1990

Thermal-hydraulic transient analysis of a packed particle bed reactor fuel element

Casey, William Emerson

THERMAL-HYDRAULIC TRANSIENT ANALYSIS OF A PACKED PARTICLE BED REACTOR FUEL ELEMENT

by
WILLIAM EMERSON CASEY
B.S., Engineering Physics
U.S. Naval Academy, Annapolis, MD
(1976)

SUBMITTED TO THE DEPARTMENT OF OCEAN ENGINEERING AND THE DEPARTMENT OF NUCLEAR ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES OF

NAVAL ENGINEER

and

MASTER OF SCIENCE IN NUCLEAR ENGINEERING

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May, 1990

THERMAL-HYDRAULIC TRANSIENT ANALYSIS OF A PACKED PARTICLE BED REACTOR FUEL ELEMENT

by

WILLIAM EMERSON CASEY

Submitted to the Department of Ocean Engineering and the
Department of Nuclear Engineering on May 11, 1990, in
partial fulfillment of the requirements for the degrees of
Naval Engineer and Master of Science in Nuclear Engineering.

ABSTRACT

A model which describes the thermal-hydraulic behavior of a packed particle bed reactor fuel element is developed and compared to a reference standard (Tuddenham, 1989). The model represents a step toward a thermal-hydraulic module for a real-time, autonomous reactor power controller.

The general configuration of the fuel element is similar in construction to a design studied by Brookhaven National Laboratory and Sandia National Laboratory. A bed of small (diameter ≈ 500 μm) fuel particles are packed between concentrically mounted retention cylinders referred to as frits. The element is cooled by parahydrogen which flows axially through the inlet and outlet plenums and radially inward through the fuel particle bed.

The momentum integral approach used in the MINET code (Van Tuyle, et Al, 1984) is applied to this model to balance the fundamental mass, energy and momentum conservation relationships. The element is divided into only three control volumes: the inlet plenum and cold frit define the first control volume, the fuel particle bed defines a second control volume, and the outlet plenum and hot frit define the third control volume. The solid phase of the particle bed is represented by a single node.

This simple model was validated against the reference standard and compared favorably. As a demonstration of the model's flexibility, a number of variations were analyzed. These included variations in fuel element geometry and the initial and final values of inlet temperature, inlet pressure, and outlet pressure. As a final demonstration, a cluster of nineteen, 1 meter long fuel elements, arranged to form a core, were analyzed for an up-power transient from 0 MWt to approximately 18 MWt.

The simple model significantly decreases the time necessary to perform a single analysis. A transient of 10 s with a timestep of 10 ms, for example, takes approximately 45 s of computation on a desktop computer equipped with an 80386 microprocessor.

Thesis Supervisor: J. E. Meyer
Title: Professor of Nuclear Engineering

ACKNOWLEDGEMENTS

I would like to thank and express my appreciation for the support of all those who helped bring this thesis to fruition. A special thanks goes to my thesis advisor, Professor J. E. Meyer, whose expert supervision and promethean patience provided a never ending source of assistance, motivation and inspiration. Another special thanks is extended to Cdr Rich Celotto whose limitless assistance was invaluable as well. Finally, and most importantly, I would like to thank my wife, Deidre, who helped me maintain perspective and our cat, Charlie, who kept me company during those long nights in front of the keyboard.

Table of Contents

Table of Figures

Table of Tables

CHAPTER 1

INTRODUCTION

1.1 STATEMENT OF PROBLEM

A packed particle bed reactor (PBR) is a gas-cooled reactor similar in nature to a high temperature gas-cooled reactor (HTGR). Unlike the HTGR, however, the PBR packs small fuel particles between inner and outer retention elements, designated as frits. The PBR is appropriate for a number of gas-cooled reactor applications and, in particular, seems to be most appropriate for use in space because of its compactness, high outlet temperature, and wide range of delivered power.

The PBR is proposed for use as a power source for a variety of systems in space. The requirements of these systems range from relatively low power levels, on the order of kilowatts, to multi-megawatt applications (A-1). These systems include nuclear propulsion rockets as well as systems which may require large, rapid power transients. The proposed PBR's will initially be unmanned and will therefore require an integrated autonomous control system.

The Advanced Controls Group at MIT has worked with the Sandia National Laboratory to develop a control system for a PBR which is to operate in space. The system which is currently being considered uses a control algorithm based upon a real-time model which must accurately represent the reactor so that proper control signals are generated. As a step toward the development of the real-time model, a thermal-hydraulic model was formulated by Tuddenham (T-1) to calculate the behavior of a PBR fuel element during transient operations. This model was developed to provide detailed information of fuel element behavior and to serve as the standard for future models. The

objective of this investigation is to continue the development of the model-based algorithm by evolving the thermal-hydraulic model of a PBR fuel element and begin coupling the element with other system components.

The automatic reactor controller will rely on the thermal-hydraulic model for performance predictions based on temperature and pressure inputs from the system. This, in turn, will allow the controller to calculate projected fuel temperatures and allow operation without exceeding safety limits within the fuel. Further, the model provides information on the temperature and pressure of the hydrogen coolant which is used to obtain the amount of reactivity feedback from changes in coolant conditions during a transient.

1.2 APPROACH

The major emphasis of this investigation is the simplification of the thermal-hydraulic reference model provided by Tuddenham (T-1). Although his model produces a very detailed analysis, exceedingly long computational times are required to reach a solution. The simplified model allows for a method of analysis which is sufficient in detail to give an accurately calculation of reactivity feedback within the fuel and yet is sufficiently simple so as not to require excessive computation.

Once the less complicated model is developed, it must be validated by comparing the model results with the reference standard. The reference standard provides calculated state variables for some steady state and transient conditions.

In addition to building a simplified model of the fuel element behavior, preliminary steps are taken to couple the fuel element with other fuel elements within the reactor as

well as with other core and power plant components. Future developments can then proceed with exact modeling of a complete system once specifics become available for each separate component.

1.3 CONCERNS

The major concerns involved with the simplification of the reference standard are those which deal with the effects of reducing the number of control volumes within the model and increasing the size of the time increment. The number of control volumes is decreased from approximately fifty to only three. The size of the time step depends on the numerical method used and must be sized to provide stability in the formulation. An implicit method is used in the simple model and allows the time step to be increased from .05 ms to 100 ms without loss of stability. Unfortunately, spatial resolution is decreased with an increase in the size of the control volumes and, if spatial resolution is important, the accuracy of the model will suffer.

Specific control volume concerns include:

a. Coolant Flow: Can coolant flow through the inlet plenum, particle bed, outlet plenum and the retention elements be simplified into a one dimensional flow compared to a two dimensional flow represented in the reference standard?

b. Flow Distribution: Can simplifying assumptions be made to use a uniform flow distribution across the frits and particle bed?

c. Sources of Pressure Loss: Can the sources of pressure loss in many small control volumes be combined in calculations for larger control volumes?

d. Heat Transfer: Can the heat transfer prediction of the reference standard be well represented by using less control volumes?

Specific numerical concerns include whether the stability advantages of an implicit model are sufficient to counterbalance the longer computational times required.

The following chapters describe the development and proposed application of the PBR and then describe the development of a less complex model to be used to analyze thermal-hydraulic transient response of a PBR fuel element.

CHAPTER 2

BACKGROUND

2.1 GENERAL

The concept of operating a nuclear reactor system in space is not new. The requirement to supply a reliable, sustained source of power to satellites and deep space probes was identified early in the US space program. Two programs were started to investigate the possible applications of nuclear reactors in space. The NERVA (Nuclear Engine for Rocket Vehicle Application) program was started in the early 1960's and was primarily concerned with the development of nuclear technology for use as a means of propulsion in space. The SNAP (Space Nuclear Applications Program) program and Advanced Liquid Metal Cooled Reactor (ALMCR) program were developed in parallel with NERVA and were concerned with the development of power supplies for satellites and deep space probes (1960 through 1973). Although much of the SNAP program was restricted to the design and deployment of RTG (Radio-isotope Thermal Generator) power sources, an operating liquid metal reactor, the SNAP-10A, was launched in 1965 and subsequently tested at criticality in orbit.

High power, pulsed, reactor concepts are being explored for orbital applications and general boost applications. With the renewed emphasis on space exploration, NTR (Nuclear Thermal Rocket) technology is being reexamined for possible uses to minimize IMEO (Initial Mass in Earth Orbit) for long duration missions (L-2). Although the NERVA program has already established an NTR technology base, other advanced concepts are being studied as attractive alternatives to this 1960's technology.

2.2 PARTICLE BED REACTORS

2.2.1 Applications

Placing any payload into LEO (Low Earth Orbit) requires extensive planning and support. This is usually measured by the cost of placing a unit weight into orbit (ie, $/kg) and places an emphasis on minimizing IMEO. Budget limitations drive the need for a compact, lightweight power system which can provide adequate energy for its application. Conventional terrestrial reactor designs are too bulky for use in space so advanced reactor concepts are required. One of these advanced concepts is the PBR which is projected to provide up to 50% greater specific impulse, for a propulsion application, than the current NERVA design (L-2).

PBR's are being investigated by the Idaho National Engineering Laboratory (INEL) Multi-Megawatt Project Office, Brookhaven National Laboratory (BNL) and Sandia National Laboratory (SNL) (H-1, L-1, V-1). Open cycle systems (Figure 2.1) (M-1) as well as closed cycle systems (Figure 2.2) (G-1) are being examined. Open cycle systems are typical of a PFNTR (pressure fed nuclear thermal rocket) (H-2) while closed cycle systems are typical of pulsed power applications being studied by SNL within the scope of the PIPE experiments being conducted in the Annular Core Research Reactor (ACRR) (V-1). PBR concepts show promise in that direct cooled particles within each fuel element provide a greater power density and a relatively large surface area for the heating of the working gas resulting in lighter and smaller systems (P-1,2).

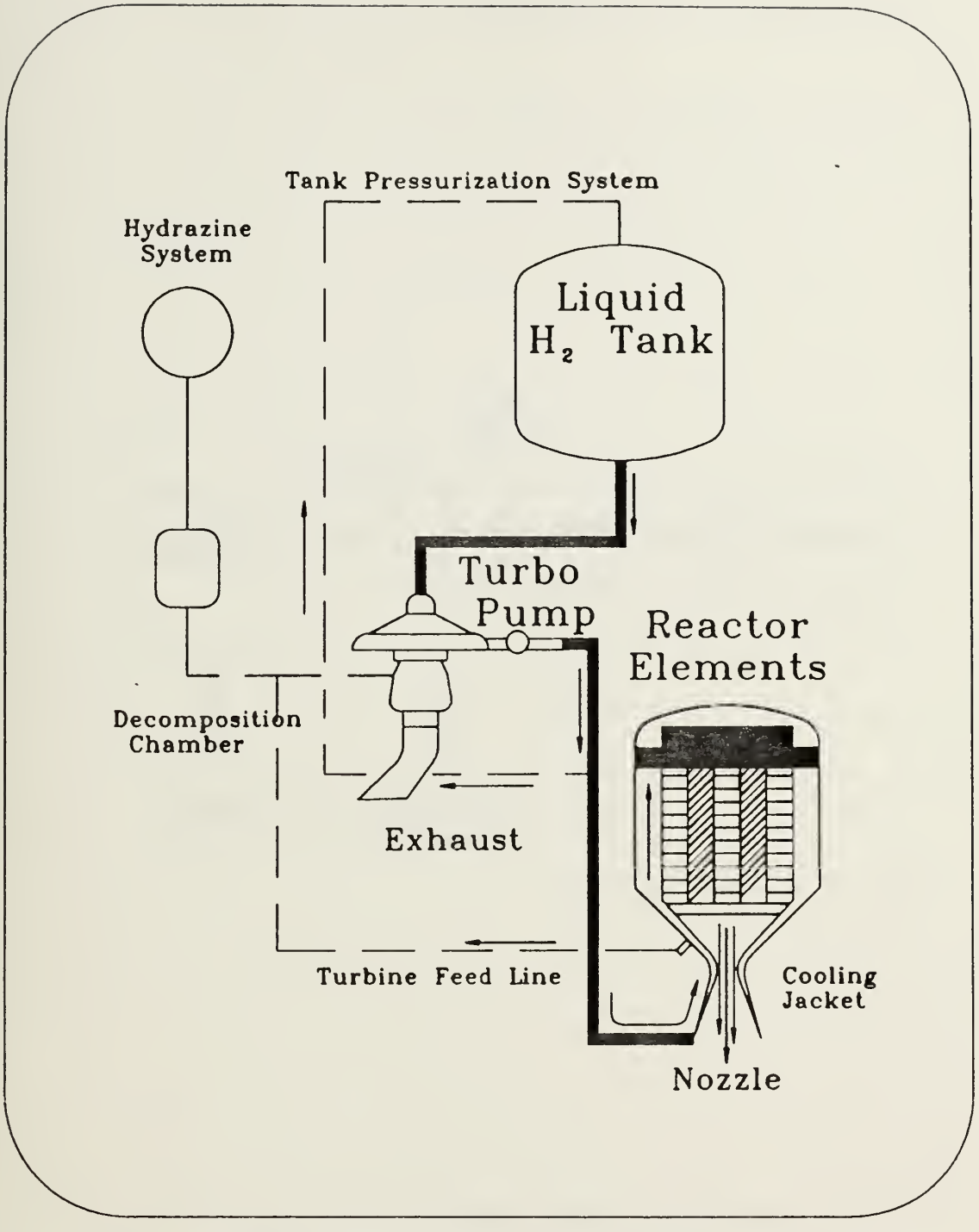


Figure 2.1: Typical Open Cycle Nuclear Thermal Rocket System

(Adapted from M-1)

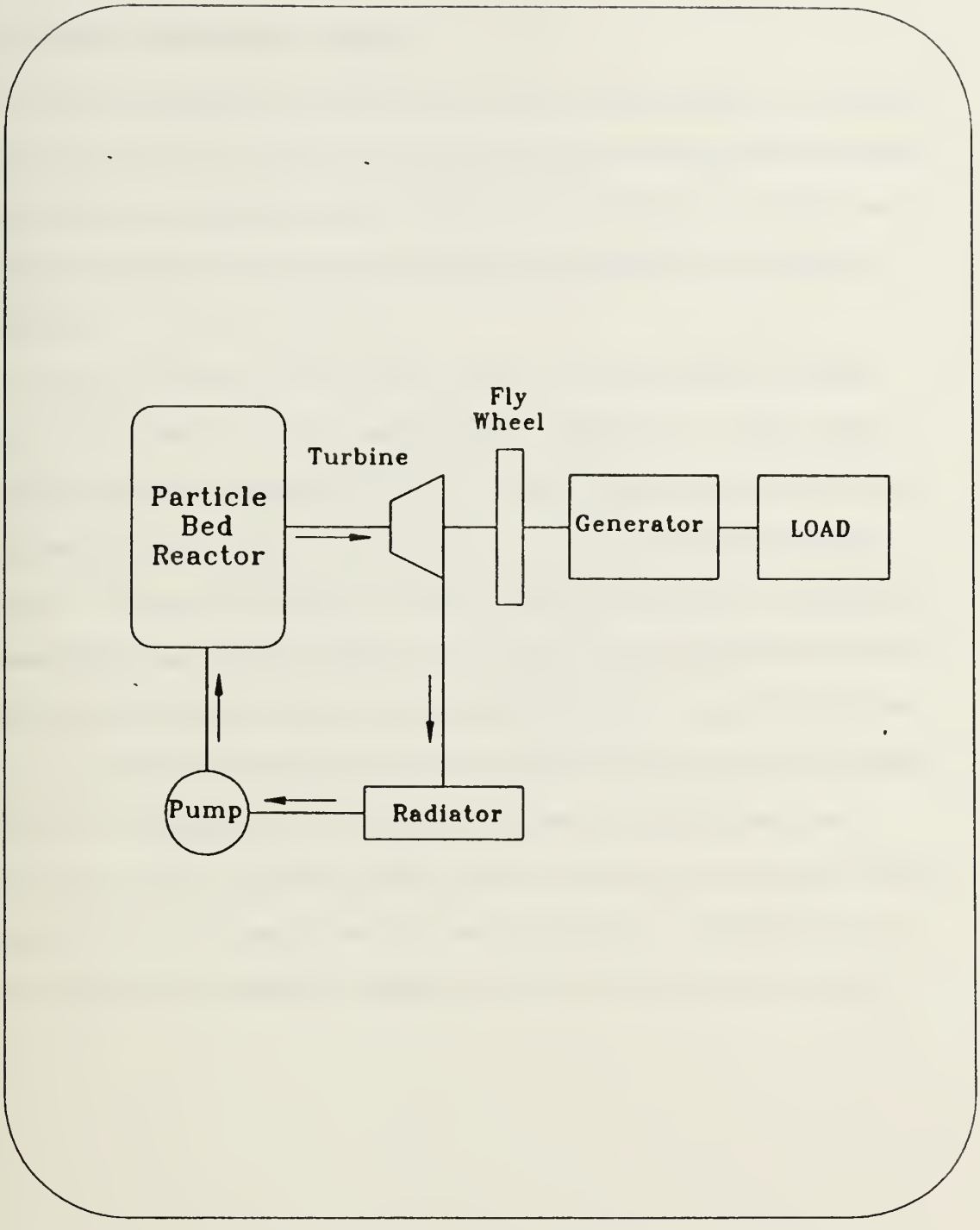


Figure 2.2: Typical Closed Cycle Space Nuclear Power System

(Adapted from G-1)

2.2.2 Particle Bed Reactor Design

The previous investigation of PBR thermal-hydraulic characteristics was conducted by Tuddenham (T-1) and addressed the complications that arise during pulsed operations. Pulsed operations are applicable to open as well as closed cycle systems. A pulse associated with the open cycle system would be analogous to a timed burn in a chemical rocket system.

The open cycle pulsed reactor usually consists of a coolant reservoir, a coolant pump, a preheating stage, the reactor assembly and the exhaust nozzle. These systems are discussed in detail in references B-2, L-2, P-1 and P-2. The coolant reservoir stores liquid hydrogen which is used as a coolant and eventually as the exhaust gas from the NTR nozzle. To maintain a relatively constant inlet pressure to the reactor assembly and provide sufficient mass flowrate through the core, a turbo-pump is placed in the system to force the coolant through the nozzle cooling jacket and then to the core. The hydrogen coolant is preheated becoming supercritical as it passes through the nozzle cooling jacket and innerpass cooling channels within the core. This provides efficient heat removal from the nozzle which is subjected to high temperature exhaust gases and ensures that the hydrogen coolant is in the gaseous state prior to entering the core. The hydrogen then passes through the fuel assemblies, is heated, and finally exhausted into space via the nozzle.

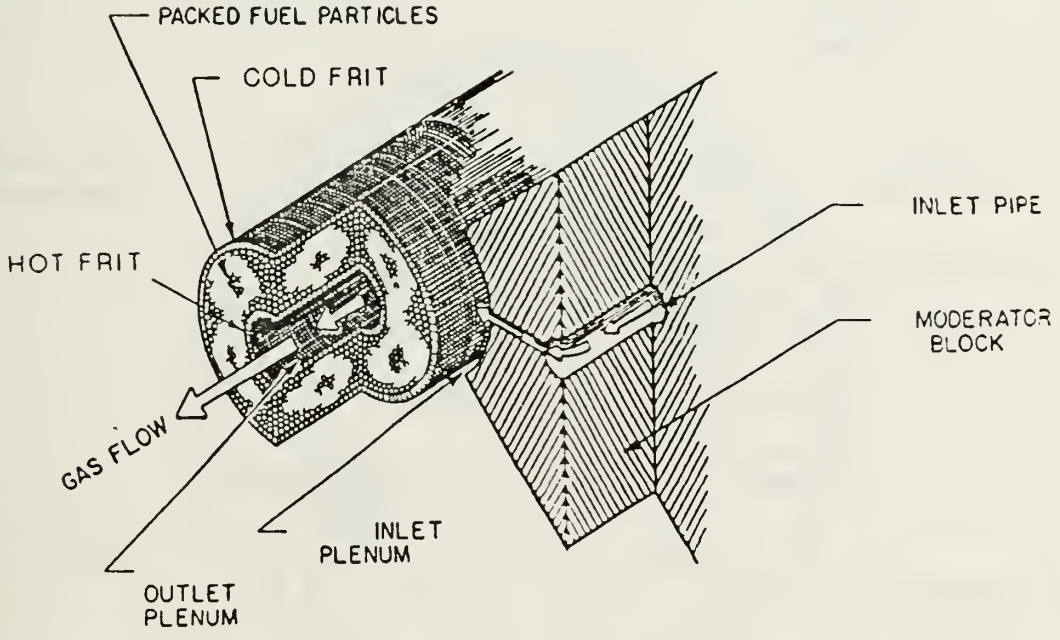


Figure 2.3: Packed Particle Bed Fuel Element

(Adapted from L-3)

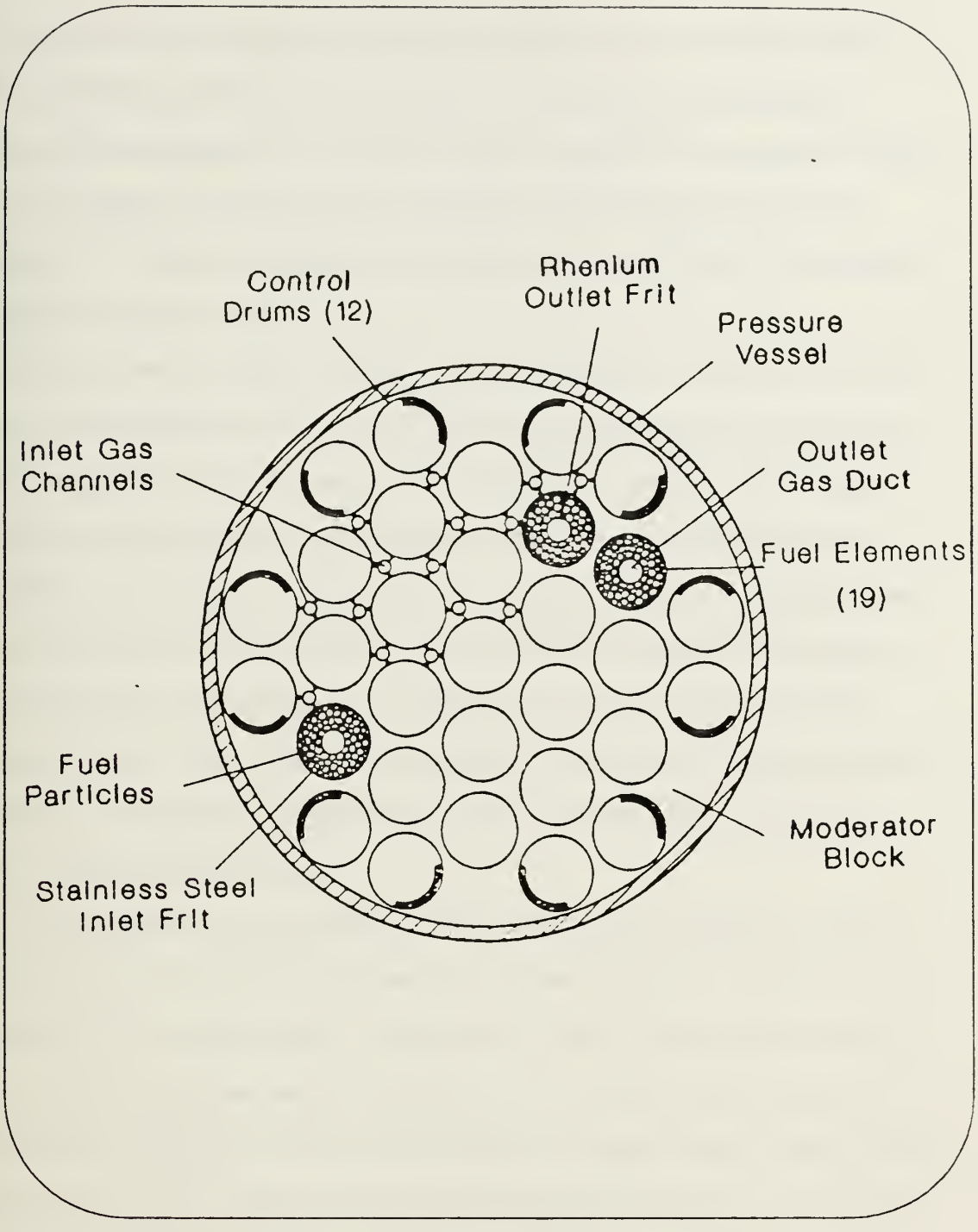


Figure 2.4: Packed Particle Bed Reactor Cross Section

(Adapted from P-3)

As the hydrogen coolant passes through the fuel elements, it experiences several directional changes as shown in Figure 2.3 (L-3). The fuel elements are clustered, packed particle beds mounted in a moderator support assembly and surrounded by control drums and reflectors. A pressure vessel, most likely aluminum, encloses the entire assembly (P-2). The fuel elements may be clustered as shown in Figure 2.4 in a number necessary for the power required.

Each fuel element assembly consists of fuel particles packed between the cold frit, the outer retention element, and the hot frit, the inner retention element. These retention elements are porous to allow the coolant to flow through the assembly but are fabricated to be robust enough to retain the fuel particles between them at elevated temperatures. The cold frit serves to distribute the coolant flow axially as well as to retain the fuel particles. This allows better cooling of more power dense regions of the particle bed and prevent preferential flow through specific regions. The cold frit is made by sintering many small (2.5 μm) stainless steel particles together. The hot frit, however, is made of a high temperature resistant material, usually rhenium, which has evenly spaced holes drilled to attain a desired porosity.

