1 ) then
*     ... UO2
          ccp1 = fcp(T)
          return
      else if ( M1 .eq. 2 ) then
*     ... W
          ccp1 = wcp(T)
          return
      else if ( M1 .eq. 3 ) then
*     ... Nb
          ccp1 = ncp(T)
          return
      else if ( M1 .eq. 4 ) then
*     ... Nb1Zr
          ccp1 = Cpdata(4)
          return
      else if ( M1 .eq. 5 ) then
*     ... Mo
          call intrpl(8,11,Motm,Mocp,1,T,ccp1)
          return
      else if ( M1 .eq. 6 ) then
*     ... Re
          call intrpl(8,10,Retm,Recp,1,T,ccp1)
          return
      else if ( M1 .eq. 7 ) then
*     ... Cs
          ccp1 = Cpdata(7)
          return
      else if ( M1 .eq. 8 ) then
*     ... Al2O3
          call intrpl(8,7,Altm,Alcp,1,T,ccp1)
      else
          write(8,100) 'Invalid material'
100       format(10x,'***',a22,'***')
          stop
      end if

      return
      end
      double precision Function rho(I,R2)
***       Units = Kg/cm^3  ***

      Integer I,Rmesh(9),M(5)
      double precision R, Rbound(10), Temp, R2, Rhdata(8)
      Common /Rdata/ Rbound,Rmesh,M
*                                       Density Kg/cm^3
*      Material #1 = UO2                1.098d-2
*      Material #2 = W                  19.3d-3
*      Material #3 = Nb                 8.57d-3
*      Material #4 = Nb1Zr
*      Material #5 = Mo                 10.2d-3
*      Material #6 = Re                 20.d-3
*      Material #7 = Cs                 1.9d-3
*      Material #8 = Al2O3              3.6264d-3
*
       Data Rhdata/1.098d-2,  19.3d-3  ,  8.57d-3,
     2             0.d0    ,  10.2d-3  ,  20.0d-3,
     3             1.9d-3  ,  3.6264d-3/

      if (R2.LT.Rbound(1)) then
*               *** Void ( assumed to be air )
          Temp = 1.293d-6
          Goto 100
        endif
      if (R2.EQ.Rbound(1)) then
*               *** Void/Pellet boundary
          Temp = (1.293d-6 + Rhdata(M(1)))/2
          Goto 100
        endif
```

```
      if ((R2.GT.Rbound(1)).AND.(R2.LT.Rbound(2))) then
          Temp = Rhdata(M(1))
*                *** Fuel pellet
          Goto 100
        endif
      if (R2.EQ.Rbound(2)) then
*                *** Fuel Pellet outer surface
***        Does a gap exist between the fuel and the emmiter?  ***
          if (Rbound(2).NE.Rbound(3)) then
*                *** Gap
              Temp = (1.293D-6*(R(I+1)-R(I)) +
     a                 Rhdata(M(1))*(R(I)-R(I-1)))
     b                       /(R(I+1)-R(I-1))
            else
*                *** No Gap
              Temp = (Rhdata(M(2))*(R(I+1)-R(I)) +
     a                  Rhdata(M(1))*(R(I)-R(I-1)))
     b                          /(R(I+1)-R(I-1))
            endif
          Goto 100
        endif
      if ((Rbound(2).NE.Rbound(3)).AND.((R2.GT.Rbound(2))
     a            .AND.(R2.LT.Rbound(3)))) then
*                 *** Gap
          Temp = 1.293d-6
          Goto 100
        endif
      if ((R2.EQ.Rbound(3)).AND.(Rbound(2).NE.Rbound(3))) then
          Temp = (Rhdata(M(2))*(R(I+1)-R(I)) + 1.293D-6*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
*                 *** Emitter inner surface
          Goto 100
        endif
      if ((R2.GT.Rbound(3)).AND.(R2.LT.Rbound(4))) then
          Temp = Rhdata(M(2))
*                 *** Emitter
          Goto 100
        endif
      if (R2.EQ.Rbound(4)) then
*                 *** Emitter outer surface (Cs vapor used in gap)
          Temp = (1.9D-3*(R(I+1)-R(I)) + Rhdata(M(2))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
          Goto 100
        endif
      if ((R2.GT.Rbound(4)).AND.(R2.LT.Rbound(5))) then
*                  *** Gap (Cesium vapor)
          Temp = 1.9D-3
          Goto 100
        endif
      if (R2.EQ.Rbound(5)) then
*                  *** Collector inner surface
          Temp = (Rhdata(M(3))*(R(I+1)-R(I)) + 1.9D-3*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
          Goto 100
        endif
      if ((R2.GT.Rbound(5)).AND.(R2.LT.Rbound(6))) then
*                  *** Collector
          Temp = Rhdata(M(3))
          Goto 100
        endif
      if (R2.EQ.Rbound(6)) then
*                  *** Collector outer surface
***        Does a gap exist between the collector and the insulator?  ***
          if (Rbound(6).NE.Rbound(7)) then
*                  *** Gap
              Temp = (1.293d-6*(R(I+1)-R(I)) +
     a                  Rhdata(M(3))*(R(I)-R(I-1)))
     a                     /(R(I+1)-R(I-1))
            else
*                  *** No Gap
              Temp = (Rhdata(M(4))*(R(I+1)-R(I)) +
     a                  Rhdata(M(3))*(R(I)-R(I-1)))
     a                     /(R(I+1)-R(I-1))
            endif
          Goto 100
        endif
      if ((Rbound(6).NE.Rbound(7)).AND.((R2.GT.Rbound(6))
     a            .AND.(R2.LT.Rbound(7)))) then
*                  *** Gap
          Temp = 1.293D-6
          Goto 100
        endif
      if ((R2.EQ.Rbound(7)).AND.(Rbound(6).NE.Rbound(7))) then
*                  *** Insulator inner surface
          Temp = (Rhdata(M(4))*(R(I+1)-R(I)) + 1.0D-2*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
```

```
          Goto 100
        endif
      if ((R2.GT.Rbound(7)).AND.(R2.LT.Rbound(8))) then
*                   *** Insulator
          Temp = Rhdata(M(4))
          Goto 100
        endif
      if (R2.EQ.Rbound(8)) then
*                   *** Insulator outer surface
***       Does a gap exist between the insulator and the clad?  ***
          if (Rbound(8).NE.Rbound(9)) then
*                   *** Gap
              Temp = (1.293D-6*(R(I+1)-R(I)) +
     a                  Rhdata(M(4))*(R(I)-R(I-1)))
     a                     /(R(I+1)-R(I-1))
            else
*                   *** No Gap
              Temp = (Rhdata(M(5))*(R(I+1)-R(I)) +
     a                  Rhdata(M(4))*(R(I)-R(I-1)))
     a                     /(R(I+1)-R(I-1))
            endif
          Goto 100
        endif
      if ((Rbound(8).NE.Rbound(9)).AND.((R2.GT.Rbound(8))
     a            .AND.(R2.LT.Rbound(9)))) then
*                   *** Gap
          Temp = 1.293D-6
          Goto 100
        endif
      if ((R2.EQ.Rbound(9)).AND.(Rbound(8).NE.Rbound(9))) then
*                   *** Clad inner surface
          Temp = (Rhdata(M(5))*(R(I+1)-R(I)) + 1.293D-6*(R(I)-R(I-1)))
     a                     /(R(I+1)-R(I-1))
          Goto 100
        endif
      if ((R2.GT.Rbound(9)).AND.(R2.LT.Rbound(10))) then
*                   *** Clad
          Temp = Rhdata(M(5))
          Goto 100
        endif
      if (R2.EQ.Rbound(10)) then
*                   *** Clad outer surface
*                   *** Clad
          Temp = Rhdata(M(5))
          Goto 100
        endif

  100 rho = Temp

      end
      subroutine output(time,imax,jmax, v, qec, jdens, emheat,
     1                  tcoolant, colheat, qch, qrad, qcscond,
     2                  t, current, prob, ThPower, ipout)
      implicit double precision (a-h,o-z)
***********************************************************************
*                                                                     *
*          Write the results to the file tfehx.out                    *
*                                                                     *
***********************************************************************

      double precision tr, itop, ibottom, tcoolant(jmax)
      double precision ems, phie, v(jmax), z, qec(jmax), jdens(jmax)
      double precision emheat(jmax), colheat(jmax), qch(jmax), current
      double precision qrad(jmax), qcscond(jmax), time, t(imax,jmax)
      double precision Tinlet, De, G1, W, Dout, Din, mdot, ThPower
      integer i, j, prob, ipout
      character*80 title
      common /input/ tr, ems, phie, itop, ibottom, title
      common /coolprop/  tinlet, de, g1, w, dout, din, mdot

      write(8,1300)
      write(8,1400) title
      write(8,1500) time
      if ( ipout .eq. 1 ) then
      write(8,1600)
c
c... output temp. distribution for fuel
         do i=1,imax
            do j=1,jmax
               write(8,200) i,j, t(i,j)
            end do
         end do
 200     format (' t(',i3,',',i3,') = ',f13.7)
         do j=1,jmax
            write(8,1900) j,tcoolant(j)
         end do
```

```
      end if
      write(8,1700) tcoolant(jmax)
      if ( prob .eq. 3 ) then
           write(8,1800) mdot
      end if
      write(8,300) v(1),v(jmax)
      write(8,400) current, v(1)*ibottom+v(jmax)*itop
 300  format(' Voltage across bottom of cell: ',F13.7/
     a        ' Voltage across top of cell: ',F13.7)
 400  format(' Output current = ',F13.7/' Output electrical power = ',
     a         F13.7)
      write(8,450) ThPower
 450  format(' Total Thermal power = ',F13.7)
      write(8,500)
 500  format(//8X,'Z',19X,'V',18X,'Qec',16X,'Jdens'/X,3(15('-'),5X),
     A         15('-')/)
      do 600 j=1,jmax
         Write(8,700) Z(j), V(j), Qec(j), Jdens(j)
 600  continue
 700  format(X,3(F15.8,5X),F15.8)
      write(8,800)
 800  format(//8X,'Z',17X,'EmHeat',13X,'ColHeat'/X,2(15('-'),5X),
     A        15('-')/)
      do 900 j=1,jmax
         write(8,1000) z(j), emheat(j), colheat(j)
 900  continue
 1000 format(X,2(F15.8,5X),F15.8)
      write(8,1100)
 1100 format(//8X,'Z',18X,'Qch',16X,'Qrad',15X,'QCsCond'/
     A        X,3(15('-'),5X),15('-')/)
      do 1200 j=1,jmax
         write(8,700) Z(j), Qch(j), Qrad(j), QCsCond(j)
 1200  continue


 1300 format ('1',20('*'),' TFETC ',20('*')/
     a         ' ***** RESULTS FOR THE FOLLOWING CASE:')
 1400 format (A80)
 1500 format ( ' TIME = ',f15.8,/)
 1600 format (' TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---'/)
 1700 Format (/' Temperature of coolant at core exit: ',F10.3,
     a            ' degrees K.')
1800  Format (/' Mass flow rate = ', F10.3 )
1900  Format (' tcool(',i3,') = ',f13.7 )

      return
      end
      subroutine helium(v,qec,qch,jdens,emheat,colheat)
      parameter ( jmax = 10 )
      implicit double precision (a-h,o-z)
      double precision v(jmax),qec(jmax)
      double precision jdens(jmax),emheat(jmax),colheat(jmax)
      double precision qch(jmax)
      integer i


      do i=1,jmax
           qec(i)     = 0.d0
           v(i)       = 0.d0
           jdens(i)   = 0.d0
           emheat(i)  = 0.d0
           colheat(i) = 0.d0
           qch(i)     = 0.d0
      end do

      return
      end

      double precision Function GapCond(Te,Tc,Tstop,Tstart,Prob,Tr,D)

      integer Prob
      double precision Te,Tc,Tstop,Tr,Tr1,D,Pcs,K,Hcond,Tstart
      external Hcond

      if ( Prob .eq. 1 .and. Te .le. Tstop ) then
*          ... helium heating phase
           Tr1 = (Te+Tc)/2.0d0
           K = Hcond(Tr1)
           GapCond = K*(Te-Tc)/D
*          if ( GapCond .lt. 0.0d0 ) GapCond = 0.0d0
      else
           K = 5.5D-5
*          ... electron-cooling phase
           if ( Te .lt. Tstart ) then
                Tr1 = Te * Tr/Tstart
           else
```

```
            Trl = Tr
         end if
         Pcs = 2.45D+8 * exp(-8910.D0/Trl)/SQRT(Trl)
         GapCond = K*(Te-Tc)/(D + 1.15D-5*(Te-Tc)/Pcs)
*        if ( GapCond .lt. 0.0d0 ) GapCond = 0.0d0
      end if

      End

      double precision function HCond(T)
************************************************************************
*                                                                      *
*     Calculate the thermal conductivity of helium as a function of   *
*     temperature T, in Kelvin. Data is taken from "Introduction to    *
*     Heat Tranfer", by F. P. Incropera and D. P. DeWitt,              *
*     Table A.4 page 683.                                              *
*                                                                      *
*                                                                      *
*     Units are in W/cm.K                                              *
************************************************************************

      double precision k(22), temp(22), result, T
      data temp / 100.0d0, 120.0d0, 140.0d0, 160.0d0, 180.0d0, 200.0d0,
     &            220.0d0, 240.0d0, 260.0d0, 280.0d0, 300.0d0, 350.0d0,
     &            400.0d0, 450.0d0, 500.0d0, 600.0d0, 650.0d0, 700.0d0,
     &            750.0d0, 800.0d0, 900.0d0,1000.0d0 /
      data k    /  73.0d0,  81.9d0,  90.7d0,  99.2d0, 107.2d0, 115.1d0,
     &            123.1d0, 130.0d0, 137.0d0, 145.0d0, 152.0d0, 170.0d0,
     &            187.0d0, 204.0d0, 220.0d0, 252.0d0, 264.0d0, 278.0d0,
     &            291.0d0, 304.0d0, 330.0d0, 354.0d0 /

c
c
      call intrpl(8,22,temp,k,1,T,result)
      HCond = result * 1.0d-5

      return
      end
      Subroutine TConvect(tnow,dt,tfcool)
**********************************************************
*                   Subroutine TConvect                  *
*            Written by: Abdullah S. Al-Kheliewi         *
*                     --------------                     *
*                        6/10/1993                       *
*                                                       *
*     Computes the temperature of the coolant within     *
*     cylindrical flow channels by solving the transien  *
*     partial differential equation for temperature rise*
*     through the core ( equation 6.8.6 in Elements of   *
*     Nuclear Reactor Design, J. Weisman ed., Kreiger    *
*     Publishing Company, 1983, with CpdT substituted    *
*     for dh.)  An explicit finite difference scheme is  *
*     solve the partial differential equation.           *
*     This treatment allows the temperature dependences  *
*     of the coolant properties to be included in the    *
*     analysis.  The output for this module is the axial*
*     temperature profile of the coolant within the      *
*     flow channel.                                      *
*                                                        *
*                                                        *
**********************************************************

      implicit double precision (a-h,o-z)
      parameter ( N = 500 , jmax = 10 )
*     double precision T, Told, Tinlet, h, z, mdot
      double precision T, Tinlet, h, z, mdot,tfcool(jmax)
      double precision Cp, HeatFlux, Rbound(10), zcool(jmax), hcool
      double precision De, G, W, CoolTbl(2000,2), Zmax, Zmin, D2, D1
      double precision tnow, dt, rhonak, dum1, dum2, dum3, dum4, dum5
      double precision TOLD(N), HF(N), dt1
      Integer i, j, Kmax, M
      Integer Rmesh(9), Mat(5)
      Common /CoolProp/  Tinlet, De, G, W, D2, D1, mdot
      Common /TTAB/ CoolTbl
      Common /Zdata/ Zmin, Zmax, Kmax
      Common /Rdata/ Rbound, Rmesh, Mat
      external rhonak


      dt1 = 0.5
      if ( dt .lt. dt1 ) dt1 = dt/10.0d0
      Kmax = 10
      h = (Zmax-Zmin)/(N-1)
      CoolTbl(1,1) = Zmin
      CoolTbl(1,2) = Tinlet
```

```
      hcool = (Zmax-Zmin)/(Jmax-1)
      do j=1,jmax
         zcool(j) = (j-1)*hcool + Zmin
      end do

      if ( tnow .lt. dt ) then
           do j=1,N
              z = (j-1)*h + Zmin
              CoolTBl(j,1) = z
           end do
           call intrpl(8,jmax,zcool,tfcool,N,CoolTBl(1,1),CoolTBl(1,2))
      end if

      do j=1,n
         z = (j-1)*h + Zmin
        HF(j) = HeatFlux(z)
      end do

      M = idint( dt/dt1 )
      do i=1,M

      do j=1,N
        TOLD(j) = CoolTbl(j,2)
      end do

      do 100 j=2, N
          dum1 = HF(j)
          dum2 = Cp(TOLD(j),W)
          dum3 = RhoNak(TOLD(j),W)
          dum4 = dum3/dt1
          dum5 = G/h

          T =  ( 4.0*dum1/dum2/De + dum4*TOLD(j) + dum5*CoolTbl(j-1,2) )
     a          / ( dum4 + dum5 )

          CoolTbl(j,2) = T

  100 Continue

      end do

      End
      double precision Function RhoNaK(T,W)
*************************************************************
*      Uses correlations from the Sodium-NaK Engineering     *
*      Handbook (O. Foust, ed.; vol. 1 pp. 16-17) to return  *
*      the value of the density of the NaK-78 coolant for    *
*      a given temperature T and potassium weight fraction W  *
*      for the coolant (e.g. eutectic NaK-78 has W=78%.)      *
*      Only single phase coolants are modeled.  If the        *
*      temperature of the coolant is higher than the boiling  *
*      point of NaK at the given sodium-potassium composition,*
*      this routine reports the error and halts the program.  *
*                                                             *
*      Units are in kilogram/cm^3                             *
*                                                             *
*************************************************************
      double precision T, BoilingPt, W

      BoilingPt = ((756.5-881.4)*W + 881.4) + 273.1
      If (T.GT.BoilingPt) then
c         Write(*,100) T, INT(W*100)
          Write(8,100) T, INT(W*100)
          Stop
        EndIf

      RhoNak = 9.4971d-4 - 2.473d-7 * T

  100 Format(' The temperature T=',F7.1,' degrees K is higher than '/
     a        ' the boiling point for NaK-',I3,//
     b        ' Execution terminated in RhoNaK.'//)
      End
      Subroutine TFEHX(Tcoolant,t,Isolver)

C     Implicit NONE
      Integer Imax, Kmax, N, J, TabFlag, Isolver                      KHEL 7/18/93
      Parameter (Imax = 10, Kmax = 10)
      Real*8 T(Imax,Kmax), C1, Kcond, R, Z, Temp, R1, Z1
C     Real*8 DeltaT
      Real*8 R2, Z2, R3, Z3, Temp3, G, H, T1, Temm(Kmax), Tcol(Kmax)
      Real*8 DeltaZ, C3, T2, Tcoolant(Kmax), ErrSqrd, V(Kmax), Zmin
      Real*8 A(Imax*Kmax+1,Imax*Kmax), X(Imax*Kmax), Tinlet, V0
      Real*8 Rbound(10), Sig, Ems, QGapCond, Tr, Qch(Kmax), mdot      KHEL 4/26/93
      Real*8 HeffE(Kmax), HeffC(Kmax),  Current, Qec(Kmax), Jdens(Kmax)
      Real*8 QTable(Kmax), De, G1, W, Pi, PhiE, EmHeat(Kmax)
      Real*8 ColHeat(Kmax), Zmax, Itop, Ibottom, Cden_av1, Cden_av2
```

```
      Real*8 Vguess(Kmax), Te0, Tc0, TeAv(Kmax), TcAv(Kmax)
      Real*8 Dout, Din, Q3ave, Error, PowerTabl(2,100), Seconds
      Real*8 Qrad(Kmax),QCsCond(Kmax),MaxError, Timer(3), TotalTime
      Integer I, K, IC, I1, I2, I3, I4, J2, Rmesh(9), K2, I9, Mat(5)
      Character*11 Start,Step1,Step2,Stop
      Character*80 Title
      Common /GaussMAIN/ A, X, N
      Common /Rdata/ Rbound,Rmesh,Mat
      Common /Zdata/ Zmin,Zmax,K2
      Common /QTAB/ QTable
      Common /Input/ Tr, Ems, PhiE, Itop, Ibottom, Title
      Common /CoolProp/  Tinlet, De, G1, W, Dout, Din, mdot          KHEL 4/26/93
      Common /Steady/ Emheat,ColHeat,Qch,Qrad,QcsCond,Qec,Jdens,     KHEL 6/30/93
     &                Current                                        KHEL 6/30/93
      Common /PowerData/ Q3ave, PowerTabl, TabFlag
      Data Pi/3.1415926D0/

      Call TIME(Start)
      Do 10 I=1,3
 10      Timer(I)=0

      Te0 = 1900D0
      Tc0 = 880D0
      Sig = 5.67D-12
      V0 = 0.60D0
      N = Imax*Kmax
      IC = 1

      Do 100 K=1, Kmax
           Temp = Tinlet
           Do 100 I=1, Imax
                If (R(I).EQ.Rbound(5)) then
                     T(I,K) = Tc0
                   ElseIf (R(I).GT.Rbound(5)) then
                     T(I,K) = (2.0D0)*(Rbound(10)-R(I))
     a                  /(Rbound(10)-Rbound(5)) + Temp
                   Else
                     T(I,K) = Te0
                   EndIf
 100    Continue
       Do 101 K=1, Kmax
 101      Vguess(K) = V0

 102  Do 105 K=1,Kmax
          QTable(K) = Kcond(Imax,R(Imax),T(Imax,K))
     a                *(T(Imax-1,K) - T(Imax,K))/(R(Imax) - R(Imax-1))
 105    Continue

*     Write(*,107)
      Call TIME(Step1)
      Call Convect
      Call CoolantTemp(Tcoolant)
*     Write(*,108)
      Call TIME(Step2)
* 107 Format(' Entering Convect/CoolantTemp......')
* 108 Format(' .......Leaving Convect/CoolantTemp')
      Timer(1) = Timer(1) + Seconds(Step2) - Seconds(Step1)

 110  I1 = 1
      Do 120 I9=1,2
 120      I1 = I1 + Rmesh(I9)
      I2 = 1
      Do 125 I9=1,3
 125      I2 = I2 + Rmesh(I9)
      I3 = I2+1
      I4 = 1
      Do 128 I9=1,5
 128      I4 = I4 + Rmesh(I9)
      Do 150 K=1,Kmax
          Temm(K) = T(I2,K)
          Tcol(K) = T(I2+1,K)
          TeAv(K) = T(I1,K)*(R(I1+1)**2+2*R(I1+1)*R(I1)-3*R(I1)**2)/4
          TcAv(K) = T(I3,K)*(R(I3+1)**2+2*R(I3+1)*R(I3)-3*R(I3)**2)/4
          Do 130 I=I1+1,I2-1
              TeAv(K) = TeAv(K) + T(I,K)*(R(I+1)**2+2*R(I)*
     a                        (R(I+1)-R(I-1))-R(I-1)**2)/4
 130        Continue
          Do 140 I=I3+1,I4-1
              TcAv(K) = TcAv(K) + T(I,K)*(R(I+1)**2+2*R(I)*
     a                        (R(I+1)-R(I-1))-R(I-1)**2)/4
 140        Continue
          TeAv(K) = TeAv(K) + T(I2,K)*(3*R(I2)**2-2*R(I2)*R(I2-1)
     a                    -R(I2-1)**2)/4
          TcAv(K) = TcAv(K) + T(I4,K)*(3*R(I4)**2-2*R(I4)*R(I4-1)
     a                    -R(I4-1)**2)/4
          TeAv(K) = TeAv(K)/(R(I2)**2-R(I1)**2)
```

```
          TcAv(K) = TcAv(K)/(R(I4)**2-R(I3)**2)
  150   Continue
        Cden_av1 = Ibottom/(Pi*R(I2)*2*(Zmax-Zmin)/2)
        Cden_av2 = Itop/(Pi*R(I2)*2*(Zmax-Zmin)/2)

*       Write(*,154)
        Call TIME(Step1)
        Call Cylcon6(Temm,TeAv,Tcol,TcAv,Tr,PhiE,R(I3)-R(I2),Cden_av1,
       1 Cden_av2,Zmax-Zmin,2*R(I2),R(I2)-R(I1),R(I4)-R(I3),Kmax,
       2 Vguess, V, Qec, Jdens, EmHeat, ColHeat)
        Call TIME(Step2)
*       Write(*,155)
* 154 Format(' Entering CYLCON6.....')
* 155 Format(' ......Leaving CYLCON6')
        Timer(2) = Timer(2) + Seconds(Step2) - Seconds(Step1)

        Current = Itop+Ibottom
        Do 180 K=1,Kmax
            Vguess(K) = V(K)
  180      Qch(K) = Qec(K)-Jdens(K)*V(K)


        Do 2005 K=1, Kmax
        Do 2000 I=1, Imax
            I2 = (K-1)*Imax + I
            Do 250 J=1,Imax*Kmax+1
                A(J,I2) = 0.0D000
  250         Continue
**        I = 1, K = 2, Kmax -1  ****
            If ((I.EQ.1).AND.((K.NE.1).AND.(K.NE.Kmax))) Then
                R1 = R(I)
                R3 = R(I+1)
                R2 = (R3 + R1)/2
                Z1 = Z(K)
                DeltaZ = (Z(K+1)-Z(K-1))/2
                T1 = (T(I+1,K) + T(I,K))/2
                C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
                C1 = C3
                J2 = (K-1)*Imax + I+1
                A(J2,I2) = C3

                Temp3 = (R3**2 + 2*R1*R3 - 3*R1**2)/4
                Z3 = Z(K+1)
                Z2 = (Z3 + Z1)/2
                T1 = (T(I,K+1) + T(I,K))/2
                C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
                C1 = C1 + C3
                J2 = K*Imax + I
                A(J2,I2) = C3

                Z3 = Z(K-1)
                Z2 = (Z3 + Z1)/2
                T1 = (T(I,K-1) + T(I,K))/2
                C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
                C1 = C1 + C3
                J2 = (K-2)*Imax + I
                A(J2,I2) = C3

                A(Imax*Kmax+1,I2) = - G((R1+R2)/2,Z1)*DeltaZ * Temp3

                Goto 1000

              EndIf

            If ((I.EQ.1).AND.(K.EQ.1)) Then
**        I = 1, K = 1  ****
                R1 = R(I)
                R3 = R(I+1)
                R2 = (R3 + R1)/2
                Z1 = Z(K)
                Z3 = Z(K+1)
                DeltaZ = (Z3-Z1)/2
                T1 = (T(I+1,K) + T(I,K))/2
                C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
                C1 = C3
                J2 = (K-1)*Imax + I+1
                A(J2,I2) = C3

                Temp3 = (R3**2 - 3*R1**2 + 2*R1*R3)/4
                Z2 = (Z3 + Z1)/2
                T1 = (T(I,K+1) + T(I,K))/2
                C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
                C1 = C1+C3
                J2 = K*Imax + I
                A(J2,I2) = C3
```

```
          A(Imax*Kmax+1,I2) = - G((R1+R2)/2,(Z1+Z2)/2)*DeltaZ
     a           * Temp3
          Goto 1000

        EndIf

      If ((I.EQ.1).AND.(K.EQ.Kmax)) Then
**    I = 1 and K = Kmax ****
          R1 = R(I)
          R3 = R(I+1)
          R2 = (R3 + R1)/2
          Z1 = Z(K)
          Z3 = Z(K-1)
          DeltaZ = (Z1-Z3)/2
          T1 = (T(I+1,K) + T(I,K))/2
          C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
          C1 = C3
          J2 = (K-1)*Imax + I+1
          A(J2,I2) = C3

          Temp3 = (R3**2 - 3*R1**2 + 2*R1*R3)/4
          Z2 = (Z3 + Z1)/2
          T1 = (T(I,K-1) + T(I,K))/2
          C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
          C1 = C1 + C3
          J2 = (K-2)*Imax + I
          A(J2,I2) = C3

          A(Imax*Kmax+1,I2) = - G((R1+R2)/2,(Z1+Z2)/2)*DeltaZ
     a           * Temp3
          Goto 1000