The fuel particles are approximately 500 μm in diameter and consist of a central fuel kernel surrounded by two pyrographite layers and an outer layer of zirconium carbide (Figure 2.5). Uranium carbide is used as the fuel and is enriched to approximately 93% for compactness. These particles are packed directly in the annular region between the cold frit and the hot frit without being imbedded into a graphite matrix similar to high temperature gas reactors. Direct packing provides better heat transfer to the coolant and is expected to behave well during rapid power transients associated with pulsed power operations (P-3).

2.2.3 PIPE Experiment

Pulsed Irradiation of a Packed Bed Element (PIPE) experiments have been conducted by Sandia National Laboratory (V-1). The PIPE experiments attempted to evaluate the performance of a packed particle bed fuel element in the areas of temperature characteristics, flow characteristics, fuel/coolant interaction, and power output. A fuel element test assembly (Figures 2.6 and 2.7) is then placed into the annular core research reactor (ACRR) for evaluation.

Because previous modeling of a PBR fuel element concentrated on the specific geometry of the PIPE experiment, it is important to describe the experimental apparatus used for the PIPE experiments. The test assembly is a right cylinder 4.19 m long with a diameter of .35 m. Parahydrogen is introduced to the inlet plenum of the fuel element at a pressure of 2 MPa and a temperature of 300 K by a series of blowers which circulate the coolant through the test assembly. Flow enters the inlet plenum axially then flows radially inward through the cold frit, the particle bed and the hot frit. The hydrogen then exits the outlet plenum into a heat sink composed of stainless steel ball, through the flow meter and back to the blowers for another pass (V-1).

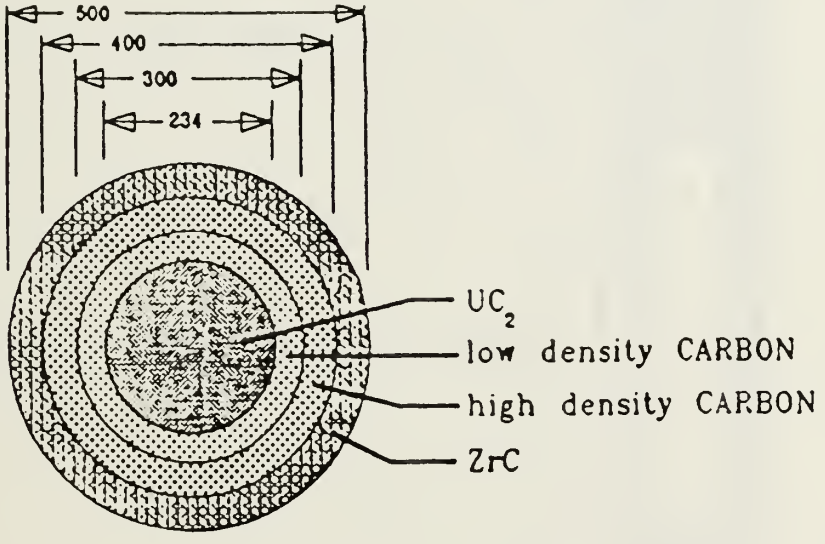


Figure 2.5: 500 Micron Fuel Particle

(Adapted from V-1)

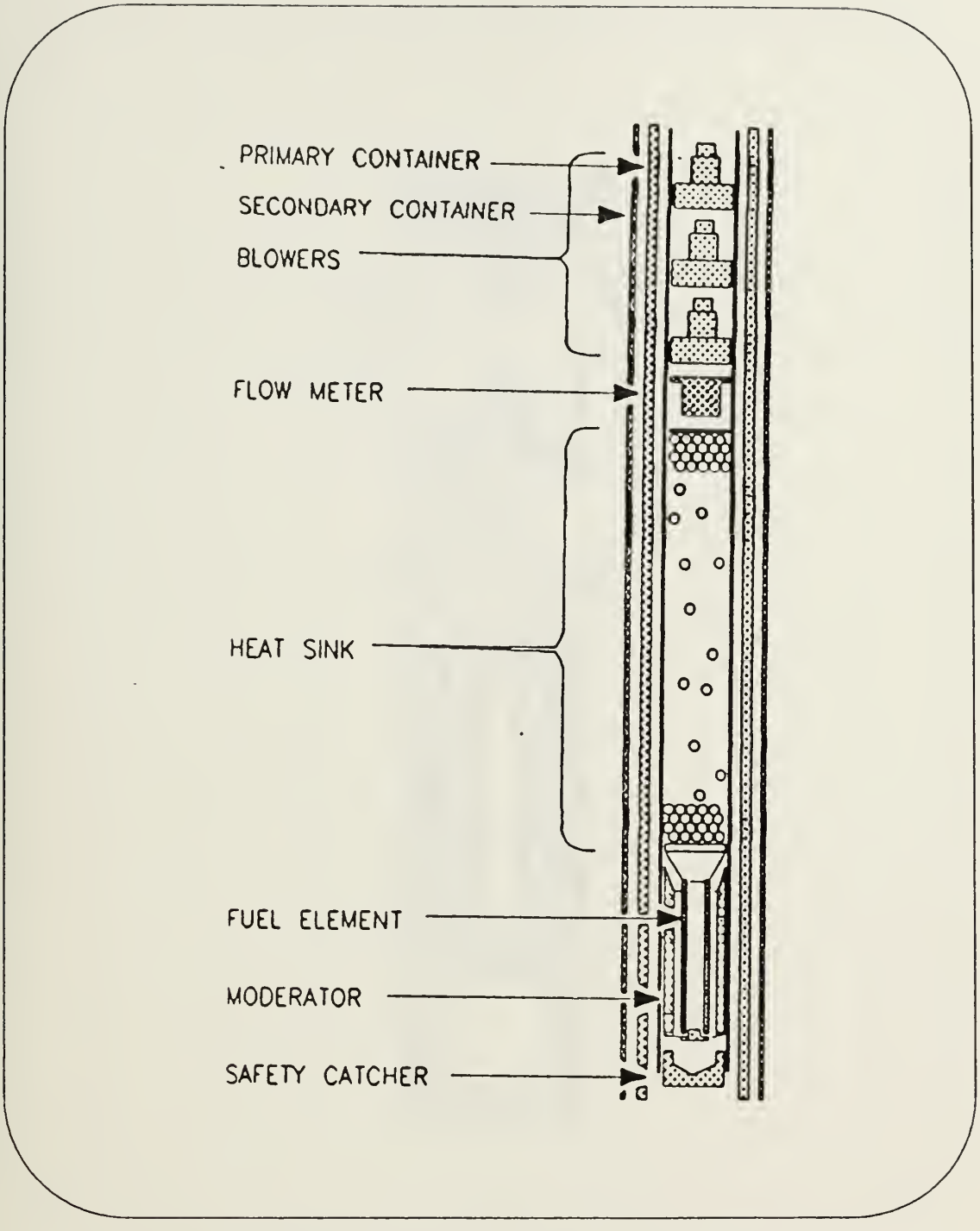


Figure 2.6: PIPE Experiment Fuel Element Test Assembly

(Adapted from V-1)

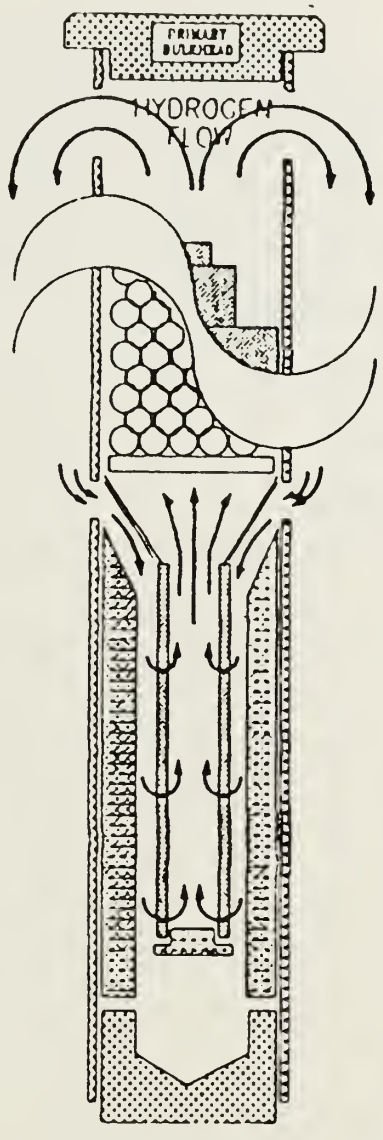


Figure 2.7: PIPE Experiment Fuel Element Test Subassembly

(Adapted from V-1)

Figure 2.8 shows detailed dimensions of the fuel element tested by Sandia National Laboratory. In addition to the measurements shown in Figure 2.6, other useful parameters are:

•Cold Frit Thickness	1.70 to 2.36 mm
•Cold Frit Porosity	68.5%
•Cold Frit Particle Diameter	2.5 μm
•Cold Frit Material	316 Stainless Steel
•Hot Frit Thickness	.76 mm
•Hot Frit Porosity	23.3%
•Hot Frit Material	Rhenium

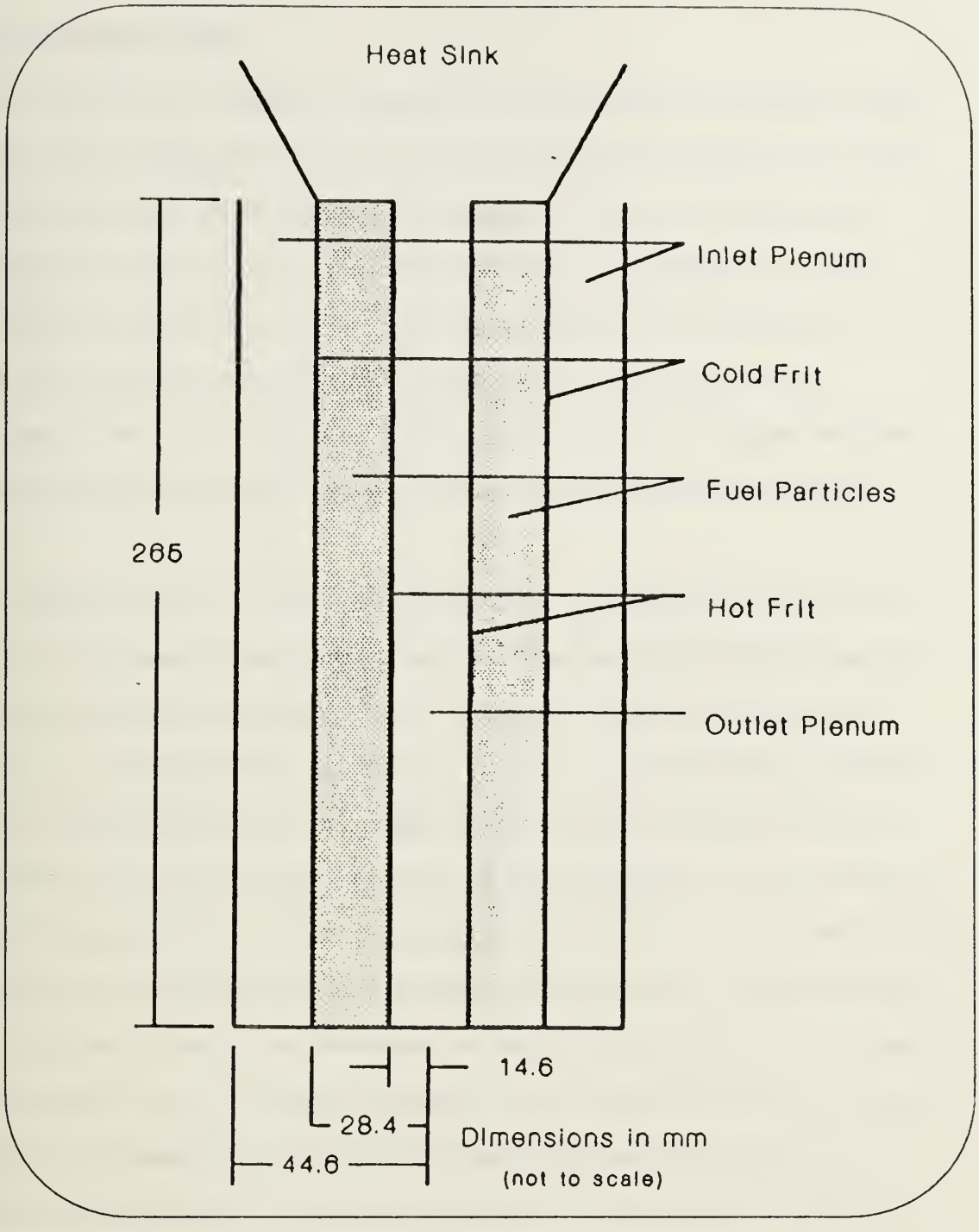


Figure 2.8: Dimensions of PIPE Experiment Fuel Element

(Adapted from T-1)

2.2.4 Existing Models

In his thesis (T-1), Tuddenham develops an analytical model to calculate the behavior of the PIPE experiment particle bed fuel element. His model, which will serve as the reference standard and which will henceforth be referred to as such, divides the fuel element into 50 control volumes. The reference standard is separated into 10 subdivisions in the radial direction and 5 subdivisions in the axial direction (Figure 2.9). Further, it uses a staggered grid arrangement (Figure 2.10) for discretization of the conservation laws. Table 2.1 specifies the geometry of the reference standard and shows how a varying cold frit thickness affects the volume of the inlet plenum and the particle bed.

Although several other models exist for advanced space reactors (B-2, G-1, L-2), the reference standard addresses only the thermal-hydraulics of a PBR fuel element. It is more universal than one of the models used by Brookhaven National Laboratory (B-1), which assumes that power and coolant flow is matched in each control volume. The reference standard separates these and provides detailed information pertaining to coolant temperature, pressure, and density for later use by a neutronic model. Also, the reference standard attempts to focus on actual operating conditions by using only instrumented system parameters, such as inlet and outlet temperatures and pressures, as sources of input.

Tuddenham balances mass, momentum, and energy in each of his control volumes at each advance in time. A maximum timestep of 50 μs is used by the reference standard. Although very detailed, the calculations tend to take a prolonged length of time. To examine a 6 second transient, for example, a single run would take approximately 48 hours of computational time on an AT compatible desktop computer (80286). On the other hand, the detail provided in the reference model allows modeling of axial and radial

crossflows between the control volumes. Because the simple model combines many of the axial and radial control volumes into a single control volume, the particle bed for instance, specific information concerning crossflow is lost. If crossflow calculations are of primary importance, the reference standard should be used.

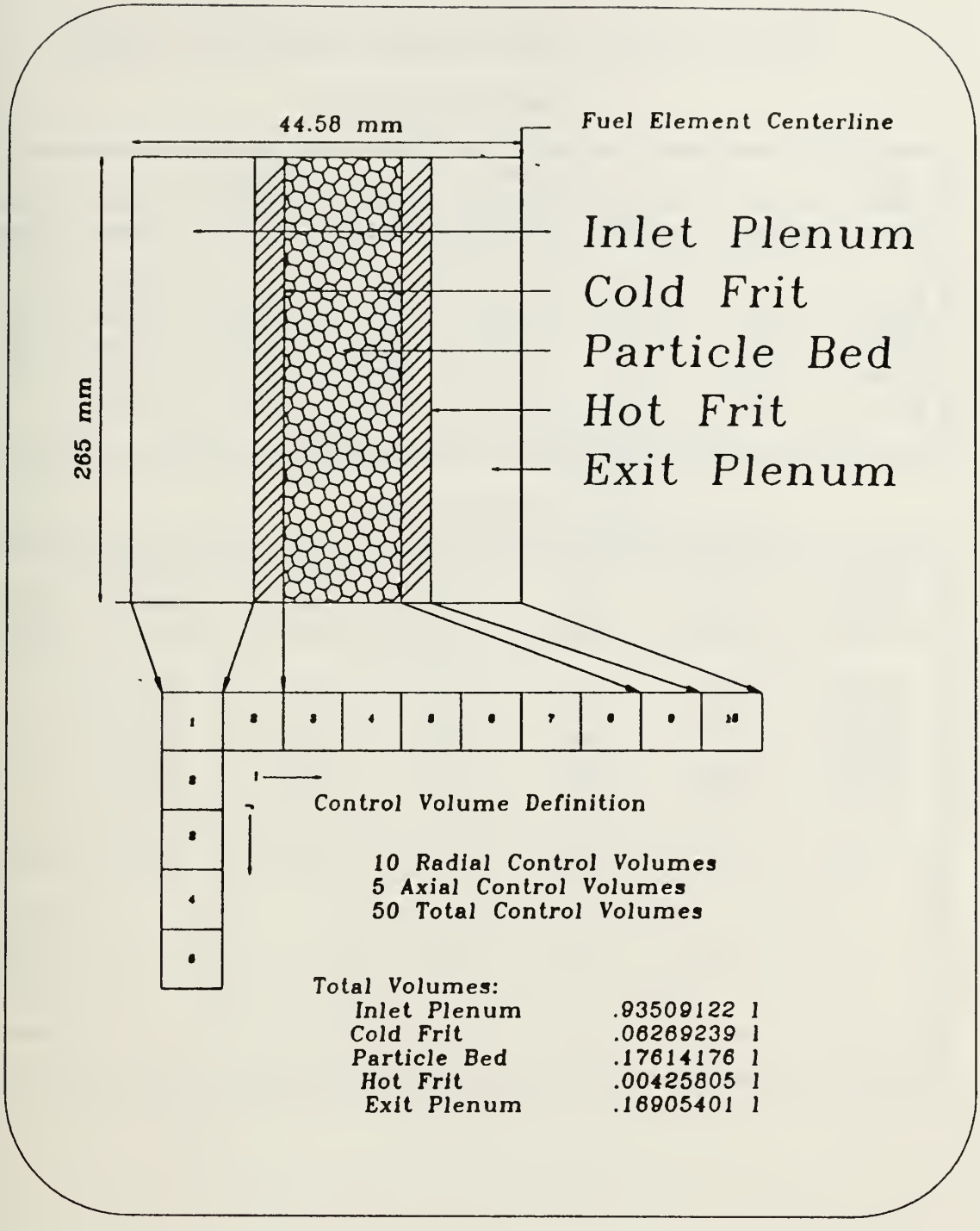


Figure 2.9: Control Volume Definition for the Reference Standard

Table 2.1: Reference Standard Control Volume Geometry

(measurements in mm)

Axial Position	Outer Radius	Cold Frit Outer Radius	Cold Frit Thickness	Outer PBED Radius	Inner PBED Radius	Hot Frit Thickness	Exit Plenum Radius
1	44.58	29.56	2.26	27.30	15.01	0.76	14.250
2	44.58	29.39	1.93	27.47	15.01	0.76	14.250
3	44.58	29.33	1.79	27.53	15.01	0.76	14.250
4	44.58	29.32	1.78	27.54	15.01	0.76	14.250
5	44.58	29.38	1.91	27.48	15.01	0.76	14.250

(Volumes in Liters)

Axial Position	Height of Control Volume	Volume of Inlet Plenum	Volume of Cold Frit	Volume of PBED	Volume of Hot Frit	Volume of Exit Plenum
1	53.00	0.185	0.021	0.087	0.0037	0.0338
2	53.00	0.187	0.018	0.088	0.0037	0.0338
3	53.00	0.188	0.017	0.089	0.0037	0.0338
4	53.00	0.188	0.017	0.089	0.0037	0.0338
5	53.00	0.187	0.018	0.088	0.0037	0.0338
TOTAL:	265.00	0.935	0.092	0.440	0.0185	0.169
Void Fraction:		1	0.685	0.4	0.23	1
Void Volume:		0.935	0.063	0.176	0.0042	0.169

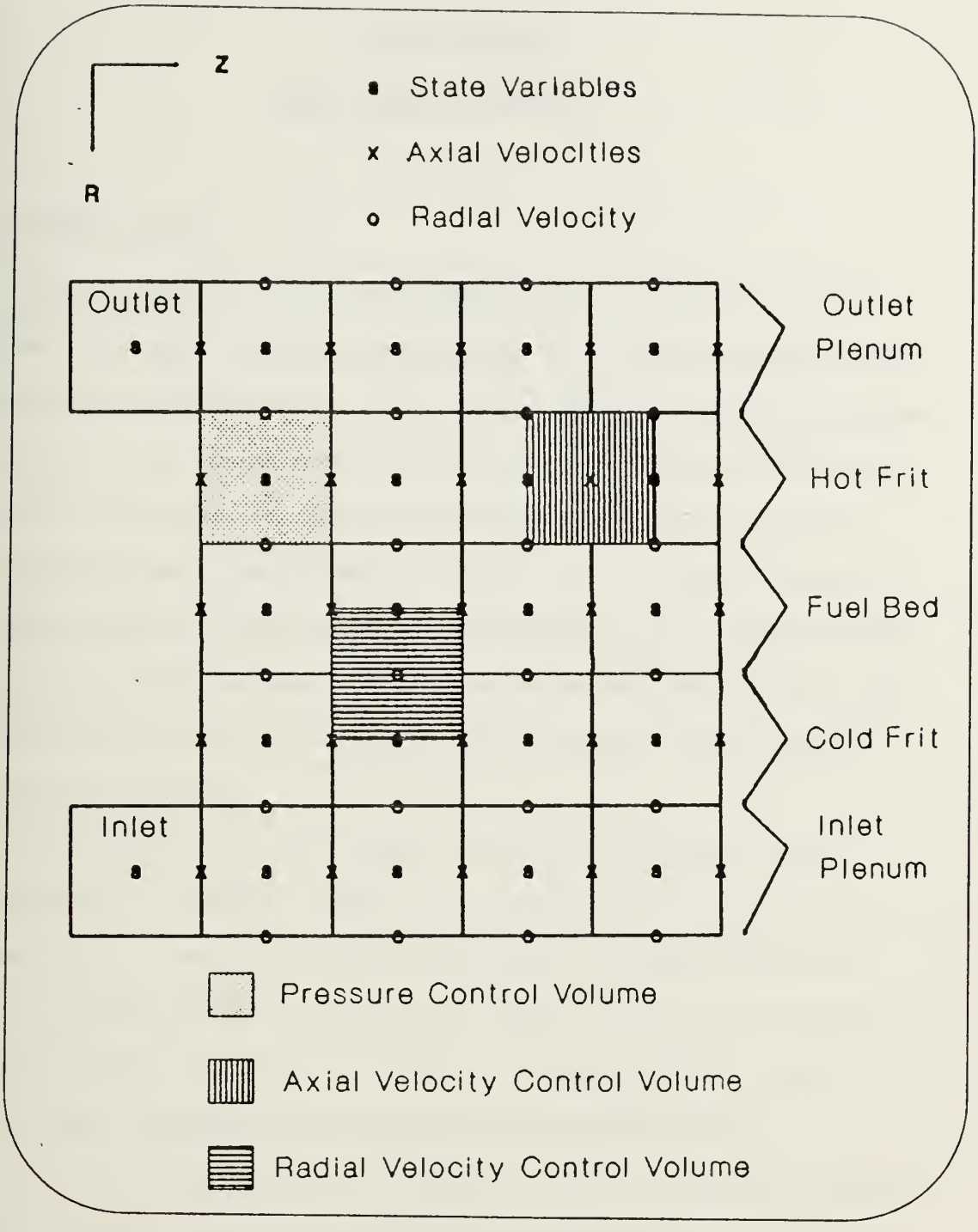


Figure 2.10: Reference Standard Staggered Grid Arrangement

(Adapted from T-1)

CHAPTER 3

THE SIMPLE MODEL

3.1 MODEL BASIS

A proposed particle bed reactor (PBR) which is to be placed in low earth orbit (LEO) will be subjected to multiple power transients. These transients may take the form of start-up and shutdown operations or rapid power transients during pulsed power operations (B-1, G-1, L-1). Regardless of the form, any transient will require precise control of the reactor. As a part of the control system development, an accurate as well as a timely simulation must be accomplished via a digital model of the reactor. When coupled with an appropriate neutronic model, the reactor may be safely controlled by analyzing control module input parameters. Because fuel element neutronics are closely tied to the thermal-hydraulic behavior of the fuel element, it is important to develop a good thermal-hydraulic model.

The basis of this research is to examine a method of analysis for the thermal-hydraulic response of a PBR fuel element in sufficient detail as to calculate reactivity feedback within the fuel for a subsequent neutronic model. The analysis should be, however, assiduously simple so as not to require excessive programming computations experienced with the reference standard model. Additionally, preliminary steps are to be taken to couple a single fuel element with other fuel elements within the core.

The simple model will have some advantages over the reference standard, the most important of which is computational speed. Because the simple model is faster than the reference standard, many more fuel element variations may be investigated. This permits

a wider scope of transient analyses and allows a departure from PIPE geometry. Further, the output data is presented quicker, granting the user a better understanding of cause-and-effect during the fuel element design phase.

The following sections discuss the formulation of the simple model and the simplifications made to improve computational performance over the reference standard with respect to the amount of time necessary to perform a single analysis. Also in this chapter, the solid phase (fuel particles) and the gas phase (coolant) are described in detail. Finally, a method of solution is presented which combines the gas and solid phases into one model.

3.2 SIMPLIFICATIONS

One of the goals of the simplified model is to be able to model the thermal-hydraulic response of a fuel element in such a manner as to develop a real-time analysis. A real-time model is one which is able to assess a system's response in a period of time less than or equal to the time it would take for an actual response. To do this, simplifications must be made to the reference standard.