        EndIf

************* Start of appended text ****************
**    Collector surface, K = 2, Kmax -1  ****
      If ((R(I).EQ.Rbound(5)).AND.((K.NE.1).AND.(K.NE.Kmax))) Then
          R1 = R(I)
          R3 = R(I+1)
          R2 = (R3 + R1)/2
          Z1 = Z(K)
          DeltaZ = (Z(K+1)-Z(K-1))/2
          T1 = (T(I+1,K) + T(I,K))/2
          C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
          C1 = C3
          J2 = (K-1)*Imax + I+1
          A(J2,I2) = C3

          Temp3 = (R3**2 + 2*R1*R3 - 3*R1**2)/4
          Z3 = Z(K+1)
          Z2 = (Z3 + Z1)/2
          T1 = (T(I,K+1) + T(I,K))/2
          C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
          C1 = C1 + C3
          J2 = K*Imax + I
          A(J2,I2) = C3

          Z3 = Z(K-1)
          Z2 = (Z3 + Z1)/2
          T1 = (T(I,K-1) + T(I,K))/2
          C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
          C1 = C1 + C3
          J2 = (K-2)*Imax + I
          A(J2,I2) = C3

          HeffC(K) = (Qch(K)
     b    + Sig*Ems*((T(I-1,K))**4-(T(I,K))**4)*R(I-1)/R(I)
     c    + QGapCond(T(I-1,K),T(I,K),Tr,R(I)-R(I-1)))
     d    *2*R(I-1)*DeltaZ/(T(I-1,K)-T(I,K))
          C1 = C1 + HeffC(K)
          J2 = (K-1)*Imax + I-1
          A(J2,I2) = HeffC(K)

          A(Imax*Kmax+1,I2) = -(G((R1+R2)/2,Z1)+ColHeat(K))
     a           *DeltaZ*Temp3
          Goto 1000

        EndIf

**    Collector surface, K = 1  ****
      If ((R(I).EQ.Rbound(5)).AND.(K.EQ.1)) Then
          R1 = R(I)
          R3 = R(I+1)
          R2 = (R3 + R1)/2
          Z1 = Z(K)
          Z3 = Z(K+1)
```

```
            DeltaZ = (Z3-Z1)/2
            T1 = (T(I+1,K) + T(I,K))/2
            C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
            C1 = C3
            J2 = (K-1)*Imax + I+1
            A(J2,I2) = C3

            Temp3 = (R3**2 - 3*R1**2 + 2*R1*R3)/4
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K+1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
            C1 = C1+C3
            J2 = K*Imax + I
            A(J2,I2) = C3

            HeffC(K) = (Qch(K)
     b      + Sig*Ems*((T(I-1,K))**4-(T(I,K))**4)*R(I-1)/R(I)
     c      + QGapCond(T(I-1,K),T(I,K),Tr,R(I)-R(I-1)))
     d      *2*R(I-1)*DeltaZ/(T(I-1,K)-T(I,K))
            C1 = C1 + HeffC(K)
            J2 = (K-1)*Imax + I-1
            A(J2,I2) = HeffC(K)

            A(Imax*Kmax+1,I2) = -(G((R1+R2)/2,Z1)+ColHeat(K))
     a          *DeltaZ*Temp3
            Goto 1000

          EndIf

**      Collector surface and K = Kmax ****
          If ((R(I).EQ.Rbound(5)).AND.(K.EQ.Kmax)) Then
            R1 = R(I)
            R3 = R(I+1)
            R2 = (R3 + R1)/2
            Z1 = Z(K)
            Z3 = Z(K-1)
            DeltaZ = (Z1-Z3)/2
            T1 = (T(I+1,K) + T(I,K))/2
            C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
            C1 = C3
            J2 = (K-1)*Imax + I+1
            A(J2,I2) = C3

            Temp3 = (R3**2 - 3*R1**2 + 2*R1*R3)/4
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K-1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
            C1 = C1 + C3
            J2 = (K-2)*Imax + I
            A(J2,I2) = C3

            HeffC(K) = (Qch(K)
     b      + Sig*Ems*((T(I-1,K))**4-(T(I,K))**4)*R(I-1)/R(I)
     c      + QGapCond(T(I-1,K),T(I,K),Tr,R(I)-R(I-1)))
     d      *2*R(I-1)*DeltaZ/(T(I-1,K)-T(I,K))
            C1 = C1 + HeffC(K)
            J2 = (K-1)*Imax + I-1
            A(J2,I2) = HeffC(K)

            A(Imax*Kmax+1,I2) = -(G((R1+R2)/2,Z1)+ColHeat(K))
     a          *DeltaZ*Temp3
            Goto 1000

          EndIf

**      Emitter surface, K = 2, Kmax -1  ****
          If ((R(I).EQ.Rbound(4)).AND.((K.NE.1).AND.(K.NE.Kmax))) Then
            R1 = R(I)
            Z1 = Z(K)
            DeltaZ = (Z(K+1)-Z(K-1))/2
            R3 = R(I-1)
            R2 = (R3 + R1)/2
            T1 = (T(I-1,K) + T(I,K))/2
            C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
            C1 = C3
            J2 = (K-1)*Imax + I-1
            A(J2,I2) = C3

            Temp3 = (3*R1**2 - R3**2 - 2*R1*R3)/4
            Z3 = Z(K+1)
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K+1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
            C1 = C1 + C3
            J2 = K*Imax + I
            A(J2,I2) = C3
```

```
            Z3 = Z(K-1)
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K-1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
            C1 = C1 + C3
            J2 = (K-2)*Imax + I
            A(J2,I2) = C3

            HeffE(K) = (Qec(K)
     b      + Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
     c      + QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I)))
     d      *2*R(I)*DeltaZ/(T(I,K)-T(I+1,K))
            Qrad(K)    = Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
            QCsCond(K) = QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I))
            C1 = C1 + HeffE(K)
            J2 = (K-1)*Imax + I+1
            A(J2,I2) = HeffE(K)

            A(Imax*Kmax+1,I2) = -(G((R1+R2)/2,Z1)+EmHeat(K))
     a           *DeltaZ*Temp3
            Goto 1000

          EndIf

**      Emitter surface, K = 1  ****
          If ((R(I).EQ.Rbound(4)).AND.(K.EQ.1)) Then
            R1 = R(I)
            Z1 = Z(K)
            Z3 = Z(K+1)
            DeltaZ = (Z3-Z1)/2
            R3 = R(I-1)
            R2 = (R3 + R1)/2
            T1 = (T(I-1,K) + T(I,K))/2
            C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
            C1 = C3
            J2 = (K-1)*Imax + I-1
            A(J2,I2) = C3

            Temp3 = (3*R1**2 - R3**2 +2*R1*R3)/4
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K+1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
            C1 = C1 + C3
            J2 = K*Imax + I
            A(J2,I2) = C3

            HeffE(K) = (Qec(K)
     b      + Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
     c      + QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I)))
     d      *2*R(I)*DeltaZ/(T(I,K)-T(I+1,K))
            Qrad(K)    = Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
            QCsCond(K) = QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I))
            C1 = C1 + HeffE(K)
            J2 = (K-1)*Imax + I+1
            A(J2,I2) = HeffE(K)

            A(Imax*Kmax+1,I2) = -(G((R1+R2)/2,(Z1+Z2)/2)+EmHeat(K))
     a           *DeltaZ*Temp3
            Goto 1000

          EndIf

**      Emitter surface, K = Kmax  ****
          If ((R(I).EQ.Rbound(4)).AND.(K.EQ.Kmax)) Then
            R1 = R(I)
            Z1 = Z(K)
            Z3 = Z(K-1)
            DeltaZ = (Z1-Z3)/2
            R3 = R(I-1)
            R2 = (R3 + R1)/2
            T1 = (T(I-1,K) + T(I,K))/2
            C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
            C1 = C3
            J2 = (K-1)*Imax + I-1
            A(J2,I2) = C3

            Temp3 = (3*R1**2 - R3**2 +2*R1*R3)/4
            Z2 = (Z3 + Z1)/2
            T1 = (T(I,K-1) + T(I,K))/2
            C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
            C1 = C1 + C3
            J2 = (K-2)*Imax + I
            A(J2,I2) = C3

            HeffE(K) = (Qec(K)
     b      + Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
```

```
c        + QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I)))
d        *2*R(I)*DeltaZ/(T(I,K)-T(I+1,K))
         Qrad(K)    = Sig*Ems*((T(I,K))**4 - (T(I+1,K))**4)
         QCsCond(K) = QGapCond(T(I,K),T(I+1,K),Tr,R(I+1)-R(I))
         C1 = C1 + HeffE(K)
         J2 = (K-1)*Imax + I+1
         A(J2,I2) = HeffE(K)

         A(Imax*Kmax+1,I2) = -(G((R2+R1)/2,(Z1+Z2)/2)+EmHeat(K))
a              *DeltaZ*Temp3
         Goto 1000

       EndIf

     If (((I.NE.1).AND.(I.NE.Imax)).AND.(K.EQ.1)) Then
**   K = 1 and I = 2, Nmax -1 ****
         R1 = R(I)
         R3 = R(I+1)
         R2 = (R3 + R1)/2
         Z1 = Z(K)
         Z3 = Z(K+1)
         DeltaZ = (Z3-Z1)/2
         T1 = (T(I+1,K) + T(I,K))/2
         C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
         C1 = C3
         J2 = (K-1)*Imax + I+1
         A(J2,I2) = C3

         R3 = R(I-1)
         R2 = (R3 + R1)/2
         T1 = (T(I-1,K) + T(I,K))/2
         C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
         C1 = C1 + C3
         J2 = (K-1)*Imax + I-1
         A(J2,I2) = C3

         R2 = R(I+1)
         Temp3 = (R2**2 - R3**2 +2*R1*(R2-R3))/4
         Z2 = (Z3 + Z1)/2
         T1 = (T(I,K+1) + T(I,K))/2
         C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
         C1 = C1 + C3
         J2 = K*Imax + I
         A(J2,I2) = C3

         A(Imax*Kmax+1,I2) = - G(R1,(Z1+Z2)/2)*DeltaZ * Temp3

         Goto 1000

       EndIf

       If (((I.NE.1).AND.(I.NE.Imax)).AND.(K.EQ.Kmax)) Then
**   K = Kmax and I = 2, Imax -1 ****
         R1 = R(I)
         R3 = R(I+1)
         R2 = (R3 + R1)/2
         Z1 = Z(K)
         Z3 = Z(K-1)
         DeltaZ = (Z1-Z3)/2
         T1 = (T(I+1,K) + T(I,K))/2
         C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
         C1 = C3
         J2 = (K-1)*Imax + I+1
         A(J2,I2) = C3

         R3 = R(I-1)
         R2 = (R3 + R1)/2
         T1 = (T(I-1,K) + T(I,K))/2
         C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
         C1 = C1 + C3
         J2 = (K-1)*Imax + I-1
         A(J2,I2) = C3

         R2 = R(I+1)
         Temp3 = (R2**2 - R3**2 +2*R1*(R2-R3))/4
         Z2 = (Z3 + Z1)/2
         T1 = (T(I,K-1) + T(I,K))/2
         C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
         C1 = C1 + C3
         J2 = (K-2)*Imax + I
         A(J2,I2) = C3

         A(Imax*Kmax+1,I2) = - G(R1,Z2)*DeltaZ * Temp3

         Goto 1000
```

```
            EndIf

          If ((I.EQ.Imax).AND.((K.NE.1).AND.(K.NE.Kmax))) Then
**      I = Imax, K = 2, Kmax -1  ****
             R1 = R(I)
             Z1 = Z(K)
             DeltaZ = (Z(K+1)-Z(K-1))/2
             T2 = Tcoolant(K)
             C3 = H(T2)*R1*DeltaZ*2
             C1 = C3
             A(Imax*Kmax+1,I2) = -C3*T2

             R3 = R(I-1)
             R2 = (R3 + R1)/2
             T1 = (T(I-1,K) + T(I,K))/2
             C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
             C1 = C1 + C3
             J2 = (K-1)*Imax + I-1
             A(J2,I2) = C3

             Temp3 = (3*R1**2 - R3**2 - 2*R1*R3)/4
             Z3 = Z(K+1)
             Z2 = (Z3 + Z1)/2
             T1 = (T(I,K+1) + T(I,K))/2
             C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
             C1 = C1 + C3
             J2 = K*Imax + I
             A(J2,I2) = C3

             Z3 = Z(K-1)
             Z2 = (Z3 + Z1)/2
             T1 = (T(I,K-1) + T(I,K))/2
             C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
             C1 = C1 + C3
             J2 = (K-2)*Imax + I
             A(J2,I2) = C3

             A(Imax*Kmax+1,I2) = A(Imax*Kmax+1,I2)
     a       - G(R2,Z1)*DeltaZ * Temp3

             Goto 1000

            EndIf

          If ((I.EQ.Imax).AND.(K.EQ.1)) Then
**      I = Imax, K = 1  ****
             R1 = R(I)
             Z1 = Z(K)
             Z3 = Z(K+1)
             DeltaZ = (Z3-Z1)/2
             T1 = T(I,K)
             T2 = Tcoolant(K)
             C3 = H(T2)*R1*DeltaZ*2
             C1 = C3
             A(Imax*Kmax+1,I2) = -C3*T2

             R3 = R(I-1)
             R2 = (R3 + R1)/2
             T1 = (T(I-1,K) + T(I,K))/2
             C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
             C1 = C1 + C3
             J2 = (K-1)*Imax + I-1
             A(J2,I2) = C3

             Temp3 = (3*R1**2 - R3**2 +2*R1*R3)/4
             Z2 = (Z3 + Z1)/2
             T1 = (T(I,K+1) + T(I,K))/2
             C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
             C1 = C1 + C3
             J2 = K*Imax + I
             A(J2,I2) = C3

             A(Imax*Kmax+1,I2) = A(Imax*Kmax+1,I2)
     a       - G(R2,Z2)*DeltaZ * Temp3

             Goto 1000

            EndIf

          If ((I.EQ.Imax).AND.(K.EQ.Kmax)) Then
**      I = Imax, K = Kmax  ****
             R1 = R(I)
             Z1 = Z(K)
             Z3 = Z(K-1)
             DeltaZ = (Z1-Z3)/2
             T1 = T(I,K)
```

```
              T2 = Tcoolant(K)
              C3 = H(T2)*R1*DeltaZ*2
              C1 = C3
              A(Imax*Kmax+1,I2) = -C3*T2

              R3 = R(I-1)
              R2 = (R3 + R1)/2
              T1 = (T(I-1,K) + T(I,K))/2
              C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
              C1 = C1 + C3
              J2 = (K-1)*Imax + I-1
              A(J2,I2) = C3

              Temp3 = (3*R1**2 - R3**2 +2*R1*R3)/4
              Z2 = (Z3 + Z1)/2
              T1 = (T(I,K-1) + T(I,K))/2
              C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
              C1 = C1 + C3
              J2 = (K-2)*Imax + I
              A(J2,I2) = C3

              A(Imax*Kmax+1,I2) = A(Imax*Kmax+1,I2)
     a         - G(R2,Z2)*DeltaZ * Temp3

              Goto 1000

            EndIf

**      All other points ***
              R1 = R(I)
              R3 = R(I+1)
              R2 = (R3 + R1)/2
              Z1 = Z(K)
              DeltaZ = (Z(K+1)-Z(K-1))/2
              T1 = (T(I+1,K) + T(I,K))/2
              C3 = Kcond(I,R2,T1)*(R3+R1)/(R3-R1)*DeltaZ
              C1 = C3
              J2 = (K-1)*Imax + I+1
              A(J2,I2) = C3

              R3 = R(I-1)
              R2 = (R3 + R1)/2
              T1 = (T(I-1,K) + T(I,K))/2
              C3 = Kcond(I,R2,T1)*(R3+R1)/(R1-R3)*DeltaZ
              C1 = C1 + C3
              J2 = (K-1)*Imax + I-1
              A(J2,I2) = C3

              R2 = R(I+1)
              Temp3 = (R2**2 - R3**2 +2*R1*(R2-R3))/4
              Z3 = Z(K+1)
              Z2 = (Z3 + Z1)/2
              T1 = (T(I,K+1) + T(I,K))/2
              C3 = Kcond(I,R1,T1)/(Z3-Z1)*Temp3
              C1 = C1 + C3
              J2 = K*Imax + I
              A(J2,I2) = C3

              Z3 = Z(K-1)
              Z2 = (Z3 + Z1)/2
              T1 = (T(I,K-1) + T(I,K))/2
              C3 = Kcond(I,R1,T1)/(Z1-Z3)*Temp3
              C1 = C1 + C3
              J2 = (K-2)*Imax + I
              A(J2,I2) =  C3

              A(Imax*Kmax+1,I2) = - G(R1,Z1)*DeltaZ * Temp3

 1000 A(I2,I2) = - C1
 2000 Continue
 2005 Continue

c      Do 2015 J=1,Imax*Kmax
c      Do 2010 I=1,Imax*Kmax+1
c          Write (8,2020) I,J, A(I,J)
c 2010   Continue
c 2015   Continue
c 2020 Format (' A(',I3,',',I3,') =',F15.5)

*     Write(*,2024)
      Call TIME(Step1)
      if ( Isolver .eq. 1 ) then                                  KHEL 7/18/93
         Call Gauss                                               KHEL 7/18/93
      else                                                        KHEL 7/18/93
         Call SGauss                                              KHEL 7/18/93
      end if                                                      KHEL 7/18/93
```

```
*     Write(*,2025)
      Call TIME(Step2)
*2024 Format(' Entering GAUSS......')
*2025 Format(' .......Leaving GAUSS')
      Timer(3) = Timer(3) + Seconds(Step2) - Seconds(Step1)


      MaxError = 0.0D0
      Error = 0.0D0
      ErrSqrd = 0.0D000
      Do 2500 K=1,Kmax
      Do 2500 I=1,Imax
          I2 = (K-1)*Imax + I
          MaxError = MAX(MaxError,X(I2)-T(I,K))
          Error = Error + (X(I2) - T(I,K))
 2500     ErrSqrd = ErrSqrd + (X(I2) - T(I,K))**2.0D0

      Do 2600 I=1,Imax
      Do 2600 K=1,Kmax
          I2 = (K-1)*Imax + I
          T(I,K) = (T(I,K)+X(I2))/2.0D0
C         Write(8,2620) I,K,T(I,K)
 2600   Continue
 2620 Format (' T(',I3,',',I3,') =',G15.5)
C         DeltaT = 100.0D0
C         If (ABS(X(I2)-T(I,K)).LT.DeltaT) DeltaT =
C    a        ABS(X(I2)-T(I,K))
C         If (X(I2).LT.T(I,K)) DeltaT = -DeltaT
C 2600    T(I,K) =  T(I,K) + DeltaT

C ******* OUTPUT RESULTS TO THE FILE TFEHX.OUT *******

      Write(8,4100)
      Write(8,4110) Title
      Write(8,4120)
c     Write(*,2550) IC, SQRT(ErrSqrd/(Imax*Kmax)), Error/(Imax*Kmax),
c    a              MaxError
      Write(8,2550) IC, SQRT(ErrSqrd/(Imax*Kmax)), Error/(Imax*Kmax),
     a              MaxError
 2550 Format (' Iteration :',I3,'  RMS error = ',F13.7,'  Ave Diff. = ',
     a           F13.7/20X,' Max. Error =',F13.7)
      IC = IC + 1

      If (SQRT(ErrSqrd/(Imax*Kmax)).LT.1.0D-1) Goto 2900
      Goto 102
 2900 Write(8,4100)
      Write(8,4110) Title
      Write(8,4130)
C***** Output Temp. distribution for fuel *****
      Do 3000 I=1,Imax
      Do 3000 K=1,Kmax
          I2 = (K-1)*Imax + I
 3000     Write(8,3010) I,K, X(I2)
 3010 Format (' T(',I3,',',I3,') = ',F13.7)
      Write (8,3015) V(1),V(Kmax)
      Write(8,3020) Current, V(1)*Ibottom+V(Kmax)*Itop
 3015 Format(' Voltage across bottom of cell: ',F13.7/
     a       ' Voltage across top of cell: ',F13.7)
 3020 Format(' Output current = ',F13.7/' Output electrical power = ',
     a          F13.7)
      Write(8,3030)
 3030 Format(//8X,'Z',19X,'V',18X,'Qec',16X,'Jdens'/X,3(15('-'),5X),
     A          15('-')/)
      Do 3040 K=1,Kmax
         Write(8,3050) Z(K), V(K), Qec(K), Jdens(K)
 3040  Continue
 3050 Format(X,3(F15.8,5X),F15.8)
      Write(8,3060)
 3060 Format(//8X,'Z',17X,'EmHeat',13X,'ColHeat'/X,2(15('-'),5X),
     A         15('-')/)
      Do 3070 K=1,Kmax
         Write(8,3080) Z(K), EmHeat(K), ColHeat(K)
 3070  Continue
 3080 Format(X,2(F15.8,5X),F15.8)
      Write(8,3090)
 3090 Format(//8X,'Z',18X,'Qch',16X,'Qrad',15X,'QCsCond'/
     A         X,3(15('-'),5X),15('-')/)
      Do 3095 K=1,Kmax
         Write(8,3050) Z(K), Qch(K), Qrad(K), QCsCond(K)
 3095  Continue
      Call TIME(Stop)
      TotalTime = Seconds(Stop)-Seconds(Start)
      Write(8,4000) INT(TotalTime/60),TotalTime-INT(TotalTime)
      Write(8,4010) 'Convect/CoolantTemp', INT(Timer(1)/60),
     a        Timer(1)-INT(Timer(1)/60)*60, Timer(1)/TotalTime*100D0
      Write(8,4020) 'CYLCON6', INT(Timer(2)/60),
```

```
     a      Timer(2)-INT(Timer(2)/60)*60, Timer(2)/TotalTime*100D0
      Write(8,4030) 'Gauss', INT(Timer(3)/60),
     a      Timer(3)-INT(Timer(3)/60)*60,Timer(3)/TotalTime*100D0

 4000 Format(' Total computational time required:',I3,' min.,',F6.2,
     a       ' sec.')
 4010 Format(' Time spent in ',A19,':',I3,' min.,',F6.2,' sec. (',
     a        F4.1,'%)')
 4020 Format(' Time spent in ',A7,':',I3,' min.,',F6.2,' sec. (',
     a        F4.1,'%)')
 4030 Format(' Time spent in ',A5,':',I3,' min.,',F6.2,' sec. (',
     a        F4.1,'%)')
 4100 Format ('1',20('*'),' TFEHX ',20('*')/
     a         ' ***** RESULTS FOR THE FOLLOWING CASE:')
 4110 Format (A80)
 4120 Format (' ITERATION HISTORY -- '/20X,'Converging the RMS error to'
     a         'less than 0.1 K.')
 4130 Format (' TEMPERATURE DISTRIBUTION FOR THE FUEL REGION ---'/)
 4140 Format (3X,'\R='3X,9(F4.2,4X:))
 4142 Format (2X,'Z=\',3X,9(A6,2X:))
 4150 Format (X,F5.2,X,'|',9(X,F6.1,X:))

      Return
      End
      Real*8 Function Seconds(Time)
      Character*11 Time

      Seconds = (ICHAR(Time(1:1))-48)*36D3 + (ICHAR(Time(2:2))-48)*36D2
     a       + (ICHAR(Time(4:4))-48)*600.0 + (ICHAR(Time(5:5))-48)*60.0
     b       + (ICHAR(Time(7:7))-48)*10.0  + (ICHAR(Time(8:8))-48)
     c       + (ICHAR(Time(10:10))-48)*.1 + (ICHAR(Time(11:11))-48)*.01

      End
      Subroutine GAUSS

      Integer Imax, Kmax
      Parameter (Imax = 10,Kmax = 10)
      Real*8 A(Imax*Kmax+1,Imax*Kmax), X(Imax*Kmax)
      INTEGER PIVOT_ROW, N
      COMMON /GaussMAIN/ A,X,N

      DO 10 PIVOT_ROW=1,N
          CALL PAR_PIVOT(PIVOT_ROW)
          CALL FWD_ELIM(PIVOT_ROW)
   10   CONTINUE

      CALL BACK_SOLN

c     DO 20 INDEX=1,N
c         WRITE(UNIT=*,FMT=25) INDEX,X(INDEX)
c  25     FORMAT(' X(',I2,')= ',F16.10)
c  20   CONTINUE

      END
      SUBROUTINE PAR_PIVOT(K)

      Integer Imax, Kmax
      Parameter (Imax = 10,Kmax = 10)
      Real*8 A(Imax*Kmax+1,Imax*Kmax), X(Imax*Kmax), SCRATCH
      INTEGER L,I,K,N
      COMMON /GaussMAIN/ A,X,N

      DO 10 L=K+1,N
          IF(ABS(A(K,L)).GT.ABS(A(K,K))) THEN
              DO 20 I=K,N+1
                  SCRATCH=A(I,K)
                  A(I,K)=A(I,L)
                  A(I,L)=SCRATCH
   20           CONTINUE
            ENDIF

   10   CONTINUE

      RETURN
      END
      SUBROUTINE FWD_ELIM(K)

      Integer Imax, Kmax
      Parameter (Imax = 10,Kmax = 10)
      Real*8 A(Imax*Kmax+1,Imax*Kmax), X(Imax*Kmax)
      INTEGER I,J,K,N
      COMMON /GaussMAIN/A,X,N

      DO 10 J=K+1,N+1
          A(J,K)=A(J,K)/A(K,K)
   10   CONTINUE
```

```
      DO 20 J=K+1,N
      DO 30 I=K+1,N+1
              A(I,J)=A(I,J)-A(K,J)*A(I,K)
   30   CONTINUE
   20   CONTINUE

      RETURN
      END
      SUBROUTINE BACK_SOLN

      Integer Imax, Kmax
      Parameter (Imax = 10,Kmax = 10)
      Real*8 A(Imax*Kmax+1,Imax*Kmax), X(Imax*Kmax)
      INTEGER I,J,N
      COMMON /GaussMAIN/ A,X,N

      DO 20 J=N,1,-1
          X(J)=A(N+1,J)
          DO 10 I=J+1,N
              X(J)=X(J)-A(I,J)*X(I)
   10         CONTINUE
   20   CONTINUE

      RETURN
      END
      Real*8 Function H(T)
C*************************************************************
C                                                           *
C   Uses the convective heat transfer correlations given    *
C   for liquid metal flows through annular channels, as     *
C   presented in M.M. El-Wakil, Nuclear Heat Transport,     *
C   p. 269, equations 10-6 and 10-7.                        *
C                                                           *
C*************************************************************
      Real*8 T, K, A, B, C, D, T1, Cp
      Real*8 Tinlet, De, G, W, D2, D1
      Real*8 Nu, Pe, mdot                                              KHEL 4/26/93
      Common /CoolProp/  Tinlet, De, G, W, D2, D1, mdot                KHEL 4/26/93
      Data A/6.20D-2/, B/7.204D-4/, C/-8.343D-7/, D/3.060D-10/
      K(T1) = A + B*T1 + C*T1**2 + D*T1**3

      Pe = De*G*Cp(T,W)/K(T)

      If (D2/D1.LT.1.4) then
          Nu = 5.8D0 + 0.02D0*(Pe**0.8D0)
        Else
          Nu = 5.25D0 + 0.0188D0*(Pe**0.8D0)*(D2/D1)**0.3D0
        EndIf

      H = Nu*K(T)/De

      End
      Real*8 Function G(R,Z)

      Integer Imax
      Parameter (Imax = 10)
      Real*8 R,Z, Rbound(10), Zmin, Zmax, Pi, Temp, A, B
      Real*8 Gave, PowerTabl(2,100)
      Integer TabFlag, I
      Integer Rmesh(9),Kmax, M(5)
      Common /Rdata/ Rbound,Rmesh,M
      Common /Zdata/ Zmin,Zmax,Kmax
      Common /PowerData/ Gave, PowerTabl, TabFlag
      Data Pi/3.1414526/

      If ((R.GE.Rbound(1)).AND.(R.LE.Rbound(2))) then
           If (TabFlag.EQ.1) then
                I=0
    5         I=I+1
                If ((Z.GE.PowerTabl(1,I)).AND.
     a            (Z.LT.PowerTabl(1,I+1))) then
                    Temp=Gave*(PowerTabl(2,I+1)-PowerTabl(2,I))/
     a                      (PowerTabl(1,I+1)-PowerTabl(1,I))*
     b                      (Z - PowerTabl(1,I))
                  Else
                    Goto 5
                  EndIf
             Else
                A = PowerTabl(1,1)
                B = PowerTabl(2,1)
                Temp = Gave*(A + B*SIN((Z-Zmin)*Pi/(Zmax-Zmin)))
             EndIf
         Else
           Temp = 0.00D0
         EndIf
      G = Temp
```

```
      End
      Real*8 Function Kcond(I,R2,T)
***       Units = W/cm/K  ***

      Integer Imax
      Parameter (Imax = 10)
      Integer I,Rmesh(9),M(5), M1
      Real*8 T, R, Rbound(10), Temp, R2, T1
      Real*8 K, Kdata(4,8)
      Common /Rdata/ Rbound,Rmesh,M
*
*      Material #1 = UO2
*      Material #2 = W
*      Material #3 = Nb
*      Material #4 = Nb1Zr
*      Material #5 = Mo
*      Material #6 = Re
*      Material #7 = Cs
*      Material #8 = Al2O3
*
      Data Kdata/2.414D-2, - 2.478D-5, 1.094D-8, - 1.67D-12,
     a           0.4886D0, - 3.057D-4, 1.237D-7, - 1.72D-11,
     b          0.11104D0,   4.870D-6, 1.281D-8,      0.0D0,
     c          0.11104D0,   4.870D-6, 1.281D-8,      0.0D0,
     d           0.3602D0, - 1.141D-4, 2.050D-8,      0.0D0,
     e              0.0D0,      0.0D0,    0.0D0,      0.0D0,
     f              0.0D0,      0.0D0,    0.0D0,      0.0D0,
     g           0.1858D0, - 4.169D-4, 3.469D-7, - 9.74D-11/

      K(T1,M1) = (Kdata(1,M1) + Kdata(2,M1)*T1 + Kdata(3,M1)*T1**2
     a             + Kdata(4,M1)*T1**3)*4.184


      If (R2.LT.Rbound(1)) then
*                *** Void
          Temp = 1.0D-2
          Goto 100
        EndIf
      If (R2.EQ.Rbound(1)) then
*                *** Void/Pellet boundary
          Temp = (1.0D-2 + K(T,M(1)))/2
          Goto 100
        EndIf
      If ((R2.GT.Rbound(1)).AND.(R2.LT.Rbound(2))) then
          Temp = K(T,M(1))
*                *** Fuel pellet
          Goto 100
        EndIf
      If (R2.EQ.Rbound(2)) then
*                *** Fuel Pellet outer surface
***        Does a gap exist between the fuel and the emmiter?  ***
          If (Rbound(2).NE.Rbound(3)) then
*                *** Gap
              Temp = (1.0D-2*(R(I+1)-R(I)) +
     a                 K(T,M(1))*(R(I)-R(I-1)))
     b                        /(R(I+1)-R(I-1))
            Else
*                *** No Gap
              Temp = (K(T,M(2))*(R(I+1)-R(I)) +
     a                  K(T,M(1))*(R(I)-R(I-1)))
     b                         /(R(I+1)-R(I-1))
            EndIf
          Goto 100
        EndIf
      If ((Rbound(2).NE.Rbound(3)).AND.((R2.GT.Rbound(2))
     a            .AND.(R2.LT.Rbound(3)))) then
*                 *** Gap
          Temp = 1.0D-2
          Goto 100
        EndIf
      If ((R2.EQ.Rbound(3)).AND.(Rbound(2).NE.Rbound(3))) then
          Temp = (K(T,M(2))*(R(I+1)-R(I)) + 1.0D-2*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
*                 *** Emitter inner surface
          Goto 100
        EndIf
      If ((R2.GT.Rbound(3)).AND.(R2.LT.Rbound(4))) then
          Temp = K(T,M(2))
*                 *** Emitter
          Goto 100
        EndIf
      If (R2.EQ.Rbound(4)) then
*                 *** Emitter outer surface
          Temp = (1.0D-3*(R(I+1)-R(I)) + K(T,M(2))*(R(I)-R(I-1)))
     a                  /(R(I+1)-R(I-1))
          Goto 100
```

```
          EndIf
       If ((R2.GT.Rbound(4)).AND.(R2.LT.Rbound(5))) then
*                  *** Gap
            Temp = 1.0D-3
            Goto 100
          Endif
       If (R2.EQ.Rbound(5)) then
*                  *** Collector inner surface
            Temp = (K(T,M(3))*(R(I+1)-R(I)) + 1.0D-3*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
            Goto 100
          EndIf
       If ((R2.GT.Rbound(5)).AND.(R2.LT.Rbound(6))) then
*                  *** Collector
            Temp = K(T,M(3))
            Goto 100
          EndIf
       If (R2.EQ.Rbound(6)) then
*                  *** Collector outer surface
***         Does a gap exist between the collector and the insulator?  ***
            If (Rbound(6).NE.Rbound(7)) then
*                  *** Gap
                Temp = (1.0D-2*(R(I+1)-R(I)) + K(T,M(3))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
              Else
*                  *** No Gap
                Temp = (K(T,M(4))*(R(I+1)-R(I)) + K(T,M(3))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
              EndIf
            Goto 100
          EndIf
       If ((Rbound(6).NE.Rbound(7)).AND.((R2.GT.Rbound(6))
     a           .AND.(R2.LT.Rbound(7)))) then
*                  *** Gap
            Temp = 1.0D-2
            Goto 100
          EndIf
       If ((R2.EQ.Rbound(7)).AND.(Rbound(6).NE.Rbound(7))) then
*                  *** Insulator inner surface
            Temp = (K(T,M(4))*(R(I+1)-R(I)) + 1.0D-2*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
            Goto 100
          EndIf
       If ((R2.GT.Rbound(7)).AND.(R2.LT.Rbound(8))) then
*                  *** Insulator
            Temp = K(T,M(4))
            Goto 100
          EndIf
       If (R2.EQ.Rbound(8)) then
*                  *** Insulator outer surface
***         Does a gap exist between the insulator and the clad?  ***
            If (Rbound(8).NE.Rbound(9)) then
*                  *** Gap
                Temp = (1.0D-2*(R(I+1)-R(I)) + K(T,M(4))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
              Else
*                  *** No Gap
                Temp = (K(T,M(5))*(R(I+1)-R(I)) + K(T,M(4))*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
              EndIf
            Goto 100
          EndIf
       If ((Rbound(8).NE.Rbound(9)).AND.((R2.GT.Rbound(8))
     a           .AND.(R2.LT.Rbound(9)))) then
*                  *** Gap
            Temp = 1.0D-2
            Goto 100
          EndIf
       If ((R2.EQ.Rbound(9)).AND.(Rbound(8).NE.Rbound(9))) then
*                  *** Clad inner surface
            Temp = (K(T,M(5))*(R(I+1)-R(I)) + 1.0D-2*(R(I)-R(I-1)))
     a                   /(R(I+1)-R(I-1))
            Goto 100
          EndIf
       If ((R2.GT.Rbound(9)).AND.(R2.LT.Rbound(10))) then
*                  *** Clad
            Temp = K(T,M(5))
            Goto 100
          EndIf
       If (R2.EQ.Rbound(10)) then
*                  *** Clad outer surface
*                  *** Clad
            Temp = K(T,M(5))
            Goto 100
          EndIf
```

```
  100 Kcond = Temp

      End
      Real*8 Function Z(K)

      Integer K,Kmax
      Real*8 Zmin,Zmax
      Common /Zdata/ Zmin,Zmax,Kmax
      Kmax = 10

      Z = (Zmax-Zmin)*(K-1)/(Kmax-1) + Zmin

      End
      Real*8 Function R(I)

      Integer I,Mesh1,Mesh2,Rmesh(9), J, Imax
      Parameter (Imax = 10)
      Integer Mat(5)
      Real*8 Rbound(10),Temp

      Common /Rdata/ Rbound,Rmesh,Mat

      Mesh2=1
      J=1

   10 Temp = Rbound(J)

      Mesh1 = Mesh2
      Mesh2 = Mesh2 + Rmesh(J)

      If ((I.GE.Mesh1).AND.(I.LE.Mesh2)) then
          R = (Rbound(J+1) - Rbound(J))/(Mesh2-Mesh1)*(I-Mesh1) + Temp
          Goto 1000
        Endif

      J=J+1
      Goto 10

 1000 Return
      End
      Subroutine CoolantTemp(Tcoolant)
*************************************************************
*      Uses linear interpolation between values in a table to *
*      return the value of the NaK coolant temperature at the *
*      given axial position z.  If the axial position is out  *
*      of the range of the table, this routine returns the    *
*      coolant temperature at the bottom or the top of the    *
*      core, whichever is appropriate.                        *
*                                                               *
*      Units are degrees K.                                   *
*************************************************************
      Integer Kmax
      Parameter (Kmax = 10)
      Real*8 Z, CoolTbl(2000,2), Zh, Zl, Th, Tl, Tcoolant(Kmax)
      Integer K, K1
      Common /TTAB/ CoolTbl

      Do 500 K1=1,Kmax
      If (CoolTbl(1,1).GE.Z(K1)) then
          Tcoolant(1) = CoolTbl(1,2)
          Goto 500
        EndIf

      K = 2

   10 If (CoolTbl(K,1).GE.Z(K1)) then
          Zh = CoolTbl(K,1)
          Zl = CoolTbl(K-1,1)
          Th = CoolTbl(K,2)
          Tl = CoolTbl(K-1,2)
          Tcoolant(K1) = (Th-Tl)/(Zh-Zl)*(Z(K1)-Zl) + Tl
          Goto 500
        Else
          K = K+1
        EndIf
      Goto 10
  500 Continue
      End
      SUBROUTINE cylcon6(te,teav,tc,tcav,tr,phi0,gap,cden_av1,
     &  cden_av2,length,edout,ethick,cthick,n,vguess,v,qel,curden,
     &  emheat,colheat)
C     IMPLICIT NONE
      INTEGER n,Kmax
      PARAMETER(Kmax=10)
      REAL*8 te(Kmax),teav(Kmax),tc(Kmax),tcav(Kmax),tr,phi0,gap,
     &  cden_av1,cden_av2,length,edout,ethick,cthick,vguess(Kmax),
```

```
     &  v(Kmax),qel(Kmax),curden(Kmax),PI,emheat(Kmax),colheat(Kmax)
      PARAMETER(PI=3.141592654)
C
C  Uses relax, resis_w, resis_nb
C
C*************************************************************
C                                                            *
C  CYLCON6 cylindrical converter model                       *
C  Control #C-853-001-A-100290                               *
C  Author:  John B. McVey                                    *
C  Rasor Associates, Inc.                                    *
C  (408) 734-1622                                            *
C                                                            *
C  Model to solve for the voltage and current density        *
C  distributions along the length of a long cylindrical      *
C  thermionic converter with non-negligible resistance in the *
C  converter electrodes.  Emitter and collector temperature  *
C  distributions are specified.  Boundary conditions use     *
C  specified currents at cell ends.                          *
C                                                            *
C   INPUTS:                                                  *
C           te      vector of length n holding emitter temper- *
C                   ature (K) values at mesh points (temper-  *
C                   ature of emitter outer surface)          *
C           teav    vector holding values of radially averaged *
C                   emitter temperatures (K)                 *
C           tc      vector of length n holding collector     *
C                   inner surface temperature (K) values at  *
C                   mesh points                              *
C           tcav    vector of radially averaged collector    *
C                   temperatures (K)                         *
C           tr      cesium reservoir temperature in K        *
C           phi0    emitter bare work function in eV         *
C           gap     interelectrode gap in cm.                *
C           cden_av1  total current at z=0 end divided by the *
C                     area of a half-cell.  (A/cm2)          *
C           cden_av2  total current at z=L end divided by the *
C                     area of a half-cell.  (A/cm2)          *
C           length    total length L of a cell in cm.        *
C           edout     emitter outer diameter (cm)            *
C           ethick    emitter clad thickness (cm)            *
C           cthick    collector clad thickness (cm)          *
C           n         number of mesh points                  *
C           vguess    vector of length n holding initial     *
C                     guesses for values of v at mesh points *
C                                                            *
C   OUTPUTS:                                                 *
C           v       vector of converged values of inter-     *
C                   electrode voltage                        *
C           qel     vector of values for emitter electron    *
C                   cooling (W/cm2)                          *
C           curden    vector of values of current density at *
C                     mesh points (A/cm2)                    *
C           emheat    volumetric heat generation rate in     *
C                     emitter clad due to ohmic heating (W/cm3)*
C           colheat   volumetric heat generation rate in     *
C                     collector due to ohmic heating (W/cm3) *
C                                                            *
C*************************************************************
C
      INTEGER i
      REAL*8 c(Kmax),p(Kmax),dz,edav,cdin,cdav,resis_w,resis_nb,
     &  param1,param2,s(Kmax),rhoe(Kmax),rhoc(Kmax),emcur(Kmax)
C
      dz=length/(n-1)
      edav=edout-ethick
      cdin=edout+2.d0*gap
      cdav=cdin+cthick
C.....Compute "C" coefficients using local resistivities......
      do i=1,n
        rhoe(i)=resis_w(teav(i))
        rhoc(i)=resis_nb(tcav(i))
        c(i)=rhoe(i)*edout/(ethick*edav)+rhoc(i)*edout/(cthick*cdav)
      end do
C......Compute "P" coefficients using expressions for derivative
C      of "C".......
      p(1)=(-3.d0*c(1)+4.d0*c(2)-c(3))/(4.d0*c(1))
      do i=2,n-1
        p(i)=(c(i+1)-c(i-1))/(4.d0*c(i))
      end do
      p(n)=(3.d0*c(n)-4.d0*c(n-1)+c(n-2))/(4.d0*c(n))
C......v is initially set to vguess.....
      do i=1,n
        v(i)=vguess(i)
      end do
C.....Boundary condition parameters......
```

```
      param1=c(1)*cden_av1*length*dz
      param2=c(N)*cden_av2*length*dz
C.....Call primary routine for solution of differential equations.....
      call relax(v,c,p,dz,param1,param2,n,te,tc,tr,phi0,gap,qel,
     &  curden)
C.......Compute ohmic heat generation rates........
      s(1)=0.d0
      emcur(1)=-cden_av1*PI*edout*length/2.d0
      emcur(n)=cden_av2*PI*edout*length/2.d0
      do i=2,n-1
        s(i)=(curden(i)+curden(i-1))*dz/2.d0+s(i-1)
        emcur(i)=PI*edout*s(i)+emcur(1)
      end do
      do i=1,n
C***** Changed by Ron Pawlowski 11/14/90
C       emheat(i)=emcur(i)*emcur(i)*rhoe(i)/(PI*edav*ethick)**2
C       colheat(i)=emcur(i)*emcur(i)*rhoc(i)/(PI*cdav*cthick)**2
        emheat(i)=emcur(i)*emcur(i)*rhoe(i)/(PI*edav*ethick)*dz
        colheat(i)=emcur(i)*emcur(i)*rhoc(i)/(PI*cdav*cthick)*dz
C **** End of changes
      end do
C
      END
C
C
      SUBROUTINE relax(v,c,p,dz,param1,param2,n,te,tc,tr,phi0,gap,
     &  qel,curden)
C     IMPLICIT NONE
      INTEGER n,Kmax,ITMAX
      PARAMETER(Kmax=10)
      REAL*8 v(Kmax),c(Kmax),p(Kmax),dz,te(Kmax),tc(Kmax),tr,phi0,gap,
     &  qel(Kmax),curden(Kmax),ONE,TWO,THREE,FOUR,cden,TOL,SMALLV,
     &  param1,param2
      PARAMETER(ONE=1.d0,TWO=2.d0,THREE=3.d0,
     &  FOUR=4.d0,ITMAX=100,TOL=1.e-4,SMALLV=1.d-3)
C
C  Uses cden, tridag
C
C
C*****************************************************************
C                                                               *
C  Primary routine for solving the differential equation        *
C  and boundary conditions for the interelectrode voltage.      *
C  Newton-Raphson method is used to solve equation set          *
C  resulting from discretization.                               *
C                                                               *
C   INPUTS:                                                     *
C            v       initial guesses for v(i)'s                 *
C                    vector returns converged values on output  *
C            c       vector of "C" coefficients                 *
C            p       vector of "P" coefficients                 *
C            dz      mesh spacing (cm)                          *
C            param1   a parameter used in the boundary condition*
C                     at z=0                                    *
C            param2   as for param1, for z=L                    *
C            n       number of mesh points                      *
C            te      vector of length n holding emitter temper- *
C                    ature (K) values at mesh points            *
C            tc      vector of length n holding collector temp- *
C                    erature (K) values at mesh points          *
C            tr      cesium reservoir temperature in K          *
C            phi0    emitter bare work function in eV           *
C            gap     interelectrode gap in cm.                  *
C                                                               *
C   OUTPUTS:                                                    *
C            v       vector of converged values of inter-       *
C                    electrode voltage                          *
C            qel     vector of values for emitter electron      *
C                    cooling (W/cm2)                            *
C            curden    vector of values of current density at   *
C                      mesh points (A/cm2)                      *
C                                                               *
C*****************************************************************
C
      INTEGER i,iter
      REAL*8 dif,jguess,djdv(Kmax),curpls,v2
      REAL*8 f(Kmax),aa(Kmax),bb(Kmax),cc(Kmax),delta(Kmax)
      LOGICAL convrg
C **** Commented out by R.A. Pawlowski, 10/31/90 **
C **** Jguess will be set to zero for all calls to cden **
C     SAVE jguess
C
C.....sub-diagonal elements of Jacobian.....
      do i=2,n-1
        aa(i)=ONE+p(i)
      end do
```

```
      aa(n)=TWO
C......super-diagonal elements......
      cc(1)=TWO
      do i=2,n-1
        cc(i)=ONE-p(i)
      end do
C.....Begin iteration.......
      do iter=1,ITMAX
C.....diagonal elements, including dJ/dV calculation.......
        do i=1,n
          jguess=0.0D0
          curden(i)=cden(te(i),tc(i),tr,phi0,gap,v(i),qel(i),jguess)
C         jguess=curden(i)
          v2=v(i)+SMALLV
          curpls=cden(te(i),tc(i),tr,phi0,gap,v2,qel(i),jguess)
          djdv(i)=(curpls-curden(i))/SMALLV
          bb(i)=c(i)*dz*dz*djdv(i)-TWO
        end do
C.....compute function values..........
        do i=2,n-1
          f(i)=v(i-1)*(ONE+p(i))+v(i+1)*(ONE-p(i))+c(i)*dz*dz*
     &      curden(i)-TWO*v(i)
        end do
        f(1)=TWO*v(2)-param1*(ONE+p(1))+c(1)*curden(1)*dz*dz-
     &    TWO*v(1)
        f(n)=TWO*v(n-1)-param2*(ONE-p(N))+c(n)*dz*dz*curden(n)-
     &    TWO*v(n)
C.....solve for corrections delta using tridiagonal routine.......
        call tridag(1,n,aa,bb,cc,f,delta)
C.....update voltages.........
        do i=1,n
C *** Additions by R.A. Pawlowski 11/6/90 ****
          If (ABS(Delta(i)).le.0.1D0) then
             v(i)=v(i)-delta(i)
          else
             v(i)=v(i)-DSIGN(0.1D0,delta(i))
           endIf
c          Write(*,16) i,v(i)
   16     Format(' V(',I2,') =',F10.7)
C *** End of additions ****
        end do
C......check for convergence......
        convrg=.true.
        do i=1,n
          dif=dabs(delta(i))
          if (dif.gt.TOL) convrg=.false.
        end do
        if (convrg) goto 10
      end do
      if (iter.gt.ITMAX) pause 'No convergence'
C.......end of routine - calculate current density using converged v's
C  10 jguess=curden(1)
   10 jguess=0.0D0
      do i=1,n
        curden(i)=cden(te(i),tc(i),tr,phi0,gap,v(i),qel(i),jguess)
C       jguess=curden(i)
      end do
C.....jguess updated for any subsequent call to relax, as when
C     calculating an I-V curve........
C **** Commented out by R.A. Pawlowski on 10/31/90 **
C **** Sometimes updating jguess led to problems with calls to UNIG **
C     jguess=curden(1)
      return
      END
C
C
      REAL*8 FUNCTION resis_w(t)
C     IMPLICIT NONE
      REAL*8 t
C
C......cubic fit to resistivity of tungsten vs. temperature...
C......input is temperature in K........
C
      REAL*8 a(4),r
      DATA a /-.2285507d0,0.01808205d0,6.64431d-6,-7.479135d-10/
      r=a(1)+t*(a(2)+t*(a(3)+t*a(4)))
      resis_w=1.d-6*r
      return
      END
C
C
      REAL*8 FUNCTION resis_nb(t)
C     IMPLICIT NONE
      REAL*8 t
C
C......quadratic fit to resistivity of niobium vs. temperature...
```

```
C......input is temperature in K........
C
      REAL*8 a(3),r
      DATA a /-1.451331d0,0.04999382d0,-4.867492d-6/
      r=a(1)+t*(a(2)+t*a(3))
      resis_nb=1.d-6*r
      return
      END
C
C
      REAL*8 FUNCTION resis_mo(t)
C     IMPLICIT NONE
      REAL*8 t
C
C......linear fit to resistivity of molybdenum vs. temperature...
C......input is temperature in K........
C......This is used for cases with a molybdenum collector and is
C       thrown in free of charge.....
C
      REAL*8 a(2),r
      DATA a /-.506d0,0.022d0/
      r=a(1)+t*a(2)
      resis_mo=1.d-6*r
      return
      END
C
C
      SUBROUTINE tridag(f,l,a,b,c,d,v)
C     IMPLICIT NONE
      INTEGER f,l,Kmax
        PARAMETER (Kmax=10)
      REAL*8 a(Kmax),b(Kmax),c(Kmax),d(Kmax),v(Kmax)
C
C.......Subroutine for solving a tridiagonal linear system of
C       equations.....
C     f = index of first equation
C     l = index of last equation
C     a = sub-diagonal vector
C     b = diagonal vector
C     c = super-diagonal vector
C     d = vector of right-hand side values
C     v = solution
C
      INTEGER fp1,last,k,i
      REAL*8 beta(101),gamma(101)
      beta(f)=b(f)
      gamma(f)=d(f)/beta(f)
      fp1=f+1
      do i=fp1,l
        beta(i)=b(i)-a(i)*c(i-1)/beta(i-1)
        gamma(i)=(d(i)-a(i)*gamma(i-1))/beta(i)
      end do
      v(l)=gamma(l)
      last=l-f
      do k=1,last
        i=l-k
        v(i)=gamma(i)-c(i)*v(i+1)/beta(i)
      end do
      return
      END
      REAL*8 FUNCTION cden(te,tc,tr,phi0,d,v,qel,jguess)
C     IMPLICIT NONE
      REAL*8 te,tc,tr,phi0,d,v,qel,jguess
C
C  Uses jvbrac, jvfind, ndsphi
C
C***************************************************************
C                                                              *
C   The function cden returns current density as a function of *
C   voltage utilizing thermionic models which compute voltage  *
C   as a function of current density.  A combination of the    *
C   TECMDL and UNIG thermionic models are used, which are called*
C   by the routines jvbrac and jvfind.                         *
C                                                              *
C      Input values -                                          *
C      te          Emitter temperature (K)                     *
C      tc          Collector temperature (K)                   *
C      tr          Cesium reservoir temperature (K)            *
C      phi0        Emitter bare work function (eV)             *
C      d           Interelectrode spacing (cm)                 *
C      v           Output voltage                              *
C      jguess      Guess for current density (amps/cm2)        *
C                                                              *
C      Output values -                                         *
C      cden        Current density (amps/cm2)                  *
C      qel         Emitter electron cooling (watts/cm2)        *
```

```
C                                                                *
C*****************************************************************
C
      REAL*8 jvfind,phie,phic,j1,j2,f1,f2,ndsphi
      LOGICAL success
C
C ....Get the cesiated work functions....
      phie=ndsphi(te,tr,phi0)
      phic=1.9104+(tc/tr)*(2.2963+(tc/tr)*(-3.1364+
     &  (tc/tr)*.98039))
      if ((te.le.1300.d0).and.(v.ge.0.5d0)) then
        cden=0.d0
        qel=0.d0
        return
      endif
C ....Try to bracket the desired current density
      j1=jguess
      call jvbrac(te,tc,tr,phie,phic,d,v,j1,j2,f1,f2,success)
C.....Zero in on the correct current density value......
      if (success) then
        cden=jvfind(te,tc,tr,phie,phic,d,v,j1,j2,f1,f2,qel)
      else
        pause ' no match'
      end if
C
      return
      END
C
C
      SUBROUTINE jvbrac(te,tc,tr,phie,phic,d,v0,x1,x2,f1,f2,success)
C     IMPLICIT NONE
      REAL*8 te,tc,tr,phie,phic,d,v0,x1,x2,f1,f2,XKE,FACTOR
      LOGICAL success
      PARAMETER(XKE=8.6175d-5,FACTOR=1.6d0)
C
C Uses jvcurve
C
C*****************************************************************
C                                                                *
C   The subroutine jvbrac searches for two current density      *
C   values which will bracket the desired solution for output   *
C   voltage.                                                     *
C                                                                *
C      Input values -                                            *
C      te          Emitter temperature (K)                       *
C      tc          Collector temperature (K)                     *
C      tr          Cesium reservoir temperature (K)              *
C      phie        Cesiated emitter work function (eV)           *
C      phic        Cesiated collector work function (eV)         *
C      d           Interelectrode spacing (cm)                   *
C      v0          Desired value for output voltage              *
C      x1          Input as first guess for current density      *
C                                                                *
C      Output values -                                           *
C      x1          Output as one of the bracketing values of     *
C                  current density                               *
C      x2          The other bracketing value of current density*
C      f1          The value of v - v0 at x1                     *
C      f2          The value of v - v0 at x2                     *
C                                                                *
C*****************************************************************
C
      REAL*8 dx,x3,f3,v,qel,xjc
      INTEGER bad
C
C.....First set the search step to 1 A/cm2.......
      dx=1.
C.....Call the combined thermionic model......
      call jvcurve(te,tc,tr,phie,phic,d,x1,v,qel)
      f1=v-v0
C.....Increment current density in the correct direction...
      x2=x1+dsign(dx,f1)
      x2=dmax1(x2,0.d0)
C.....Compute voltage at new current density.......
      call jvcurve(te,tc,tr,phie,phic,d,x2,v,qel)
      f2=v-v0
      bad=0
C.....Find the back emission current density for lower search limit..
      xjc=120.d0*tc*tc*dexp(-11604.5d0*phic/tc)
C.....Continue searching until solution is bracketed.....
      do while (f1*f2.gt.0.d0)
C.......Increase size of search step.......
        dx=dx*FACTOR
        x3=x2+dsign(dx,f2)
C.......Count number of times that current density tries to go
C       below the back emission level.  If 2 or more, return
```

```
C       without a successful solution......
        if (x3.lt.-xjc) bad=bad+1
        if (bad.ge.2) then
          success=.false.
          return
        end if
        x3=dmax1(x3,-xjc)
        call jvcurve(te,tc,tr,phie,phic,d,x3,v,qel)
        f3=v-v0
        x1=x2
        f1=f2
        x2=x3
        f2=f3
      end do
      success=.true.
      return
      END
C
C
      REAL*8 FUNCTION jvfind(te,tc,tr,phie,phic,d,v0,j1,j2,f1,f2,qel)
C     IMPLICIT NONE
      REAL*8 te,tc,tr,phie,phic,d,v0,j1,j2,f1,f2,qel,TOL2
      PARAMETER(TOL2=1.d-5)
C
C  Uses jvcurve
C
C*************************************************************
C                                                            *
C   The function jvfind uses the modified regula falsi method  *
C   to find a value for current density which yields a desired *
C   voltage.  The solution must already have been bracketed.   *
C                                                            *
C      Input values -                                        *
C      te           Emitter temperature (K)                  *
C      tc           Collector temperature (K)                *
C      tr           Cesium reservoir temperature (K)         *
C      phie         Cesiated emitter work function (eV)      *
C      phic         Cesiated collector work function (eV)    *
C      d            Interelectrode spacing (cm)              *
C      v0           Desired value for output voltage         *
C      j1           One of the bracketing values of current  *
C                   density.                                 *
C      j2           The other bracketing value of current density*
C      f1           The value of v - v0 at j1                *
C      f2           The value of v - v0 at j2                *
C                                                            *
C      Output values -                                       *
C      jvfind       The solution for the current density     *
C      qel          The electron cooling at the solution point *
C                                                            *
C*************************************************************
C
      REAL*8 toll,save_f,j3,f3,v
C
      toll=1.d-5
      save_f=f1
   10 continue
        j3=j2-f2*(j2-j1)/(f2-f1)
        call jvcurve(te,tc,tr,phie,phic,d,j3,v,qel)
        f3=v-v0
C.....Re-assign whichever endpoint has the same sign of f as the most
C     recent point.  If an endpoint has been stagnant for 2 passes,
C     replace f with f/2 there.........
        if (f3*f1.lt.0.d0) then
          j2=j3
          f2=f3
          if (f3*save_f.gt.0.d0) f1=f1/2.d0
        else
          j1=j3
          f1=f3
          if (f3*save_f.gt.0.d0) f2=f2/2.d0
        end if
        save_f=f3
      if (.not.((dabs(j1-j2).le.toll).or.(dabs(f3).le.TOL2))) goto 10
      jvfind=j3
      return
      END
C
C
      SUBROUTINE jvcurve(te,tc,tr,phie,phic,dcm,j,v,qel)
C     IMPLICIT NONE
      REAL*8 te,tc,tr,phie,phic,dcm,j,v,qel,JLOW,JHIGH
      PARAMETER(JLOW=0.1d0, JHIGH=3.d0)
C  Uses tecmdl, unig
C
C*************************************************************
```