Merely loading the reference standard code into a larger, faster computer (ie, Cray) would certainly reduce the amount of time necessary for a single run. However, the cost of analyzing the many variations would preclude the necessary translation. On the other hand, simplification of the mathematical method of the reference standard would produce a convenient program which could be executed on a typical desktop computer in a reasonable time frame. Further, a version of the model could be considered as part of a spacecraft power controller.

The two major simplifications incorporated into the simple model are the reduction of control volumes and the increasing of the time step. Reducing the number of control volumes provides a proportional decrease in the amount of computation required and increasing the time interval, reduces the number of times these calculations must be made during a specified simulation time interval. The overall effect is to reduce the net number of calculations.

An example of the effect of the numerical simplification is readily seen when contrasting the two models on a similar computer (80386). It takes approximately 8 hours to compute a 6 second simulated transient using the reference standard with a timestep of .05 ms but only about 2 minutes for a 10 second simulation transient using the simple model with a timestep of 100 ms. This includes a reduction in the number of control volumes for the element from fifty in the reference standard to only three in the simple model. A computational advantage of approximately 240 to 1 is realized with the simple model which allows analysis of 240 variations in the same time, on a comparable desktop computer, that it takes to do a single variation using the reference standard.

The simple model uses only three control volumes compared to the fifty defined by the reference standard. The three control volumes are exhibited in Figures 3.1, 3.2 and 3.3. The first control volume (Figure 3.1) includes the inlet plenum and the cold frit. The second control volume includes only the packed particle bed consisting of the fuel particles. Lastly, the third control volume includes the hot frit and the exit plenum. Combining the cold and hot frits with the inlet and exit plenums respectively allows a more precise model of the particle bed while defining conditions at the inlet and outlet of the fuel.

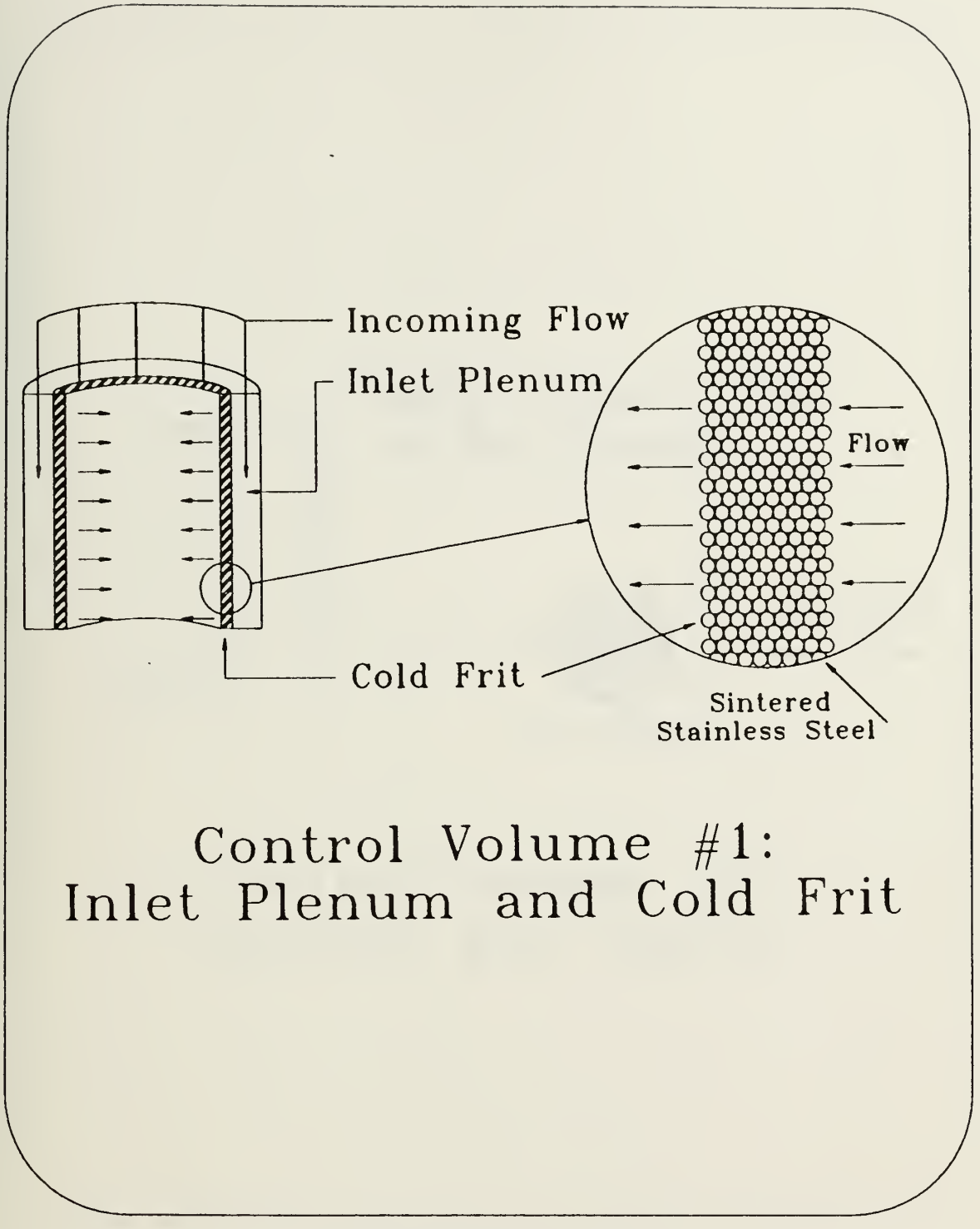
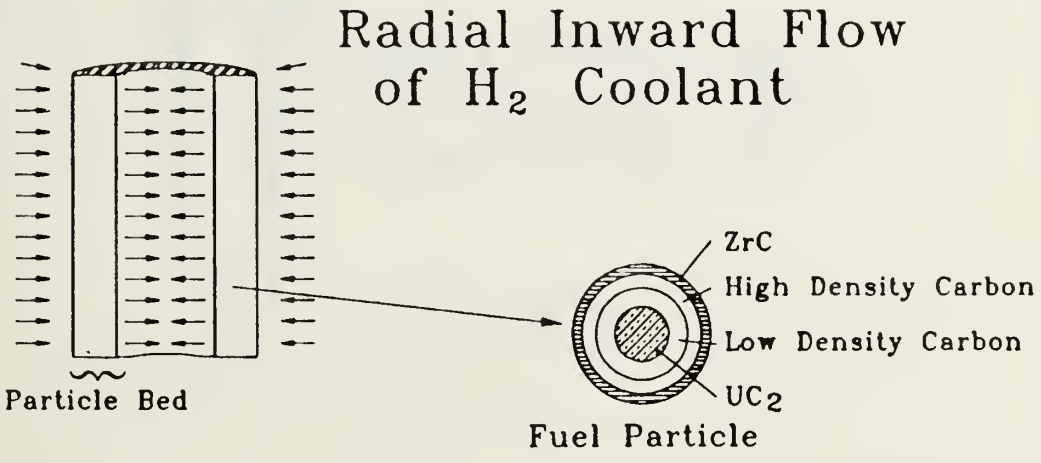


Figure 3.1: Control Volume 1



Control Volume #2:
Particle Bed Region

Figure 3.2: Control Volume 2

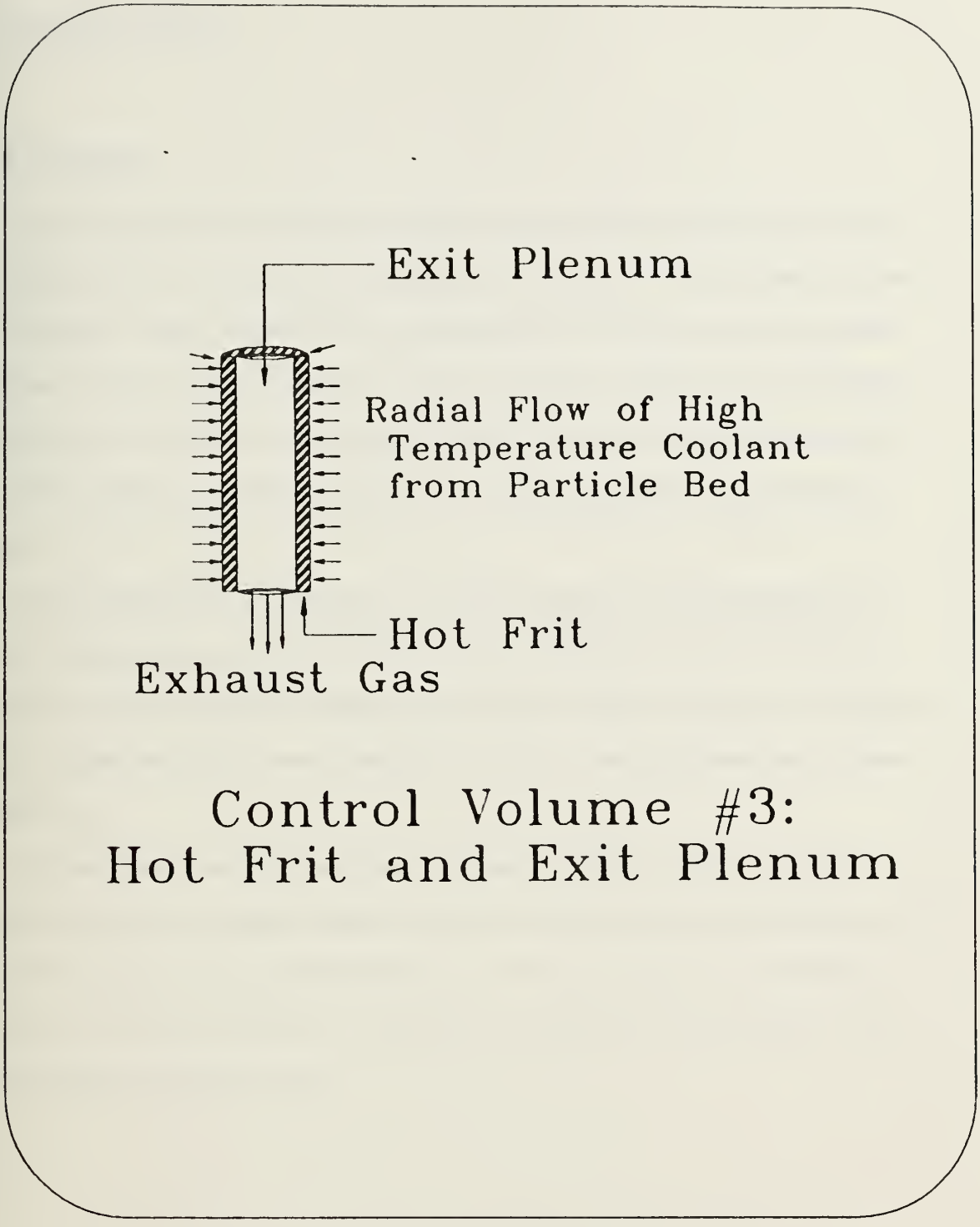


Figure 3.3: Control Volume 3

3.3 SOLID PHASE

3.3.1 General

The solid phase of this model represents the fuel particles packed between the cold frit and the hot frit. For simplicity, the particle bed is modeled as a single control volume and, to simplify further, the particle bed is represented by a single, average fuel particle. Additionally, this average fuel particle is centered axially and radially in the particle bed. This approach is similar in nature to the one Tuddenham used for each control volume of the reference standard and assumes that all of the fuel particles behave in a analogous manner.

Fuel particle dimensions and physical properties similar to the PIPE experiments are used in the modeling of the solid phase as shown in Figure 3.4. Once the physical properties of the fuel particle are defined, an overall heat transfer coefficient is developed from a weighted average of these properties and a surface fuel temperature may be determined (M-2).

Heat deposition rate also depends on the physical properties of the fuel particles used in the model. For a power transient, the particle bed power density is increased over a specific transient time to a predetermined level. The heat energy is not immediately transferred to the coolant however. The effects of heat deposition, heat storage, and heat removal must be taken into account.

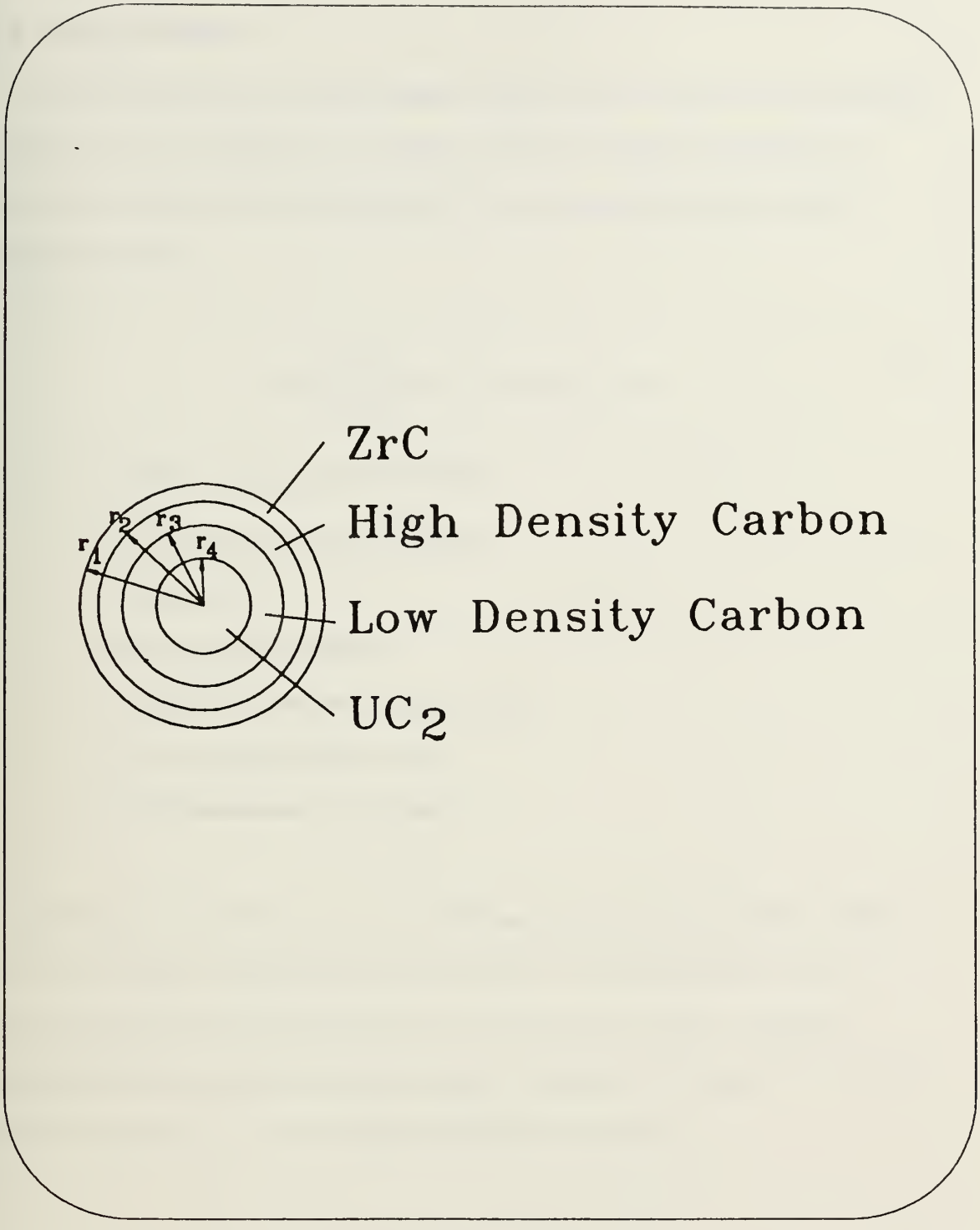


Figure 3.4: Fuel Particle Geometry

3.3.2 Energy Balance

The fuel particles are the source of heat to the hydrogen coolant and can be modeled as a separate control volume from the coolant. Since all of the fuel is contained in a single control volume, the following equation for the conservation of energy may be employed for the fuel:

$$\left(m\overline{C}_p\frac{\overline{dT}}{dt}\right)_{CV} = q - hA_V V_{CV}(T_S - T_G) \qquad 3.1$$

$\overline{T}$ = effective fuel particle temperature

T_S = fuel particle surface temperature

q = heat source

h = heat transfer coefficient

A_V = particle surface area per unit volume

V_{CV} = size of the control volume

T_G = bulk temperature of coolant

Modeling of the solid phase for this model parallels the reference standard. Reference M-2 uses a single node analysis for the particle bed. This allows the use of an assumed steady state temperature distribution through the particle bed. In addition, material properties are assumed to remain constant. Reference M-2 simplifies the general heat balance equation in 3.1 by developing weighted average value:

$$m_a\overline{C}_p\frac{d\overline{T}}{dt} = q'' - \overline{U}(\overline{T} - T_G) \qquad 3.2$$

m_a = particle mass per unit surface area (kg/m^2)

$\overline{C}_p$ = average particle specific heat (J/kgK)

q'' = heat deposition per surface area (W/m^2)

$\overline{U}$ = effective over all heat transfer coefficient (W/m^{2}K)

It must be pointed out that $\overline{T}$ does not represent a temperature at a specific point in the fuel particle but is an average temperature for use in equation 3.2. The remaining values are determined in accordance with the procedures of M-2 which uses i = 1 ... 4 to designated the shells of figure 3.4:

$$m_{ai} = \frac{(r_i^3 - r_{i+1}^3)\rho_i}{3r_1^2} \qquad m_a = \sum_{i=1}^{4} m_{ai} \tag{3.3a}$$

$$\overline{C}_p = \sum_{i=1}^{4}\left(\frac{m_{ai}C_{pi}}{m_a}\right) \tag{3.3b}$$

$$U_i = \left(\frac{r_i r_{i+1}}{r_i - r_{i+1)}}\right)\frac{k_i}{r_1^2} \qquad U_4 = \frac{2k_4}{3r_1} \qquad U_T = \left(\sum_{i=1}^{4}\frac{1}{U_i}\right)^{-1} \tag{3.3c}$$

$$f_1 = \frac{U_T}{U_1} \qquad f_2 = \frac{U_T}{U_2} + f_1 \qquad f_3 = \frac{U_T}{U_3} + f_2 \tag{3.3d}$$

$$\bar{f} = \frac{1}{2m_a\overline{C}_p}\{m_{a1}C_{p1}f_1 + m_{a2}C_{p2}(f_1+f_2) + m_{a3}C_{p3}(f_2+f_3) + m_{a4}C_{p4}(f_3+1)\} \tag{3.3e}$$

$$\overline{U} = \frac{U_T h}{\bar{f}h + U_T} \tag{3.3f}$$

Although the properties associated with the fuel particle vary with temperature, the simple model, as well as the reference standard, hold these material properties constant. The exception is the heat transfer coefficient, h, which is allowed to vary with temperature and with coolant properties. h is shown in the following equation as a function of the Nusselt number and inversely with the sampling position in the particle bed. This relationship will be discussed in greater detail in the next section.

$$h_{coolant} = Nu_p \frac{k_{coolant}}{x} \qquad 3.4$$

where: x = position in the particle bed (in this case x will be half of the particle bed thickness)

Conductive heat transfer between adjacent fuel particles and control volume boundaries as well as radiative heat transfer, which can occur at elevated temperatures, are neglected in the simple model. These forms of heat transfer are addressed and discussed at length in reference T-1 which concludes that their effects are minor compared to the convective heat transfer.

3.3.3 Discretized Form

In order to analyze the thermal-hydraulic response of the PBR fuel element, numerical approximations must be made to the ordinary differential equations and the convective equations used in the model. The solid phase heat balance equation addressed in equation 3.2 may be solved by using a discretized form of the equation.

As with the reference standard, each side of equation 3.2 is multiplied by the control volume size, V_{CV}, and the particle surface area per unit volume, A_V, to produce total power terms (Watts). This is shown in the following expression:

$$m_a \overline{C}_p V_{CV} A_V \frac{d\overline{T}}{dt} = q'' V_{CV} A_V - \overline{U} V_{CV} A_V (\overline{T} - T_G) \tag{3.5}$$

where A_V can be expressed as a function of the void fraction and the fuel particle diameter, D_P:

$$A_V = \frac{6(1-\varepsilon)}{D_P} \tag{3.6}$$

To simplify this expression, incorporate equations 3.7, 3.8, and 3.9:

$$\text{Let } A = \overline{U} V_{CV} A_V \tag{3.7}$$

$$\text{Let } B = m_a \overline{C}_p V_{CV} A_V \tag{3.8}$$

$$\text{Let } Q = q'' V_{CV} A_V \tag{3.9}$$

These substitutions in equation 3.5 result in:

$$B \frac{d\overline{T}}{dt} = Q - A(\overline{T} - T_G) \tag{3.10}$$

A simple implicit scheme is used to place equation 3.10 into a discretized form. This helps to provide stability at larger timesteps. In addition to using $\overline{T}$ at time n+1, Q is also advanced to n+1. Q^{n+1} represents the driving element in this relationship. If Q were to remain unchanged, the result would be a null transient. Further, because the A (equation 3.7) is a function of properties which change with temperature, it is evaluated at time n for the discretized form of 3.10 which can be expressed as:

$$\frac{\overline{T}^{n+1}-\overline{T}^{n}}{\delta t}=\frac{Q^{n+1}}{B}-\frac{A^{n}}{B}\left(\overline{T}^{n+1}-T_{G}^{n}\right) \tag{3.11}$$

When this is solved for $\overline{T}^{n+1}$, a simple expression results:

$$\overline{T}^{n+1}=\frac{\overline{T}^{n}+\frac{\delta t}{B}(Q^{n+1}+A^{n}T_{G}^{n})}{1+\delta t\frac{A^{n}}{B}} \tag{3.12}$$

Equation 3.14 is incorporated into the simple model to advance the average temperature of the fuel. The interphase heat energy transferred from the solid phase to the gas phase can now be calculated from the temperature difference between the average fuel temperature and the bulk temperature of the coolant. The interphase heat transfer, Q_I, is expressed as:

$$Q_{I}^{n+1}=\overline{U}A_{V}V_{CV}\left(\overline{T}^{n+1}-T_{G}^{n}\right) \tag{3.13}$$

3.4 GAS PHASE

3.4.1 General

Although less complicated in nature than the reference standard, many of the features incorporated in modeling the gas phase in the reference standard are included in the simple model. The gas phase is, however, more complex than the solid phase in that it includes both hydraulic and thermal characteristics. Complications arise, however, in defining the thermal-hydraulic balances within each control volume.

The general structure of the gas phase equations balances energy, mass, and momentum for each control volume. As enthalpy and mass flowrate are solved for each control volume, these values are advanced in time and forwarded to the adjoining control volume. This allows the model to generate a time response for the fuel element during a simulated transient.

With the exception of the total number of control volumes, the simple model is based on the same assumptions and significant features which affect the element response addressed in the reference standard. As in the reference standard, the simple model is cylindrical and includes axial and radial flows. Cross axial flow in the particle bed is not a variable for the simple model since the flow vectors are contained entirely within the second control volume.

In the previous section, it was mentioned that the heat transfer coefficient, h, for the hydrogen coolant varies with temperature and velocity. Reference E-1 provides an expression for the heat transfer coefficient in terms of the Nusselt number, the heat conductivity of hydrogen and the sample position within the particle bed:

$$h = \frac{Nu_p k}{x} \qquad 3.14$$

where: Nu_p = Nusselt number;

k = coolant heat conductivity;

x = .5 (particle bed thickness)

Reference E-1 also provides an expression for the Nusselt number:

$$Nu_p = 0.8 Re^{.7} Pr^{.33} \qquad 3.15$$

where: $Re = \frac{\rho v d_p}{\mu}$

$d_p = \frac{6}{S_p}$

$S_p = \frac{6(1-\varepsilon)}{D_p}$

D_p = particle diameter;

d_p = effective particle diameter (E-1)

The rest of this section discusses the energy, mass, and momentum balance techniques and equations used in the simple model. Once the basic formulations are identified, a discretized form of the relationships will be stated and terms specific to each control volume will be assembled for final analysis.

3.4.2 Energy and Mass Balance

The primary energy transfer mechanism within the particle bed is convection heat transfer and it is assumed that all of the heat deposited in the fuel particles will be transferred to the hydrogen coolant. In order to accurately calculate the response of the entire fuel element to changes in input heat energy, an energy balance and mass balance must be performed on each control volume. Since there are fewer control volumes in the simple model as compared to the reference standard, the calculations involved will be fewer as well.

The interphase heat transfer provided by equation 3.13 represents the heat energy being convected to the hydrogen coolant. This serves to couple the solid phase with the gas phase of the model. The rate at which heat is transferred from the solid phase to the gas phase depends on the rate of increase of fuel heat deposition and coolant properties.

For a control volume with fixed volume, the time rate of change of enthalpy is the sum of the heat energy source, the time rate of change of control volume pressure and volume, and the sum of enthalpy flux terms. This is similar to equation 3.22 in reference T-1:

$$\left(\frac{d(MH)}{dt}\right) = Q_I + V_{CV}\frac{dP}{dt} + \sum_{i=1}^{I} W_i H_i \qquad 3.16$$

where: M = mass within the control volume;

Q_I = heat transfer from solid to coolant for control volume;

P = average pressure within the control volume;

H = specific enthalpy;

W = mass flowrate.