```
C                                                              *
C  The routine jvcurve combines the outputs of TECMDL and UNIG  *
C  to produce a single, physically reasonable, well-behaved     *
C  volt-ampere curve.  The limits JHIGH and JLOW are used to    *
C  save computational effort by only calling both models in the *
C  interval bounded by these limits.  Above JHIGH only TECMDL   *
C  is called, below JLOW only UNIG is called.                   *
C                                                              *
C      Input values -                                          *
C      te            Emitter temperature (K)                   *
C      tc            Collector temperature (K)                 *
C      tr            Cesium reservoir temperature (K)          *
C      phie          Cesiated emitter work function (eV)       *
C      phic          Cesiated collector work function (eV)     *
C      dcm           Interelectrode spacing (cm)               *
C      j             Current density (amps/cm2)                *
C                                                              *
C      Output values -                                         *
C      v             Output voltage                            *
C      qel           Emitter electron cooling (watts/cm2)      *
C                                                              *
C***************************************************************
      INTEGER sheaths
      REAL*8 dmm,vig,qelig,vun,qelun,ji,jel,old
C
      dmm=dcm*10.d0
      if (j.lt.JLOW) then
        jel=j
C.....A simple iteration is necessary when calling unig as it
C     accepts the electron current as an independent variable,
C     whereas the total current = electron current - ion current....
  10    continue
          old=jel
          call unig(te,tc,tr,dcm,phie,phic,jel,ji,v,qel,sheaths)
          jel=j+ji
         if (dabs((jel-old)/jel).gt.1.d-5) go to 10
C         print*,'v =',v,'unig j =',jel-ji
      else if (j.gt.JHIGH) then
         call tecmdl(te,tc,tr,phie,phic,dmm,j,v,qel)
C         print*,'v =',v,'tecmdl j =',j
      else
        jel=j
  20    continue
          old=jel
          call unig(te,tc,tr,dcm,phie,phic,jel,ji,vun,qelun,sheaths)
          jel=j+ji
        if (dabs((jel-old)/jel).gt.1.d-6) go to 20
C         print*,'v =',v,'unig j =',jel-ji
         call tecmdl(te,tc,tr,phie,phic,dmm,j,vig,qelig)
C         print*,'v =',v,'tecmdl j =',j
C.....Select whichever voltage (and corresponding electron cooling)
C     is higher........
        if (vig.ge.vun) then
          v=vig
          qel=qelig
        else
          v=vun
          qel=qelun
        end if
C         print*,'v =',v,'chosen j =',j
      end if
      return
      END
      SUBROUTINE tecmdl(te,tc,tr,phie,phic,d,j,vout,qel)
C     IMPLICIT NONE
      REAL*8 te,tc,tr,phie,phic,d,j,vout,qel,VI,B,H,BP,XK,TWO,
     &  AR,HALF
      PARAMETER (VI=3.2d0,B=30.d0,H=5.d0,BP=50.d0,XK=8.6175d-5,
     &           TWO=2.d0,AR=120.d0,HALF=0.5d0)
C
C     Uses ndsphi,obstr,obstr2,satur,satur2
C
C***********************************************************
C                                                         *
C      C-563-002-G-082988                                 *
C      Written by                                         *
C      John B. McVey                                      *
C      Rasor Associates, Inc.                             *
C      (408) 734-1622 X-315                               *
C                                                         *
C      TECMDL is an implementation of the "phenomenological *
C      model" of the ignited mode converter described in  *
C      N.S. Rasor, "Thermionic Energy Conversion", Chapter *
C      5 of Applied Atomic Collision Physics, vol. 5,     *
C      Massey, McDaniel, and Bederson eds., Academic Press, *
C      1982.  The article is available on request from    *
```

```
C      Rasor Associates.  The following physics has been     *
C      added to the model:                                   *
C             A.  Plasma energy loss by radiation.           *
C             B.  Improved description of saturation         *
C                 region.                                    *
C             C.  Provision for ion-retaining collector      *
C                 sheath.                                    *
C      The equations are documented in the Rasor document    *
C      E-563-002-A-063087, which is available on request.    *
C                                                            *
C      This routine calls four subroutines, two for the      *
C      obstructed mode (positive and negative collector      *
C      sheath) and two for the saturation region (positive   *
C      and negative collector sheath) as required.  It       *
C      calculates the output voltage and emitter electron    *
C      cooling.                                              *
C                                                            *
C      Input values -                                        *
C      TE          Emitter temperature (K)                   *
C      TC          Collector temperature (K)                 *
C      TR          Cesium reservoir temperature (K)          *
C      PHIE        Emitter work function (eV).               *
C      PHIC        Collector work function (eV).             *
C      D           Interelectrode spacing (mm)               *
C      J           Current density (amps/cm2)                *
C                                                            *
C      Output values -                                       *
C      VOUT        Output voltage (volts)                    *
C      QEL         Net emitter electron cooling (watts/cm2)  *
C                                                            *
C   Version G is special for use in CYLCON6.                 *
C   - calculation of cesiated work functions removed         *
C   - changed to double precision                            *
C   - parameter jconfdnc removed                             *
C                                                            *
C*************************************************************
C
      REAL*8 jsp,jc,jcj,pcs,ta,
     &        pd,tee,tec,ve,vc,vd,vrad,jej,je,dv,
     &        jsj,jij,js,phis
C
C......Calculate saturation current density......
      jsp=AR*te*te*dexp(-phie/(XK*te))
C
C......Calculate back emission current density and ratio......
      jc=AR*tc*tc*dexp(-phic/(XK*tc))
      jcj=jc/j
C
C......Calculate cesium pressure in torr......
      pcs=2.45d+8*dexp(-8910.d0/tr)/dsqrt(tr)
C
C......Average neutral and ion temperature......
      ta=(te+tc)/TWO
C
C......Calculate Pd......
      pd=pcs*d
C
C......Call the obstructed region calculation
C      (can't be obstructed if current density is
C      above saturation)......
      if (j.le.jsp) then
C
C......Calculation for positive collector sheath......
        call obstr(VI,B,H,j,jcj,te,tc,tr,pd,d,ta,tee,tec,ve,vc,
     &    vd,vrad,jej)
C
C......If collector sheath was negative in previous calculation,
C      use appropriate calculation......
        if (vc.lt.0.d0) call obstr2(VI,B,H,j,jcj,te,tc,tr,pd,d,ta,
     &    tee,tec,ve,vc,vd,vrad,jej)
C
C......Calculate effective emitted current density......
        je=jej*j
C
C......Calculate sheath barrier height......
        dv=XK*te*dlog(jsp/je)
C
C......Calculate output voltage......
        vout=phie-phic-vd+dv
C
C......Calculate net electron cooling from emitter
C      (includes plasma radiation)......
        qel=j*(phie+dv+TWO*XK*tee)-je*TWO*XK*(tee-te)-
     &          HALF*j*vrad
C
      endif
```

```
C
C......Call the saturation region calculation if above saturation
C       or if obstructed region calculation failed......
      if ((j.gt.jsp).or.(dv.lt.0.d0)) then
C
C......Calculation for positive collector sheath......
        call satur(VI,B,BP,H,j,jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,
     &      ve,vc,vd,vrad,jsj,jij)
C
C......Calculation for negative collector sheath if previous
C       calculation gave negative value......
        if (vc.lt.0.d0) call satur2(VI,B,BP,H,j,
     &     jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,ve,vc,vd,vrad,jsj,
     &     jij)
C
C......Calculate effective Schottky saturation current density......
        js=jsj*j
C
C......Calculate Schottky reduced emitter work function......
        phis=XK*te*dlog(AR*te*te/js)
C
C......Calculate output voltage......
        vout=phis-phic-vd
C
C......Calculate net emitter electron cooling (includes
C       ion heating and plasma radiation)......
        qel=j*(phis+TWO*XK*tee)-js*TWO*XK*(tee-te)+j*jij*(ve+3.89d0+
     &      TWO*XK*tee)-HALF*j*vrad
C
      endif
      return
      END
C
C
      SUBROUTINE obstr(vi,b,h,j,jcj,te,tc,tr,pd,d,ta,tee,tec,ve,vc,
     &    vd,vrad,jej)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 vi,b,h,j,jcj,te,tc,tr,pd,d,ta,tee,tec,ve,vc,vd,vrad,
     &    jej,XK,HALF,TOL,AR,EMIS,ONE,TWO,THREE,TINY
      PARAMETER (XK=8.6175d-5,TWO=2.d0,HALF=.5d0,TOL=1.d-5,AR=120.d0,
     &           EMIS=.4d0,MAXITR=50,ONE=1.d0,THREE=3.d0,TINY=1.d-32)
C
C     Uses tsc, ltec
C
C************************************************************
C                                                          *
C      OBSTR is called by TECMDL to implement the          *
C      phenomenological equations for                      *
C      the obstructed region of the ignited mode volt-     *
C      ampere curve with a positive (electron retaining)   *
C      sheath at the collector.  This is the formulation   *
C      described in Massey, McDaniel, and Bederson.        *
C      Equations (24)-(29) and (31) are used.  The emitter *
C      side and collector side electron temperatures are   *
C      subject to LTE (Local Thermodynamic Equilibrium)    *
C      constraints which are implemented by the functon    *
C      TSC and the subroutine LTE.  Equations (24) for     *
C      JE/J and (25) for Vd are coupled, and are solved    *
C      iteratively using a secant method search.           *
C                                                          *
C      Input values -                                      *
C      VI         Effective ionization energy (eV)         *
C      B          Ionizability factor                      *
C      H          Collector current factor                 *
C      J          Current density (amps/cm2)               *
C      JCJ        Ratio of back emission to current density *
C      TE         Emitter temperature (K)                  *
C      TC         Collector temperature (K)                *
C      TR         Cesium reservoir temperature (K)         *
C      PD         Pressure-spacing product (torr-mm)       *
C      D          Interelectrode spacing (mm)              *
C      TA         Average neutral and ion temperature (K)  *
C                                                          *
C      Output values -                                     *
C      TEE        Emitter side electron temperature (K)    *
C      TEC        Collector side electron temperature (K)  *
C      VE         Emitter sheath height (eV)               *
C      VC         Collector sheath height (eV)             *
C      VD         Arc drop (eV)                            *
C      VRAD       Plasma radiation component of arc drop (eV)*
C      JEJ        Ratio JE/J of effective emitted current  *
C                 density to working current density       *
C                                                          *
C************************************************************
C
```

```
      INTEGER iter
      REAL*8 dlea,phinc,jnc,hs,tsc,telect,dl,r,ans,dif,
     &    oldj,olddif,newj,param1,param2,param3,param4
      LOGICAL first,ltec
C
C......Calculate ratio of spacing to electron-neutral
C       mean free path.......
      dlea=35.d0/((te+tc)/2000.d0)*pd
C
C......Calculate emitter side electron temperature......
      tee=VI/(TWO*XK*dlog(B*dlea))
C
C......Calculate collector side electron temperature......
      tec=(THREE*tee+TWO*tc*jcj)/(dlog((H+HALF)/
     &   (ONE+jcj))+TWO*jcj+THREE)
C
C......Calculate neutralization potential and current density......
      phinc=1.7d0+.383d0*tec/tr
      jnc=AR*tec*tec*dexp(-phinc/(XK*tec))
C
C......Assign value to LTE limit for H and check
C       to see if LTE limits the electron temperature
C       at the collector edge - if so, calculate a new
C       TEC value......
      hs=jnc/j
      if (hs.lt.h) then
        tec=tsc(tee,tc,tr,hs,j,jcj)
        ltec=.true.
      else
        ltec=.false.
      endif
C
C......Calculate average electron temperature......
      telect=(tee+tec)/TWO
C
C......Calculate ratio of spacing to total
C       electron mean free path, including
C       ion scattering......
      dl=dlea+3.4d+7*j*d/(telect**2.5d0)
C
C......If LTE has occured at the collector edge, the routine
C       LTE is called to check that the average electron
C       temperature is above the bulk LTE limit.  If not,
C       the LTE routine will calculate new emitter side,
C       collector side,  and average electron temperatures
C       The ratio of spacing to total mean free path is also
C       recalculated......
      if (ltec) call lte(tee,tec,telect,tc,tr,hs,j,jcj,dl,dlea,d)
C
C......Calculate collector sheath height......
      vc=THREE*XK*(tee-tec)-TWO*xk*(tec-tc)*jcj
C
C......Calculate collector reflection factor......
      r=(ONE+jcj)*dexp(vc/(XK*tec))-ONE
C
C......Calculate radiation component of arc drop......
      vrad=9.65d+5*pd/(j*ta)*dexp(-2.d0/(XK*telect))*(ONE+.069d0
     &  *dexp(.58d0/(XK*telect))*(EMIS/dsqrt(d/10.d0)+HALF))
C
C......Guess JEJ and enter secant method iteration......
      jej=TWO
      first=.true.
C
C.......First compute some parameters in order to save time in the
C          iteration loop..........
      param1=TWO*XK*(tec-te+(tec-tc)*jcj)+vrad
      param2=TWO*XK*(tee-te)
      param3=.75d0*dl+r
      param4=-ONE/(XK*tee)
C......Start iteration........
      do iter=1,MAXITR
C......Calculate arc drop......
        vd=param2*(jej-ONE)+param1
C
C......Calculate emitter sheath height......
        ve=vd+vc
C
C......Calculate answer for JEJ and compute difference from
C       guess for JEJ......
        if (ve*param4.le.dlog(TINY)) then
C.........Case for ve so large that the exp function would underflow..
          ans=ONE
        else
C.........Normal case........
          ans=ONE+param3*dexp(ve*param4)
        endif
```

```
        dif=jej-ans
      if (dabs(dif).lt.TOL) go to 10
C
C......Update value of JEJ until convergence......
        if (first) then
          oldj=jej
          olddif=dif
          jej=jej-dsign(.2d0,dif)
          first=.false.
        else
          newj=(oldj*dif-jej*olddif)/(dif-olddif)
          oldj=jej
          olddif=dif
          jej=newj
         endif
      if (dabs(jej-oldj).lt.1.d-5*jej) go to 10
      enddo
 10   if (iter.gt.MAXITR) pause 'Exceeded maximum iterations in
     &obstr'
      return
      END
C
C
      SUBROUTINE satur(vi,b,bp,h,j,jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,
     &     ve,vc,vd,vrad,jsj,jij)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 vi,b,bp,h,j,jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,ve,vc,vd,
     &     vrad,jsj,jij,XK,TWO,HALF,TOL1,TOL2,AR,EMIS,ONE,THREE,TINY
      PARAMETER (XK=8.6175d-5,TWO=2.d0,HALF=.5d0,TOL1=1.d-6,TOL2=1.d-5,
     &           MAXITR=100,AR=120.d0,EMIS=.4d0,ONE=1.d0,THREE=3.d0,
     &          TINY=1.d-32)
C
C     Uses tsc, ltec
C
C**********************************************************************
C                                                                    *
C      SATUR is called by TECMDL to implement the phenomen-          *
C      ological model equations in the saturation region,           *
C      with a positive collector sheath.  The formulation           *
C      given by eqs. (33) to (35) in Massey, McDaniel, and          *
C      Bederson has been improved so that is consistent             *
C      with the level of complexity used in the obstructed          *
C      region calculation.  The ion current into the                *
C      emitter is now included in all equations in the form         *
C      of the parameter JIJ (Ji/J).   An additional                 *
C      iteration over what was needed in the obstructed             *
C      mode calculation is required for finding the value           *
C      of JIJ.  A modified linear interpolation method is           *
C      used.  The iteration for finding VD and JEJ is               *
C      nested within this new iteration.                            *
C                                                                    *
C      Input values -                                                *
C      VI        Effective ionization energy (eV)                    *
C      B         Ionizability factor                                 *
C      BP        Temperature increase parameter                      *
C      H         Collector current factor                            *
C      J         Current density (amps/cm2)                          *
C      JCJ       Ratio of back emission to current density           *
C      TE        Emitter temperature (K)                             *
C      TC        Collector temperature (K)                           *
C      TR        Cesium reservoir temperature (K)                    *
C      PD        Pressure-spacing product (torr-mm)                  *
C      D         Interelectrode spacing (mm)                         *
C      TA        Average neutral and ion temperature (K)             *
C                                                                    *
C      Output values -                                               *
C      TEE       Emitter side electron temperature (K)               *
C      TEC       Collector side electron temperature (K)             *
C      VE        Emitter sheath height (eV)                          *
C      VC        Collector sheath height (eV)                        *
C      VD        Arc drop (eV)                                       *
C      VRAD      Plasma radiation component of arc drop (eV)         *
C      JSJ       Ratio JS/J of effective emitted current             *
C                density to working current density                  *
C      JIJ       Ratio Ji/J of additional ion current to             *
C                the emitter to working current density              *
C                                                                    *
C**********************************************************************
C
      INTEGER iwhch,iter1,iter2
      REAL*8 dlea,phinc,jnc,hs,tsc,telect,dl,r,ans,f,oldj,
     &       oldf,newj,param1,param2,param3,param4,js,g,
     &       x1,x2,x3,y1,y2,y3,ys
      LOGICAL first,ltec
C
```

```
C......Guess ion current ratio......
      jij=0.d0
C
C......Set iteration counter......
      iwhch=1
C
C......Calculate ratio of gap to electron-neutral mean free path......
      dlea=35.d0/((te+tc)/2000.d0)*pd
C
C......Begin modified linear interpolation search for JIJ......
C
      do iter1=1,MAXITR
C......Calculate emitter side electron temperature......
        tee=vi/(TWO*XK*dlog(b*dlea)-XK*dlog(ONE-bp*jij))
C
C......Calculate collector side electron temperature......
        tec=(THREE*tee+TWO*tc*jcj)/(dlog((h+HALF)/
     &    (ONE+jcj))+TWO*jcj+THREE)
C
C......Calculate neutralization potential and current density......
        phinc=1.7d0+.383d0*tec/tr
        jnc=AR*tec*tec*dexp(-phinc/(XK*tec))
C
C......Assign value to LTE limit for H and check
C      to see if LTE limits the electron temperature
C      at the collector edge - if so, calculate a new
C      TEC value......
        hs=jnc/j
        if (hs.lt.h) then
          tec=tsc(tee,tc,tr,hs,j,jcj)
          ltec=.true.
        else
          ltec=.false.
        endif
C
C......Calculate average electron temperature......
        telect=(tee+tec)/TWO
C
C......Calculate ratio of spacing to total
C      electron mean free path, including
C      ion scattering......
        dl=dlea+3.4d+7*j*d/(telect**2.5d0)
C
C......If LTE has occured at the collector edge, the routine
C      LTE is called to check that the average electron
C      temperature is above the bulk LTE limit.  If not,
C      the LTE routine will calculate new emitter side,
C      collector side,  and average electron temperatures
C      The ratio of spacing to total mean free path is also
C      recalculated......
        if (ltec) call lte(tee,tec,telect,tc,tr,hs,j,jcj,dl,dlea,d)
C
C......Calculate collector sheath height......
        vc=THREE*XK*(tee-tec)-TWO*XK*(tec-tc)*jcj
C
C......Calculate collector reflection factor......
        r=(ONE+jcj)*dexp(vc/(XK*tec))-ONE
C
C......Calculate radiation component of arc drop......
        vrad=9.65d+5*pd/(j*ta)*dexp(-2.d0/(XK*telect))*
     &    (ONE+.069d0*dexp(.58d0/(XK*telect))*
     &    (EMIS/dsqrt(d/10.d0)+HALF))
C
C......Guess JSJ and enter secant method iteration......
        jsj=2.d0
        first=.true.
C
C........First compute some parameters in order to save time
C          in the iteration loop
        param1=TWO*XK*(tec-te+(tec-tc)*jcj)+vrad-jij*(vc+
     &         3.89d0+TWO*XK*tee)
        param2=TWO*XK*(tee-te)
        param3=.75d0*dl+r
        param4=-ONE/(XK*tee)
C.......Start iteration.........
        do iter2=1,MAXITR
C......Calculate arc drop......
          vd=(param2*(jsj-ONE)+param1)/(ONE+jij)
C
C......Calculate emitter sheath height......
          ve=vd+vc
C
C......Calculate answer for JSJ and compute difference from
C      guess for JSJ......
          if (ve*param4.le.dlog(TINY)) then
C..........Case for ve so large that exp function would underflow......
```

```
            ans=ONE+jij
          else
C...........Normal case.........
            ans=ONE+(param3-HALF*jij)*dexp(ve*param4)+jij
          endif
          f=jsj-ans
        if (dabs(f).lt.TOL1) go to 20
C
C......Update value of JSJ until convergence......
          if (first) then
            oldj=jsj
            oldf=f
            jsj=jsj-dsign(.2d0,f)
            first=.false.
          else
            newj=(oldj*f-jsj*oldf)/(f-oldf)
            oldj=jsj
            oldf=f
            jsj=newj
          endif
        if (dabs(jsj-oldj).lt.1.d-5*jsj) go to 20
        enddo
 20     if (iter2.gt.MAXITR) pause 'Exceeded maximum iterations in
     &SATUR for finding current ratio'
C
C......Calculate value of JS from eqn. (35) of Massey,
C      McDaniel, and Bederson......
        js=jsp*dexp(612.d0*dsqrt(dsqrt(-j*jij*dsqrt(ve)))/te)
C
C......Compute error term......
        g=js/j-jsj
      if (dabs(g).lt.TOL2) go to 30
C
C......Update JIJ to make error small......
        if (iwhch.eq.1) then
          if ((g.gt.0.d0).and.(vc.le.0.d0)) return
          x1=jij
          y1=g
          jij=-.1d0
          x2=jij
          iwhch=2
        else if (iwhch.eq.2) then
          x2=jij
          y2=g
          if (y1*y2.gt.0.d0) then
            x1=x2
            y1=y2
            jij=jij+dsign(.1d0,g)
C.........Prevent jij from becoming equal to -1.  Make it the
C         nearest larger number.....
            if (jij.le.-ONE) jij=-.999999d0
          else
            iwhch=3
            ys=y2
            jij=(x1*y2-x2*y1)/(y2-y1)
          endif
        else if (iwhch.eq.3) then
          x3=jij
          y3=g
          if (y3*y1.lt.0.d0) then
            x2=x3
            y2=y3
            if (y3*ys.gt.0.d0) y1=y1/TWO
          else
            x1=x3
            y1=y3
            if (y3*ys.gt.0.d0) y2=y2/TWO
          endif
          ys=y3
          jij=(x1*y2-x2*y1)/(y2-y1)
        endif
      enddo
 30   if (iter1.gt.MAXITR) then
c       write(*,'(a)') ' Maximum iterations exceeded in
c    &SATUR2 for finding ion current'
        write(8,'(a)') ' Maximum iterations exceeded in
     &SATUR2 for finding ion current'
c       write(*,*) j,tr
        write(8,*) j,tr
        stop
      endif
      return
      END
C
C
      REAL*8 FUNCTION tsc(tee,tc,tr,hs,j,jcj)
```

```
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 tee,tc,tr,hs,j,jcj,XK,ONE,TWO,THREE,TOL,AR,HALF
      PARAMETER (XK=8.6175d-5,ONE=1.d0,TWO=2.d0,THREE=3.d0,TOL=
     &            1.d-5,AR=120.d0,MAXITR=50,HALF=.5d0)
C
C*********************************************************
C                                                         *
C      The function TSC is called by the subroutines OBSTR  *
C      and SATUR in order to compute the collector side     *
C      electron temperature when LTE conditions exist at the*
C      collector.  A secant method iteration is used.       *
C                                                         *
C      Input values -                                      *
C      TEE           Emitter side electron temperature (K)  *
C      TC            Collector temperature (K)              *
C      TR            Cesium reservoir temperature (K)       *
C      HS            Ratio of neutralization current Jn to  *
C                    Current density J                      *
C      J             Current density (amps/cm2)             *
C      JCJ           Ratio of back emission to current den- *
C                    sity                                   *
C                                                         *
C      Output values -                                     *
C      TSC           LTE value for electron temperature at  *
C                    collector edge of plasma               *
C                                                         *
C*********************************************************
C
      INTEGER iter
      REAL*8 r1,dh,phinc,jnc,param1,param2,hss,dif,oldh,
     &        olddif,newh
      LOGICAL first,goon
C
C......Calculate numerator......
      r1=THREE*tee+TWO*tc*jcj
C
C......Enter iteration
C
      first=.true.
      goon=.false.
      dh=ONE
      param1=TWO*jcj+THREE
      param2=ONE+jcj
      do iter=1,MAXITR
C......Calculate collector edge electron temperature......
        tsc=r1/(dlog((hs+HALF)/param2)+param1)
C
C......Calculate neutralization work function and current density......
        phinc=1.7d0+.383d0*tsc/tr
        jnc=AR*tsc**2*dexp(-phinc/(XK*tsc))
C
C......Find answer for HS, difference between guess and answer......
        hss=jnc/j
        dif=hs-hss
      if (dabs(dif).lt.TOL) go to 40
C
C......Update HS to make difference small......
        if (first) then
          oldh=hs
          olddif=dif
          hs=hs-dsign(dh,dif)
          hs=dmax1(hs,1.0d-12)
          first=.false.
          dh=dh*1.6d0
        else
          if (.not.goon) then
            if (dif*olddif.gt.0.d0) then
              newh=hs-dsign(dh,dif)
              newh=dmax1(newh,1.d-12)
              dh=1.6d0*dh
            else
              goon=.true.
              newh=(oldh*dif-hs*olddif)/(dif-olddif)
            endif
          else
            newh=(oldh*dif-hs*olddif)/(dif-olddif)
          endif
          oldh=hs
          olddif=dif
          hs=newh
        endif
      if (dabs(hs-oldh).lt.1.d-5*hs) go to 40
      enddo
 40   if (iter.gt.MAXITR) pause 'Exceeded maximum iterations in TSC'
      return
```

```
      END
C
C
      SUBROUTINE lte(tee,tec,tav,tc,tr,hs,j,jcj,dl,dlea,d)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 tee,tec,tav,tc,tr,hs,j,jcj,dl,dlea,d,XK,
     &       ONE,TWO,THREE,TOL,AR,HALF
      PARAMETER (XK=8.6175d-5,ONE=1.d0,TWO=2.d0,THREE=3.d0,TOL=
     &           1.d-5,AR=120.d0,MAXITR=50,HALF=.5d0)
C
C***********************************************************
C                                                          *
C     The routine LTE is called by OBSTR and SATUR in order *
C     to check, and possibly recalculate, the electron     *
C     temperatures in order to keep the average electron   *
C     temperature above the LTE limit for the bulk plasma. *
C     This is briefly discussed in Appendix B of Chapter 5 *
C     in Massey, McDaniel, and Bederson.                   *
C                                                          *
C     Input values -                                       *
C     TEE         Electron temperature at emitter edge (K) *
C     TEC         Electron temperature at collectore edge (K)*
C     TAV         Average electron temperature (K)         *
C     TC          Collector temperature (K)                *
C     TR          Cesium reservoir temperature (K)         *
C     HS          Ratio of neutalization current to current *
C                 density                                  *
C     J           Current density (amps/cm2)               *
C     JCJ         Ratio of back emission to current density *
C     DL          Ratio of gap to total electron mean free *
C                 path                                     *
C     DLEA        Ratio of gap to electron-neutral mean    *
C                 free path                                *
C     D           Interelectrode gap (mm)                  *
C                                                          *
C     Output values (recalculated) -                       *
C     TEE                                                  *
C     TEC                                                  *
C     TAV                                                  *
C     DL                                                   *
C                                                          *
C***********************************************************
C
      INTEGER iter1,iter2
      REAL*8 ts,dl1,tss,dif,oldt,olddif,newt,r1,dh,
     &       phinc,jnc,hss,oldh,newh,tsc,param1,param2
      LOGICAL first,goon
C
C......First guess for TS......
      ts=tav
      first=.true.
      dh=TWO
C
C......Enter secant method search for TS......
C
C......Calculate new value for ratio of gap to mean free path......
      do iter1=1,MAXITR
        dl1=dlea+3.4d+7*j*d/ts**2.5d0
C
C......Calculate an answer for TS......
        tss=1.7d0/(XK*dlog((AR*ts*ts)/(j*dl1))-.383d0/tr)
C
C......Find difference between guess and answer.....
        dif=ts-tss
      if (dabs(dif).lt.TOL) go to 50
C
C......Update TS to make difference small......
        if (first) then
          oldt=ts
          olddif=dif
          ts=ts-dsign(50.d0,dif)
          first=.false.
        else
          newt=(oldt*dif-ts*olddif)/(dif-olddif)
          oldt=ts
          olddif=dif
          ts=newt
        endif
      if (dabs(ts-oldt).lt.1.d-5*ts) go to 50
      enddo
 50   if (iter1.gt.MAXITR) pause 'Max. iterations exceeded in LTE'
C
C......Check to see if average electron temperature is above
C     the limit.  If so, return without altering any
C     values......
```

```
      if (tav.ge.ts) return
C
C......If bulk LTE is in effect, replace TAV and DL with their
C       proper LTE values......
      tav=ts
      dl=dll
C
C......TEC must now be recalculated, since it depends on TEE,
C       which will change to keep the average temperature
C       above its limit.  An iteration like that in the function
C       TSC is used......
      first=.true.
      goon=.false.
      r1=TWO*THREE*ts+TWO*tc*jcj
      dh=TWO
C
C......Begin iteration......
      param1=TWO*(jcj+THREE)
      param2=ONE+jcj
      do iter2=1,MAXITR
C......Calculate collector edge electron temperature......
        tsc=r1/(dlog((hs+HALF)/param2)+param1)
        phinc=1.7d0+.383d0*tsc/tr
        jnc=AR*tsc**2*dexp(-phinc/(XK*tsc))
        hss=jnc/j
        dif=hs-hss
      if (dabs(dif).lt.TOL) go to 60
        if (first) then
          oldh=hs
          olddif=dif
          hs=hs-dsign(dh,dif)
          hs=dmax1(hs,1.d-12)
          first=.false.
          dh=1.6d0*dh
        else
          if (.not.goon) then
            if (dif*olddif.gt.0.d0) then
              newh=hs-dsign(dh,dif)
              newh=dmax1(newh,1.d-12)
              dh=1.6d0*dh
            else
              goon=.true.
              newh=(oldh*dif-hs*olddif)/(dif-olddif)
            endif
          else
            newh=(oldh*dif-hs*olddif)/(dif-olddif)
          endif
          oldh=hs
          olddif=dif
          hs=newh
        endif
      if (dabs(hs-oldh).lt.1.d-5*hs) go to 60
      enddo
 60   if (iter2.gt.MAXITR) pause 'Max. iterations exceeded in LTE'
C......Recompute TEC......
      tec=tsc
C
C......Recompute TEE......
      tee=TWO*ts-tsc
      return
      END
C
C
      SUBROUTINE obstr2(vi,b,h,j,jcj,te,tc,tr,pd,d,ta,tee,tec,ve,vc,
     &    vd,vrad,jej)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 vi,b,h,j,jcj,te,tc,tr,pd,d,ta,tee,tec,ve,vc,vd,vrad,
     &    jej,XK,HALF,TOL,AR,EMIS,ONE,TWO,THREE,TINY
      PARAMETER (XK=8.6175d-5,TWO=2.d0,HALF=.5d0,TOL=1.d-5,AR=120.d0,
     &           EMIS=.4d0,MAXITR=50,ONE=1.d0,THREE=3.d0,TINY=1.d-32)
C
C     Uses tsc2, ltec2
C
C***********************************************************
C                                                          *
C      OBSTR2 is called by TECMDL to  implement the         *
C      phenomenological equations for the obstructed        *
C      region of the ignited mode volt-ampere curve with    *
C      a negative (ion retaining) sheath at the collector.  *
C      The emitter side and collector side electron         *
C      temperatures are subject to LTE (Local Thermodynamic *
C      Equilibrium) constraints which are implemented by    *
C      the functon TSC2 and the subroutine LTE2.  The sub-  *
C      routine is very similar to OBSTR except that the     *
C      equations for TEC and VC and the LTE routines are    *
```

```
C      different.                                                 *
C                                                                 *
C      Input values -                                             *
C      VI         Effective ionization energy (eV)                *
C      B          Ionizability factor                             *
C      H          Collector current factor                        *
C      J          Current density (amps/cm2)                      *
C      JCJ        Ratio of back emission to current density       *
C      TE         Emitter temperature (K)                         *
C      TC         Collector temperature (K)                       *
C      TR         Cesium reservoir temperature (K)                *
C      PD         Pressure-spacing product (torr-mm)              *
C      D          Interelectrode spacing (mm)                     *
C      TA         Average neutral and ion temperature (K)         *
C                                                                 *
C      Output values -                                            *
C      TEE        Emitter side electron temperature (K)           *
C      TEC        Collector side electron temperature (K)         *
C      VE         Emitter sheath height (eV)                      *
C      VC         Collector sheath height (eV)                    *
C      VD         Arc drop (eV)                                   *
C      VRAD       Plasma radiation component of arc drop (eV)*
C      JEJ        Ratio JE/J of effective emitted current         *
C                 density to working current density              *
C                                                                 *
C*****************************************************************
C
      INTEGER iter
      REAL*8 dlea,zetac,phinc,jnc,hs,tsc2,telect,dl,r,ans,dif,
     &    oldj,olddif,newj,param1,param2,param3,param4
      LOGICAL first,ltec
C
C......Calculate ratio of spacing to electron-neutral
C       mean free path.......
      dlea=35.d0/((te+tc)/2000.d0)*pd
C
C......Calculate emitter side electron temperature......
      tee=vi/(TWO*XK*dlog(b*dlea))
C
C......Calculate collector sheath attenuation factor......
      zetac=(-(h+HALF)+dsqrt((h+HALF)**2+8.d0*jcj*h))/
     &   (TWO*jcj)
C
C......Calculate collector sheath height......
      vc=XK*tc*dlog(zetac)
C
C......Calculate collector emission factor......
      r=jcj*dexp(vc/(XK*tc))
C
C......Calculate collector side electron temperature......
      tec=(THREE*tee+TWO*tc*r)/(TWO*r+THREE)
C
C......Calculate neutralization potential and current density......
      phinc=1.7d0+.383d0*tec/tr
      jnc=AR*tec*tec*dexp(-phinc/(XK*tec))
C
C......Assign value to LTE limit for H and check
C       to see if LTE limits the electron temperature
C       at the collector edge - if so, calculate new
C       values for TEC, VC, and R......
      hs=jnc/j
      if ((r+HALF).gt.hs) then
        tec=tsc2(tee,tc,tr,hs,j,jcj,vc)
        r=hs-HALF
        ltec=.true.
      else
        ltec=.false.
      endif
C
C......Calculate average electron temperature......
      telect=(tee+tec)/TWO
C
C......Calculate ratio of spacing to total
C       electron mean free path, including
C       ion scattering......
      dl=dlea+3.4d+7*j*d/(telect**2.5d0)
C
C......If LTE has occured at the collector edge, the routine
C       LTE is called to check that the average electron
C       temperature is above the bulk LTE limit.  If not,
C       the LTE routine will calculate new emitter side,
C       collector side,  and average electron temperatures
C       The ratio of spacing to total mean free path is also
C       recalculated......
      if (ltec) call lte2(tee,tec,telect,tc,tr,hs,j,jcj,vc,dl,dlea,
     &  d,r)
```

```
C
C......Calculate radiation component of arc drop......
      vrad=9.65d+5*pd/(j*ta)*dexp(-2.d0/(XK*telect))*
     &   (ONE+.069d0*dexp(.58d0/(XK*telect))
     &   *(EMIS/dsqrt(d/10.d0)+HALF))
C
C......Guess JEJ and enter secant method iteration......
      jej=TWO
      first=.true.
C
C.......First compute some parameters in order to save time in the
C          iteration........
      param1=TWO*XK*(tec-te+(tec-tc)*r)+vrad
      param2=TWO*XK*(tee-te)
      param3=.75d0*dl+r
      param4=-ONE/(XK*tee)
C.......Start iteration.......
      do iter=1,MAXITR
C......Calculate emitter sheath height......
        ve=param2*(jej-ONE)+param1
C
C......Calculate answer for JEJ and compute difference from
C       guess for JEJ......
        if (ve*param4.le.dlog(TINY)) then
C.........Case where ve is so large that it would cause exp function
C            to underflow..........
          ans=ONE
        else
C.........Normal case..........
          ans=ONE+param3*dexp(ve*param4)
        endif
        dif=jej-ans
      if (dabs(dif).lt.TOL) go to 70
C
C......Update value of JEJ until convergence......
        if (first) then
          oldj=jej
          olddif=dif
          jej=jej-dsign(.2d0,dif)
          first=.false.
        else
          newj=(oldj*dif-jej*olddif)/(dif-olddif)
          oldj=jej
          olddif=dif
          jej=newj
        endif
      if (dabs(jej-oldj).lt.1.d-5*jej) go to 70
      enddo
 70   if (iter.gt.MAXITR) pause 'Exceeded maximum iterations in
     &OBSTR2'
C
C......Calculate arc drop......
      vd=ve-vc
      return
      END
C
C
      SUBROUTINE satur2(vi,b,bp,h,j,jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,
     &     ve,vc,vd,vrad,jsj,jij)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 vi,b,bp,h,j,jcj,jsp,te,tc,tr,pd,d,ta,tee,tec,ve,vc,vd,
     &     vrad,jsj,jij,XK,TWO,HALF,TOL1,TOL2,AR,EMIS,ONE,THREE,TINY
      PARAMETER (XK=8.6175d-5,TWO=2.d0,HALF=.5d0,TOL1=1.d-6,TOL2=1.d-5,
     &          MAXITR=100,AR=120.d0,EMIS=.4d0,ONE=1.d0,THREE=3.d0,
     &          TINY=1.d-32)
C
C     Uses tsc2, ltec2
C
C*********************************************************
C                                                        *
C      SATUR2 is called by TECMDL to implement the phenomen-*
C      ological model equations in the saturation region,   *
C      with a negative collector sheath.  The formulation   *
C      is very similar to SATUR except that the equations   *
C      for TEC and VC and the LTE routines are different.   *
C                                                        *
C      Input values -                                    *
C      VI        Effective ionization energy (eV)        *
C      B         Ionizability factor                     *
C      BP        Temperature increase parameter          *
C      H         Collector current factor                *
C      J         Current density (amps/cm2)              *
C      JCJ       Ratio of back emission to current density  *
C      TE        Emitter temperature (K)                 *
C      TC        Collector temperature (K)               *
```

```
C       TR          Cesium reservoir temperature (K)          *
C       PD          Pressure-spacing product (torr-mm)        *
C       D           Interelectrode spacing (mm)               *
C       TA          Average neutral and ion temperature (K)   *
C                                                             *
C       Output values -                                       *
C       TEE         Emitter side electron temperature (K)     *
C       TEC         Collector side electron temperature (K)   *
C       VE          Emitter sheath height (eV)                *
C       VC          Collector sheath height (eV)              *
C       VD          Arc drop (eV)                             *
C       VRAD        Plasma radiation component of arc drop (eV)*
C       JSJ         Ratio JS/J of effective emitted current   *
C                   density to working current density        *
C       JIJ         Ratio Ji/J of additional ion current to   *
C                   the emitter to working current density    *
C                                                             *
C*************************************************************
C
      INTEGER iwhch,iter1,iter2
      REAL*8 dlea,zetac,phinc,jnc,hs,tsc2,telect,dl,r,ans,f,oldj,
     &       oldf,newj,param1,param2,param3,param4,js,g,
     &       x1,x2,x3,y1,y2,y3,ys
      LOGICAL first,ltec
C
C......Guess ion current ratio......
      jij=0.d0
C
C......Set iteration counter......
      iwhch=1
C
C......Calculate ratio of gap to electron-neutral mean free path......
      dlea=35.d0/((te+tc)/2000.d0)*pd
C
C......Begin modified linear interpolation search for JIJ......
C
      do iter1=1,MAXITR
C......Calculate emitter side electron temperature......
        tee=vi/(TWO*XK*dlog(b*dlea)-XK*dlog(ONE-bp*jij))
C
C......Calculate collector sheath attenuation factor......
        zetac=(-(h+HALF)+dsqrt((h+HALF)**2+8.d0*jcj*h))/
     &    (2.d0*jcj)
C
C......Calculate collector sheath height......
        vc=XK*tc*dlog(zetac)
C
C......Calculate collector emission factor......
        r=jcj*dexp(vc/(XK*tc))
C
C......Calculate collector side electron temperature......
        tec=(THREE*tee+TWO*tc*r)/(TWO*r+THREE)
C
C......Calculate neutralization potential and current density......
        phinc=1.7d0+.383d0*tec/tr
        jnc=AR*tec*tec*dexp(-phinc/(XK*tec))
C
C......Assign value to LTE limit for H and check
C      to see if LTE limits the electron temperature
C      at the collector edge - if so, calculate new
C      values for TEC, VC, and R......
        hs=jnc/j
        if ((r+HALF).gt.hs) then
          tec=tsc2(tee,tc,tr,hs,j,jcj,vc)
          r=hs-HALF
          ltec=.true.
        else
          ltec=.false.
        endif
C
C......Calculate average electron temperature......
        telect=(tee+tec)/TWO
C
C......Calculate ratio of spacing to total
C      electron mean free path, including
C      ion scattering......
        dl=dlea+3.4d+7*j*d/(telect**2.5d0)
C
C......If LTE has occured at the collector edge, the routine
C      LTE is called to check that the average electron
C      temperature is above the bulk LTE limit.  If not,
C      the LTE routine will calculate new emitter side,
C      collector side,  and average electron temperatures
C      The ratio of spacing to total mean free path is also
C      recalculated......
        if (ltec) call lte2(tee,tec,telect,tc,tr,hs,j,jcj,vc,dl,dlea,
```

```
     &    d,r)
C
C......Calculate radiation component of arc drop......
        vrad=9.65d+5*pd/(j*ta)*dexp(-2.d0/(XK*telect))*
     &    (ONE+.069d0*dexp(.58d0/(XK*telect))*
     &    (EMIS/dsqrt(d/10.d0)+HALF))
C
C......Guess JSJ and enter secant method iteration......
        jsj=TWO
        first=.true.
C.......First calculate some parameters in order to save time
C         in the iteration.......
        param1=TWO*XK*(tec-te+(tec-tc)*r)+vrad-jij*(
     &          3.89d0+TWO*XK*tee)
        param2=TWO*XK*(tee-te)
        param3=.75d0*dl+r
        param4=-ONE/(XK*tee)
C........Begin iteration..........
        do iter2=1,MAXITR
C......Calculate emitter sheath height......
          ve=(param2*(jsj-ONE)+param1)/(ONE+jij)
C
C......Calculate answer for JSJ and compute difference from
C      guess for JSJ......
          if (ve*param4.le.dlog(TINY)) then
C............Case where ve is so large that the exp function would
C              underflow.........
            ans=ONE+jij
          else
C...........Normal case............
            ans=ONE+(param3-HALF*jij)*dexp(-ve/(XK*tee))+jij
          endif
          f=jsj-ans
        if (dabs(f).lt.TOL1) go to 80
C
C......Update value of JSJ until convergence......
          if (first) then
            oldj=jsj
            oldf=f
            jsj=jsj-dsign(.2d0,f)
            first=.false.
          else
            newj=(oldj*f-jsj*oldf)/(f-oldf)
            oldj=jsj
            oldf=f
            jsj=newj
          endif
        if (dabs(jsj-oldj).lt.1.d-5*jsj) go to 80
        enddo
 80     if (iter2.gt.MAXITR) pause 'Max. iterations exceeded in
     &SATUR2 for finding current ratio'
C
C......Calculate value of JS from eqn. (35) of Massey,
C      McDaniel, and Bederson......
        js=jsp*dexp(612.d0*dsqrt(dsqrt(-j*jij*dsqrt(ve)))/te)
C
C......Compute error term......
       g=js/j-jsj
      if (dabs(g).lt.TOL2) go to 90
C
C......Update JIJ to make error small......
       if (iwhch.eq.1) then
          if ((jij.eq.0.d0).and.(g.gt.0.d0)) then
            pause ' No solution in SATUR2'
            return
          endif
          x1=jij
          y1=g
          jij=-.1d0
          x2=jij
          iwhch=2
       else if (iwhch.eq.2) then
         x2=jij
         y2=g
          if (y1*y2.gt.0.d0) then
            x1=x2
            y1=y2
            jij=jij+dsign(.1d0,g)
C.........Prevent jij from becoming equal to -1.  Make it the
C          nearest larger number.....
            if (jij.le.-ONE) jij=-.999999d0
         else
           iwhch=3
           ys=y2
           jij=(x1*y2-x2*y1)/(y2-y1)
         endif
```

```
          else if (iwhch.eq.3) then
            x3=jij
            y3=g
            if (y3*y1.lt.0.d0) then
              x2=x3
              y2=y3
              if (y3*ys.gt.0.d0) y1=y1/TWO
            else
              x1=x3
              y1=y3
              if (y3*ys.gt.0.d0) y2=y2/TWO
            endif
            ys=y3
            jij=(x1*y2-x2*y1)/(y2-y1)
          endif
        enddo
 90     if (iter1.gt.MAXITR) then
c         write(*,'(a)') ' Maximum iterations exceeded in
c      &SATUR2 for finding ion current'
          write(8,'(a)') ' Maximum iterations exceeded in
       &SATUR2 for finding ion current'
c         write(*,*) j,tr
          write(8,*) j,tr
          stop
        endif
C
C......Calculate arc drop......
        vd=ve-vc
        return
        END
C
C
        REAL*8 FUNCTION tsc2(tee,tc,tr,hs,j,jcj,vc)
C       IMPLICIT NONE
        INTEGER MAXITR
        REAL*8 tee,tc,tr,hs,j,jcj,vc,XK,ONE,TWO,THREE,TOL,AR,HALF
        PARAMETER (XK=8.6175d-5,ONE=1.d0,TWO=2.d0,THREE=3.d0,TOL=
       &           1.d-5,AR=120.d0,MAXITR=50,HALF=.5d0)
C
C**********************************************************
C                                                          *
C       The function TSC2 is called by the subroutines OBSTR2*
C       and SATUR2 in order to compute the collector side   *
C       electron temperature when LTE conditions exist at the*
C       collector and the collector sheath is negative.  It *
C       is very similar to the function TSC.  One difference *
C       is that the collector sheath is recalculated by TSC2.*
C                                                          *
C       Input values -                                     *
C       TEE           Emitter side electron temperature (K)  *
C       TC            Collector temperature (K)            *
C       TR            Cesium reservoir temperature (K)     *
C       HS            Ratio of neutralization current Jn to *
C                     Current density J                    *
C       J             Current density (amps/cm2)           *
C       JCJ           Ratio of back emission to current den- *
C                     sity                                 *
c       vc            collector sheath height (ev)         *
C                                                          *
C       Output values -                                    *
C       TSC2          LTE value for electron temperature at *
C                     collector edge of plasma (K)         *
C       VC            Recalculated value for collector     *
C                     sheath (eV)                          *
C                                                          *
C**********************************************************
C
        INTEGER iter
        REAL*8 dh,phinc,jnc,hss,dif,oldh,olddif,newh
        LOGICAL first,goon
C
C......Enter iteration (first guess for HS has already been
C       calculated by calling routine)......
C
C......Calculate collector edge electron temperature......
        first=.true.
        goon=.false.
        dh=ONE
        do iter=1,MAXITR
          tsc2=(THREE*tee+(TWO*hs-ONE)*tc)/(TWO*(hs+ONE))
C
C......Calculate neutralization work function and current density......
          phinc=1.7d0+.383d0*tsc2/tr
          jnc=AR*tsc2**2*dexp(-phinc/(XK*tsc2))
C
C......Find answer for HS, difference between guess and answer......
```

```
        hss=jnc/j
        dif=hs-hss
      if (dabs(dif).lt.TOL) go to 100
C
C......Update HS to make difference small......
        if (first) then
          oldh=hs
          olddif=dif
          hs=hs-dsign(dh,dif)
          hs=dmax1(hs,1.d-12)
          first=.false.
          dh=1.6d0*dh
        else
          if (.not.goon) then
            if (dif*olddif.gt.0.d0) then
              newh=hs-dsign(dh,dif)
              newh=dmax1(newh,1.d-12)
              dh=1.6d0*dh
            else
              goon=.true.
              newh=(oldh*dif-hs*olddif)/(dif-olddif)
            endif
          else
            newh=(oldh*dif-hs*olddif)/(dif-olddif)
          endif
          oldh=hs
          olddif=dif
          hs=newh
        endif
      if (dabs(hs-oldh).lt.1.d-5*hs) go to 100
      enddo
 100  if (iter.gt.MAXITR) pause 'Exceeded max. iterations in TSC2'
C
C......Recalculate VC......
      vc=XK*tc*dlog((hs-HALF)/jcj)
      return
      END
C
C
      SUBROUTINE lte2(tee,tec,tav,tc,tr,hs,j,jcj,vc,dl,dlea,d,r)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 tee,tec,tav,tc,tr,hs,j,jcj,vc,dl,dlea,d,r,XK,
     &       ONE,TWO,THREE,TOL,AR,HALF
      PARAMETER (XK=8.6175d-5,ONE=1.d0,TWO=2.d0,THREE=3.d0,TOL=
     &           1.d-5,AR=120.d0,MAXITR=50,HALF=.5d0)
C
C***************************************************************
C                                                              *
C      The routine LTE2 is called by OBSTR2 and SATUR2 to      *
C      perform checking and, if necessary,recomputation of     *
C      the electron temperatures when the collector sheath     *
C      is negative.  It is very similar to the routine         *
C      LTE, however, the collector sheath is re-               *
C      calculated in LTE2.                                     *
C                                                              *
C     Input values -                                           *
C     TEE         Electron temperature at emitter edge (K)     *
C     TEC         Electron temperature at collectore edge (K)*
C     TAV         Average electron temperature (K)             *
C     TC          Collector temperature (K)                    *
C     TR          Cesium reservoir temperature (K)             *
C     HS          Ratio of neutalization current to current    *
C                 density                                      *
C     J           Current density (amps/cm2)                   *
C     JCJ         Ratio of back emission to current density    *
C     VC          Collector sheath height (eV)                 *
C     DL          Ratio of gap to total electron mean free     *
C                 path                                         *
C     DLEA        Ratio of gap to electron-neutral mean        *
C                 free path                                    *
C     D           Interelectrode gap (mm)                      *
C                                                              *
C     Output values (recalculated) -                           *
C     TEE                                                      *
C     TEC                                                      *
C     TAV                                                      *
C     DL                                                       *
C     VC                                                       *
C                                                              *
C***************************************************************
C
      INTEGER iter1,iter2
      REAL*8 ts,dl1,tss,dif,oldt,olddif,newt,dh,
     &       phinc,jnc,hss,oldh,newh,tsc
      LOGICAL first,goon
```

```
C
C......First guess for TS......
      ts=tav
C
C......Enter secant method search for TS......
C
      first=.true.
      dh=TWO
      do iter1=1,MAXITR
C......Calculate new value for ratio of gap to mean free path......
        dl1=dlea+3.4d+7*j*d/ts**2.5d0
C
C......Calculate an answer for TS......
        tss=1.7d0/(XK*dlog((120.d0*ts*ts)/(j*dl1))-.383d0/tr)
C
C......Find difference between guess and answer.....
        dif=ts-tss
      if (dabs(dif).lt.TOL) go to 110
C
C......Update TS to make difference small......
        if (first) then
          oldt=ts
          olddif=dif
          ts=ts-dsign(50.d0,dif)
          first=.false.
        else
          newt=(oldt*dif-ts*olddif)/(dif-olddif)
          oldt=ts
          olddif=dif
          ts=newt
        endif
      if (dabs(ts-oldt).lt.1.d-5*ts) go to 110
      enddo
 110  if (iter1.gt.MAXITR) pause 'Max. iterations exceeded in LTE'
C
C......Check to see if average electron temperature is above
C       the limit.  If so, return without altering any
C       values......
      if (tav.ge.ts) return
C
C......If bulk LTE is in effect, replace TAV and DL with their
C       proper LTE values......
      tav=ts
      dl=dl1
C
C......TEC must now be recalculated, since it depends on TEE,
C       which will change to keep the average temperature
C       above its limit.  An iteration like that in the function
C       TSC2 is used......
      first=.true.
      goon=.false.
      dh=TWO
C
C......Begin iteration......
      do iter2=1,MAXITR
C......Calculate collector edge electron temperature......
        tsc=(TWO*THREE*ts+(TWO*hs-ONE)*tc)/(TWO*(hs+ONE)+
     &      THREE)
        phinc=1.7d0+.383d0*tsc/tr
        jnc=AR*tsc**2*dexp(-phinc/(XK*tsc))
        hss=jnc/j
        dif=hs-hss
      if (dabs(dif).lt.TOL) go to 120
        if (first) then
          oldh=hs
          olddif=dif
          hs=hs-dsign(dh,dif)
          hs=dmax1(hs,1.d-12)
          first=.false.
          dh=1.6d0*dh
        else
          if (.not.goon) then
            if (dif*olddif.gt.0.d0) then
              newh=hs-dsign(hs,dif)
              newh=dmax1(newh,1.d-12)
              dh=1.6d0*dh
            else
              newh=(oldh*dif-hs*olddif)/(dif-olddif)
              goon=.true.
            endif
          else
            newh=(oldh*dif-hs*olddif)/(dif-olddif)
          endif
          oldh=hs
          olddif=dif
          hs=newh
```