For the simple model fewer control volumes limit the number of enthalpy flux terms and, as a result, equation 3.16 may be written as:

$$\frac{d(MH)}{dt} = Q_I + V_{CV}\frac{dP}{dt} + W_{in}H_{in} - W_{out}H_{out} \qquad 3.17$$

The general mass balance equation may be expressed as:

$$\frac{dM}{dt} = W_{in} - W_{out} \qquad 3.18$$

Equations 3.17 and 3.18 may be combined to produce a more direct equation which will allow us to advance enthalpy. This is accomplished by expanding the left hand side of equation 3.17, multiplying both sides of equation 3.18 by enthalpy, H, and subtracting equation 3.18 from 3.17. Also assuming that H_{out} is representative of the enthalpy in the control volume (a donor cell method) and mass flowrate, W, is the same throughout the control volume, a general relationship is produced:

$$M\frac{dH_{out}}{dt} = V_{CV}\frac{dP}{dt} + Q_I + W(H_{in} - H_{out}) \qquad 3.19$$

In the discretized form of equation 3.19, and adopting an implicit formulation, values at the current timestep may be used to compute the enthalpy for the subsequent timestep. An implicit numerical formulation is also used with the enthalpy to provide numerical stability at large timesteps. This results in:

$$\frac{(H_{out}^{n+1} - H_{out}^{n})}{\delta t} = \frac{1}{M^n}\left\{V_{CV}\left(\frac{\delta P}{\delta t}\right)^n + Q_I^{n+1} + W^n(H_{in}^{n+1} - H_{out}^{n+1})\right\} \quad 3.20$$

Values of Q_I^{n+1} and H_{in}^{n+1} are determined from the previous control volume (or inlet boundary) and may be evaluated at $n+1$. For example, H_{in}^{n+1} for the first control volume represents the enthalpy of the coolant being supplied to the control volume. The source code allows a change in the inlet temperature during a transient which would subsequently affect the enthalpy of the coolant leaving the control volume.

3.4.3 Momentum Balance

In addition to a balance of mass and energy, a balance of momentum must be dovetailed into the response of the fuel element. For the simple model, the framework of the MINET (Momentum Integral Network) code (V-2) is used to determine the effects of momentum on the flow of coolant. The MINET code develops equivalent forces which act to resist the flow of the coolant through each control volume. These resistive forces can be divided into the significant mechanisms which contribute to the head loss through a fuel element. Unlike the MINET code, which accounts for spatial changes in mass flowrate, the simple model assumes that mass flowrate is the same everywhere (at any time t).

The basic momentum integral equation (V-2) used by the simple model is

$$I_{CV}\frac{dW}{dt} = P_{in} - P_{out} - \sum_{i=1}^{N}(F_{eq})_i \quad 3.21$$

where: $I_{CV} = L/A$ (control volume inertia)

L = average length of travel for each control volume

A = cross sectional area perpendicular to flow

F_{eq} = equivalent resistive pressure drop

This equation is equivalent to a constitutive relationship which relates a change in pressure to the flowrate through a control volume (C-1):

$$\Delta P = R_{Total} W^2 \tag{3.22}$$

where: R_{Total} = total equivalent resistance;

ΔP = pressure drop.

Equation 3.21 may now be expressed as a function of the total pressure drop across the control volume and the sum of flow resistances:

$$I_{CV} \frac{dW}{dt} = \Delta P_{CV} - \sum_{i=1}^{N} (R_{eq})_i W^2 \tag{3.23}$$

This is a convenient expression because all of the terms may be determined from existing relationships. These include relationships for an equivalent pressure drop due to friction, spatial acceleration, expansion or contraction, manifold mixing, and packed particle beds. Although the MINET code includes gravity terms, they are considered negligible. Expressions which differ from the reference standard will be examined more closely.

The equivalent pressure drop due to friction is calculated for the inlet and the outlet plenums only and takes the form of equation 3.24:

$$F_f = f_f\left(\frac{L_p}{D_{eq}}\right)\frac{W^2}{2\rho A^2} \qquad 3.24$$

where: F_f = equivalent pressure drop due to friction;

L_p = plenum half-length;

$f_f = 0.138Re^{-0.151}$;

$$Re = \frac{\rho v D_{eq}}{\mu} \qquad 3.25$$

$$D_{eq} = \frac{4A}{P_{wet}}$$

Since frictional pressure drop is directly related to the distance that the coolant must travel through the control volume, an average path length is defined for the inlet and outlet plenum which is half of the overall element length. The cross-sectional area used in this calculation is the area in the plane perpendicular to the axial axis of the fuel element. This area is used because the path length in the axial direction (for the inlet and outlet plenums) is so much greater than the radial path length. Density is the average of the density calculated at the inlet and outlet of each plenum.

Equation 3.25 relates a correlation line which was fit to the log-log Moody plot of friction factors as a function of Reynold's number, Re, and roughness in the channel caused by the frits (T-1). This improves on the relationships developed by McAdams and

Blasius to correlate friction factors for smooth pipe and is the relationship used by the reference standard. Resistance due to friction is determined in the inlet plenum and outlet plenum only since the Ergun (E-2) relation includes this effect for the particle bed.

Spatial acceleration in a duct which allows cross sectional area to change while holding coolant density constant will produce a change in pressure across a control volume. These conditions are experienced in the inlet and outlet plenums and is:

$$F_a = \left(\frac{1}{A_o^2} - \frac{1}{A_i^2} \right) \frac{W^2}{2\rho} \qquad 3.26$$

where: F_a = equivalent pressure drop due to acceleration.

The cross-sectional areas in this case are the areas associated with the inlet and outlet of the plenum control volumes. For example, for the outlet plenum, A_o is the area at the exit of the fuel element and A_i is the cross-sectional area at the inlet of the hot frit. A_o for in the inlet plenum is the cross-sectional area at the outlet of the cold frit and A_i is the area at the inlet to the fuel element. As in the calculation for the frictional pressure drop, the density used in equation 3.26 is the average of the density at the inlet and outlet of the control volume in question.

Equation 3.26 can not be used for the control volume containing the fuel particles because the coolant experiences a change in duct area and density. A general form of the change in pressure caused by spatial acceleration in a duct which allows cross sectional area and density to change is addressed in Appendix A. This case describes the changes to the hydrogen coolant as it travels radially through the particle bed. The coolant will experience a change in cross sectional flow area as well as a change in density due to the transfer of heat. This takes the form of:

$$(F_a)_{PBED} = \left(\frac{A_1 + A_2}{2A_1A_2}\right) W v_2 - \left(\frac{A_1 + A_2}{2A_1A_2}\right) W v_1 \qquad 3.27$$

where: $(F_a)_{PBED}$ = spatial acceleration losses in the particle bed;

A_1 = cross-sectional flow area at the inlet of the particle bed;

A_2 = cross-sectional flow area at the outlet of the particle bed;

v_1 = coolant velocity at the inlet of the particle bed;

v_2 = coolant velocity at the outlet of the particle bed.

Another source of resistance within the fuel element is the sudden expansions and contractions experienced by the coolant as it passes through the cold frit and hot frit. These expansion/contractions would occur when the coolant enters and exits each frit because of the area change on both sides of the frits. For example, the coolant would experience a first contraction as it entered the cold frit and a second contraction as it leaves the cold frit and enters the particle bed. The coolant would experience a first expansion as it leaves the particle bed and passes into the hot frit and a second expansion as it enters the outlet plenum. The general form of this equivalent pressure drop is adopted from T-2 and is given as:

$$F_k = K_k \frac{W^2}{2\rho A_{small}^2} \qquad 3.28$$

where: F_k = equivalent loss due to a sudden expansion or contraction;

$K_k = K_e$ for expansions;

$K_e = (1 - B)^2$;

$K_k = K_c$ for contractions;

$K_c = \frac{1}{2}(1 - B)^2$;

$B = \frac{smaller\ area}{larger\ area}$;

A_{small} = the smaller of the two areas involved.

The value of density used in equation 3.28 is the density at the point of expansion or contraction. In the case of the cold frit, the density calculated at the entrance of the particle bed is used to evaluate the equivalent pressure drop due to sudden contractions. The density calculated at the exit of the particle bed, on the other hand, is used for the expansions observed at the hot frit.

The final relationship to be used to describe the hydraulics of the fuel element is for the mixing effects experienced in the inlet and outlet plenums. This is known as manifold mixing. The general relationship was derived by Bajura (B-4, 5, 6) and improved by Datta and Majumdar (D-1, M-3), and is used in the reference standard. When put into a form compatible with equation 3.22, the manifold effect is expressed as:

$$F_M = \Theta\Sigma(F_{M-radial} + F_{M-axial}) \qquad 3.29$$

$$F_{M-axial} = \frac{W^2}{\rho A_{axial}^2}$$

$$F_{M-radial} = \frac{W^2}{\rho A_{radial}^2}$$

where: F_M = equivalent pressure drop due to the manifold effect

Θ = .95 for the inlet plenum and 1.1 for the exit plenum

The value of density used in this case is, again, the average of the inlet and outlet densities for the control volume being analyzed. In addition, it must be noted that although the reference standard uses a value of 2.66 for exit plenum Θ in the calculations,

T-1 states that the values of Θ may be adjusted to take into account differences in mixing. Because of the difference in how the simple model treats crossflow within a control volume, Θ for the outlet plenum is reduced from 2.66 to 1.1. This allows a better approximation of the results obtained by the reference standard. Although this is minor contributor to the total flow resistance, it serves to illustrate how the model may be fine tuned and results in a better agreement for pressure drop-flow relations.

To complete the set of expression used in the simple model, the Ergun relation is used to determine the equivalent pressure drop across the cold frit and the particle bed. Since the cold frit is made of small particles, modeling it as a particle bed is a good approximation. The Ergun relation treats the flow through a packed particle bed as flow through tubes with irregular cross sections vice flow around many objects and is the generally accepted approach to modeling particle beds (B-6, E-2). The general form of the Ergun relation is:

$$F_{Pbed} = L_b\left\{\frac{150\mu\vec{v}_o(1-\varepsilon)^2}{D_p^2\varepsilon^3}\right\} + L_b\left\{\frac{1.75\rho\vec{v}_o\,|\,\vec{v}_o\,|\,(1-\varepsilon)}{D_p\varepsilon^3}\right\} \qquad 3.30$$

where: F_{Pbed} = equivalent pressure drop due particle bed effects;

L_b = thickness of the particle bed;

D_p = particle diameter;

$\vec{v}_o$ = superficial velocity;

μ = viscosity;

ε = void fraction or porosity;

Using the conservation of mass relation, this expression may be put into a form which is compatible with equation 3.27 and will appear as:

$$F_{Pbed} = \left\{ L_b \left(\frac{150\mu(1-\varepsilon)^2}{D_p^2 \varepsilon^3 \rho A_o W} \right) + L_b \left(\frac{1.75(1-\varepsilon)}{D_p \varepsilon^3 \rho A_o^2} \right) \right\} W^2 \qquad 3.31$$

where: $A_o = \frac{W}{\rho \vec{v}_o}$

Once more, the density used in this relationship an average of the density at the particle bed inlet and the density at the exit of the particle bed.

Having defined all of the significant terms used to describe the pressure loss through the fuel element, the next step is to discretize equation 3.21. An implicit scheme may be used if W^2 is separated into a combination of the mass flowrate evaluated at n and n+1 to give:$W^n W^{n+1}$. The discretized form of equation 3.21 is then:

$$\frac{(W^{n+1} - W^n)}{\delta t} = \frac{1}{I_{CV}} (\Delta P_{CV}^n - R_{Total}^n W^n W^{n+1}) \qquad 3.32$$

Solving for W^{n+1}:

$$W^{n+1} = \frac{W^n I_{CV} + \delta t \Delta P_{CV}^n}{I_{CV} + \delta t R_{Total}^n W^n} \qquad 3.33$$

This section has described the process by which the simple model is able to advance enthalpy and mass flowrate. The following section will describe how this is coupled with the advance of fuel temperature and interphase heat transfer to produce a full portrayal of the thermal-hydraulic response of a typical fuel element.

3.5 METHOD OF SOLUTION

A simple flow chart of the transient solution is shown in Figure 3.5. As can be seen in the schematic, initial conditions are established by the user and an initial, steady state condition is calculated. This provides initial state variables to the transient portion of the source code. Next, enthalpy and mass flowrate is advanced for each control volume. In addition, in control volume 2, calculations include the advance of fuel temperature. This in turn provides the calculated value of the interphase heat transfer.

Of note, the control variables (heat deposition, inlet pressure, outlet pressure, and inlet temperature) are also advanced in accordance with the transient parameters selected at the initiation of the simulation.

Coolant properties, such as density, viscosity and heat transfer coefficient, are recalculated for each control volume based on new temperatures and pressures corresponding to the value of enthalpy and mass flowrate. Finally, fuel element information is sent to the output file and time is advanced by the chosen timestep for the next set of calculations.

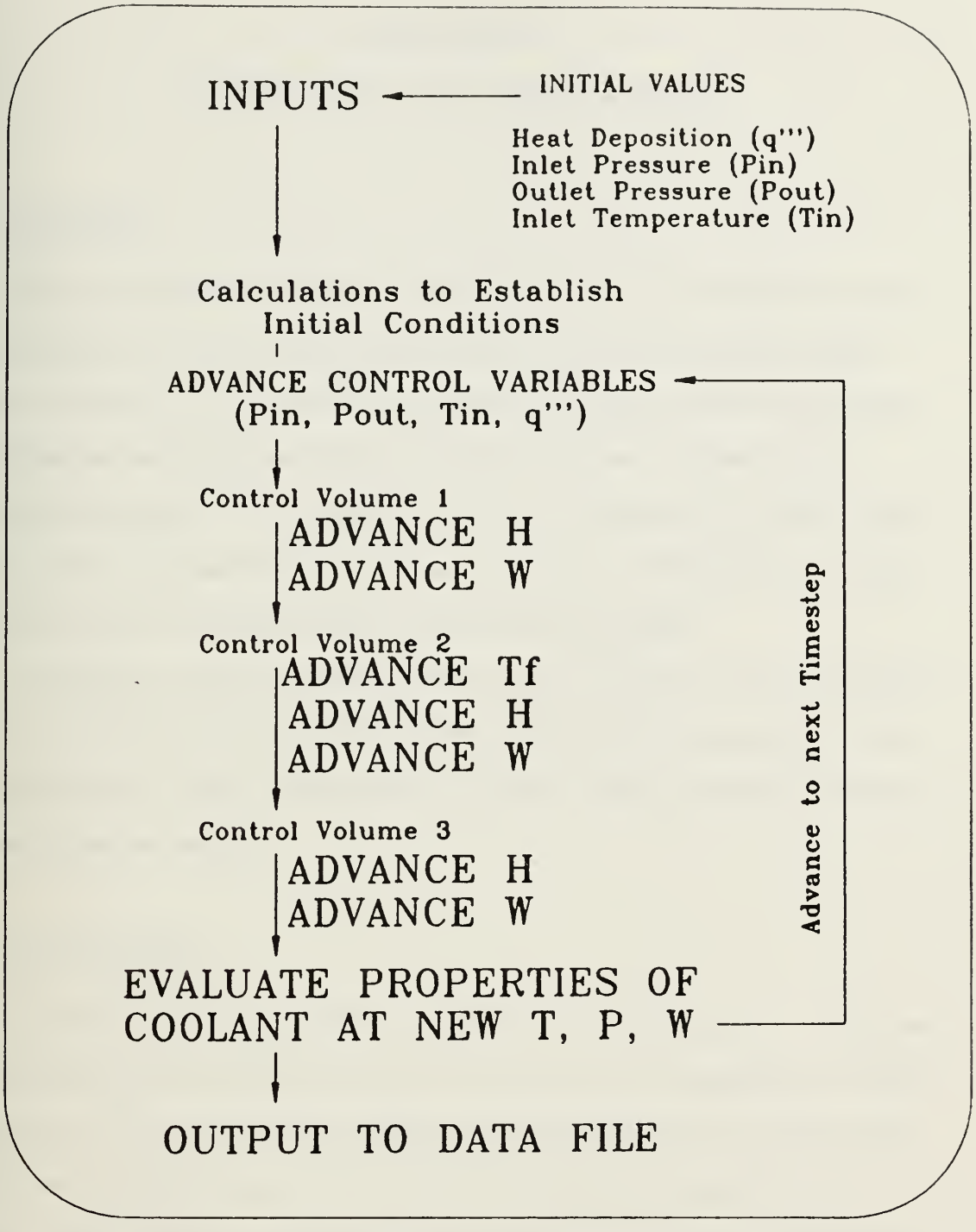


Figure 3.5: Flow Chart for Transient Solution

CHAPTER 4

VALIDATION AND APPLICATIONS

4.1 GENERAL

Although originally intended as an interim step toward the development of a real-time controller for a particle bed reactor, the simple model also lends itself serendipitously as a design tool. Because the time necessary for a single analysis is significantly shorter than that required by the reference standard, many more variations may be examined and compared. Prior to using the source code to examine these different variations, the source code must first be verified as a valid model. This may be done by comparing the results from the simple model with the reference standard.

An additional feature of the simple model is that it is capable of analyzing transients in which control variables change (inlet pressure, outlet pressure, inlet temperature, and heat deposition rate). That is, inlet pressure, for example, may be permitted to increase or decrease during the transient.

4.2 VALIDATION

Validation of the simple model is accomplished by comparing results of steady state and transient calculations provided by each model. Also, as a verification that the numerical scheme is valid, null transients are analyzed at three different power density levels. Overall, the simple model agrees very closely with the reference standard but lags slightly during a baseline 1 second transient from 0 GW/m^3 to 2 GW/m^3. Null transients are performed at 0, 1, and 2 GW/m^3.

A more in depth discussion of the validation procedure is attached as Appendix C and includes graphical comparisons of the simple model and the reference standard.

After completing a successful validation, the simple model may be used to examine variations other than the baseline (PIPE) configuration used by the reference standard and the PIPE experiment. Examples of which are supplied in the following sections.

4.3 DESIGN VARIATIONS IN FUEL ELEMENT GEOMETRY

The source code for analyzing the response of the packed particle bed fuel element allows changes in the dimensions of the fuel element. Geometry variables which may be changed include outer element radius, outer cold frit radius, cold frit thickness, particle bed thickness, hot frit thickness, and length. Other parameters which may be changed which influence the hydrodynamic characteristic of the fuel element include particle diameter, frit porosity, manifold mixing coefficients, and fuel void fraction.

As a demonstration, mass flowrates for a range of fuel element power densities were determined for a number of different fuel element lengths and plotted against the thermal power that they would deliver. Figure 4.1 shows the results of these calculations for lengths of .265 m (baseline), .500 m, .750 m, and 1.000 m. In each case, inlet pressure was maintained at 2000 kPa; outlet pressure was maintained at 1915 kPa; and inlet temperature was maintained at 300 K. Also, each curve has eleven points of data which represents calculations based on different power densities ranging from 0 GW/m^3 (thermal power equal zero) to 1 GW/m^3 (the end point of each curve) at .1 GW/m^3 intervals.

For an open-cycle nuclear thermal rocket application, Figure 4.1 could aide the designer chose a range of power operations given specific mass flowrate restrictions. Or, on the other hand, the designer could calculate the amount of hydrogen fuel/coolant

required given power level and fuel element size restraints. Figure 4.1 could be paired with a plot of exit temperature verses thermal power to ensure component temperature limits are not exceeded.

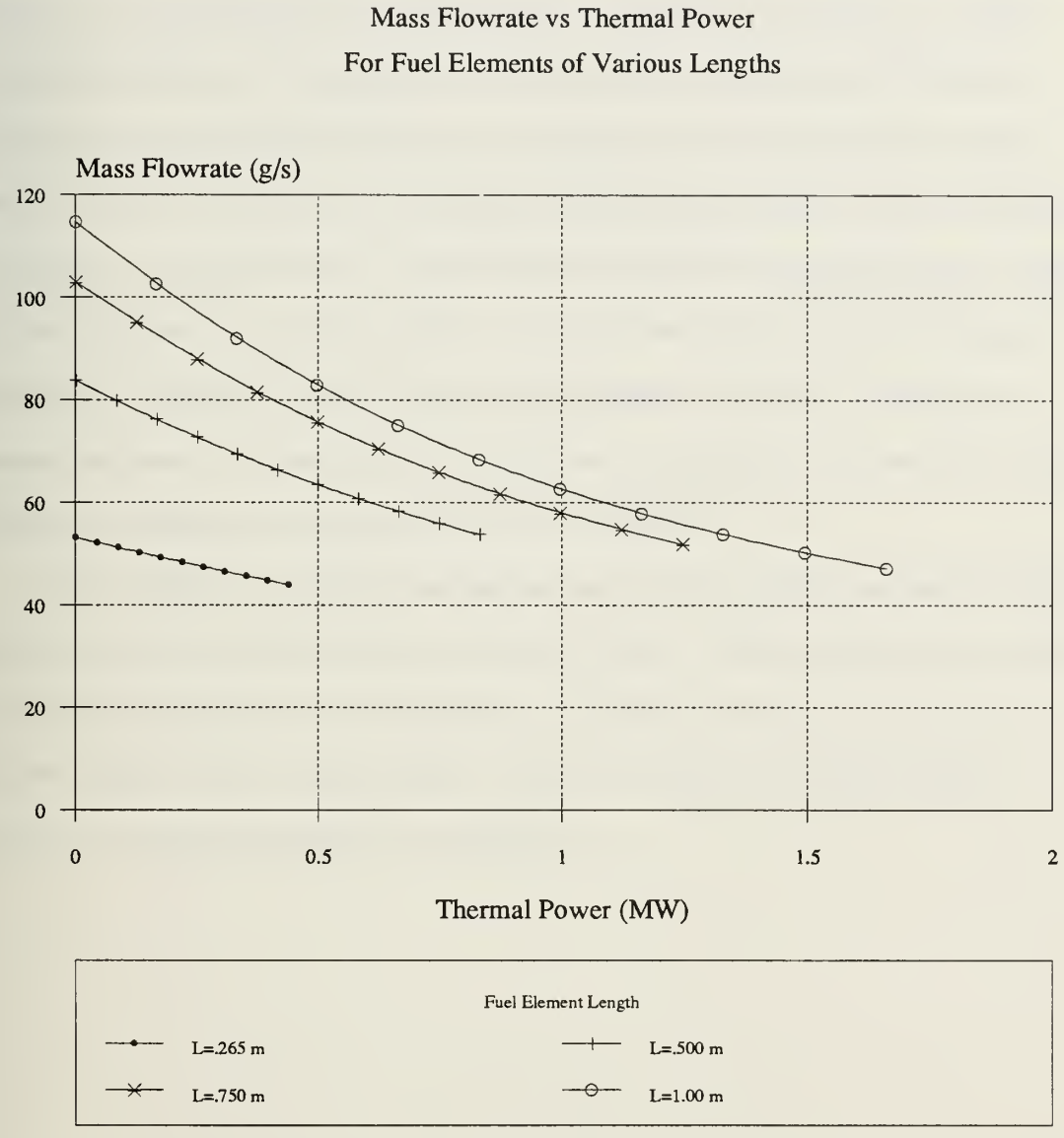


Figure 4.1: Mass Flowrate vs Power for Various Element Lengths

4.4 VARIATIONS IN THE MANNER OF TRANSIENT CONTROL

The code has the provisions to fix the initial and final values of inlet and outlet pressure and inlet temperature at a single value. They may also be changed linearly during a portion of a transient of specified duration. The time during which one of these variables may change will typically start when the power transient begins but is not coupled to the same transient duration. For example, power density may be increased from 0 GW/m^3 to 1 GW/m^3 in 1 second and inlet pressure decreased 90 kPa over a period of 5 seconds. To conserve fuel, a nuclear thermal rocket may be operated in such a fashion, that is, the reactor is started up prior to admitting the hydrogen coolant through the fuel element. Figures 4.2 and 4.3 illustrate this operating mode for a 1 m long fuel element and show how mass flowrate and delivered thermal power (the product of mass flowrate and enthalpy change) respond to such a transient.

Of note, Figure 4.2 indicates that thermal power will overshoot the required neutron power due to the extended decrease in the outlet pressure. While an increase in neutron power tends to increase the outlet temperature, a decrease in outlet pressure tends to increase mass flowrate. Thermal power continues to increase until outlet pressure reaches the required operating value of 1900 kPa.