```
          endif
        if (dabs(hs-oldh).lt.1.d-5*hs) go to 120
        enddo
 120    if (iter2.gt.MAXITR) pause 'Max. iterations exceeded in LTE2'
C
C......Recompute TEC......
        tec=tsc
C
C......Recompute VC......
        vc=XK*tc*dlog((hs-HALF)/jcj)
C
C......Recompute TEE......
        tee=TWO*ts-tsc
        r=hs-HALF
        return
        END
C
        SUBROUTINE unig(te,tc,tr,d,phie,phic,j,ji,v,qe,sheath)
C       IMPLICIT NONE
        REAL*8 te,tc,tr,d,phie,phic,ji,v,qe,
       +  j,PI,XKE,TFACT,XNFACT,TOL,XK,ME,MI,DEFAULT,AR
        INTEGER sheath,TRY,ITMAX
        PARAMETER (PI=3.141592654,XKE=8.61753d-5,TFACT=1.05d0,
       +  XNFACT=0.8d0,TOL=1.d-5,XK=1.3807d-16,ME=9.1095d-28,
       +  MI=2.207d-22,DEFAULT=-99.d0,TRY=4,AR=120.d0,ITMAX=30)
C
C       UNIG
C       John McVey 30 March 1990
C       DOCUMENT CONTROL #C-568-006-D-033090
C
C       Rev. C: Modifications for use in CYLCON6
C          Calculation of cesiated work functions removed.
C          Changed to double precision.
C
C       Rev. D:  Eliminated problem with divide by zero in update routine.
C
C       Unig is a subroutine package for calculating the output voltage of
C       a thermionic converter operating in the diffusion-dominated
C       unignited mode.
C
C       INPUTS:
C              te              Emitter temperature in K
C              tc              Collector temperature in K
C              tr              Cesium reservoir temperature in K.
C              d               Interelectrode gap in centimeters
C              phie            Emitter work function in eV.
C              phic            Collector work function in eV.
C              j               Net electron current density in Amps/sq. cm.
C        OUTPUTS:
C              ji              Ion current density in Amps/sq. cm.
C              v               Output voltage in volts.
C              qe              Electron cooling in Watts/sq. cm.
C              sheath          Integer indicating sheath configuration.
C                              0 = no solution
C                              1 = DU
C                              2 = DD
C                              3 = UU
C                              4 = UD
C
C       uses du,dd,uu,ud,dn1,dn2,denav,tcalc,update,coefs
C
        INTEGER izone,itert,iwhich
        REAL*8 taav,na0,na1,naav,veli,nav,r,vele,alpha,
       +  dife,difi,lambde,lambdi,e,i,zetae,zetcpr,vevc,zetac,
       +  zetepr,xiepr,nenc,psi,telans,navans,f,g,x1,x2,x3,
       +  y1,y2,y3,f1,f2,f3,g1,g2,g3,xnew,ynew,update,tel,
       &  ve,vc,vp,js,jie,jc,pcs,dlam,teff,arate,ionprob,
       &  mue,mui,emob,imob,jion,heat
        js=AR*te*te*dexp(-phie/(XKE*te))
        jc=AR*tc*tc*dexp(-phic/(XKE*tc))
        pcs=2.45d+8*dexp(-8910.d0/tr)/dsqrt(tr)
        taav=(te+tc)/2.d0
        na0=1333.2d0*pcs/(XK*te)
        na1=1333.2d0*pcs/(XK*tc)
        naav=(na0+na1)/2.d0
        veli=dsqrt(8.d0*XK*taav/(PI*MI))
        dlam=d*1.2d-14*naav
        if (dlam.lt.1.d0) then
          teff=taav
        else if ((dlam.ge.1.d0).and.(dlam.le.10.d0)) then
          teff=taav+(dlam-1.d0)*(te/tc)/18.d0
        else
          teff=te
        end if
        arate=1333.2d0*pcs/dsqrt(2.d0*PI*MI*XK*teff)
        ionprob=1.d0/(1.d0+2.d0*dexp((3.89d0-phie)/(XKE*te)))
```

```
      jie=ionprob*arate*1.6022d-19
      tel=1.1d0*te
      if (j.lt.-jc) then
        v=-DEFAULT
       sheath=0
        return
      endif
      if (j.gt.js) then
        v=DEFAULT
       sheath=0
       return
      endif
      nav=1.d+11
      izone=0
      iwhich=1
      do itert=1,ITMAX
        r=tel/taav
        vele=dsqrt(8.d0*XK*tel/(PI*ME))
        alpha=vele/veli
       mue=emob(tel,naav,nav)
       mui=imob(taav,naav,nav)
        dife=mue*tel*XKE
        difi=mui*taav*XKE
        lambde=3.d0*dife/vele
        lambdi=3.d0*difi/veli
        e=.75d0*r/(r+1.d0)*d/lambde
        i=.75d0/(r+1.d0)*d/lambdi
       call du(j,js,jc,jie,tel,te,i,e,alpha,zetae,zetcpr,ve,
     +         vc,vevc)
        if ((ve.gt.0.d0).and.(vc.gt.0.d0)) then
          sheath=1
          goto 12
       endif
       call dd(j,js,jc,jie,te,tc,i,e,alpha,zetae,zetac,ve,vc,vevc)
        if ((ve.gt.0.d0).and.(vc.gt.0.d0)) then
          sheath=2
          goto 12
       endif
       call uu(j,js,jc,jie,tel,te,i,e,alpha,zetepr,zetcpr,
     +         xiepr,ve,vc,vevc)
        if ((ve.gt.0.d0).and.(vc.gt.0.d0)) then
          sheath=3
          goto 12
       endif
       call ud(j,js,jc,jie,tel,te,tc,i,e,alpha,zetepr,zetac,xiepr,
     +         ve,vc,vevc)
        if ((ve.gt.0.d0).and.(vc.gt.0.d0)) then
          sheath=4
          goto 12
       endif
       sheath=0
       izone=izone+1
        if (izone.lt.TRY) then
          tel=TFACT*tel
          nav=XNFACT*nav
          iwhich=1
          goto 100
       else
          v=-dsign(DEFAULT,j)
          return
       endif
12     if ((sheath.eq.2).or.(sheath.eq.4)) then
          call dn1(j,jc,i,e,zetac,lambde,d,nenc,psi)
       else
          call dn2(j,jc,i,e,zetcpr,lambde,d,nenc,psi)
       endif
        vp=XKE*tel*((psi-1.d0)*dlog(nenc))
13     call tcalc(te,tc,j,js,jc,ve,vc,vp,zetae,zetac,telans,sheath)
       call denav(j,jc,zetcpr,zetac,nenc,vele,navans,sheath)
       f=tel-telans
       g=nav-navans
       if (iwhich.eq.1) then
         x1=nav
         y1=tel
         f1=f
         g1=g
         tel=tel-dsign(20.d0,f)
         iwhich=2
       elseif (iwhich.eq.2) then
         x2=nav
          y2=tel
         f2=f
         g2=g
         nav=nav-dsign(1.d+9,g)
         iwhich=3
       elseif (iwhich.eq.3) then
```

```
          if ((dabs(f/tel).lt.TOL).and.(dabs(g/nav).lt.TOL))
     +      goto 200
          x3=nav
          y3=tel
          f3=f
          g3=g
          xnew=update(x1,x2,x3,f1,f2,f3,g1,g2,g3)
          if (xnew.lt.0.d0) xnew=x3
          ynew=update(y1,y2,y3,f1,f2,f3,g1,g2,g3)
          if (dabs(ynew-y3).gt.y3/2.d0) ynew=y3*(1.d0+dsign(.5d0,ynew-
     +         y3))
          x1=x2
          y1=y2
          f1=f2
          g1=g2
          x2=x3
          y2=y3
          f2=f3
          g2=g3
          nav=xnew
          tel=ynew
        endif
 100  enddo
C
 200  if (itert.gt.ITMAX) then
        v=-dsign(DEFAULT,j)
        return
      end if
      ji=jion(j,jc,zetcpr,zetac,alpha,sheath)
      v=phie-phic+vevc-vp
      qe=heat(j,js,ji,phie,te,tel,ve,zetae,sheath)
C
      return
      END
C
C
      REAL*8 FUNCTION update(x1,x2,x3,f1,f2,f3,g1,g2,g3)
C     IMPLICIT NONE
      REAL*8 x1,x2,x3,f1,f2,f3,g1,g2,g3
C
C      Updates parameters for the two-dimensional secant method
C      iteration in UNIG used to find the average electron temperature
C      and plasma density.
C
      REAL*8 r,u
      r=x1*(f2*g3-f3*g2)+x2*(f3*g1-f1*g3)+x3*(f1*g2-f2*g1)
      u=f2*g3-f3*g2+f3*g1-f1*g3+f1*g2-f2*g1
      if (u.eq.0.d0) pause ' U is zero, chuckie!'
      update=r/u
      return
      END
C
C
      SUBROUTINE du(j,js,jc,jie,tel,te,i,e,alpha,zetae,zetcpr,ve,
     +    vc,vevc)
C     IMPLICIT NONE
      REAL*8 j,js,jc,jie,tel,te,i,e,alpha,zetae,zetcpr,ve,vc,vevc,
     +  XKE,ZERO,ONE,TWO,DEFAULT
      PARAMETER (XKE=8.61753d-5,ZERO=0.d0,ONE=1.d0,TWO=2.d0,
     +  DEFAULT=-99.d0)
C
C      Solves for the emitter and collector sheath heights in the
C      condition where an ion retaining sheath is at the emitter
C      and an electron retaining one is at the collector.
C
      REAL*8 a,b,c,disc
      LOGICAL bad1

      bad1 = .false.
      a=TWO*i*js
      b=TWO*js+(e-i)*j
      c=-(j*(ONE+TWO*e)+alpha*jie*(ONE+TWO*i))
      disc=b*b-4.d0*a*c
      if (disc.lt.ZERO) bad1=.true.
      disc=dmax1(disc,ZERO)
      zetae=(-b+dsqrt(disc))/(TWO*a)
      if ((zetae.le.ZERO).or.(bad1)) then
        ve=DEFAULT
      else
        ve=-XKE*te*dlog(zetae)
      endif
      zetcpr=(j+jc)*(TWO-zetae)/(j-js*zetae*zetae+alpha*jie)
      if (zetcpr.gt.ZERO) then
        vc=-XKE*tel*dlog(zetcpr)
      else
        vc=DEFAULT
```

```
      endif
      vevc=ve+vc
      return
      END
C
C
      SUBROUTINE dd(j,js,jc,jie,te,tc,i,e,alpha,zetae,zetac,ve,
     +    vc,vevc)
C     IMPLICIT NONE
      INTEGER ITMAX
      REAL*8 j,js,jc,jie,te,tc,i,e,alpha,zetae,zetac,ve,vc,vevc,
     +  XKE,ZERO,ONE,TWO,DEFAULT,FOUR,TOL
      PARAMETER (XKE=8.61753d-5,ZERO=0.d0,ONE=1.d0,TWO=2.d0,
     +  DEFAULT=-99.d0,ITMAX=40,FOUR=4.d0,TOL=1.d-5)
C
C      Solves for the emitter and collector sheath heights in the
C      condition where there are ion retaining sheaths at both
C      electrodes.
C
      INTEGER iter
      REAL*8 a,b1,c1,b,c,disc1,q,disc2,f,x1,x2,f1,f2
      LOGICAL bad1,bad2
      a=(TWO*i-ONE)*js
      b1=TWO*js+(ONE+e-i)*j
      c1=-TWO*((ONE+e)*j+i*alpha*jie)
      zetac=ZERO
      do iter=1,ITMAX
        bad1=.false.
        bad2=.false.
        b=b1+jc*zetac
         c=c1-TWO*jc*zetac
         disc1=b*b-FOUR*a*c
         if (disc1.lt.ZERO) bad1=.true.
         disc1=dmax1(disc1,ZERO)
         zetae=(-b+dsqrt(disc1))/(TWO*a)
         q=(TWO*alpha*jie-zetae*(TWO*js*zetae-j))/(TWO-zetae)
         disc2=(j+q)*(j+q)+16.d0*jc*q
         if (disc2.lt.ZERO) bad2=.true.
         disc2=dmax1(disc2,ZERO)
         f=zetac-(-(j+q)+dsqrt(disc2))/(FOUR*jc)
        if (iter.eq.1) then
          x1=zetac
          f1=f
           zetac=zetac-dsign(.1d0,f)
        else
          x2=zetac
          f2=f
          zetac=x2+f2*(x2-x1)/(f1-f2)
          x1=x2
          f1=f2
        endif
        if (dabs((zetac-x2)/zetac).lt.TOL) goto 100
      enddo
 100  if (iter.gt.itmax) bad2=.true.
      if ((zetae.le.ZERO).or.(bad1)) then
        ve=DEFAULT
      else
        ve=-XKE*te*dlog(zetae)
      endif
      if ((zetac.le.ZERO).or.(bad2)) then
        vc=DEFAULT
      else
        vc=-XKE*tc*dlog(zetac)
      endif
      vevc=ve-vc
      return
      END
C
C
      SUBROUTINE uu(j,js,jc,jie,tel,te,i,e,alpha,zetepr,zetcpr,
     +      xiepr,ve,vc,vevc)
C     IMPLICIT NONE
      INTEGER ITMAX
      REAL*8 j,js,jc,jie,tel,te,i,e,alpha,zetepr,zetcpr,xiepr,ve,
     +  vc,vevc,XKE,ZERO,ONE,TWO,DEFAULT,FOUR,TOL
      PARAMETER (XKE=8.61753d-5,ZERO=0.d0,ONE=1.d0,TWO=2.d0,
     +  DEFAULT=-99.d0,ITMAX=40,FOUR=4.d0,TOL=1.d-5)
C
C      Solves for the emitter and collector sheath heights in the
C      condition where there are electron retaining sheaths at both
C      electrodes.
C
      INTEGER iter
      REAL*8 a,b,c,disc,tau,zzx,fmin,f,dfdz,delta
      LOGICAL bad1
      a=(TWO*i+ONE)*alpha*jie
```

```
      b=-j*(ONE-e+i)
      c=TWO*(i+ONE)*(j-js)
      disc=b*b-FOUR*a*c
      disc=dmax1(disc,ZERO)
      zetepr=(-b+dsqrt(disc))/(TWO*a)
      zetepr=dmax1(zetepr,ZERO)
      tau=tel/te
      bad1=.false.
      if (b.le.ZERO) then
        zzx=(-b/(a*(tau+ONE)))**(ONE/tau)
        if (zzx.ge.ZERO) then
          fmin=a*zzx**(tau+ONE)+b*zzx+c
          if (fmin.gt.ZERO) then
            bad1=.true.
            zetepr=zzx
            go to 110
          endif
        endif
      endif
      do iter=1,ITMAX
        xiepr=dsign(dabs(zetepr)**tau,zetepr)
        f=a*zetepr*xiepr+b*zetepr+c
        dfdz=a*(tau+ONE)*xiepr+b
        if (dfdz.ne.ZERO) delta=-f/dfdz
        zetepr=zetepr+delta
        if (dabs(delta/zetepr).lt.TOL) goto 100
      enddo
 100  if (iter.gt.ITMAX) bad1=.true.
 110  xiepr=dsign(dabs(zetepr)**tau,zetepr)
      if ((zetepr.le.ZERO).or.(bad1)) then
        ve=DEFAULT
      else
        ve=-XKE*tel*dlog(zetepr)
      endif
      zetcpr=(j+jc)/(alpha*jie*xiepr-(js-j)/zetepr)
      if (zetcpr.gt.ZERO) then
        vc=-XKE*tel*dlog(zetcpr)
      else
        vc=DEFAULT
      endif
      vevc=vc-ve
      return
      END
C
C
      SUBROUTINE ud(j,js,jc,jie,tel,te,tc,i,e,alpha,zetepr,zetac,
     +    xiepr,ve,vc,vevc)
C     IMPLICIT NONE
      INTEGER ITMAX
      REAL*8 j,js,jc,jie,tel,te,tc,i,e,alpha,zetepr,zetac,xiepr,
     +  ve,vc,vevc,XKE,ZERO,ONE,TWO,DEFAULT,TOL
      PARAMETER (XKE=8.61753d-5,ZERO=0.d0,ONE=1.d0,TWO=2.d0,
     +  DEFAULT=-99.d0,ITMAX=50,TOL=1.d-5)
C
C      Solves for the emitter and collector sheath heights in the
C      condition where an electron retaining sheath is at the emitter
C      and an ion retaining one is at the collector.
C
      INTEGER iter,ii
      REAL*8 f(2),x(2),delta(2),pderiv(2,2),tau,a,b1,c,zetacg,
     +  zetepg,q,determ
      LOGICAL bad
      tau=tel/te
      a=TWO*i*alpha*jie
      b1=(e-i)*j
      c=(j-js)*(TWO*i+ONE)
      iter=1
      zetacg=ZERO
      zetepg=ONE
      x(1)=zetepg
      x(2)=zetacg
      do iter=1,ITMAX
        bad=.false.
        xiepr=dsign(dabs(x(1))**tau,x(1))
        f(1)=x(1)*(a*xiepr+b1+jc*x(2))+c
        q=TWO*alpha*jie*xiepr-TWO*(js-j)/x(1)-j
        f(2)=x(2)*(TWO*jc*x(2)+j+q)-TWO*q
        if ((dabs(f(1)).lt.TOL).and.(dabs(f(2)).lt.TOL)) goto 100
        pderiv(1,1)=(tau+ONE)*a*xiepr+b1+jc*x(2)
        pderiv(2,1)=(x(2)-TWO)*(TWO*alpha*tau*jie*(xiepr/x(1))
     +     +TWO*(js-j)/(x(1)*x(1)))
        pderiv(1,2)=jc*x(1)
        pderiv(2,2)=4.d0*jc*x(2)+j+q
        determ=pderiv(1,1)*pderiv(2,2)-pderiv(1,2)*pderiv(2,1)
        if (determ.eq.ZERO) then
          bad=.true.
```

```
          delta(1)=ZERO
          delta(2)=ZERO
        else
          bad=.false.
          delta(1)=(pderiv(1,2)*f(2)-pderiv(2,2)*f(1))/determ
          delta(2)=(pderiv(2,1)*f(1)-pderiv(1,1)*f(2))/determ
        endif
        if ((dabs(delta(1)).lt.TOL).and.(dabs(delta(2)).lt.
     +     TOL)) goto 100
        do ii=1,2
          x(ii)=x(ii)+delta(ii)
        enddo
      enddo
 100  if (iter.gt.itmax) bad=.true.
      zetepr=x(1)
      zetac=x(2)
      xiepr=dsign(dabs(zetepr)**tau,zetepr)
      if ((zetepr.le.ZERO).or.(bad)) then
        ve=DEFAULT
      else
        ve=-XKE*tel*dlog(zetepr)
      endif
      if ((zetac.le.ZERO).or.(bad)) then
        vc=DEFAULT
      else
        vc=-XKE*tc*dlog(zetac)
      endif
      vevc=-ve-vc
      return
      END
C
C
      SUBROUTINE dn1(j,jc,i,e,zetac,lambde,d,nenc,psi)
C     IMPLICIT NONE
      REAL*8 j,jc,i,e,zetac,lambde,d,nenc,psi,ONE,TWO
      PARAMETER(ONE=1.d0,TWO=2.d0)
C
C      Evaluates parameters NENC and PSI for computation of the
C      plasma drop VP in UNIG.  Used for cases in which there is an ion
C      retaining collector sheath.
C
      REAL*8 jczc
      jczc=TWO*jc*zetac+j
      nenc=ONE+TWO*e*j/jczc+TWO*zetac*i/(TWO-zetac)
      psi=j*(.75d0*d/lambde)/(e*j+jczc*i*zetac/(TWO-zetac))
      return
      END
C
C
      SUBROUTINE dn2(j,jc,i,e,zetcpr,lambde,d,nenc,psi)
C     IMPLICIT NONE
      REAL*8 j,jc,i,e,zetcpr,lambde,d,nenc,psi,ONE,TWO
      PARAMETER (ONE=1.d0,TWO=2.d0)
C
C      Evaluates parameters NENC and PSI for computation of the
C      plasma drop VP in UNIG.  Used for cases in which there is an
C      electron retaining collector sheath.
C
      REAL*8 jcj
      jcj=TWO*jc+(TWO-zetcpr)*j
      nenc=ONE+TWO*zetcpr*e*j/jcj+TWO*i
      psi=j*(.75d0*d/lambde)/(e*j+jcj*i/zetcpr)
      return
      END
C
C
      SUBROUTINE denav(j,jc,zetcpr,zetac,nenc,vele,nav,sheath)
C     IMPLICIT NONE
      INTEGER sheath
      REAL*8 j,jc,zetcpr,zetac,nenc,vele,nav,ONE,TWO,EC
      PARAMETER (ONE=1.d0,TWO=2.d0,EC=1.602d-19)
C
C      Calculates the average plasma density in the interelectrode
C      space.  This is used in UNIG to calculate the amount of
C      electron-ion scattering.
C
      if (sheath.eq.0) then
        nav=(j+TWO*jc)*(nenc+ONE)/(EC*vele)
      elseif ((sheath.eq.1).or.(sheath.eq.3)) then
        nav=((TWO-zetcpr)*j+TWO*jc)*(nenc+ONE)/(zetcpr*EC
     +       *vele)
      elseif ((sheath.eq.2).or.(sheath.eq.4)) then
        nav=(j+TWO*jc*zetac)*(nenc+ONE)/(EC*vele)
      endif
      return
      END
```

```
C
C
      SUBROUTINE tcalc(te,tc,j,js,jc,ve,vc,vp,zetae,zetac,tel,sheath)
C     IMPLICIT NONE
      INTEGER sheath
      REAL*8 te,tc,j,js,jc,ve,vc,vp,zetae,zetac,tel,TWOK
      PARAMETER (TWOK=5802.5d0)
C
C      Uses energy balance to calculate an average electron temperature
C      in the interelectrode space.
C
      REAL*8 y
      if (sheath.eq.1) then
        y=js*zetae+jc
        tel=(js*zetae*te+TWOK*j*(vp-vc)+jc*tc)/y
      elseif (sheath.eq.2) then
        y=js*zetae+jc*zetac
        tel=(js*zetae*te+TWOK*j*vp+jc*zetac*tc)/y
      elseif (sheath.eq.3) then
        y=js+jc
        tel=(js*te+TWOK*j*(ve+vp-vc)+jc*tc)/y
      elseif (sheath.eq.4) then
        y=js+jc*zetac
        tel=(js*te+TWOK*j*(ve+vp)+jc*zetac*tc)/y
      endif
      return
      END
C
C
      REAL*8 FUNCTION jion(j,jc,zetcpr,zetac,alpha,sheath)
C     IMPLICIT NONE
      REAL*8 j,jc,zetcpr,zetac,alpha,TWO
      PARAMETER(TWO=2.d0)
      INTEGER sheath
C
      if ((sheath.eq.1).or.(sheath.eq.3)) then
        jion=((TWO-zetcpr)*j+TWO*jc)/(alpha*zetcpr)
      else if ((sheath.eq.2).or.(sheath.eq.4)) then
        jion=zetac*(j+TWO*jc*zetac)/(alpha*(TWO-zetac))
      end if
      return
      END
C
C
      REAL*8 FUNCTION heat(j,js,ji,phie,te,tel,ve,zetae,sheath)
C     IMPLICIT NONE
      REAL*8 j,js,ji,phie,te,tel,ve,zetae,TK,VI
      PARAMETER(TK=2.d0/11604.5d0,VI=3.89d0)
      INTEGER sheath
C
      if ((sheath.eq.1).or.(sheath.eq.3)) then
        heat=j*(phie+ve+tel*TK)+js*(te-tel)*TK+ji*(VI-phie)
      else if ((sheath.eq.2).or.(sheath.eq.4)) then
        heat=j*(phie+tel*TK)+js*zetae*(te-tel)*TK+ji*(ve+VI-phie)
      end if
      return
      END
C
C
      REAL*8 FUNCTION emob(tel,na,n)
C     IMPLICIT NONE
      REAL*8 tel,na,n
C
      REAL*8 csecea,lnl,nuea,nuei,re,taue,muea,muei,mue
C       Evaluate electron-neutral cross-section (cm2)
      csecea=1.d-16*(535.d0+tel*(-.27d0+tel*5.2d-5))
C
C      Evaluate the Coulomb logarithm, electron-neutral collision
C      frequency, electron-ion collision frequency, and the ratio.
      if (n.gt.0.d0) then
        lnl=dlog(12390.d0*tel**1.5/dsqrt(n))
      else
        lnl=dlog(12390.d0*tel**1.5/1.d-16)
      end if
      nuea=7.319d+5*na*csecea*dsqrt(tel)
      nuei=dmax1(1.070d0*n*lnl/tel**1.5,1.d-16)
      re=nuei/nuea
C
C      Calculate the electron mobility (cm2/volt-sec).
      taue=(1.d0+re*(14.1d0+re*(30.6d0+re*16.3d0)))/(1.d0+re*
     +    (21.1d0+re*(37.4d0+re*16.3d0)))
      muea=5.167d+17/nuea
      muei=3.058d+17/nuei
      mue=muea*muei/(muea+muei)*taue
      emob=mue/299.8d0
C
```

```
      return
      END
C
C
      REAL*8 FUNCTION imob(ti,na,n)
C     IMPLICIT NONE
      REAL*8 ti,na,n
C
      REAL*8 csecia,lnl,nuia,nuii,ri,taui,muia,mui
C       Evaluate ion-neutral cross-section (cm2)
      csecia=1.d-16*(1667.d0+ti*(-.807d0+ti*(4.77d-4-1.047d-7*ti)))
C
C      Evaluate the Coulomb logarithm, ion-neutral collision frequency,
C      ion-ion collision frequency, and the ratio.
      if (n.gt.0.d0) then
        lnl=log(12390.d0*ti**1.5/dsqrt(n))
      else
        lnl=log(12390.d0*ti**1.5/1.d-16)
      end if
      nuia=1051.d0*na*csecia*dsqrt(ti)
      nuii=dmax1(1.537d-3*n*lnl/ti**1.5,1.d-16)
      ri=nuii/nuia
C
C      Calculate the ion mobility (cm2/volt-sec).
      taui=(1.d0+ri*(4.2d0+ri*2.86d0))/(1.d0+ri*(4.24d0+ri*2.91d0))
      muia=1.959d+12/nuia
      mui=muia*taui
      imob=mui/299.8d0
C
      return
      END

      REAL*8 FUNCTION ndsphi(te,tr,phi0)
C     IMPLICIT NONE
      INTEGER MAXITR
      REAL*8 te,tr,phi0,SMALL,ERRTOL
      PARAMETER(SMALL=1.d-5,ERRTOL=1.d-6,MAXITR=100)
C
C     Written by John McVey and Jean-Louis Desplat
C     Control #C-568-007-D-061290
C     This version uses a value of 1.95 eV for the cesium ion
C     adsorption energy rather than 2.04 eV (see functions f1 and f2).
C
C     uses f1,f2
C
C     ************************************************************
C
C            The function Nedsphi calculates the cesiated emitter
C            work function based on the emitter temperature,
C            cesium reservoir temperature (cesium pressure), and an
C            effective bare work function of the emitter surface.
C            The equations are based on the article "Correlation of
C            Emission Processes for Adsorbed Alkali Films on Metal
C            Surfaces" by N.S. Rasor and C. Warner, Journal of
C            Applied Physics, Vol. 35, #9, 1964.  This theory is
C            inaccurate for high bare work functions and low values of
C            T/TR (Phi0 above 5.5 and T/Tr below 2.5 simultaneously, for
C            example).  The theory does take into account the slight
C            non-uniqueness in T/Tr.
C
C            Inputs:
C               Te   Emitter temperature in K.
C               Tr   Cesium reservoir temperature in K.
C               Phi0 Effective emitter bare work function in eV.
C
C            Outputs:
C                    Returns cesiated emitter work function in eV.
C      Version D is double precision.
C     ************************************************************
C
      INTEGER itcnt,i
      REAL*8 x(2),f(2),p(2,2),cor(2),cov,dphi,dy,dx,xdx1,s1,
     *  ydy1,s2,determ,err1,err2,f1,f2
c
c
*      if ( te/tr .le. 2.5d0 ) then                               KHEL 4/27/93
*         ndsphi = 2.1                                            KHEL 4/27/93
*                                                   return        KHEL 4/27/93
*      end if
C     Initial guesses
      cov=dmax1((phi0-te/tr)/(phi0-1.d0),1.d-6)
      if (cov.le.0.5d0) then
        dphi=2.2d0*(phi0-1.5d0)*cov
      else
        dphi=1.1d0*(phi0-1.5d0)
      endif
```

```
      dphi=dmax1(dphi,1.d-6)
      do itcnt=1,MAXITR
         dy=dmax1(SMALL*dphi,1.d-6)
         dx=dmax1(SMALL*cov,1.d-6)
         x(1)=cov
         x(2)=dphi
C        Compute values of two functions which will be zero at solution.
         f(1)=f1(cov,dphi,te,tr,phi0)
         f(2)=f2(cov,dphi,te,phi0)
         if (cov.lt.0.2d0) then
           xdx1=cov-dx
            s1=1.d0
         else
           xdx1=cov+dx
            s1=-1.d0
         endif
         if (dphi.lt.0.2d0) then
           ydy1=dphi-dy
            s2=1.d0
         else
           ydy1=dphi+dy
            s2=-1.d0
         endif
C        Compute partial derivatives of both functions.
         p(1,1)=(f(1)-f1(xdx1,dphi,te,tr,phi0))/(dsign(dx,s1))
         p(1,2)=(f(1)-f1(cov,ydy1,te,tr,phi0))/(dsign(dy,s2))
         p(2,1)=(f(2)-f2(xdx1,dphi,te,phi0))/(dsign(dx,s1))
         p(2,2)=(f(2)-f2(cov,ydy1,te,phi0))/(dsign(dy,s2))
C        Perform Newton-Raphson.
         determ=p(1,1)*p(2,2)-p(2,1)*p(1,2)
         if (dabs(determ).le.1.0d-20) then
           pause 'No convergence in ndsphi'
            ndsphi=-1.0d-12
           return
         else
           cor(1)=(f(2)*p(1,2)-f(1)*p(2,2))/determ
           cor(2)=(f(1)*p(2,1)-f(2)*p(1,1))/determ
         endif
         do i=1,2
           x(i)=x(i)+cor(i)
         enddo
         err1=dabs(cor(1)/x(1))
         err2=dabs(cor(2)/x(2))
         cov=dmin1(x(1),.99)
         cov=dmax1(cov,0.)
         dphi=dmax1(x(2),0.)
         if (((err1.lt.ERRTOL).and.(err2.lt.ERRTOL)).or.((dabs(f(1))
     &     .lt.ERRTOL).and.(dabs(f(2)).lt.ERRTOL))) go to 10
      enddo
C     Return value of cesiated work function.
 10   if (itcnt.gt.MAXITR) then
        pause 'No convergence in ndsphi'
        ndsphi=-1.0d-12
      else
        ndsphi=phi0-dphi
      endif
      return
      END
C     ************************************************************
      REAL*8 FUNCTION f1(x,y,te,tcs,phi0)
C     IMPLICIT NONE
      REAL*8 x,y,te,tcs,phi0,PI,K,ONE,TWO,HALF,TPMK,VI,PHII0
      PARAMETER (PI=3.141592654,K=1.d0/11604.5d0,ONE=1.d0,TWO=2.d0,
     &  TPMK=TWO*PI*2.207d-22*1.381d-16,HALF=0.5d0,VI=3.89d0,PHII0=
     &  1.95d0)
C
C     **********************************************
C
C        F1 is called by ndsphi.
C        The value at solution will be near zero.
C
C     **********************************************
C