Thermal Power Response to Combined
1s Power/15s Outlet Pressure Transient
(L=1m; P-Inlet=2000 kPa; T-Inlet=300 K)

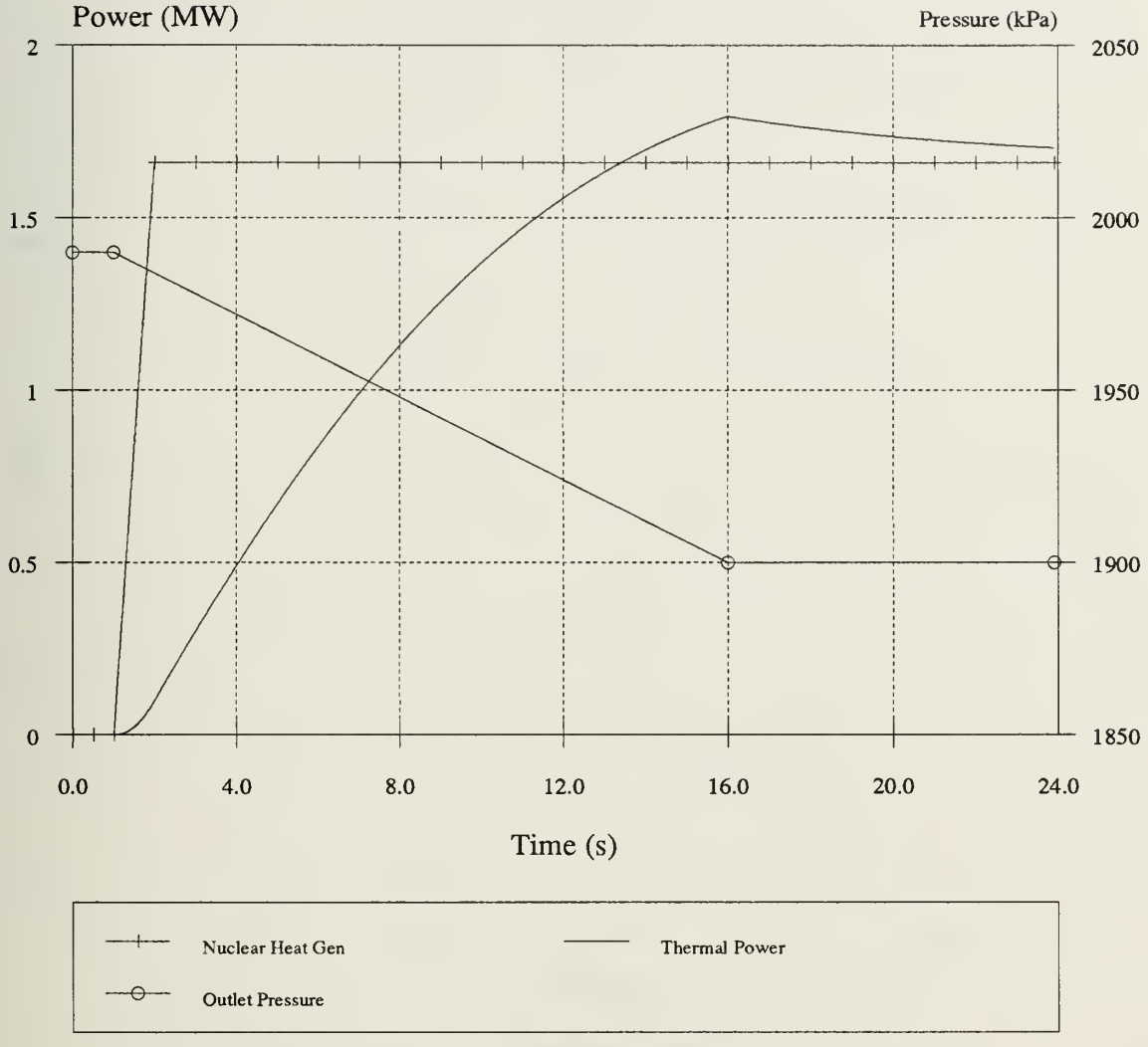


Figure 4.2: Thermal Power Response to Combined Transients

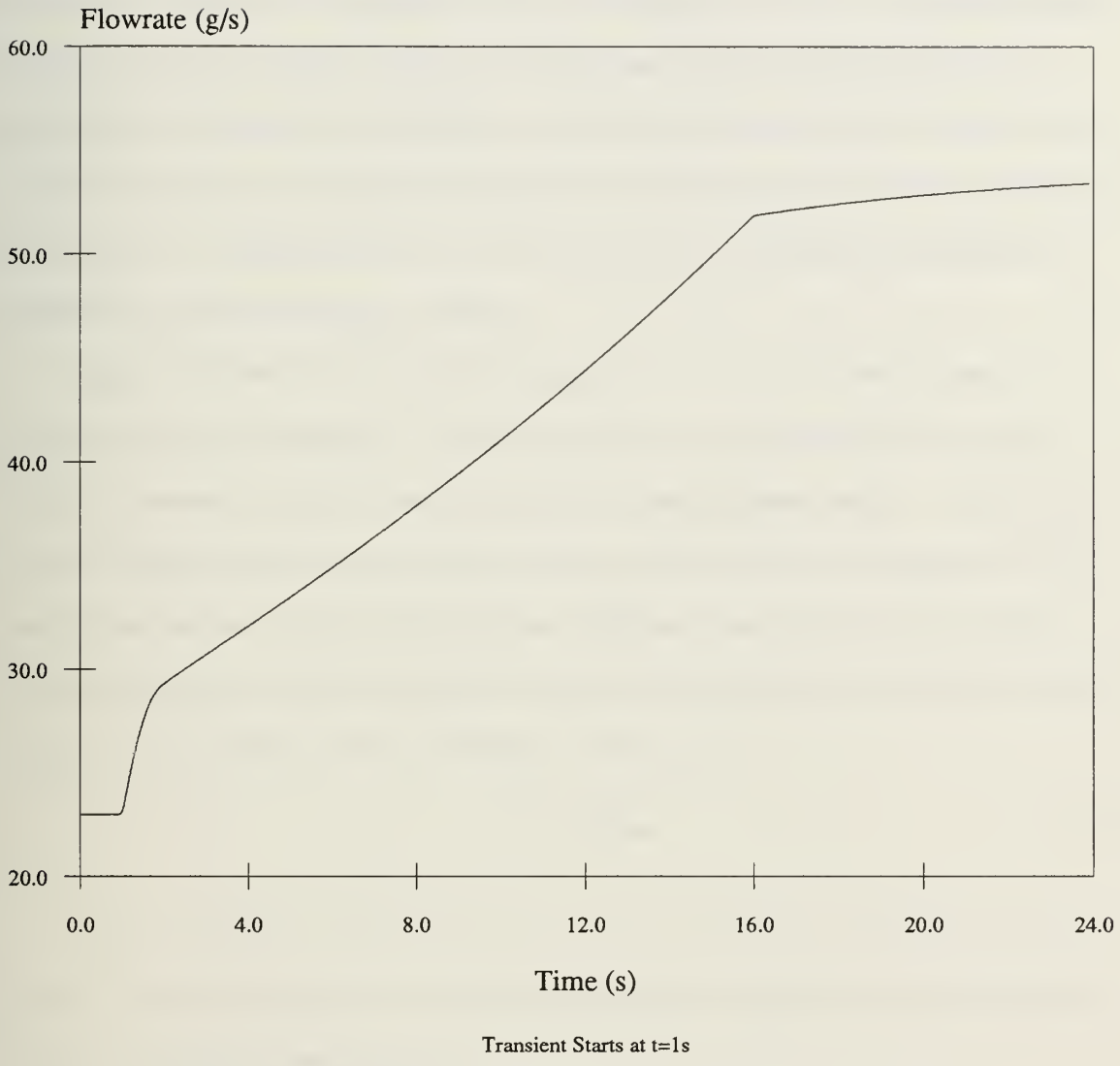


Figure 4.3: Mass Flowrate Response to Combined Transients

4.5 CORE REPRESENTATION BY MULTIPLE FUEL ELEMENTS

As a further demonstration of how the simple model may be used as a design tool, a cluster of 19 fuel elements are arranged into three rings around a center fuel element as presented in Figure 4.4. The center element is designated as O and is surrounded by the A-ring, B-ring, and C-ring. Using a zero order Bessel function of the first kind to approximate the relative power densities of each ring, the simple model was used to calculate the time behavior of each ring of the reactor for mass flowrate, thermal power, and exit temperature. Length was set to 1 m, inlet pressure to 2000 kPa, outlet pressure to 1900 kPa, and inlet temperature to 300 K.

Figure 4.5 shows the mass flowrate response to a 4 second power transient for a single fuel element in each ring. The heat deposition rate is increased from zero to 1 GW/m^3 in element O with the other rings being increased proportionally. As expected, the fuel elements with the lower power densities develop less flow resistance and, as a result, mass flowrate is greater through these elements at the final power level. Mass flowrates specific to each element may be combined to describe the overall mass flowrate response to the transient as shown in Figure 4.6 such that:

$$W_T = \sum_{i=1}^{19} W_i \qquad 4.1$$

Figure 4.7 depicts total thermal power response to the 4 second increase in neutron power. The total thermal power is found by summing the product of mass flowrate and change in enthalpy for each element:

$$\sum_{i=1}^{19} W_i \Delta H_i \qquad 4.2$$

Finally, Figure 4.8 describes exit temperatures for each ring of elements. The average exit temperature represents the temperature of the coolant which is delivered to downstream power plant components and is determined by weighting each exit temperature with the corresponding mass flowrate. Note that the center ring will be operated well above the average outlet temperature which indicates that a limit associated with fuel element material may be reached before any other component (ie, turbine blades).

As a design consideration factor, outlet temperatures may be made more uniform by changing the orificing of the inlet plenum or the thickness of the cold frit for the A, B, and C-rings. This would serve to decrease thermal gradients across regions of the core as well as improving allowable thermal power.

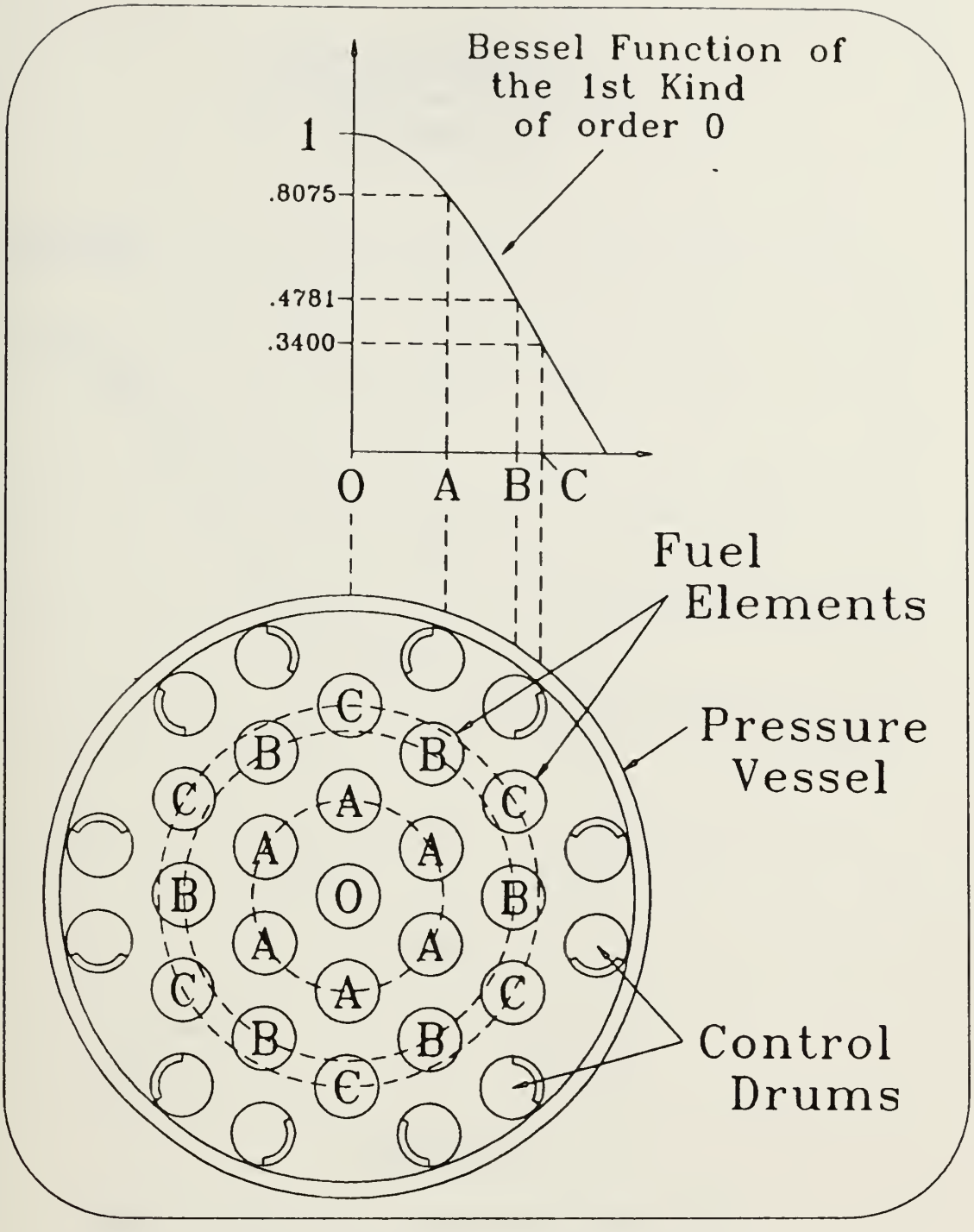


Figure 4.4: Clustered Fuel Element Arrangement for 19 Element PBR

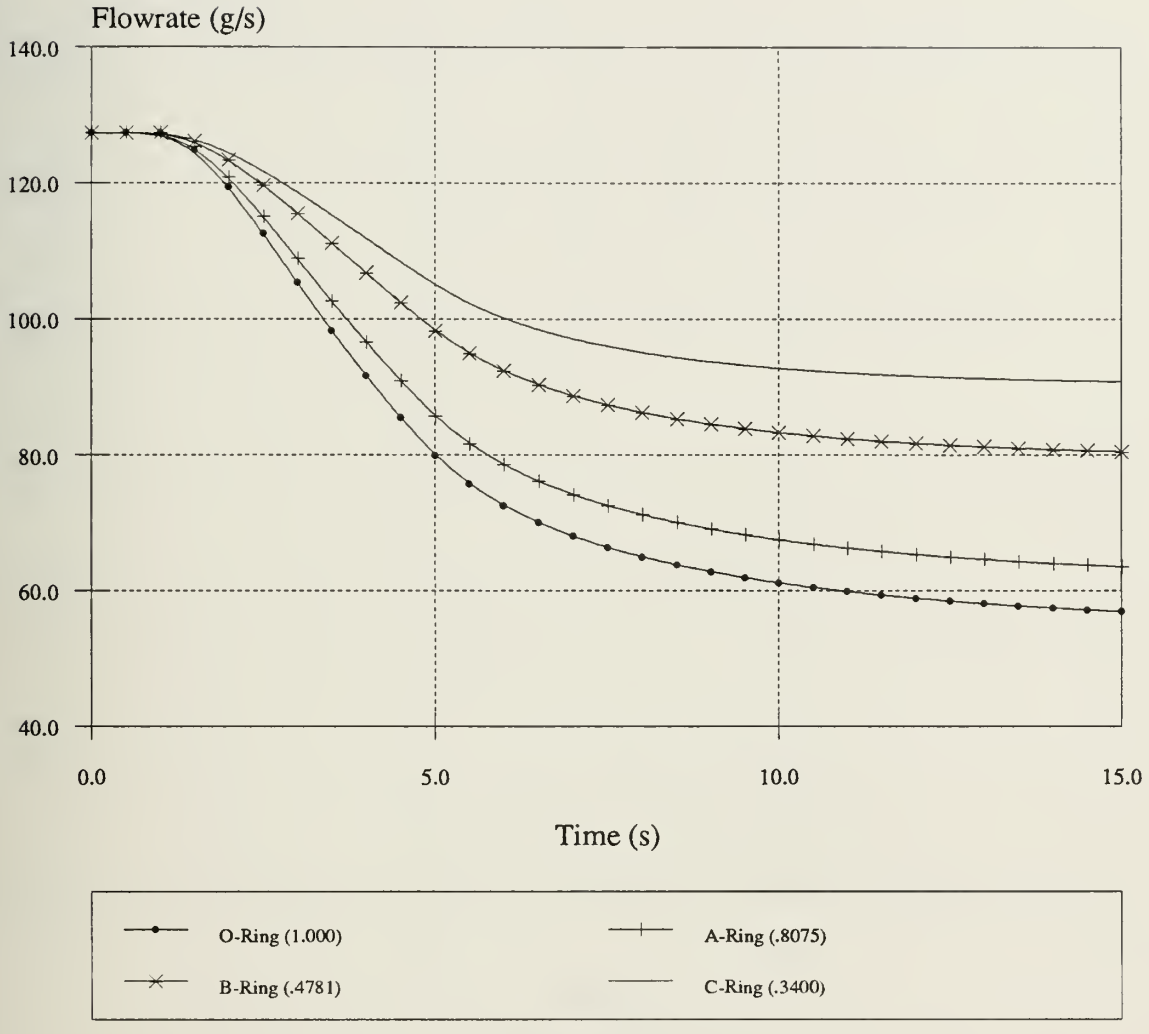


Figure 4.5: Mass Flowrate Response for 19 Element PBR

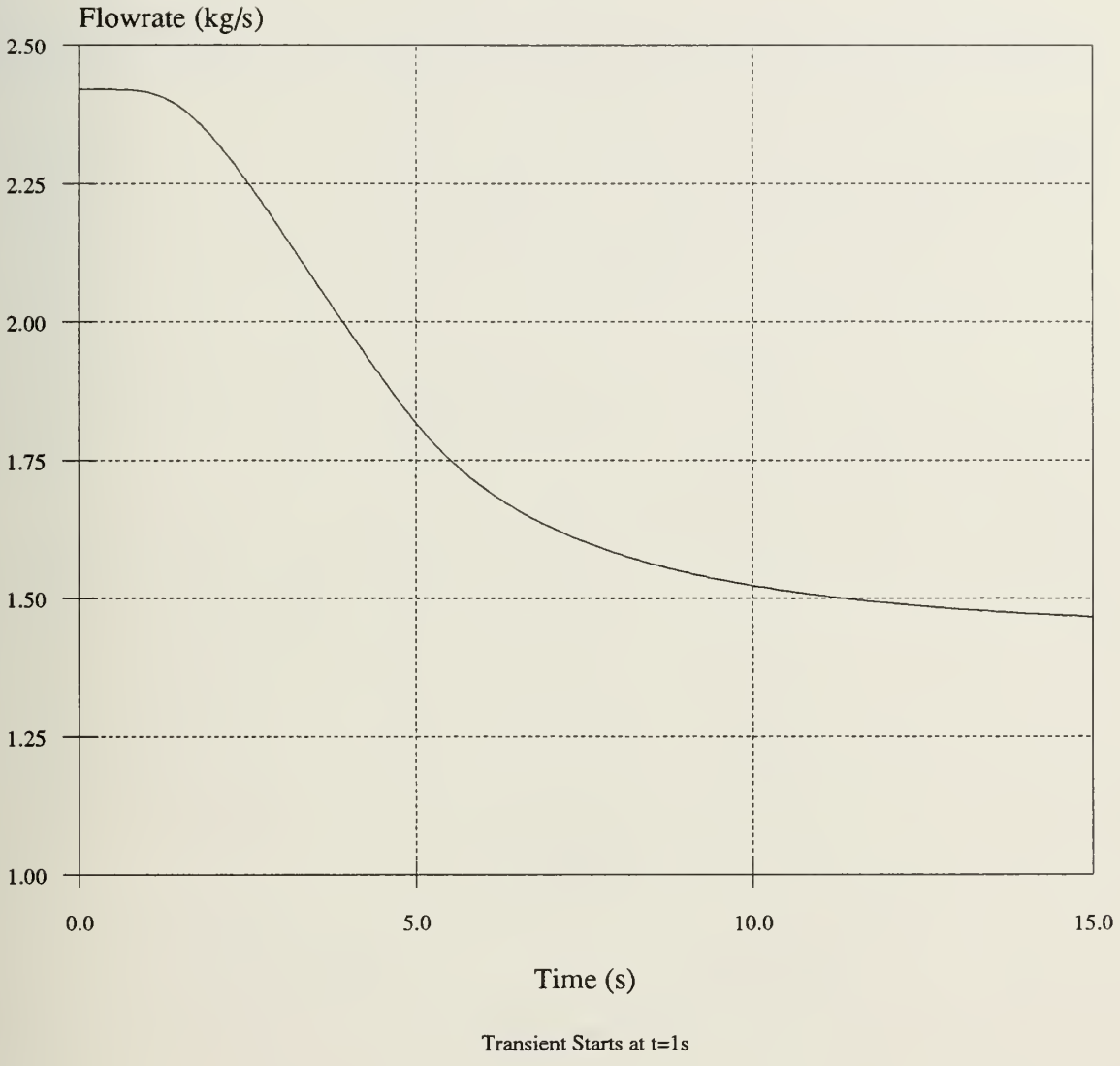


Figure 4.6: Total Mass Flowrate Response for 19 Element PBR

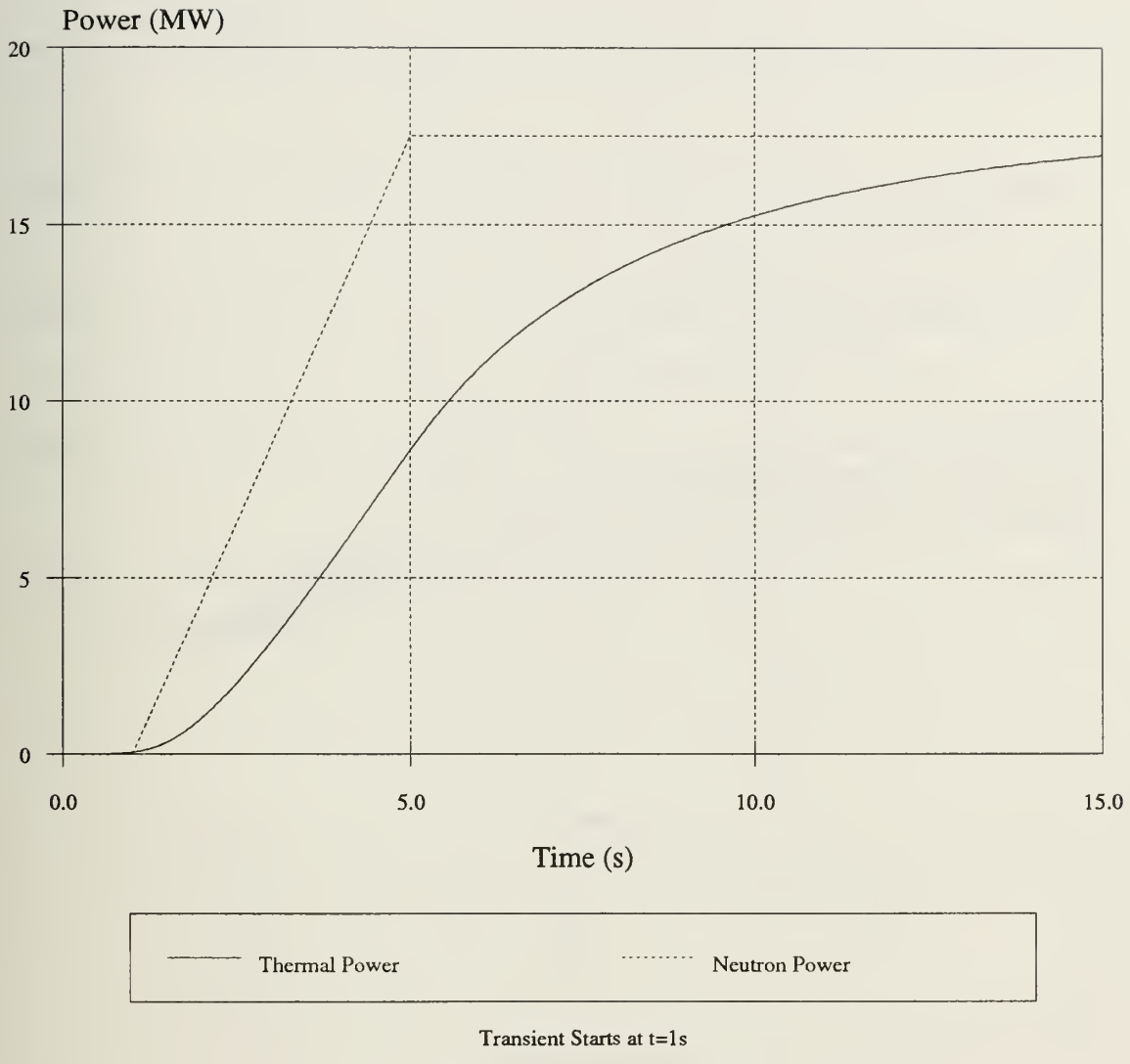


Figure 4.7: Total Power Response for 19 Element PBR

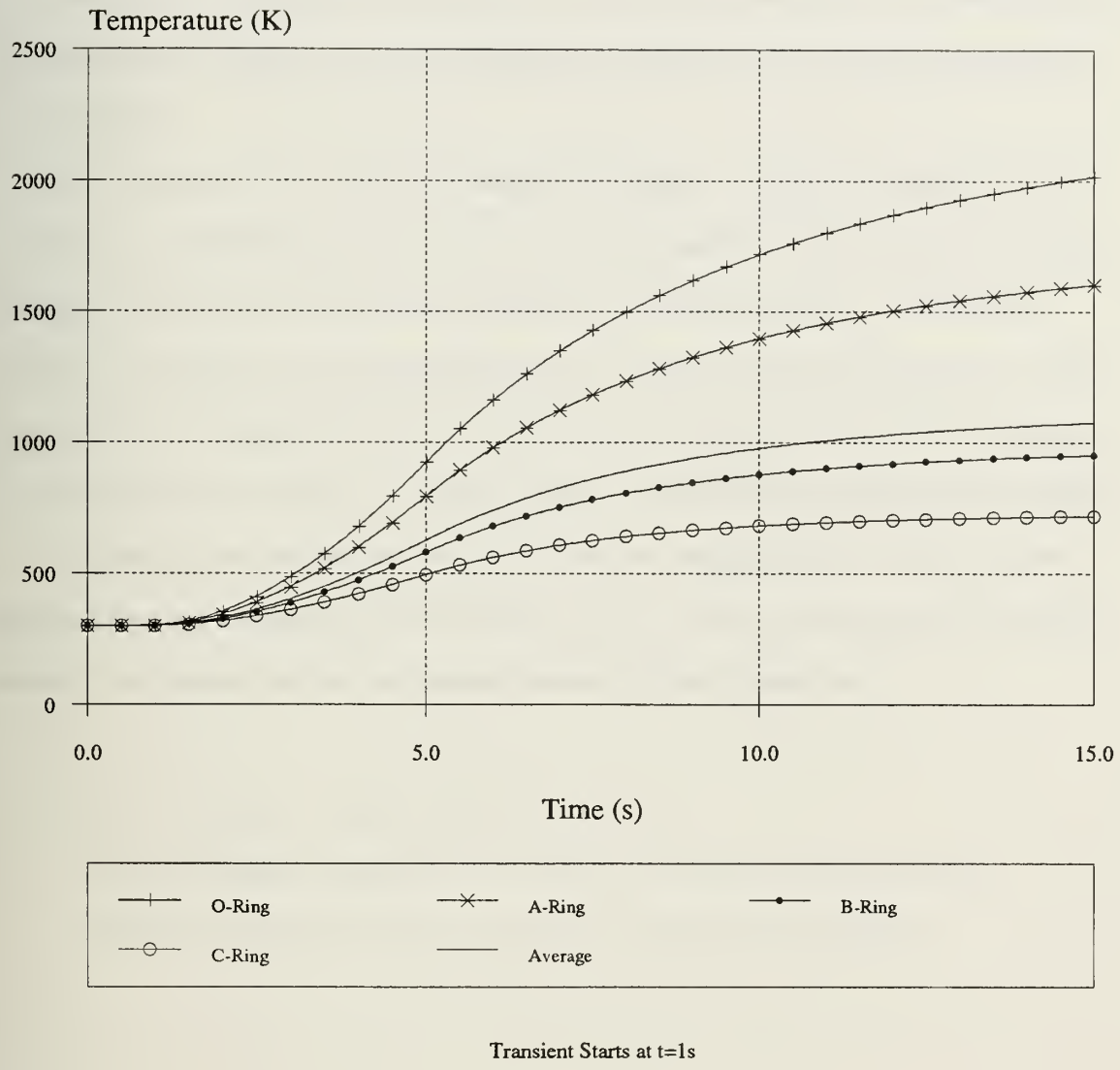


Figure 4.8: Exit Temperature Response for 19 Element PBR

4.6 REAL-TIME CONTROLLER

The simple model takes a step toward development of a real-time model for incorporation in control of a space-based reactor by significantly decreasing the time necessary for computation. Although some information is lost when compared to the data delivered by the reference standard, the overall framework which describes fuel element performance is similar.

As a demonstration of real-time control, the simple model was modified in a manner which suspended the process of writing data to a data file. The calculation time for a transient was less than the transient simulated (for a single element). A 10 second transient, for example, required approximately 3 seconds of computation time for a timestep of 100 ms and approximately 7.5 seconds for a timestep of 10 ms. The calculations were performed on a desktop computer equipped with a 30386 microprocessor (16 MHz).

Although there are reasons that these results are inappropriate for a final control application, they do indicate the plausibility of using a version of this thermal-hydraulic model. The reasons for the continued adjustment of these results are:

- Improvements in computer speed are expected;
- An optimization of calculation techniques has not been performed; and
- Multiple elements and other computations must be done in parallel.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The previous sections describe an investigation to construct a simple model which could effectively analyze the thermal-hydraulic response of a packed particle bed reactor fuel element during steady state and transient conditions. The fuel element was first divided into a solid phase and a gas phase which are modeled separately then coupled for the final solution.

The solid phase is modeled in a single control volume and a lumped parameter approach is used. This approach is essentially the same as the approach used in the reference standard and generates a fuel temperature which is representative of an average fuel temperature instead of a local fuel centerline temperature for instance. Interphase heat transfer is calculated from the product of an overall heat transfer coefficient and the difference between the average fuel temperature and the bulk temperature of the coolant. This interphase heat transfer rate provides the heat input to the coolant as it passes through the fuel particles.

The gas phase is modeled in three control volumes: the inlet plenum combined with the cold frit; the packed particle bed; and the outlet plenum combined with the hot frit. This particular arrangement allows the fuel region to be modeled separately from other components of the fuel element.

Conservation equations for the conservation of energy, mass, and momentum (in the form of the momentum integral equation) are used to ensure a balance when analyz-

ing the fuel element during a transient. Steady state values and transient values (for a 1 s baseline transient from 0 to 2 GW/m^3) are analyzed and then compared to the reference standard.

The calculations resulting from the simple model compare favorably with the reference standard and, because of the simplifications, are able to analyze the same transient significantly faster as well as compute steady state values directly. After eliminating writing to a data file, the simple model is able to achieve a faster-than-real-time thermal-hydraulic analysis of a transient for timesteps larger than 10 ms and an 80386 microprocessor.

The simple model may also be used as a design tool. It may be used on a desktop computer and many different calculations may be accomplished during a reasonable timeframe. Further, the simple model has the capability to accept variations in geometry, fuel composition, and initial/final conditions.

5.2 RECOMMENDATIONS FOR FURTHER STUDY

Development of the simple model is another step in the creation of a real-time, autonomous controller for a packed particle bed reactor. Although this model represents the thermal-hydraulic portion of the controller, it sets the stage for further investigation.

Some areas which would merit further investigation include:

• Calculations obtained by the reference model and the simple model should be compared to experimental results. At the time this investigation concluded, such experimental results were not available.

• A neutonics module needs to be developed which uses the data generated by the thermal-hydraulics module. A real-time controller may then be constructed and tested.

• A model which includes additional power generating system components should be developed. Components, such as a high temperature turbine, turbo-pump, and associated valves and piping could be coupled to the representation of the core provided by the simple model.

• Study a means to adjust the simple model to correct the observed sluggishness (Section C.3)

• Continue to compare the results obtained by the simple model with future models which may construed as a reference standard.

APPENDIX A:

DERIVATION OF THE GENERAL CASE FOR SPATIAL ACCELERATION IN A DUCT WITH NON-CONSTANT AREA AND NON-CONSTANT DENSITY

A.1 BACKGROUND

Spatial acceleration term in a momentum balance equation are usually presented as the pressure change in a duct caused by a change in density or by a change in cross sectional area. In the former case, the pressure change in a duct caused by a change in density while maintaining cross sectional area constant is expressed as:

$$P_1 - P_2 = \rho_2 v_2^2 - \rho_1 v_1^2 \qquad \text{A.1}$$

In the latter case, a change in pressure associated with spatial acceleration in a duct with constant density but with a change in duct area, is expressed as:

$$P_1 - P_2 = \left(\frac{1}{A_2^2} - \frac{1}{A_1^2} \right) \frac{W^2}{2\rho} \qquad \text{A.2}$$

where W = Mass Flowrate

In the case of the packed particle bed reactor fuel element, the working gas is subjected to a combination of cross sectional area change and density change as it travels in the radial direction. Although there are integration methods available which could accurately describe this fluid behavior, it would necessitate dividing the particle bed into many control volumes. For the simple model, it is desirable to maintain the particle bed

within a single control volume. Because neither equations A.1 nor A.2 accurately describe the thermal-hydraulics associated with variable density gas through the particle bed, a general case derivation must be formulated for the simplified model.

A.2 THE GENERAL CASE

Figure A.1 shows the general case in which the cross sectional area in the duct and the density of the gas is allowed to change as the gas travels through the duct. The objective of this derivation is to formulate an expression which will satisfy the limiting cases described in equations A.1 and A.2 above. To do this we must first define an average pressure $\overline{P}$ as a combination of P_1 and P_2:

$$\overline{P} = aP_1 + (1-a)P_2 \qquad \text{A.3}$$

Where a and $(1-a)$ represent fractional constants between 0 and 1.

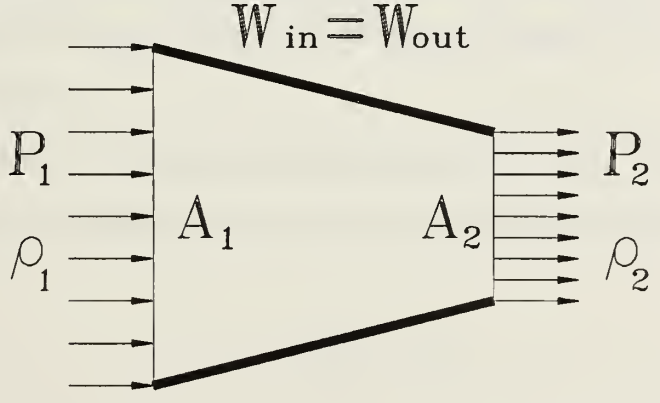


Figure A.1: Duct with changing area and changing density.

To begin the derivation, a momentum balance analysis must be performed on the control volume shown in Figure A.1. $\overline{P}$ becomes important at this point because it is allowed to act at some point within the duct on an area which is defined as the difference between A_1 and A_2 when projected onto the vertical plane. This then results in the momentum balance equation:

$$P_1A_1 - P_2A_2 + \overline{P}(A_2 - A_1) = \rho_2 v_2^2 A_2 - \rho_1 v_1^2 A_1 \qquad \text{A.4}$$

Substituting the definition of $\overline{P}$ into equation A.4 and collecting terms gives:

$$(P_1 - P_2)\{A_1 + a(A_2 - A_1)\} = \rho_2 v_2^2 A_2 - \rho_1 v_1^2 A_1 \qquad \text{A.5}$$

By inspection, equation A.5 reduces to equation A.1 when A_1 and A_2 are set equal to one another. When the duct inlet area and outlet areas remain the same, the term $a(A_2 - A_1)$ is eliminated resulting in the limiting case shown in equation A.1. For the other case in which the solution is known, the fractional constant a is solved while density remains constant.

For the case in which density is held constant, the Bernolli relationship is used to find an expression for the pressure change on the left hand side of equation A.5:

$$P_1 - P_2 = \frac{\rho}{2}(v_2^2 - v_1^2) \qquad \text{A.6}$$

and, because mass is conserved in a steady state condition, constant c is used to demote a ratio of areas and hence a ratio of velocities:

$$c = \frac{A_1}{A_2} \qquad \text{A.7}$$

Substituting the expression for $P_1 - P_2$ and letting $v_2 = c\,v_1$ results in the following expression:

$$\frac{1}{2}(c^2 - 1)\{A_1 + a(A_2 - A_1)\} = A_1(c - 1) \qquad \text{A.8}$$

The fractional constant a can now be solved by dividing both sides by $A_2(c - 1)$:

$$\begin{aligned} (c+1)\{c + a(1-c)\} &= 2c \\ c(c+1) + a(c+1)(1-c) &= 2c \\ a(c+1)(c-1) &= c(c-1) \end{aligned} \qquad \text{A.9}$$

and upon completion of the algebra, an expression is derived for the fractional constant a:

$$a = \frac{c}{c+1}$$

or

$$a = \frac{A_1}{A_1 + A_2} \qquad \text{A.10}$$

and

$$1 - a = \frac{A_2}{A_1 + A_2} \qquad \text{A.11}$$

Equations A.10 and A.11 are then substituted into equation A.3 for an expression for $\overline{P}$:

$$\overline{P} = \left(\frac{A_1}{A_1 + A_2}\right)P_1 + \left(\frac{A_2}{A_1 + A_2}\right)P_2 \qquad \text{A.12}$$

Finally, substituting this new expression for $\overline{P}$ into equation A.4 and solving for the pressure change provides an expression for the general case:

$$P_1 - P_2 = \left(\frac{A_1 + A_2}{2A_1A_2}\right)Wv_2 - \left(\frac{A_1 + A_2}{2A_1A_2}\right)Wv_1 \qquad \text{A.13}$$

APPENDIX B:

SUMMARY OF EQUATION FOR HYDRAULIC ANALYSIS OF A PBR FUEL ELEMENT

1. Constitutive Relationship:

$$\Delta P_T = R_T W^2$$

ΔP_T = Total pressure drop across element

R_T = A term representing the total resistance to fluid flow

W = Mass flowrate through the element

2. Momentum Integral Equation:

MINET CODE (Reference V-2)

$$I_m \frac{dW}{dt} = P_i - P_o + F_{PressEquivalent}$$

$$F_{PressEquivalent} = \Sigma(F_g + F_f + F_a + F_v + F_p + F_k)$$

$$F_{PressEquivalent} = \Sigma(R_g + R_f + R_a + R_v + R_p + R_k)W^2$$

$$R_T = \Sigma(R_g + R_f + R_a + R_v + R_p + R_k)$$

3. Pressure Relationships for Inlet and Outlet Plenums:

$$F_f = -f_f\left(\frac{L_p}{D_{eq}}\right)\frac{W^2}{2\rho A^2}$$

$$f_f = 0.138Re^{-0.151}$$

L_p = The half-length of the plenum

$$F_a = -\left(\frac{1}{A_o^2} - \frac{1}{A_i^2}\right)\frac{W^2}{2\rho}$$

$$F_k = -K_k\frac{W^2}{2\rho A_{small}^2}$$

$K_k = K_e for expansions$

$K_k = K_c for contractions$

$$K_e = (1-\mathrm{B})^2$$

$$K_c = \frac{1}{2}(1-\mathrm{B})^2$$

$$\mathrm{B} = \frac{smaller area}{larger area}$$

$$F_M = -\Theta\Sigma(F_{M-radial} + F_{M-axial})$$

$$F_{M-axial} = \frac{W^2}{\rho A_{axial}^2}$$

and

$$F_{M-radial} = \frac{W^2}{\rho A_{radial}^2}$$

Θ = .95 for Inlet Plenum; 1.10 for Outlet Plenum

4. Pressure Relationships for Packed Particle Bed (also Cold Frit):

(Ergun Relationship)

$$F_{Pbed} = L_b\left\{\frac{150\mu\vec{v}_o(1-\varepsilon)^2}{D_p^2\varepsilon^3}\right\} + L_b\left\{\frac{1.75\rho\vec{v}_o \mid v \mid (1-\varepsilon)}{D_p\varepsilon^3}\right\}$$

or:

$$F_{Pbed} = \left\{L_b\left(\frac{150\mu(1-\varepsilon)^2}{D_p^2\varepsilon^3\rho A_o W_o}\right) + L_b\left(\frac{1.75(1-\varepsilon)}{D_p\varepsilon^3\rho A_o^2}\right)\right\}W^2$$

$$F_a = \left\{\frac{1}{\rho_2 A_2}\left(\frac{A_1+A_2}{2A_1A_2}\right) - \frac{1}{\rho_1 A_1}\left(\frac{A_1+A_2}{2A_1A_2}\right)\right\}W^2$$

APPENDIX C:

MODEL VALIDATION

C.1 STEADY STATE

C.1.1 General

In order to provide one set of validations for the simple model, it may be compared to the reference standard. Steady state and transient results for both models are compared graphically in the following subsections. Values for reference standard parameters were drawn directly from T-1 and values for the simple model were obtained from the steady state and transient source codes attached as Appendices C and D. Also, each model used the Sandia National Laboratory PIPE experiment fuel element as the basis for its overall configuration.

C.1.2 Mass Flowrate

At a steady state condition, coolant mass flowrate will vary as a function of the particle bed power density. That is, the mass flowrate will adjust itself according to the resistance to flow produced by the element as a whole which will be a function of temperature and pressure. Figure C.1 shows how mass flowrate varies as a function of power density. Keep in mind that the average temperature of the fuel and the exit temperature increase with an increase in power density. Also shown in Figure C.1 are the mass flowrates calculated by the reference standard at 0, 1, and 2 GW/m^3. There is surprisingly little difference between the simple model and the reference standard at these power densities.

Also note that the exit plenum Θ is adjusted from 2.66 to 1.10 as discussed in section 3.62. This gives a change in calculated flow (at 2 MW/m_3) from 31 g/s (without the adjustment) to 37 g/s ($\approx$15% difference). The adjustment, however, did enable the behavior of the whole curve to be well represented.

At this point, it must be noted that the savings in computer time is substantial. The time required for the reference standard to obtain the three data points shown on Figure C.1 is approximately 24 hours (8 hours per point) while the time required to generate the same data using the simple model is approximately 12 seconds. Further, the simple model is able to provide many more data points in the region of interest.

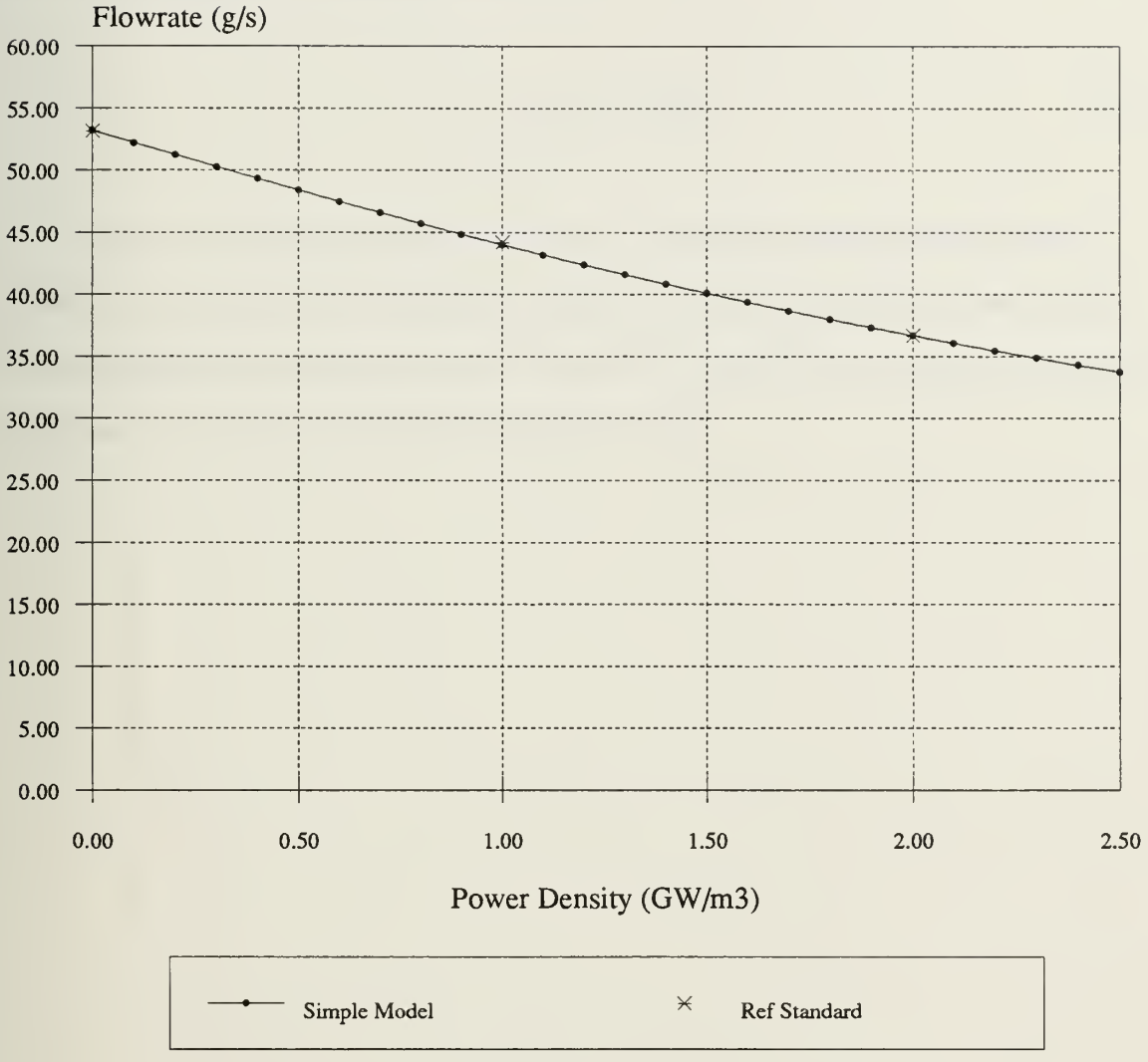


Figure C.1: Steady State Mass Flowrate vs Power Density

C.1.3 Power

The rated output of the fuel element may be measured as the thermal power of the fuel element. This is calculated by multiplying the mass flowrate through the element by the change in enthalpy across the fuel element:

$$Power = W\Delta H \qquad \text{C.1}$$

Figure C.2 shows the relationship between fuel element thermal power to power density and, as expected, the relationship is linear in nature and expresses a heat balance. The calculated value of thermal power at 2 MW/m^3 is plotted for the reference model and corresponds to the value calculated by the simple model.

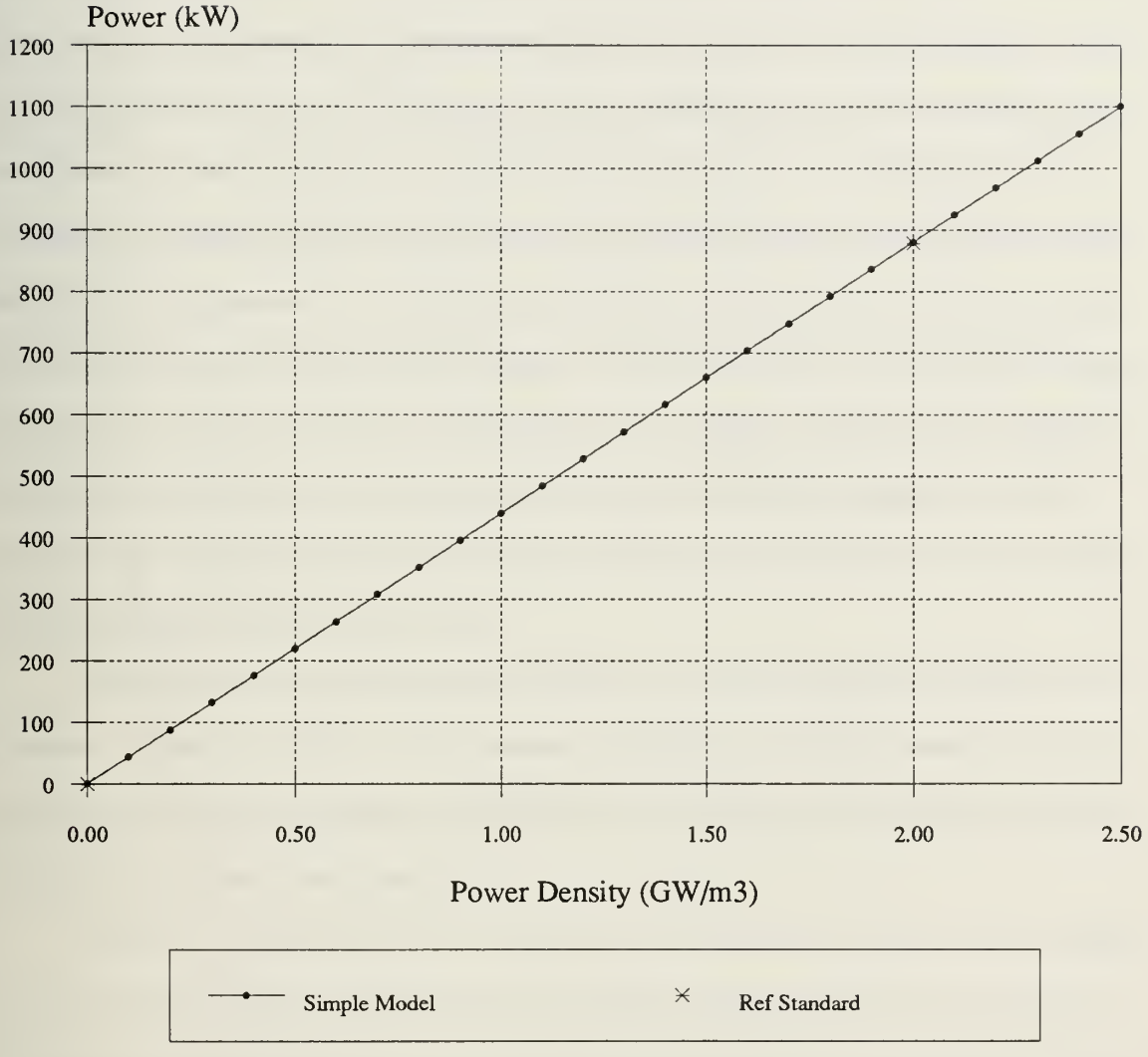


Figure C.2: Steady State Thermal Power vs Power Density

C.1.4 Exit Temperature

Figure C.3 shows how the temperature of the coolant exiting the fuel element varies with power density. Because the exit plenum is modeled as a single control volume, the exit temperature represents an average temperature of the coolant exiting the element and does not take into account any temperature differences which may occur axially. The reference standard has five axial subdivisions in the outlet plenum, each of which may have a different coolant temperature. Because of this axial separation and the ability of the reference standard to designate different axial fuel distributions, the temperatures calculated in the exit plenum for the reference standard are higher at the lower axial position and decreases as the coolant exits the plenum. The simple model can not make this same representation and, as such, exit temperatures may be slightly different between the two models even though mass flowrates, power output and enthalpy changes may be the same. In this case, values for mass flowrate at various power density levels are near reference standard values but are not exact.

The reference temperature plotted in Figure C.3 is the average outlet plenum temperature at 2 and 2.1 MW/m^3. This represents a difference of approximately 2.5% and should be considered acceptable for initial fuel element design but final analysis should be made by the more detailed model.

Velocity is measured at the exit of the plenum. Because exit velocity of the coolant is inversely proportional to the density of the coolant, and hence directly proportional to the temperature of the coolant, it is not surprising to see the reference standard exit velocity fall below the curve generated by the simple model. Exit velocity is shown as a function of power density in Figure C.4.