      REAL*8 phia0,e0,pcs,g,factr1,factr2,mucs,sigfcs
      phia0=.777d0*dsqrt(phi0)
      e0=phi0-phia0-VI+PHII0
      pcs=2.45d+8*dexp(-8910.d0/tcs)/dsqrt(tcs)
      mucs=1333.d0*pcs/dsqrt(TPMK*te)
      g=.18d0+.2d0*x
      factr1=ONE+HALF*dexp((e0-g*y)/(K*te))
      factr2=x/(ONE-x)*dexp(x/(ONE-x))
      sigfcs=HALF*dexp(62.d0+4.8d0*x*(ONE-HALF*x))
      f1=sigfcs*factr2*dexp(-phia0/(K*te))/(factr1*mucs)-ONE
      return
      END
C     ************************************************************
```

```
      REAL*8 FUNCTION f2(x,y,te,phi0)
C     IMPLICIT NONE
      REAL*8 x,y,te,phi0,EC,SIGCS,RCS,ALPHCS,K,PI,ONE,TWO,A,B,VI,PHII0
      PARAMETER (EC=4.8032d-10,SIGCS=3.56d+14,RCS=1.4d-8,ALPHCS=1.5d-23,
     &    K=1.d0/11604.5d0,PI=3.141592654,ONE=1.d0,TWO=2.d0,
     &    A=6.25d+11*4.*PI*EC*EC*SIGCS*RCS,B=TWO*PI*ALPHCS*SIGCS/RCS,
     &    VI=3.89d0,PHII0=1.95d0)
C
C     ***********************************************
C
C        F2 is called by ndsphi.
C        The value at solution will be near zero.
C
C     ***********************************************
C
      REAL*8 g,e0,phia0
      g=.18d0+.2d0*x
      phia0=.777d0*dsqrt(phi0)
      e0=phi0-phia0-VI+PHII0
      f2=y*(ONE+B*g*x+TWO*dexp((-e0+g*y)/(K*te)))-A*x
      return
      END
C     **********************************************************
      Real*8 Function QGapCond(Te,Tc,Tr,D)

      Real*8 Te,Tc,Tr,D,Pcs,Kcs

      Pcs = 2.45D+8 * exp(-8910.D0/Tr)/SQRT(Tr)
      Kcs = 5.5D-5

      QGapCond = Kcs*(Te-Tc)/(D + 1.15D-5*(Te-Tc)/Pcs)

      End
      Subroutine Convect
**********************************************************
*                                                        *
*                 Subroutine Convect                     *
*                 --------------                         *
*                                                        *
*              Written by: Ron Pawlowski                 *
*              Date      : February, 1990                *
*              --------------------------                *
*                                                         *
*     Computes the temperature of the coolant within     *
*     cylindrical flow channels by solving the           *
*     differential equation for temperature rise         *
*     through the core ( equation 6.6.8 in Elements of   *
*     Nuclear Reactor Design, J. Weisman ed., Kreiger    *
*     Publishing Company, 1983, with CpdT substituted    *
*     for dh.)  The differential equation is solved      *
*     using the fourth-order Runge-Kutta solution        *
*     technique.  This treatment allows the temperature  *
*     dependences of the coolant properties to be        *
*     included in the analysis.  The output for this     *
*     module is the axial temperature profile of the     *
*     coolant within the flow channel.                   *
*                                                        *
*      This code is hardwired to adjust the width of     *
*      the spatial intervals until the exit              *
*      temperature converges to within 0.1 degrees K.    *
*                                                        *
**********************************************************
*     Real*8 T, Told, Tinlet, h, z, f, k1, k2, k3, k4, mdot          KHEL 4/26/93
      Real*8 T, Told, Tinlet, h, z, mdot                             KHEL 5/30/93
      Real*8 Cp, HeatFlux, Rbound(10)                                KHEL 5/30/93
      Real*8 De, G, W, CoolTbl(2000,2), Zmax, Zmin, D2, D1
      Integer N, I, Kmax
      Integer Rmesh(9), Mat(5)                                       KHEL 5/30/93
      Common /CoolProp/  Tinlet, De, G, W, D2, D1, mdot              KHEL 4/26/93
      Common /TTAB/ CoolTbl
      Common /Zdata/ Zmin, Zmax, Kmax
      Common /Rdata/ Rbound, Rmesh, Mat                              KHEL 5/30/93

      Kmax = 10
      N = Kmax/2
      CoolTbl(1,1) = Zmin
      CoolTbl(1,2) = Tinlet
      Told = Tinlet

   10 T = Tinlet
      h = (Zmax-Zmin)/(N-1)
      do 100 I=1, N-1
          z = (I-1)*h
*          k1 = h*f(z,T)                                             KHEL 5/30/93
*          k2 = h*f(z+h/2,T+k1/2)                                    KHEL 5/30/93
*          k3 = h*f(z+h/2,T+k2/2)                                    KHEL 5/30/93
```

```
*           k4 = h*f(z+h,T+k3)                                          KHEL 5/30/93
*           T = T + (k1 + 2*k2 + 2*k3 + k4)/6                           KHEL 5/30/93
c          T = T + ((HeatFlux(z)*Rbound(10)*Rbound(10)*3.14159D0*h)/     KHEL 5/30/93
c     a            (Mdot*cp(T,W)))                                       KHEL 5/30/93

         T = T + 4.0d0/( De*G*cp(T,W) ) * HeatFlux(z) * h              KHEL 5/30/93

           CoolTbl(I+1,1) = z+h
           CoolTbl(I+1,2) = T
 100     Continue

       If (ABS(T-Told).GE.0.1) then
           Told = T
           N = N*2
           if (N .gt. 2000) then                                         KHEL 5/29/93
               write(8,*) ' Nonconvergence in Convect'                   KHEL 5/29/93
               write(8,*) ' Execution Terminated '                       KHEL 5/30/93
               stop                                                      KHEL 5/29/93
           end if                                                        KHEL 5/29/93
         Else
           Goto 160
         EndIf

       Goto 10

 160 Write(8,200) T                                                      KHEL 5/27/93
 200 Format (/' Temperature of coolant at core exit: ',F10.1,            KHEL 5/27/93
    a             ' degrees K.')                                         KHEL 5/27/93
       End
       Real*8 Function f(z,T)
*************************************************************
*      Returns the value of the derivative of T with respect  *
*      to z (dT/dz), as given by equation 6.6.8 of Weisman    *
*      (see reference in the comments for the main porgram).  *
*************************************************************
       Real*8 z, T, G, De, HeatFlux, Cp, Tinlet, W, D1, D2, mdot         KHEL 4/26/93
       Common /CoolProp/  Tinlet, De, G, W, D2, D1, mdot                 KHEL 4/26/93

       f = 4.0D0/(G*De*Cp(T,W)) * HeatFlux(z)

       End
       Real*8 Function Cp(T,W)
*************************************************************
*      Uses correlations from the Sodium-NaK Engineering      *
*      Handbook (O. Foust, ed.; vol. 1 pp. 52-53) to return   *
*      the value of the heat capacity of the NaK coolant for  *
*      a given temperature T and potassium weight fraction W  *
*      for the coolant (e.g. eutectic NaK-78 has W=78.)       *
*      Only single phase coolants are modeled.  If the        *
*      temperature of the coolant is higher than the boiling  *
*      point of NaK at the given sodium-potassium composition,*
*      this routine reports the error and halts the program.  *
*                                                             *
*      Units are in Joules/(kilogram*K).                      *
*                                                             *
*************************************************************
       Real*8 T, BoilingPt, CpNa, CpK, W

       BoilingPt = ((756.5-881.4)*W + 881.4) + 273.1
       If (T.GT.BoilingPt) then
c          Write(*,100) T, INT(W*100)
           Write(8,100) T, INT(W*100)
           Stop
         EndIf

       CpNa = (1.43612D0 - 5.80237D-4*T + 4.62081D-7*T**2)*1000.D0
       CpK  = (0.83850D0 - 3.67230D-4*T + 4.58980D-7*T**2)*1000.D0

       Cp   = CpNa*(1.0D0 - W) + CpK*W

 100 Format(' The temperature T=',F7.1,' degrees K is higher than '/
    a        ' the boiling point for NaK-',I3,//
    b        ' Execution terminated in Cp.'//)
       End
       Real*8 Function HeatFlux(z1)
*************************************************************
*      Uses linear interpolation between values in a table to *
*      return the value of the heat flux at the given axial   *
*      position z.  If the axial position is out of the range *
*      of the table, this routine reports the error and halts *
*      the program.                                           *
*************************************************************
       Integer Kmax
       Parameter (Kmax=10)
       Real*8 z1, QTable(Kmax), zh, zl, Qh, Ql, Tinlet, De, G
       Real*8 W, Z, D1, D2, Zmin, Zmax, mdot                             KHEL 4/26/93
```

```
      Integer K, K2
      Common /CoolProp/  Tinlet, De, G, W, D2, D1, mdot           KHEL 4/26/93
      Common /QTAB/ Qtable
      Common /Zdata/ Zmin, Zmax, K2

      K = 1

   10 if (z1 .ge. zmax) then                                          KHEL 5/22/93
          heatflux = qtable(kmax)                                     KHEL 5/22/93
          return                                                      KHEL 5/22/93
      end if                                                          KHEL 5/22/93
      If (Z(K).EQ.z1) then
          HeatFlux = QTable(K)
          Return
        ElseIf (Z(K).GT.z1) then
          zh = Z(K)
          zl = Z(K-1)
          Qh = QTable(K)
          Ql = QTable(K-1)
          HeatFlux = (Qh-Ql)/(zh-zl)*(z1-zl) + Ql
          Return
        Else
          K = K+1
        EndIf
      Goto 10
      End
C     ALGORITHM 433 COLLECTED ALGORITHMS FROM ACM.
C     ALGORITHM APPEARED IN COMM. ACM, VOL. 15, NO. 10,
C     P. 914.
      SUBROUTINE  INTRPL(IU,L,X,Y,N,U,V)
C INTERPOLATION OF A SINGLE-VALUED FUNCTION
C THIS SUBROUTINE INTERPOLATES, FROM VALUES OF THE FUNCTION
C GIVEN AS ORDINATES OF INPUT DATA POINTS IN AN X-Y PLANE
C AND FOR A GIVEN SET OF X VALUES (ABSCISSAS), THE VALUES OF
C A SINGLE-VALUED FUNCTION Y = Y(X).
C THE INPUT PARAMETERS ARE
C     IU = LOGICAL UNIT NUMBER OF STANDARD OUTPUT UNIT
C     L  = NUMBER OF INPUT DATA POINTS
C          (MUST BE 2 OR GREATER)
C     X  = ARRAY OF DIMENSION L STORING THE X VALUES
C          (ABSCISSAS) OF INPUT DATA POINTS
C          (IN ASCENDING ORDER)
C     Y  = ARRAY OF DIMENSION L STORING THE Y VALUES
C          (ORDINATES) OF INPUT DATA POINTS
C     N  = NUMBER OF POINTS AT WHICH INTERPOLATION OF THE
C          Y VALUE (ORDINATE) IS DESIRED
C          (MUST BE 1 OR GREATER)
C     U  = ARRAY OF DIMENSION N STORING THE X VALUES
C          (ABSCISSAS) OF DESIRED POINTS
C THE OUTPUT PARAMETER IS
C     V  = ARRAY OF DIMENSION N WHERE THE INTERPOLATED Y
C          VALUES (ORDINATES) ARE TO BE DISPLAYED
C DECLARATION STATEMENTS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                             KHEL 6/12/93
      DIMENSION    X(L),Y(L),U(N),V(N)
      EQUIVALENCE  (P0,X3),(Q0,Y3),(Q1,T3)
      REAL         M1,M2,M3,M4,M5
      EQUIVALENCE  (UK,DX),(IMN,X2,A1,M1),(IMX,X5,A5,M5),
     1             (J,SW,SA),(Y2,W2,W4,Q2),(Y5,W3,Q3)
C PRELIMINARY PROCESSING
   10 L0=L
      LM1=L0-1
      LM2=LM1-1
      LP1=L0+1
      N0=N
      IF(LM2.LT.0)          GO TO 90
      IF(N0.LE.0)           GO TO 91
      DO 11  I=2,L0
        IF(X(I-1)-X(I))     11,95,96
   11   CONTINUE
      IPV=0
C MAIN DO-LOOP
      DO 80  K=1,N0
        UK=U(K)
C ROUTINE TO LOCATE THE DESIRED POINT
   20   IF(LM2.EQ.0)        GO TO 27
        IF(UK.GE.X(L0))     GO TO 26
        IF(UK.LT.X(1))      GO TO 25
        IMN=2
        IMX=L0
   21   I=(IMN+IMX)/2
        IF(UK.GE.X(I))      GO TO 23
   22   IMX=I
        GO TO 24
   23   IMN=I+1
   24   IF(IMX.GT.IMN)      GO TO 21
```

```
      I=IMX
      GO TO 30
   25 I=1
      GO TO 30
   26 I=LP1
      GO TO 30
   27 I=2
C CHECK IF I = IPV
   30 IF(I.EQ.IPV)       GO TO 70
      IPV=I
C ROUTINES TO PICK UP NECESSARY X AND Y VALUES AND
C         TO ESTIMATE THEM IF NECESSARY
   40 J=I
      IF(J.EQ.1)         J=2
      IF(J.EQ.LP1)       J=L0
      X3=X(J-1)
      Y3=Y(J-1)
      X4=X(J)
      Y4=Y(J)
      A3=X4-X3
      M3=(Y4-Y3)/A3
      IF(LM2.EQ.0)       GO TO 43
      IF(J.EQ.2)         GO TO 41
      X2=X(J-2)
      Y2=Y(J-2)
      A2=X3-X2
      M2=(Y3-Y2)/A2
      IF(J.EQ.L0)        GO TO 42
   41 X5=X(J+1)
      Y5=Y(J+1)
      A4=X5-X4
      M4=(Y5-Y4)/A4
      IF(J.EQ.2)         M2=M3+M3-M4
      GO TO 45
   42 M4=M3+M3-M2
      GO TO 45
   43 M2=M3
      M4=M3
   45 IF(J.LE.3)         GO TO 46
      A1=X2-X(J-3)
      M1=(Y2-Y(J-3))/A1
      GO TO 47
   46 M1=M2+M2-M3
   47 IF(J.GE.LM1)       GO TO 48
      A5=X(J+2)-X5
      M5=(Y(J+2)-Y5)/A5
      GO TO 50
   48 M5=M4+M4-M3
C NUMERICAL DIFFERENTIATION
   50 IF(I.EQ.LP1)       GO TO 52
      W2=ABS(M4-M3)
      W3=ABS(M2-M1)
      SW=W2+W3
      IF(SW.NE.0.0)      GO TO 51
      W2=0.5
      W3=0.5
      SW=1.0
   51 T3=(W2*M2+W3*M3)/SW
      IF(I.EQ.1)         GO TO 54
   52 W3=ABS(M5-M4)
      W4=ABS(M3-M2)
      SW=W3+W4
      IF(SW.NE.0.0)      GO TO 53
      W3=0.5
      W4=0.5
      SW=1.0
   53 T4=(W3*M3+W4*M4)/SW
      IF(I.NE.LP1)       GO TO 60
      T3=T4
      SA=A2+A3
      T4=0.5*(M4+M5-A2*(A2-A3)*(M2-M3)/(SA*SA))
      X3=X4
      Y3=Y4
      A3=A2
      M3=M4
      GO TO 60
   54 T4=T3
      SA=A3+A4
      T3=0.5*(M1+M2-A4*(A3-A4)*(M3-M4)/(SA*SA))
      X3=X3-A4
      Y3=Y3-M2*A4
      A3=A4
      M3=M2
C DETERMINATION OF THE COEFFICIENTS
   60 Q2=(2.0*(M3-T3)+M3-T4)/A3
      Q3=(-M3-M3+T3+T4)/(A3*A3)
```

```
C COMPUTATION OF THE POLYNOMIAL
   70    DX=UK-P0
   80    V(K)=Q0+DX*(Q1+DX*(Q2+DX*Q3))
      RETURN
C ERROR EXIT
   90 WRITE (IU,2090)
      GO TO 99
   91 WRITE (IU,2091)
      GO TO 99
   95 WRITE (IU,2095)
      GO TO 97
   96 WRITE (IU,2096)
   97 WRITE (IU,2097)  I,X(I)
   99 WRITE (IU,2099)  L0,N0
      RETURN
C FORMAT STATEMENTS
 2090 FORMAT(1X/22H  ***    L = 1 OR LESS./)
 2091 FORMAT(1X/22H  ***    N = 0 OR LESS./)
 2095 FORMAT(1X/27H  ***    IDENTICAL X VALUES./)
 2096 FORMAT(1X/33H  ***    X VALUES OUT OF SEQUENCE./)
 2097 FORMAT(6H   I =,I7,10X,6HX(I) =,E12.3)
 2099 FORMAT(6H   L =,I7,10X,3HN =,I7/
     1          36H ERROR DETECTED IN ROUTINE    INTRPL)
      END
      SUBROUTINE  CRVFIT(IU,MD,L,X,Y,M,N,U,V)
C SMOOTH CURVE FITTING
C THIS SUBROUTINE FITS A SMOOTH CURVE TO A GIVEN SET OF IN-
C PUT DATA POINTS IN AN X-Y PLANE.  IT INTERPOLATES POINTS
C IN EACH INTERVAL BETWEEN A PAIR OF DATA POINTS AND GENER-
C ATES A SET OF OUTPUT POINTS CONSISTING OF THE INPUT DATA
C POINTS AND THE INTERPOLATED POINTS.  IT CAN PROCESS EITHER
C A SINGLE-VALUED FUNCTION OR A MULTIPLE-VALUED FUNCTION.
C THE INPUT PARAMETERS ARE
C     IU = LOGICAL UNIT NUMBER OF STANDARD OUTPUT UNIT
C     MD = MODE OF THE CURVE (MUST BE 1 OR 2)
C        = 1 FOR A SINGLE-VALUED FUNCTION
C        = 2 FOR A MULTIPLE-VALUED FUNCTION
C     L  = NUMBER OF INPUT DATA POINTS
C          (MUST BE 2 OR GREATER)
C     X  = ARRAY OF DIMENSION L STORING THE ABSCISSAS OF
C          INPUT DATA POINTS (IN ASCENDING OR DESCENDING
C          ORDER FOR MD = 1)
C     Y  = ARRAY OF DIMENSION L STORING THE ORDINATES OF
C          INPUT DATA POINTS
C     M  = NUMBER OF SUBINTERVALS BETWEEN EACH PAIR OF
C          INPUT DATA POINTS (MUST BE 2 OR GREATER)
C     N  = NUMBER OF OUTPUT POINTS
C        = (L-1)*M+1
C THE OUTPUT PARAMETERS ARE
C     U  = ARRAY OF DIMENSION N WHERE THE ABSCISSAS OF
C          OUTPUT POINTS ARE TO BE DISPLAYED
C     V  = ARRAY OF DIMENSION N WHERE THE ORDINATES OF
C          OUTPUT POINTS ARE TO BE DISPLAYED
C DECLARATION STATEMENTS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                          KHEL 6/12/93
      DIMENSION    X(L),Y(L),U(N),V(N)
      EQUIVALENCE  (M1,B1),(M2,B2),(M3,B3),(M4,B4),
     1             (X2,P0),(Y2,Q0),(T2,Q1)
      REAL         M1,M2,M3,M4
      EQUIVALENCE  (W2,Q2),(W3,Q3),(A1,P2),(B1,P3),
     1             (A2,DZ),(SW,R,Z)
C PRELIMINARY PROCESSING
   10 MD0=MD
      MDM1=MD0-1
      L0=L
      LM1=L0-1
      M0=M
      MM1=M0-1
      N0=N
      IF(MD0.LE.0)         GO TO 90
      IF(MD0.GE.3)         GO TO 90
      IF(LM1.LE.0)         GO TO 91
      IF(MM1.LE.0)         GO TO 92
      IF(N0.NE.LM1*M0+1)   GO TO 93
      GO TO (11,16), MD0
   11 I=2
      IF(X(1)-X(2))        12,95,14
   12 DO 13  I=3,L0
        IF(X(I-1)-X(I))    13,95,96
   13   CONTINUE
      GO TO 18
   14 DO 15  I=3,L0
        IF(X(I-1)-X(I))    96,95,15
   15   CONTINUE
      GO TO 18
   16 DO 17  I=2,L0
```

```
        IF(X(I-1).NE.X(I))  GO TO 17
        IF(Y(I-1).EQ.Y(I))  GO TO 97
   17   CONTINUE
   18 K=N0+M0
      I=L0+1
      DO 19  J=1,L0
        K=K-M0
        I=I-1
        U(K)=X(I)
   19   V(K)=Y(I)
      RM=M0
      RM=1.0/RM
C MAIN DO-LOOP
   20 K5=M0+1
      DO 80  I=1,L0
C ROUTINES TO PICK UP NECESSARY X AND Y VALUES AND
C          TO ESTIMATE THEM IF NECESSARY
        IF(I.GT.1)        GO TO 40
   30   X3=U(1)
        Y3=V(1)
        X4=U(M0+1)
        Y4=V(M0+1)
        A3=X4-X3
        B3=Y4-Y3
        IF(MDM1.EQ.0)     M3=B3/A3
        IF(L0.NE.2)       GO TO 41
        A4=A3
        B4=B3
   31   GO TO (33,32), MD0
   32   A2=A3+A3-A4
        A1=A2+A2-A3
   33   B2=B3+B3-B4
        B1=B2+B2-B3
        GO TO (51,56), MD0
   40   X2=X3
        Y2=Y3
        X3=X4
        Y3=Y4
        X4=X5
        Y4=Y5
        A1=A2
        B1=B2
        A2=A3
        B2=B3
        A3=A4
        B3=B4
        IF(I.GE.LM1)      GO TO 42
   41   K5=K5+M0
        X5=U(K5)
        Y5=V(K5)
        A4=X5-X4
        B4=Y5-Y4
        IF(MDM1.EQ.0)     M4=B4/A4
        GO TO 43
   42   IF(MDM1.NE.0)     A4=A3+A3-A2
        B4=B3+B3-B2
   43   IF(I.EQ.1)        GO TO 31
        GO TO (50,55), MD0
C NUMERICAL DIFFERENTIATION
   50   T2=T3
   51   W2=ABS(M4-M3)
        W3=ABS(M2-M1)
        SW=W2+W3
        IF(SW.NE.0.0)     GO TO 52
        W2=0.5
        W3=0.5
        SW=1.0
   52   T3=(W2*M2+W3*M3)/SW
        IF(I-1) 80,80,60
   55   COS2=COS3
        SIN2=SIN3
   56   W2=ABS(A3*B4-A4*B3)
        W3=ABS(A1*B2-A2*B1)
        IF(W2+W3.NE.0.0)  GO TO 57
        W2=SQRT(A3*A3+B3*B3)
        W3=SQRT(A2*A2+B2*B2)
   57   COS3=W2*A2+W3*A3
        SIN3=W2*B2+W3*B3
        R=COS3*COS3+SIN3*SIN3
        IF(R.EQ.0.0)      GO TO 58
        R=SQRT(R)
        COS3=COS3/R
        SIN3=SIN3/R
   58   IF(I-1) 80,80,65
C DETERMINATION OF THE COEFFICIENTS
   60   Q2=(2.0*(M2-T2)+M2-T3)/A2
```

```
      Q3=(-M2-M2+T2+T3)/(A2*A2)
      GO TO 70
 65   R=SQRT(A2*A2+B2*B2)
      P1=R*COS2
      P2=3.0*A2-R*(COS2+COS2+COS3)
      P3=A2-P1-P2
      Q1=R*SIN2
      Q2=3.0*B2-R*(SIN2+SIN2+SIN3)
      Q3=B2-Q1-Q2
      GO TO 75
C COMPUTATION OF THE POLYNOMIALS
 70   DZ=A2*RM
      Z=0.0
      DO 71  J=1,MM1
        K=K+1
        Z=Z+DZ
        U(K)=P0+Z
 71     V(K)=Q0+Z*(Q1+Z*(Q2+Z*Q3))
      GO TO 79
 75   Z=0.0
      DO 76  J=1,MM1
        K=K+1
        Z=Z+RM
        U(K)=P0+Z*(P1+Z*(P2+Z*P3))
 76     V(K)=Q0+Z*(Q1+Z*(Q2+Z*Q3))
 79   K=K+1
 80   CONTINUE
      RETURN
C ERROR EXIT
 90 WRITE (IU,2090)
    GO TO 99
 91 WRITE (IU,2091)
    GO TO 99
 92 WRITE (IU,2092)
    GO TO 99
 93 WRITE (IU,2093)
    GO TO 99
 95 WRITE (IU,2095)
    GO TO 98
 96 WRITE (IU,2096)
    GO TO 98
 97 WRITE (IU,2097)
 98 WRITE (IU,2098)  I,X(I),Y(I)
 99 WRITE (IU,2099)  MD0,L0,M0,N0
    RETURN
C FORMAT STATEMENTS
 2090 FORMAT(1X/31H  ***   MD OUT OF PROPER RANGE./)
 2091 FORMAT(1X/22H  ***   L = 1 OR LESS./)
 2092 FORMAT(1X/22H  ***   M = 1 OR LESS./)
 2093 FORMAT(1X/25H  ***   IMPROPER N VALUE./)
 2095 FORMAT(1X/27H  ***   IDENTICAL X VALUES./)
 2096 FORMAT(1X/33H  ***   X VALUES OUT OF SEQUENCE./)
 2097 FORMAT(1X/33H  ***   IDENTICAL X AND Y VALUES./)
 2098 FORMAT(7H   I  =,I4,10X,6HX(I) =,E12.3,
     1                 10X,6HY(I) =,E12.3)
 2099 FORMAT(7H   MD =,I4,8X,3HL =,I5,8X,
     1      3HM =,I5,8X,3HN =,I5/
     2      36H ERROR DETECTED IN ROUTINE    CRVFIT)
      END
      subroutine initial(prob,Tstop,Tecool,tr,sig,ems,
     &                   Qrad,Qcscond,t)
      parameter ( imax = 10, jmax = 10 )
      implicit double precision (a-h,o-z)
      integer prob, i, j, rmesh(9), mat(5)
      double precision tr, sig, ems, Tstop, Tecool
      double precision Qrad(jmax), t(imax,jmax)
      double precision Qcscond(jmax), rbound(10)
      double precision gapcond, r
      external gapcond, r
      common /rdata/ rbound,rmesh,mat
*********************************************************************
*                                                                   *
*  Calculate initial values of thermal power, conduction and radiation *
*                                                                   *
*********************************************************************

      do i=1,imax
         if ( r(i).eq.rbound(4) ) then
            do j=1,jmax
               qrad(j)    = sig*ems*((t(i,j))**4 - (t(i+1,j))**4)
               qcscond(j) = gapcond(t(i,j),t(i+1,j),Tstop,Tecool,Prob,
     a                              tr,r(i+1)-r(i))
            end do
         end if
      end do
```