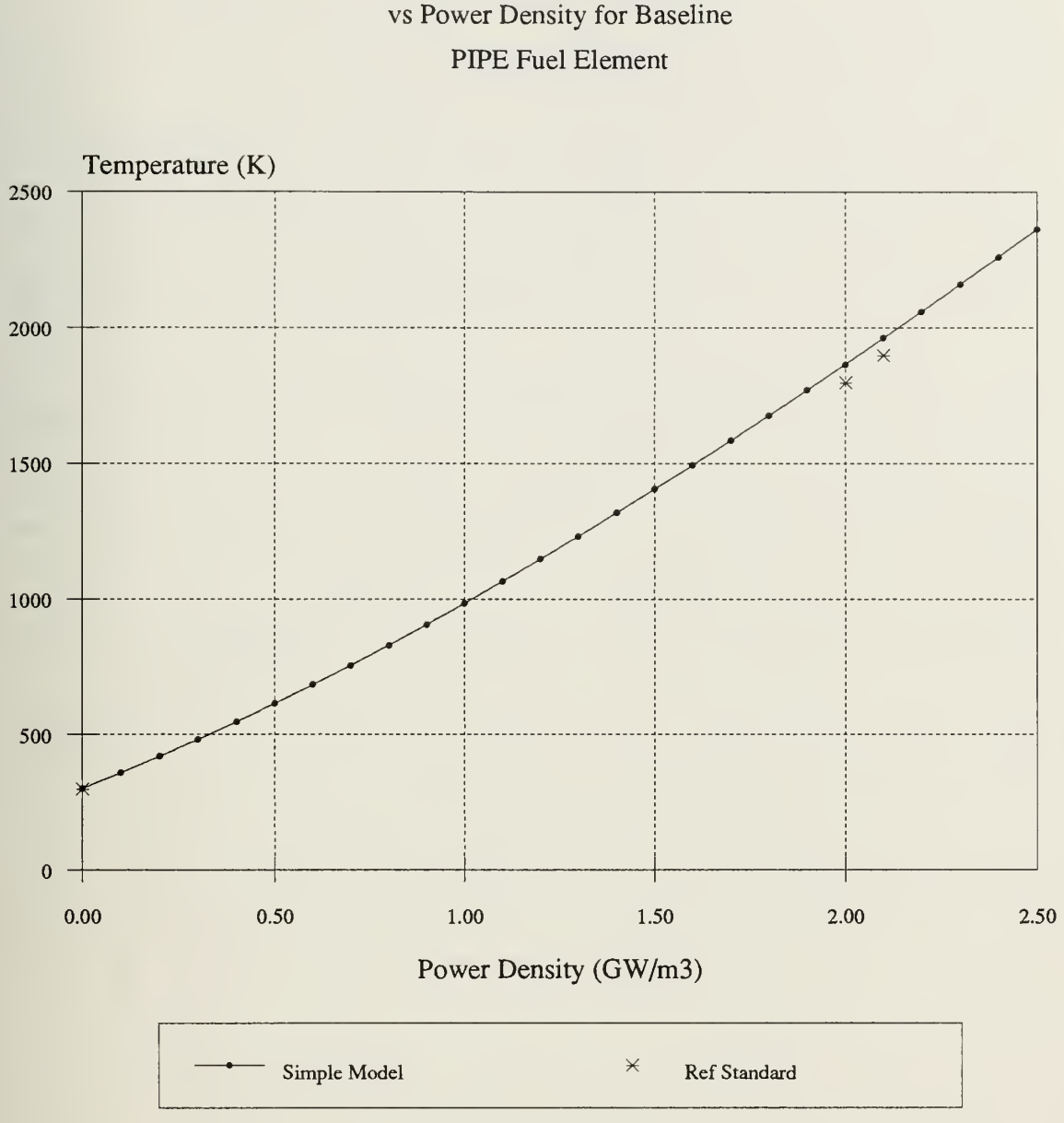


Figure C.3: Steady State Exit Temperature vs Power Density

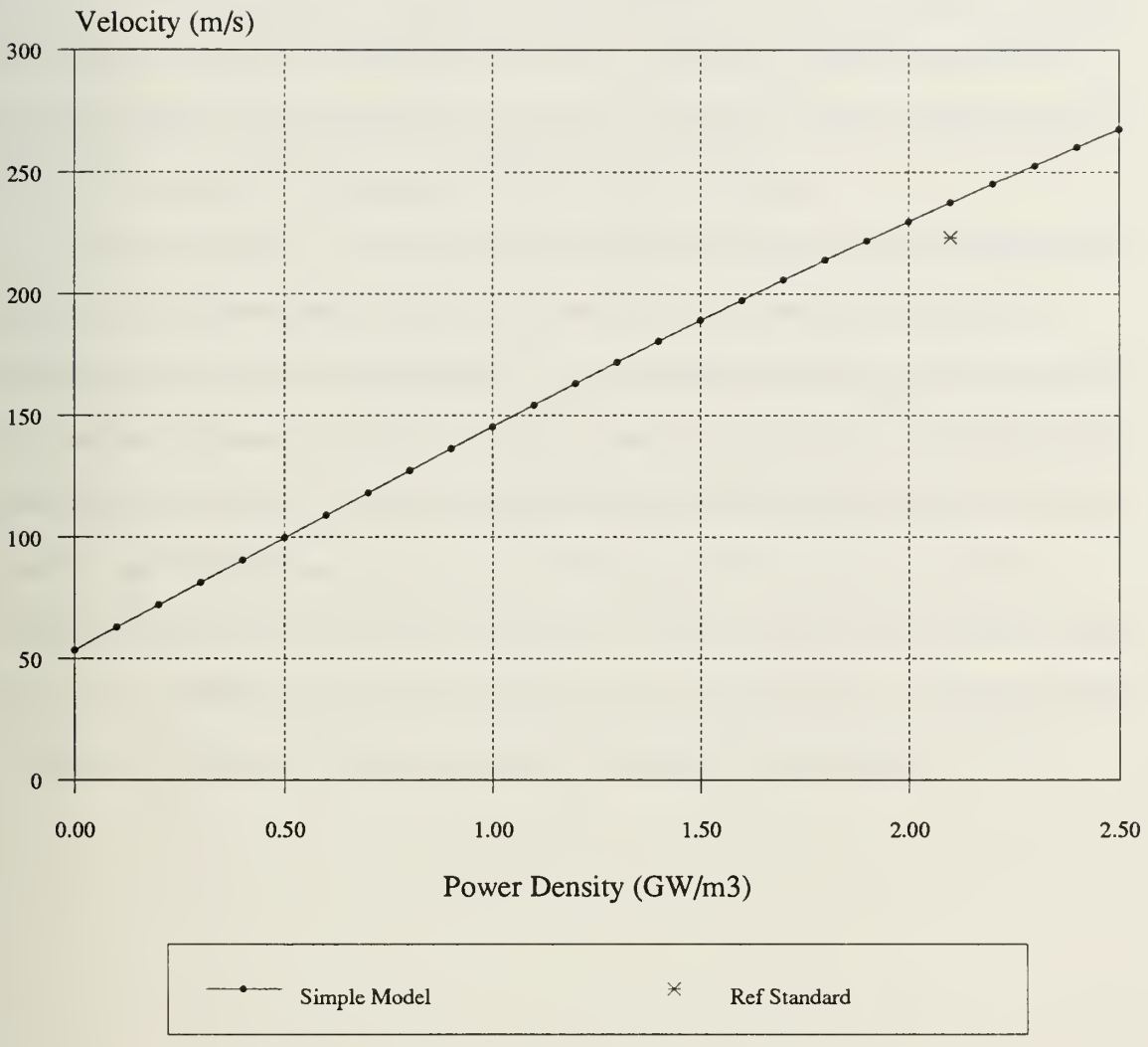


Figure C.4: Steady State Exit Velocity vs Power Density

C.1.5 Pressure

The calculations of sections C.1 are based on constant inlet pressure and constant outlet pressure. As power increases, the total flow through the fuel element drops and the fraction of the total pressure drop increases in the control volumes at the higher temperatures (lower densities). C.5 shows these fraction by giving the calculated pressures at points located at the inlet and outlet of the element and at the inlet and outlet of the fuel. This also represents the boundaries of the three control volumes.

As mentioned in T-1, the inlet plenum and the cold frit dominate the resistance at 0 GW/m^3 but as power density (and exit temperature) is increased, the outlet plenum assumes a greater share of the resistance. The resulting pressure drops across each control volume are shown in Figure C.6. When totaled, the sum of the pressure drops will remain a constant 85 MPa, which is established as an initial condition. Figure C.6 also shows values for pressure drop across the outlet plenum in the reference standard. Although values at 0 and 1 GW/m^3 are near to the values produced by the simple model, the value at 2 MW/m^3 seems to be inconsistent. This inconsistency is unexplained and is judged to be unimportant when the consistent behavior of other variables is noted.

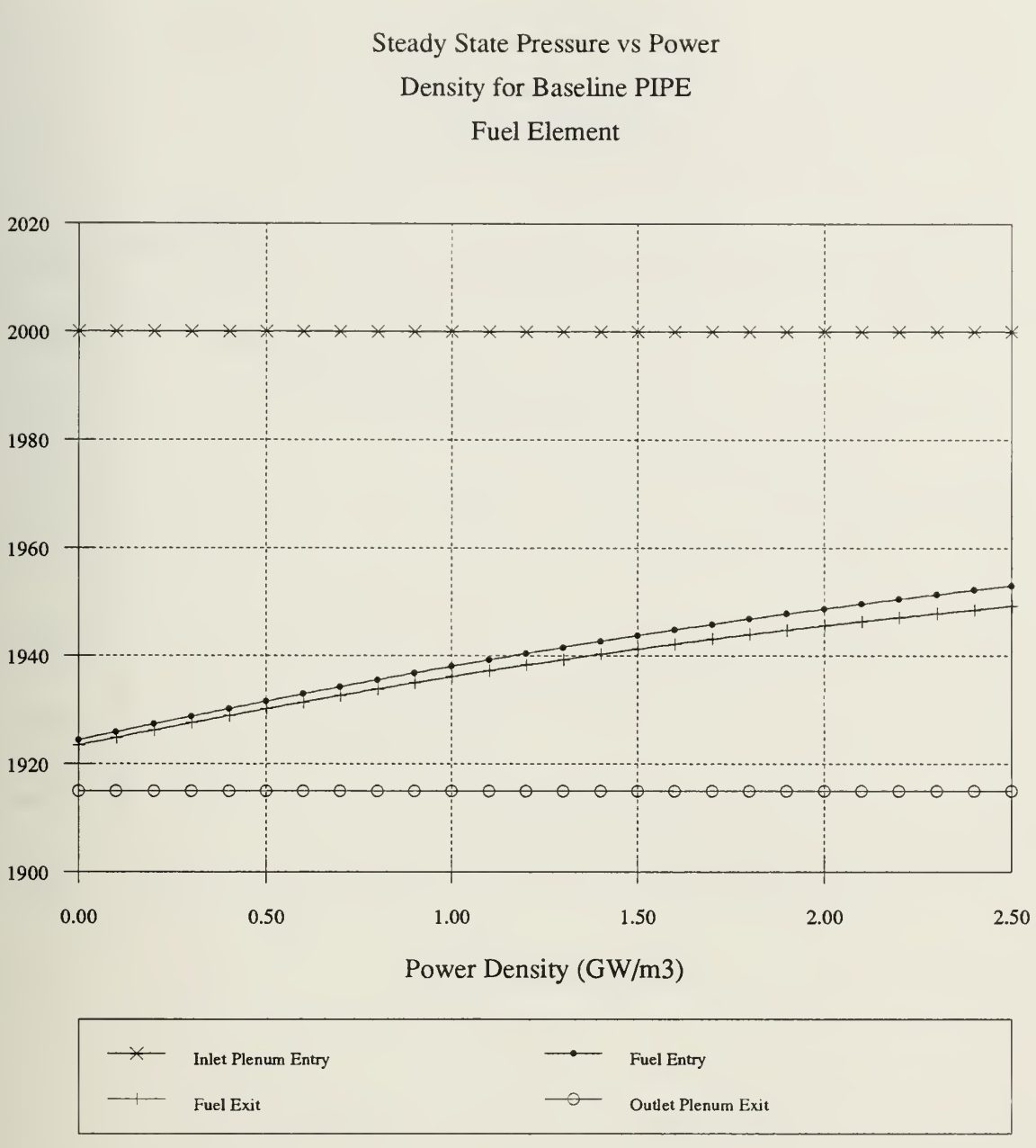


Figure C.5: Steady State Pressures vs Power Density

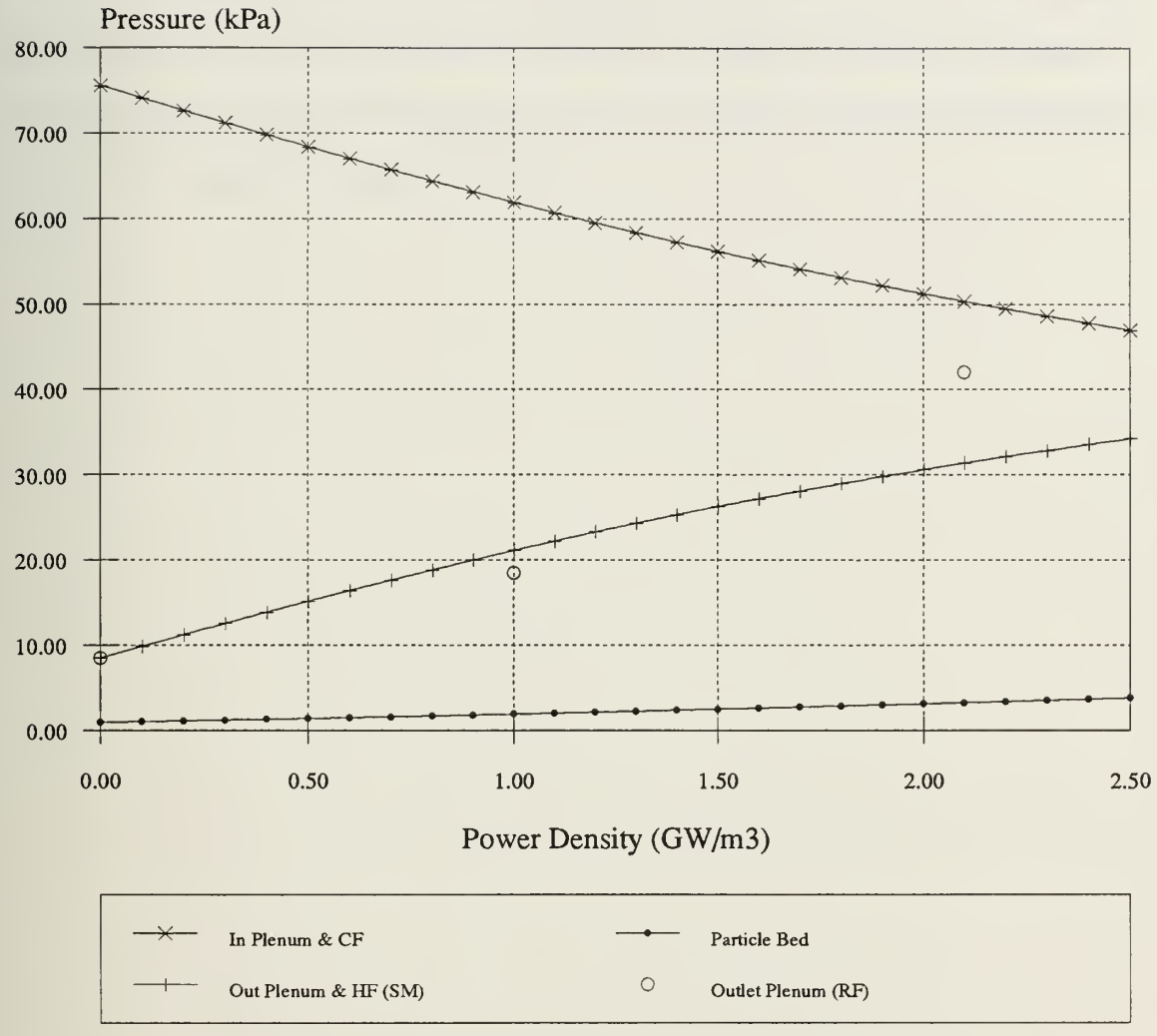


Figure C.6: Steady State Pressure Drops vs Power Density

C.2 NULL TRANSIENTS

As a test for validating the simple model, null transients were analyzed at 0, 1, and 2 MW/m^3 and subsequent mass flowrate responses were examined. Figures C.7, C.8 and C.9 show that there is a smooth behavior throughout the transient and that there is negligible difference between the mass flowrate before the transient and the mass flowrate after the transient. Each of the transients start at .01 seconds and the scale for flowrate on the y-axis is expanded for better viewing.

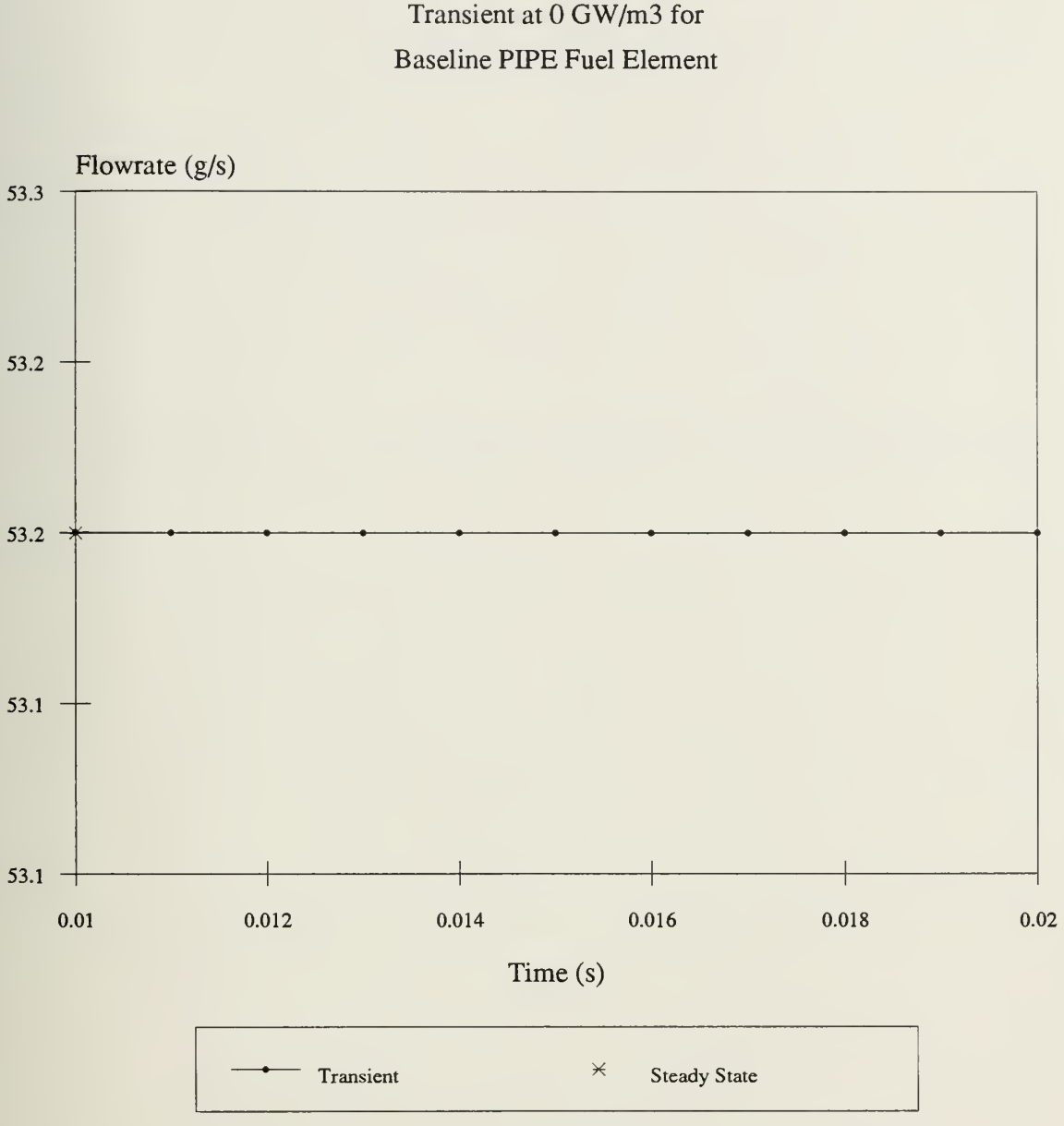


Figure C.7: Mass Flowrate Response to Null Transient at 0 GW/m^3

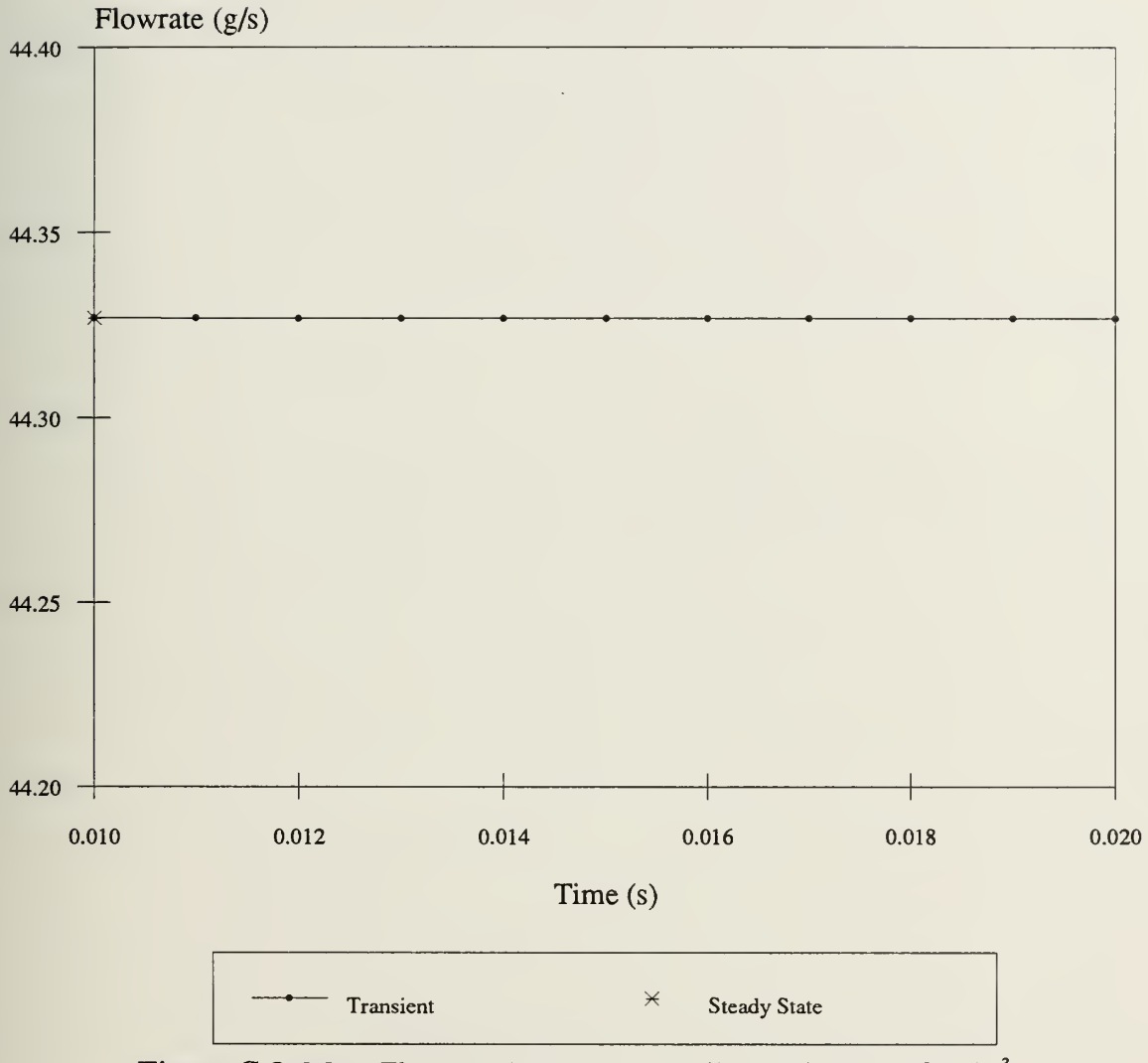


Figure C.8: Mass Flowrate Response to Null Transient at 1 GW/m^3

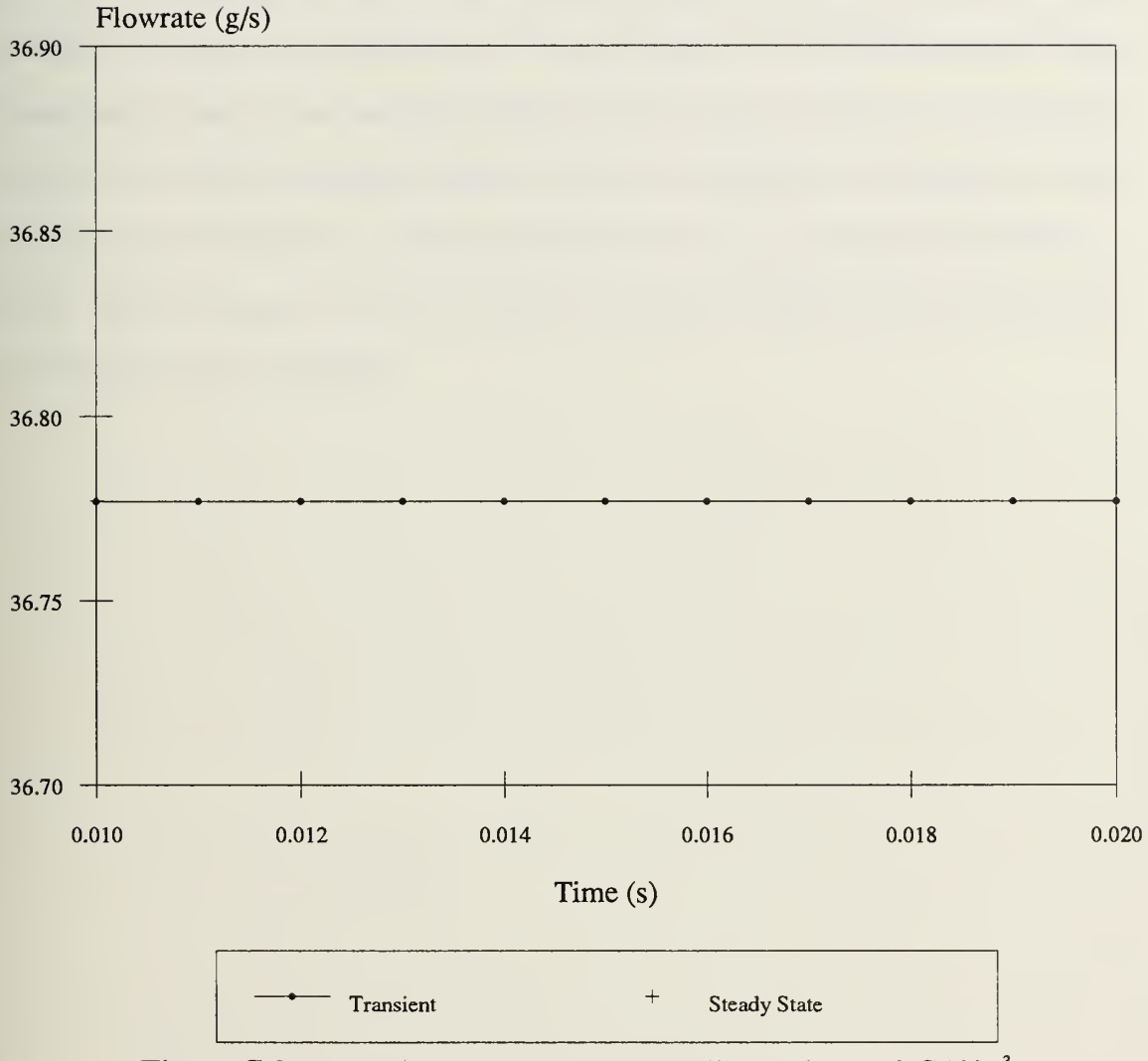


Figure C.9: Mass Flowrate Response to Null Transient at 2 GW/m^3

C.3 TRANSIENTS

In Figures C.10, C.11, and C.12, calculations for a 1 second transient from 0 GW/m^3 to 2 GW/m^3. Calculations for the simple model for mass flowrate, thermal power, and exit temperature are compared to the reference standard. As can be seen, the reference standard reaches the final set of values quicker than the simple model. Adjustment could be made to correct the sluggishness of the simple model (e.g. by increasing the fuel-to-coolant heat transfer parameter and/or by artificially decreasing the mass of coolant in the fuel element). These adjustments have not been attempted in this thesis study. Both the simple model and the reference standard respond quickly; both end at essentially the same condition.