```
      return
      end
      subroutine sgauss
      implicit double precision (a-h,o-z)
      integer imax, jmax, nn
      double precision zero
      parameter (imax = 10, jmax = 10, zero = 0.0d0)
      parameter (nn = 10*imax*jmax)
      double precision aa(imax*jmax+1,imax*jmax), x(imax*jmax)
      double precision a(nn), aflag(8), pivot(imax*jmax)
      integer i, j, z, n, snr(nn), rnr(nn), iflag(10)
      integer ha(imax*jmax,11), ifail, nn1, iha
      common /gaussmain/ aa,x,n

      z = 0
      do j=1,N
         do i=1,N
            if ( aa(i,j) .ne. zero ) then
               z = z + 1
               a(z) = aa(i,j)
               snr(z) = j
               rnr(z) = i
            end if
         end do
         x(j) = aa(n+1,j)
      end do
      nn1 = nn
      iha = n
*     print*,'nn=',nn,' z=',z

      call y12maf(n, z, a, snr, nn, rnr, nn1, pivot, ha,
     1iha,aflag,iflag,x,ifail)

      if ( ifail .ne. 0 ) then
         write(8,10) ifail
10       format(1x,' Error in sparse gaussian elimination code',/,
     &             ' error diagnostic parameter ifail = ', i4 )
         stop
      end if

      return
      end
      subroutine y12maf(n, z, a, snr, nn, rnr, nn1, pivot, ha,
     1iha,aflag,iflag,b,ifail)
      implicit double precision (a-b,g,p,t-y), integer (c,f,h-n,r-s,z)
      double precision a(nn), pivot(n), aflag(8),b(n)
      integer snr(nn), rnr(nn1), ha(iha,11), iflag(10)
      aflag(1)=16.0d0
      aflag(2)=1.d-12
      aflag(3)=1.d+16
      aflag(4)=1.d-12
      iflag(2)=2
      iflag(3)=1
      iflag(4)=0
      iflag(5)=1
      call y12mbf(n,z,a,snr,nn,rnr,nn1,ha,iha,aflag,iflag,ifail)
      if(ifail.ne.0)go to 1
      call y12mcf(n,z,a,snr,nn,rnr,nn1,pivot,b,ha,iha,aflag,iflag,
     1 ifail)
      if(ifail.ne.0)go to 1
      call y12mdf(n,a,nn,b,pivot,snr,ha,iha,iflag,ifail)
1     return
      end
      subroutine y12mbf(n, z, a, snr, nn, rnr, nn1, ha, iha, aflag,
     1 iflag,ifail)
c
c
c  the non-zero elements of a sparse matrix a are prepared  in order to
c  solve the system ax=b by use of sparse matrix technique/
c
c
      implicit double precision(a-b,g,p,t-y),integer(c,f,h-n,r-s,z)
      double precision a(nn), aflag(8)
      integer snr(nn), rnr(nn1), ha(iha,11), iflag(10)
      mode=iflag(4)
      ifail=0
      if(n.lt.2)ifail=12
      if(z.le.0)ifail=13
      if(nn.lt.2*z)ifail=5
      if(nn1.lt.z)ifail=6
      if(ifail.eq.0.and.n.gt.z)ifail=14
      if(iha.lt.n)ifail=15
      if(mode.lt.0)ifail=16
      if(mode.gt.2)ifail=16
      if(ifail.ne.0) go to 22
      gt1=0.0d0
```

```
      do 10 i=1,n
      ha(i,2)=0
      ha(i,3)=0
   10 ha(i,6)=0
c
c  find the number of the non-zero elements in each row and column;move
c  the non-zero elements in the end of the arrays a and snr;find the
c  largest non-zero element in a(in absolute value).
c
      do 20 i=1,z
      t=dabs(a(i))
      l3=rnr(i)
      l4=snr(i)
      if(l4.gt.n.or.l4.lt.1)ifail=24
      if(l3.gt.n.or.l3.lt.1)ifail=25
      ha(l3,3)=ha(l3,3)+1
      ha(l4,6)=ha(l4,6)+1
      if(t.gt.gt1)gt1=t
      a(z+i)=a(i)
   20 snr(z+i)=snr(i)
      if(ifail.gt.0)go to 22
c
c  store the information of the row starts(in ha(i,1))and of the column
c  starts(in ha(i,4)).
c
      l1=1
      l2=1
      do 40 i=1,n
      l3=ha(i,3)
      l4=ha(i,6)
      if(l3.gt.0)go to 21
      ifail=17
      go to 22
   21 if(l4.gt.0)go to 23
      ifail=18
      go to 22
   23 if(mode.eq.2)go to 30
      ha(i,9)=l3
      ha(i,10)=l4
      ha(i,11)=0
      ha(l3,2)=ha(l3,2)+1
      ha(i,5)=l3
   30 ha(i,1)=l1
      ha(i,4)=l2
      l1=l1+l3
      l2=l2+l4
      ha(i,3)=0
   40 ha(i,6)=0
c
c  store the non-zero elements of matrix a(ordered in rows) in the
c  first z locations of the array a.do the same for their column numbers
c
      do 50 i=1,z
      l1=z+i
      l3=rnr(i)
      l2=ha(l3,1)+ha(l3,3)
      a(l2)=a(l1)
      snr(l2)=snr(l1)
   50 ha(l3,3)=ha(l3,3)+1
c
c  store the row numbers of the non-zero elements ordered by columns in
c  the first z locations of the array rnr. store information about row
c  ends(in ha(i,3)).
c
      l4=1
      do 70 i=1,n
      if(mode.eq.2)go to 60
      if(ha(i,2).eq.0)go to 60
      ha(i,11)=l4
      l4=l4+ha(i,2)
      ha(i,2)=ha(i,11)
   60 ha(i,3)=ha(i,1)+ha(i,3)-1
      l1=ha(i,1)
      l2=ha(i,3)
      do 70 j=l1,l2
      l3=snr(j)
      r=ha(l3,6)
      index=ha(l3,4)+r
      rnr(index)=i
      if(r.eq.0)go to 70
      if(j.eq.l1)go to 70
      if(rnr(index-1).ne.i)go to 70
      ifail=11
      go to 22
   70 ha(l3,6)=r+1
      do 90 i=1,n
```

```
      if(mode.eq.2)go to 80
      l3=ha(i,5)
      l5=ha(l3,2)
      ha(l5,8)=i
      ha(i,7)=l5
      ha(l3,2)=ha(l3,2)+1
   80 continue
   90 ha(i,6)=ha(i,4)+ha(i,6)-1
      aflag(6)=gt1
      iflag(6)=0
      iflag(7)=0
      iflag(8)=z
      iflag(1)=-1
22    return
      end
      subroutine y12mcf(n,z,a,snr,nn,rnr,nn1,pivot,b,ha,iha, aflag,iflag
     *,ifail)
c
c  systens of linear equations are solved by use of sparse matrix tech-
c  nique and by gaussian elimination.
c
      implicit double precision(a-b,g,p,t-y),integer(c,f,h-n,r-s,z)
      double precision a(nn),b(n),pivot(n),aflag(8)
c
c  information which is necessary to begin the elimination is stored.
c
      integer snr(nn),rnr(nn1),ha(iha,11), iflag(10)
      ifail=0
      if(iflag(1).ne.-1)ifail=2
      if(aflag(1).lt.1.0d0)aflag(1)=1.0005 d0
      if(aflag(3).lt.1.0d+5)aflag(3)=1.0d+5
      if(aflag(4).lt.0.0d0)aflag(4)=-aflag(4)
      if(iflag(2).lt.1)ifail=19
      if(iflag(3).lt.0.or.iflag(3).gt.2)ifail=20
      if(iflag(5).lt.1.or.iflag(5).gt.3)ifail=21
      if(iflag(5).eq.3)ifail=22
      if(ifail.gt.0)go to 1110
      snr(z+1)=0
      rnr(z+1)=0
      n8=n+1
      n7=n-1
      u=aflag(1)
      grmin=aflag(4)*aflag(6)
c
c  use the information about fill-ins if it is possible.
c
      zz=z
      nr=n*n
      if(iflag(4).ne.2)go to 100
      if(iflag(10).gt.nn)go to 50
      l1=iflag(10)
      l5=l1+1
      if(l5.le.nn)snr(l5)=0
      do 40 i=1,n
      l=n8-i
      l2=ha(l,3)+1
      l3=l2-ha(l,1)
      do 10 j=1,l3
      snr(l5-j)=snr(l2-j)
   10 a(l5-j)=a(l2-j)
      ha(l,3)=l1
      ha(l,1)=l5-l3
      l6=l1-l3
      l5=l5-ha(l,9)
      if(l5.gt.l6)go to 30
      do 20 j=l5,l6
   20 snr(j)=0
   30 continue
   40 l1=l5-1
   50 if(iflag(9).gt.nn1)go to 100
      l2=iflag(9)
      l5=l2+1
      if(l5.le.nn1)rnr(l5)=0
      do 90 i=1,n
      l=n8-i
      l1=ha(l,6)+1
      l4=l1-ha(l,4)
      do 60 j=1,l4
   60 rnr(l5-j)=rnr(l1-j)
      ha(l,4)=l5-l4
      ha(l,6)=l2
      l6=l2-l4
      l5=l5-ha(l,10)
      if(l5.gt.l6)go to 80
      do 70 j=l5,l6
   70 rnr(j)=0
```

```
   80 continue
   90 l2=l5-1
  100 r4=ha(n,3)
      r5=ha(n,6)
      aflag(7)=aflag(6)
      aflag(8)=aflag(6)
      do 110 i=1,n
      pivot(i)=0.0 d0
      ha(i,2)=ha(i,1)
  110 ha(i,5)=ha(i,4)
      index=ha(n,8)
c
c  start of gaussian elimination.
c
      slut=ha(index,3)-ha(index,2)+1
      do 950 i=1,n7
      rr3=ha(i,2)
      rr4=ha(i,3)
      c1=ha(i,4)
      cr4=ha(i,6)
      if(iflag(3).eq.0)go to 350
      if(iflag(4).ne.2)go to 120
      rrow=ha(i,7)
      rcoll=ha(i,8)
      go to 220
  120 l4=ha(i,8)
      if(iflag(3).eq.1)go to 130
      rrow=l4
      rcoll=rrow
      rpivot=i
      go to 170
  130 r=nr
      v=0.0 d0
      index=iflag(2)
      do 160 kk=1,index
      l1=i-1+kk
      if(l1.gt.n)go to 170
      j=ha(l1,8)
      r7=ha(j,2)
      r8=ha(j,3)
      r9=r8-r7
      t=0.0 d0
      do 140 k=r7,r8
      td=dabs(a(k))
  140 if(t.lt.td)t=td
      t=t/u
      do 160 k=r7,r8
      td=dabs(a(k))
      if(td.lt.t)go to 150
      r6=snr(k)
      r3=r9*(ha(r6,6)-ha(r6,5))
      if(r3.gt.r)go to 150
      if(r3.lt.r)go to 151
      if(v.ge.td)go to 150
  151 v=td
      rrow=j
      rcoll=r6
      r=r3
      rpivot=l1
  150 continue
  160 continue
  170 r3=ha(rcoll,10)
      ha(rcoll,10)=ha(i,10)
      ha(i,10)=r3
      r3=ha(rrow,9)
      ha(rrow,9)=ha(i,9)
c
c  remove the pivot row of the list where the rows are ordered by
c  increasing numbers of non-zero elements.
c
      ha(i,9)=r3
      l1=0
      l=i
      l2=ha(l4,3)-ha(l4,2)+1
  180 l=l+1
      if(l2.gt.l1)ha(l2,11)=l
      if(l.gt.n)go to 190
      l5=ha(l,8)
      l3=ha(l5,3)-ha(l5,2)+1
      if(rpivot.lt.l)go to 190
      ha(l4,7)=l
      ha(l,8)=l4
      l4=l5
      l1=l2
      l2=l3
      l3=n8
```

```
      go to 180
  190 if(l2.eq.l1)go to 200
      if(l3.eq.l2)go to 200
      ha(l2,l1)=0
  200 l5=ha(i,7)
      if(rrow.eq.i)go to 210
      ha(l5,8)=rrow
      ha(rrow,7)=l5
  210 ha(i,7)=rrow
c
c  row interchanges.
c
      ha(i,8)=rcoll
  220 if(rrow.eq.i)go to 290
      t=b(rrow)
      b(rrow)=b(i)
      b(i)=t
      do 250 j=rr3,rr4
      l1=snr(j)
      r=ha(l1,5)-1
      r10=ha(l1,6)
  240 r=r+1
      if(rnr(r).ne.i)go to 240
      rnr(r)=rnr(r10)
  250 rnr(r10)=rrow
      rr3=ha(rrow,2)
      rr4=ha(rrow,3)
      do 270 j=rr3,rr4
      l1=snr(j)
      r=ha(l1,5)-1
  260 r=r+1
      if(rnr(r).ne.rrow)go to 260
  270 rnr(r)=i
      do 280 j=1,3
      r3=ha(rrow,j)
      ha(rrow,j)=ha(i,j)
c
c  column interchanges.
c
  280 ha(i,j)=r3
  290 if(rcoll.eq.i)go to 350
      do 310 j=c1,cr4
      l1=rnr(j)
      r=ha(l1,2)-1
      r10=ha(l1,3)
  300 r=r+1
      if(snr(r).ne.i)go to 300
      t=a(r10)
      a(r10)=a(r)
      a(r)=t
      snr(r)=snr(r10)
  310 snr(r10)=rcoll
      c1=ha(rcoll,4)
      cr4=ha(rcoll,6)
      do 330 j=c1,cr4
      l1=rnr(j)
      r=ha(l1,2)-1
  320 r=r+1
      if(snr(r).ne.rcoll)go to 320
  330 snr(r)=i
      do 340 j=4,6
      r3=ha(rcoll,j)
      ha(rcoll,j)=ha(i,j)
c
c end of the interchanges.
c the row ordered list and the column ordered list are prepared to
c begin step i of the elimination.
c
  340 ha(i,j)=r3
  350 r9=rr4-rr3
      do 360 rr=rr3,rr4
      if(snr(rr).eq.i)go to 370
  360 continue
      ifail=9
      go to 1110
  370 v=a(rr)
      pivot(i)=v
      td=dabs(v)
      if(td.lt.aflag(8))aflag(8)=td
      if(td.ge.grmin)go to 380
      ifail=3
      go to 1110
  380 r2=ha(i,1)
      a(rr)=a(rr3)
      snr(rr)=snr(rr3)
      a(rr3)=a(r2)
```

```
      snr(rr3)=snr(r2)
      snr(r2)=0
      z=z-1
      rr3=rr3+1
      ha(i,2)=rr3
      ha(i,1)=r2+1
      cr3=ha(i,5)
      if(r9.le.0)go to 431
      do 430 j=rr3,rr4
      index=snr(j)
  430 pivot(index)=a(j)
  431 r7=cr4-cr3+1
      do 880 k=1,r7
      r1=rnr(cr3-1+k)
      if(r1.eq.i)go to 870
      i1=ha(r1,1)
      rr1=ha(r1,2)
      rr2=ha(r1,3)
      l2=rr2-rr1+1
      l=rr1-1
  390 l=l+1
      if(snr(l).ne.i)go to 390
      t=a(l)/v
      if(iflag(5).eq.2)go to 400
      a(l)=a(i1)
      snr(l)=snr(i1)
      snr(i1)=0
      i1=i1+1
      ha(r1,1)=i1
      z=z-1
      go to 410
  400 a(l)=a(rr1)
      a(rr1)=t
      r3=snr(rr1)
      snr(rr1)=snr(l)
      snr(l)=r3
  410 rr1=rr1+1
      ha(r1,2)=rr1
      b(r1)=b(r1)-b(i)*t
      if(r9.le.0)go to 669
      r=rr1
      if(r.gt.rr2)go to 470
      do 460 l=r,rr2
      l1=snr(l)
      td=pivot(l1)
      if(td.eq.0.0d0)go to 450
      pivot(l1)=0.0 d0
      td=a(l)-td*t
      a(l)=td
      td1=dabs(td)
      if(td1.gt.aflag(7))aflag(7)=td1
c
c  too small element is created.remove it from the lists.
c
      if(td1.gt.aflag(2))go to 450
      z=z-1
      a(l)=a(rr1)
      snr(l)=snr(rr1)
      a(rr1)=a(i1)
      snr(rr1)=snr(i1)
      snr(i1)=0
      rr1=rr1+1
      i1=i1+1
      ha(r1,2)=rr1
      ha(r1,1)=i1
      r3=ha(l1,5)
      r2=r3-1
      l4=ha(l1,4)
      l5=rnr(l4)
      l6=rnr(r3)
  440 r2=r2+1
      if(rnr(r2).ne.r1)go to 440
      rnr(r2)=l6
      rnr(r3)=l5
      rnr(l4)=0
      ha(l1,5)=r3+1
      ha(l1,4)=l4+1
  450 continue
  460 continue
  470 continue
      do 750 j=1,r9
      r=rr3-1+j
      r2=snr(r)
      tol2=pivot(r2)
      pivot(r2)=a(r)
      if(tol2.eq.0.0d0)go to 740
```

```
      tol3=-tol2*t
      tol1=dabs(tol3)
      if(tol1.lt.aflag(2))go to 740
      c2=ha(r2,4)
      cr2=ha(r2,6)
      cr1=ha(r2,5)
      lfr=rr2-i1+2
      lfc=cr2-c2+2
      if(iflag(4).ne.1)go to 480
      if(lfr.gt.ha(r1,9))ha(r1,9)=lfr
      if(lfc.gt.ha(r2,10))ha(r2,10)=lfc
  480 if(i1.eq.1)go to 490
      if(snr(i1-1).eq.0)go to 600
  490 if(rr2.eq.nn)go to 500
      if(snr(rr2+1).eq.0)go to 580
c
c  collection in row ordered list.
c
  500 r10=nn-lfr
      if(r10.ge.r4)go to 560
      iflag(6)=iflag(6)+1
      do 520 jj=1,n
      l1=ha(jj,3)
      if(l1.lt.ha(jj,1))go to 510
      ha(jj,3)=snr(l1)
      snr(l1)=-jj
  510 continue
  520 continue
      l3=0
      l4=1
      do 550 jj=1,r4
      if(snr(jj).eq.0)go to 540
      l3=l3+1
      if(snr(jj).gt.0)go to 530
      l5=-snr(jj)
      snr(jj)=ha(l5,3)
      ha(l5,3)=l3
      l6=l4+ha(l5,2)-ha(l5,1)
      ha(l5,2)=l6
      ha(l5,1)=l4
      l4=l3+1
  530 a(l3)=a(jj)
      snr(l3)=snr(jj)
  540 continue
  550 continue
      r4=l3
      snr(l3+1)=0
      rr3=ha(i,2)
      rr4=ha(i,3)
      i1=ha(r1,1)
      rr1=ha(r1,2)
      r=rr3-1+j
      if(r10.ge.r4)go to 560
      ifail=5
c
c fill-in takes place in the row ordered list.
c
      go to 1110
  560 r8=lfr-1
      rr2=r4+lfr
      if(r8.le.0)go to 579
      l3=i1-1
      do 570 l1=1,r8
      l4=r4+l1
      l5=l3+l1
      a(l4)=a(l5)
      snr(l4)=snr(l5)
  570 snr(l5)=0
  579 rr1=r4+rr1-i1+1
      ha(r1,3)=rr2
      ha(r1,2)=rr1
      i1=r4+1
      ha(r1,1)=i1
      l1=rr2
      go to 590
  580 rr2=rr2+1
      ha(r1,3)=rr2
      l1=rr2
      if(rr2.le.r4)go to 610
  590 r4=rr2
      if(r4.lt.nn)snr(r4+1)=0
      go to 610
  600 rr1=rr1-1
      i1=i1-1
      ha(r1,1)=i1
      ha(r1,2)=rr1
```

```
      l1=rr1
      snr(i1)=snr(l1)
      a(i1)=a(l1)
  610 a(l1)=tol3
      snr(l1)=snr(r)
      td=dabs(a(l1))
      if(td.gt.aflag(7))aflag(7)=td
      z=z+1
      if(iflag(8).lt.z) iflag(8)=z
      if(c2.eq.1)go to 620
      if(rnr(c2-1).eq.0)go to 720
  620 if(cr2.eq.nn1)go to 630
      if(rnr(cr2+1).eq.0)go to 700
c
c  collection in column ordered list.
c
  630 r10=nn1-lfc
      if(r10.ge.r5)go to 680
      iflag(7)=iflag(7)+1
      do 640 jj=i,n
      l1=ha(jj,6)
      ha(jj,6)=rnr(l1)
  640 rnr(l1)=-jj
      l3=0
      l4=1
      do 670 jj=1,r5
      if(rnr(jj).eq.0)go to 660
      l3=l3+1
      if(rnr(jj).gt.0)go to 650
      l5=-rnr(jj)
      rnr(jj)=ha(l5,6)
      ha(l5,6)=l3
      l6=l4+ha(l5,5)-ha(l5,4)
      ha(l5,5)=l6
      ha(l5,4)=l4
      l4=l3+1
  650 rnr(l3)=rnr(jj)
  660 continue
  670 continue
      r5=l3
      rnr(r5+1)=0
      c2=ha(r2,4)
      cr3=ha(i,5)
      cr4=ha(i,6)
      cr1=ha(r2,5)
      if(r10.ge.r5)go to 680
      ifail=6
c
c fill-in takes place in the column ordered list.
c
      go to 1110
  680 r8=lfc-1
      cr2=r5+lfc
      if(r8.le.0)go to 699
      l3=c2-1
      do 690 l=1,r8
      l4=r5+l
      l5=l3+l
      rnr(l4)=rnr(l5)
  690 rnr(l5)=0
  699 cr1=r5+cr1-c2+1
      c2=r5+1
      ha(r2,6)=cr2
      ha(r2,4)=c2
      ha(r2,5)=cr1
      r=cr2
      go to 710
  700 cr2=cr2+1
      ha(r2,6)=cr2
      r=cr2
      if(cr2.le.r5)go to 730
  710 r5=cr2
      if(r5.lt.nn1)rnr(r5+1)=0
      go to 730
  720 cr1=cr1-1
      c2=c2-1
      ha(r2,4)=c2
      ha(r2,5)=cr1
      r=cr1
      rnr(c2)=rnr(r)
  730 rnr(r)=r1
  740 continue
  750 continue
  669 if(rr1.le.rr2)go to 760
      ifail=7
c
```

```
c  update the information in the list where the rows are ordered by
c  increasing numbers of the non-zero elements.
c
      go to 1110
  760 if(iflag(4).eq.2)go to 870
      if(iflag(3).eq.0)go to 870
      l1=rr2-rr1+1
      if(l1.eq.l2)go to 870
      l6=ha(r1,7)
      l4=ha(l2,11)
      if(l1.gt.l2)go to 820
      if(l6.gt.l4)go to 780
      if(l4.eq.n)go to 770
      l=ha(l4+1,8)
      l5=ha(l,3)-ha(l,2)+1
      if(l5.eq.l2)go to 790
  770 ha(l2,11)=0
      go to 800
  780 l5=ha(l4,8)
      l3=ha(l6,8)
      ha(l4,8)=l3
      ha(l6,8)=l5
      ha(l5,7)=l6
      ha(l3,7)=l4
      l6=l4
  790 ha(l2,11)=l4+1
  800 if(l4.eq.i+1)go to 810
      l=ha(l6-1,8)
      l2=ha(l,3)-ha(l,2)+1
      l4=ha(l2,11)
      if(l1.lt.l2)go to 780
  810 if(l1.ne.l2)ha(l1,11)=l6
      go to 870
  820 if(l6.gt.l4)go to 840
      if(l4.eq.n)go to 830
      l=ha(l4+1,8)
      l5=ha(l,3)-ha(l,2)+1
      if(l5.eq.l2)go to 840
  830 ha(l2,11)=0
  840 l2=l2+1
      if(l2.le.slut)go to 850
      l3=n
      slut=l1
      l2=l1
      go to 860
  850 l3=ha(l2,11)-1
      if(l3.eq.-1)go to 840
      if(l2.gt.l1)l2=l1
  860 ha(l2,11)=l3
      l4=ha(l3,8)
      l7=ha(l6,8)
      ha(l3,8)=l7
      ha(l6,8)=l4
      ha(l7,7)=l3
      ha(l4,7)=l6
      l6=l3
      if(l2.lt.l1)go to 840
  870 continue
  880 continue
      if(r9.le.0)go to 882
      do 881 j=rr3,rr4
      index=snr(j)
  881 pivot(index)=0.0 d0
  882 continue
      cr3=ha(i,4)
      do 890 j=cr3,cr4
  890 rnr(j)=0
      if(r9.le.0)go to 930
      l2=ha(i,2)-1
      do 920 l1=1,r9
      r=snr(l2+l1)
      r1=ha(r,5)
      r2=ha(r,6)
      if(r2.gt.r1)go to 900
      ifail=8
      go to 1110
  900 ha(r,5)=r1+1
      r3=r1-1
  910 r3=r3+1
      if(rnr(r3).ne.i)go to 910
      rnr(r3)=rnr(r1)
  920 rnr(r1)=i
  930 aflag(5)=aflag(7)/aflag(6)
      if(aflag(5).lt.aflag(3))go to 940
      ifail=4
      go to 1110
```

```
  940 continue
c
c  preparation to begin the back substitution.
c
  950 continue
      index=ha(n,2)
      pivot(n)=a(index)
      a(index)=0.0 d0
      td=dabs(pivot(n))
      if(td.gt.aflag(7))aflag(7)=td
      if(td.lt.aflag(8))aflag(8)=td
      if(td.gt.grmin)go to 960
      ifail=3
      go to 1110
  960 if(iflag(4).ne.1)go to 1060
      iflag(10)=ha(n,9)
      iflag(9)=ha(n,10)
      do 990 i=1,n7
      r1=n-i
      iflag(10)=iflag(10)+ha(r1,9)
      iflag(9)=iflag(9)+ha(r1,10)
      if(iflag(3).eq.0)go to 980
      do 970 j=9,10
      r2=ha(r1,j-2)
      r6=ha(r2,j)
      ha(r2,j)=ha(r1,j)
  970 ha(r1,j)=r6
  980 continue
  990 continue
 1060 continue
      aflag(5)=aflag(7)/aflag(6)
      iflag(1)=-2
 1110 z=zz
      return
      end
      subroutine y12mdf(n,a,nn,b,pivot,snr,ha,iha,iflag,ifail)
      implicit double precision(a-b,g,p,t-y),integer (c,f,h-n,r-s,z)
      double precision a(nn), pivot(n), b(n)
      integer snr(nn), ha(iha,11), iflag(10)
      ifail=0
      if(iflag(1).eq.-2)go to 1000
      ifail=1
      go to 1110
 1000 mode=iflag(4)
      ipiv=iflag(3)
      n8=n+1
      n7=n-1
      state=iflag(5)
c
c  solve the system with lower triangular matrix  l  (if the
c  lu-factorization is available).
c
      if(state.ne.3)go to 1051
      if(ipiv.eq.0)go to 1020
      do 1010 i=1,n7
      l1=ha(i,7)
      t=b(l1)
      b(l1)=b(i)
      b(i)=t
 1010 continue
 1020 continue
      do 1050 i=1,n
      rr1=ha(i,1)
      rr2=ha(i,2)-1
      if(rr1.gt.rr2)go to 1040
      do 1030 j=rr1,rr2
      l1=snr(j)
 1030 b(i)=b(i)-a(j)*b(l1)
 1040 continue
 1050 continue
c
c  solve the system with upper triagular matrix.
c
 1051 continue
      do 1090 i=1,n
      r1=n8-i
      rr1=ha(r1,2)
      rr2=ha(r1,3)
      if(rr2.lt.rr1)   go to 1080
      do 1070 j=rr1,rr2
      r2=snr(j)
 1070 b(r1)=b(r1)-a(j)*b(r2)
 1080 continue
 1090 b(r1)=b(r1)/pivot(r1)
c
c if interchanges were used during the  elimination then a reordering in
```

```
c lution vector is made.
c
      if(ipiv.eq.0)go to 1110
      do 1100 i=1,n7
      r1=n-i
      r2=ha(r1,8)
      t=b(r2)
      b(r2)=b(r1)
 1100 b(r1)=t
 1110 return
      end
```