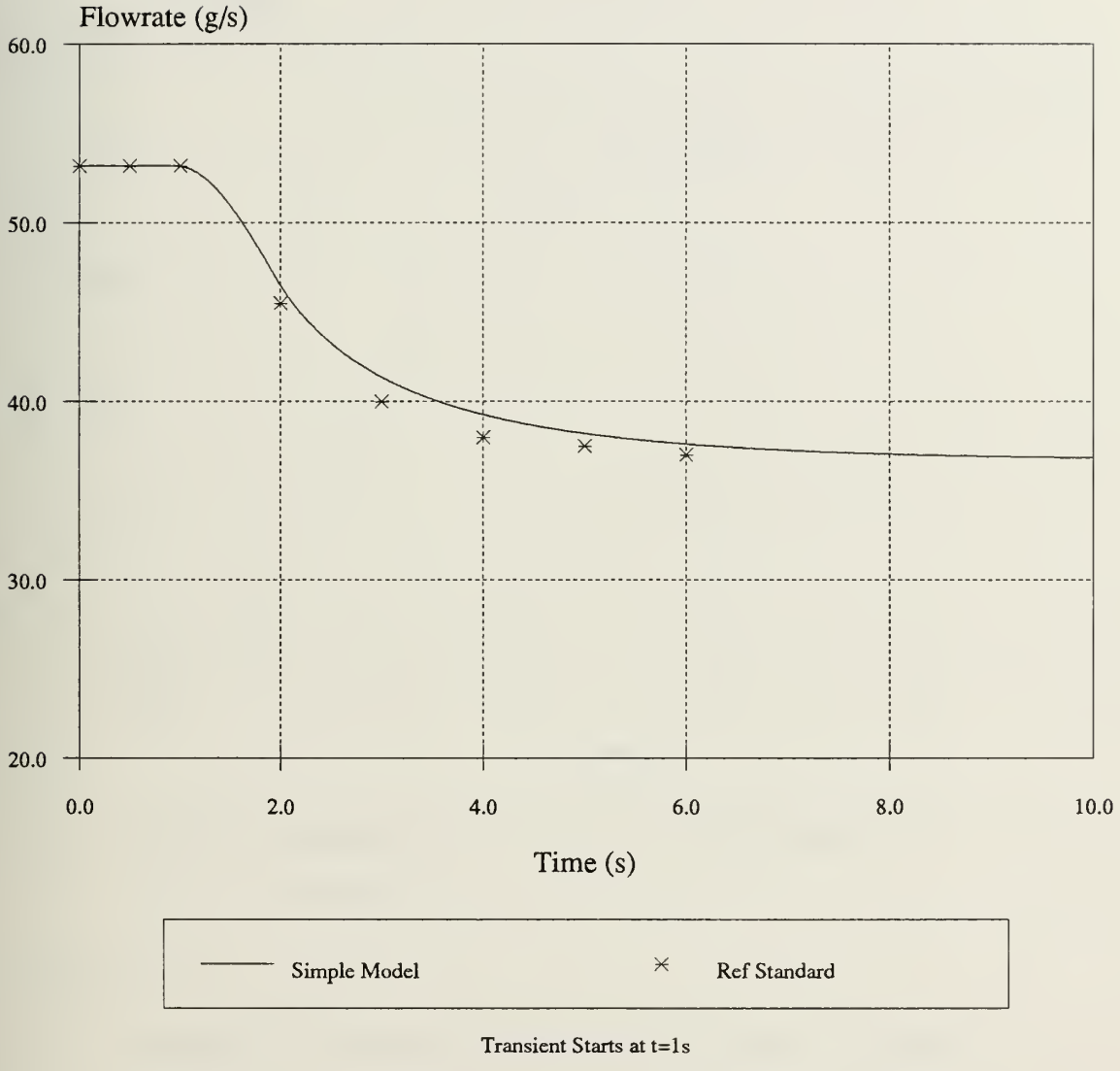


Figure C.10: Mass Flowrate Response to Baseline 1 s Transient

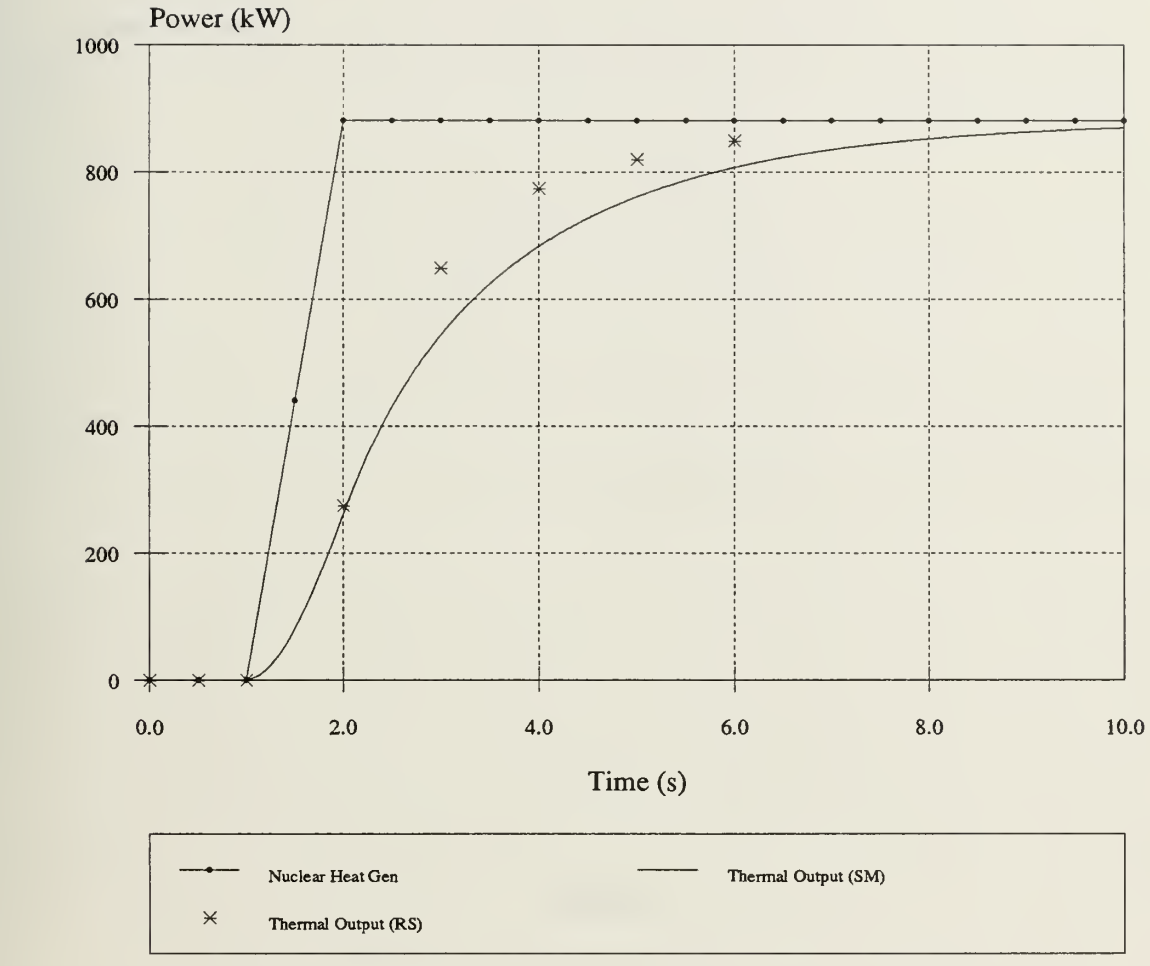


Figure C.11: Thermal Power Response to Baseline 1 s Transient

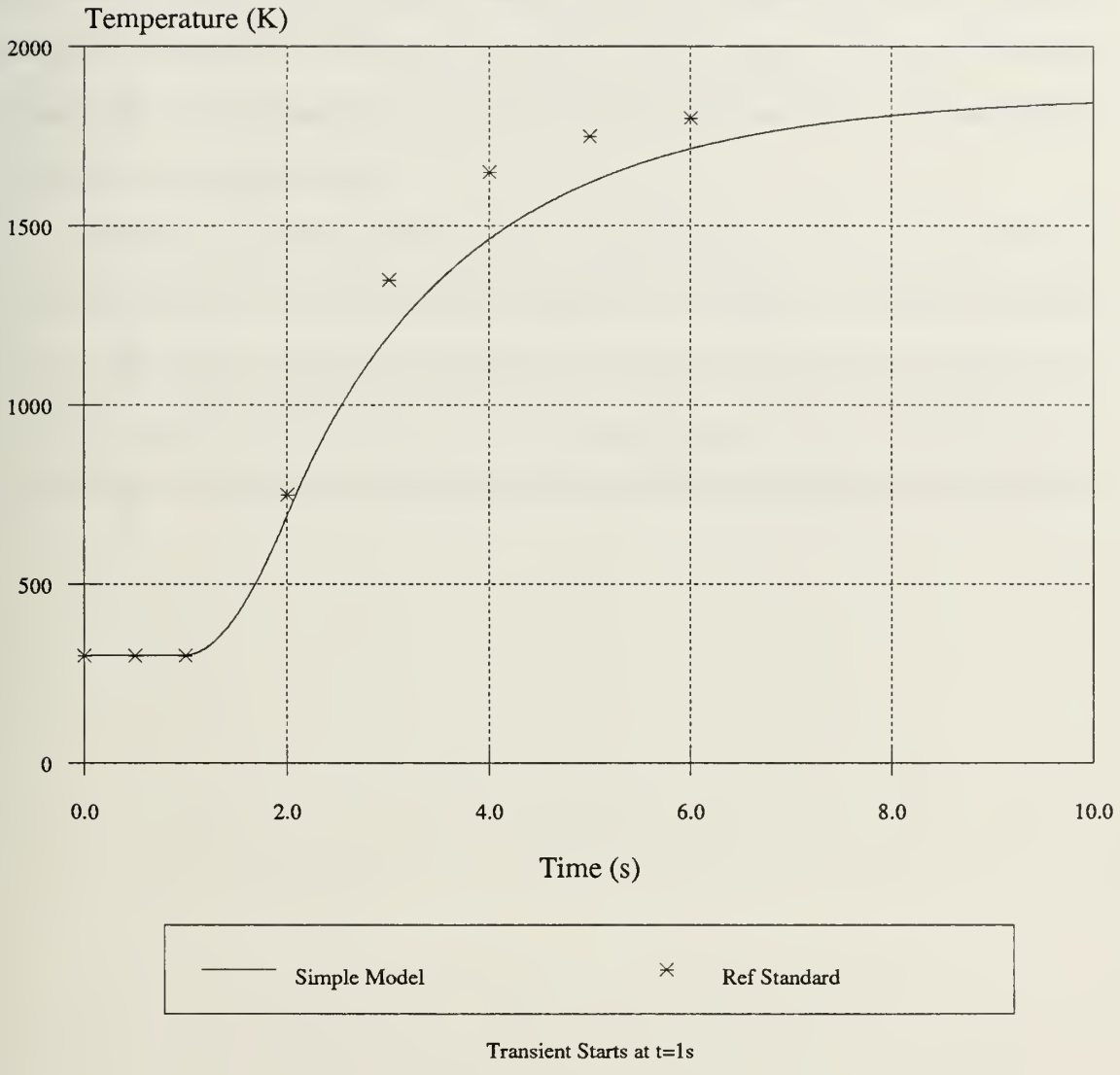


Figure C.12: Exit Temperature Response to Baseline 1 s Transient

C.4 NUMERICAL STABILITY

Accuracy of the solution of the control volume balance equations is illustrated in Figure C.13. That is, consider that the balance equations are written in terms of ordinary differential equations in time. Then consider solving these equations (as in the simple model) by discretizing the equations in the time variable. Figure C.13 gives the results related to the errors generated in this discretization. It does not address the question of discretization in space, however.

The balance equations were discretized using timesteps of 100 ms, 10 ms, and 1 ms. All of the curves are very close and do not show much variation. The plot of mass flow-rate corresponding to timesteps of 1 ms and 10 ms seem to represent the solution of the differential equations better than the curve corresponding to 100 ms. The results for 100 ms indicate an acceptable error (less than 5%) and could be used for faster calculations.

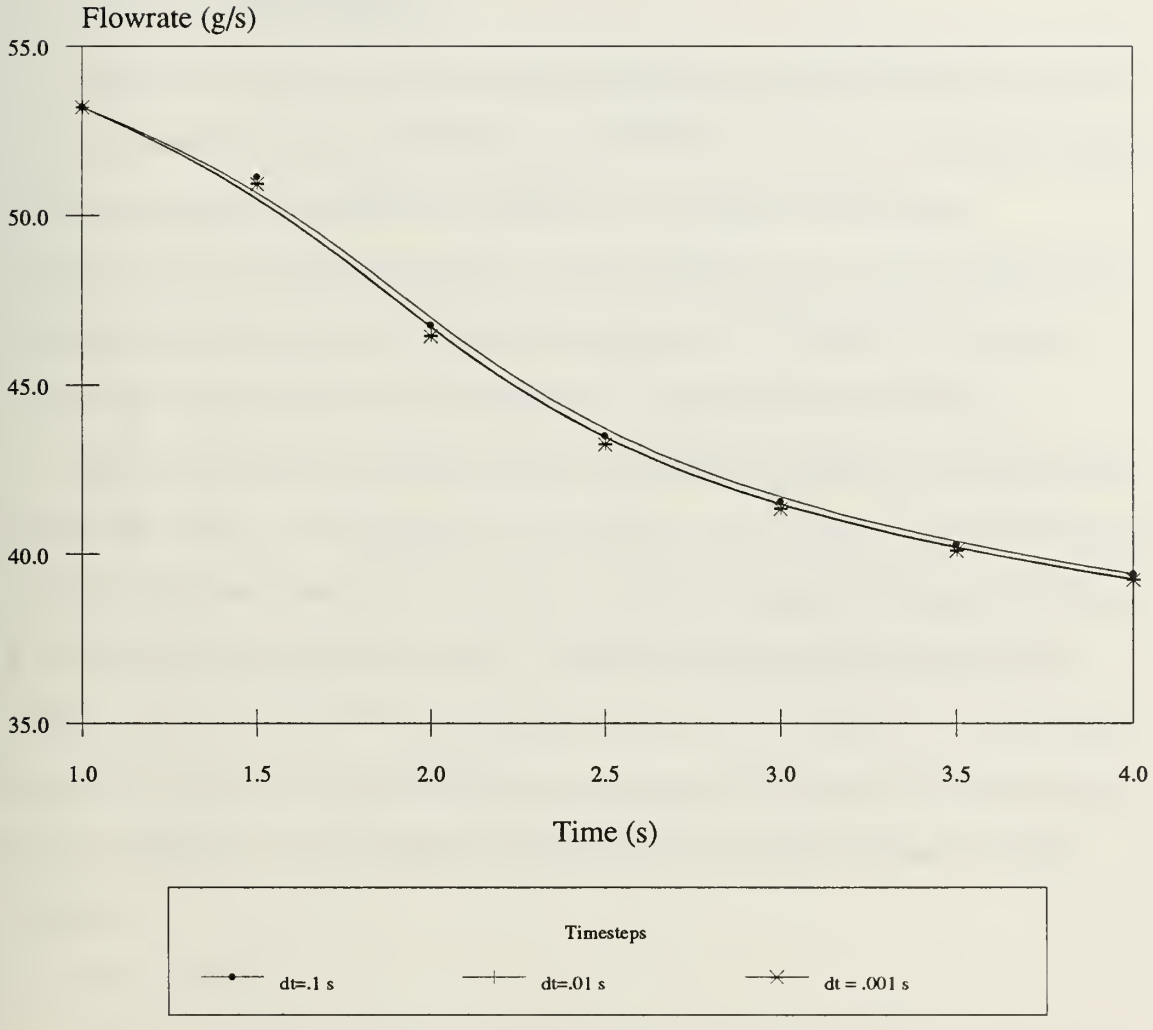


Figure C.13: Mass Flowrate Response for Various Timesteps

APPENDIX D:

SOURCE CODE AND DATA ENTRY

D.1 SOURCE CODE DESIGN

The source code for the analysis of the response of a particle bed reactor fuel element is provided in two parts: STEADY and UPPOWER. STEADY.C provides a means of calculating values at steady state conditions for a variety of fuel element configurations and operating parameters. This enables the user to analyze steady state conditions without having to run the transient program. UPPOWER.C is similar to STEADY.C but calculates transient values over a specified simulation period.

Both source codes are written in the C programming language and compiled using Microsoft™ Quick-C. The programs are intended for use on a desktop personal computer equipped with a hard drive but may run on a computer equipped with floppy disk drives if one drive is designated as the "c drive". STEADY produces the output data file STEADYST.DAT and UPPOWER produces the data file POWER.DAT. The data files generated by STEADY and UPPOWER are unformatted files which are easily imported into a spreadsheet (ie, Lotus 123™) which then allows the user to review the output graphically.

D.2 DATA ENTRY

STEADY and UPPOWER provide fuel element geometry and initial conditions which are similar to the PIPE experiments as initial values of all program variables. These values may be changed via a series of input screens presented to the user prior to

beginning an analysis. The first screen (Figure E.1) initializes all important geometry variables which may be changed by responding to the prompts under the list of parameters.

```
THE FOLLOWING ARE PARAMETERS FOR PPBR FUEL ELEMENT:

1.                  Outer Radius = 0.044580 m
2.              Cold Frit Radius = 0.029360 m
3.           Cold Frit Thickness = 0.001870 m
4.     Dia of Cold Frit Particle = 0.00000270 m
5.            Cold Frit Porosity = 0.685000
6.         Particle Bed Thickness = 0.012432 m
7.         Fuel Particle Diameter = 0.000500 m
8.    Particle Bed Void Fraction = 0.400000
9.            Hot Frit Thickness = 0.000760 m
10.            Hot Frit Porosity = 0.230000
11.        Inlet Manifold Factor = 0.950000
12.         Exit Manifold Factor = 0.888800
13.          Fuel Element Length = 0.265000 m


CHANGE A PARAMETER? (1=Y or 0=N)1

PLEASE ENTER THE PARAMETER NUMBER AND VALUE (eg 12, 1.705):
13, 1.0

CHANGE ANOTHER PARAMETER FOR ELEMENT A? (1=Y or 0=N)0
```

Figure E.1: Data Input Screen #1 for STEADY

Once the user is satisfied with the fuel element geometry, a listing of these values as they compare to the baseline PIPE fuel element is then shown on the screen (Figure E.2). This is followed by a screen which allows changes to specific fuel particle properties (Figure E.3) and a final input display which allows changes to general operating parameters (Figure E.4).

```
PARAMETER                          ELEMENT A              BASELINE

1. Outer Radius                    0.044580               0.044580 m
2. Cold Frit Radius                0.029360               0.029360 m
3. Cold Frit Thickness             0.001870               0.001870 m
4. Dia of Cold Frit Particle       0.00000270             0.00000270 m
5. Cold Frit Porosity              0.685000               0.685000
6. Particle Bed thickness          0.012432               0.012432 m
7. Fuel Particle Diameter          0.000500               0.000500 m
8. Particle Bed Void Fraction      0.400000               0.400000
9. Hot Frit Thickness              0.000760               0.000760 m
10.Hot Frit Porosity               0.230000               0.230000
11.Inlet Manifold Factor           0.950000               0.950000
12.Exit Manifold Factor            0.888800               0.888800
13.Length of Element               1.000000 m             0.265000


HIT ANY KEY TO CONTINUE
```

Figure E.2: Data Input Screen #2 for STEADY

```
1.      Fuel Material = UC2
2.   Layer 1 Material = Low D Car
3.   Layer 2 Material = High D CarZrC
4.   Layer 3 Material = ZrC
5.     Radius of Fuel = 0.000117 m
6.  Radius of Layer 1 = 0.000150 m
7.  Radius of Layer 2 = 0.000200 m
8.  Radius of Layer 3 = 0.000250 m
9.    Density of Fuel = 10500.000000 kg/m3
10.Density of Layer 1 = 1000.000000 kg/m3
11.Density of Layer 2 = 1900.000000 kg/m3
12.Density of Layer 3 = 6300.000000 kg/m3
13.        Cp of Fuel = 200.000000 J/kgK
14.     Cp of Layer 1 = 3000.000000 J/kgK
15.     Cp of Layer 2 = 3000.000000 J/kgK
16.     Cp of Layer 3 = 200.000000 J/kgK
17.        k for Fuel = 30.000000 W/m2K
18.     k for Layer 1 = 1.500000 W/m2K
19.     k for Layer 2 = 3.000000 W/m2K
20.     k for Layer 3 = 40.000000 W/m2K

CHANGE A PARAMETER? (1=Y or 0=N)0

CHANGE ANOTHER PARAMETER? (1=Y or 0=N)0
```

Figure E.3: Data Input Screen #3 for STEADY

```
THE FOLLOWING INITIAL AND FINAL CONDITIONS ARE SET

1.       Initial Inlet Temperature = 300.000000 K
2.          Initial Inlet Pressure = 2000.000000 KPa
3.         Initial Outlet Pressure = 1915.000000 KPa
4.           Initial Power Density = 0.000000 GW/m3
5.             Final Power Density = 2.500000 GW/m3
6.         Power Density Increment = 0.100000 sec
7.  Initial Guess at Mass Flowrate = 0.100000 kg/sec

CHANGE A PARAMETER? (1=Y or 0=N)1

ENTER PARAMETER NUMBER, VALUE (eg 20, 40)
3,1900

CHANGE ANOTHER PARAMETER? (1=Y or 0=N)0
```

Figure E.4: Data Input Screen #4 for STEADY

UPPOWER uses the same data input screens with the exception of input screen #4. UPPOWER allows the user to select a value for the initial and final value of inlet temperature, inlet pressure, outlet pressure and power density (Figure E.5). Further, the user may choose the duration of each transient. All transients start at the same time except for inlet temperature which has a 1 s delay to artificially simulate a preheating effect from a nuclear thermal rocket nozzle cooling jacket.

```
THE FOLLOWING INITIAL AND FINAL CONDITIONS ARE SET

1.         Initial Inlet Temperature = 300.000000 K
2.           Final Inlet Temperature = 300.000000 K
3.    Duration of Temperature Change = 1.000000 sec
4.             Initial Inlet Pressure = 2000.000000 KPa
5.               Final Inlet Pressure = 2000.000000 KPa
6.       Duration of Pressure Change = 1.000000 sec
7.            Initial Outlet Pressure = 1915.000000 KPa
8.              Final Outlet Pressure = 1915.000000 KPa
9.       Duration of Pressure Change = 1.000000 sec
10.             Initial Power Density = 0.000000 GW/m3
11.               Final Power Density = 2.000000 GW/m3
12.Duration of Power Density Change = 1.000000 sec
13.          Time Delay to Transient = 1.000000 sec
14.             Time After Transient = 8.000000 sec
15.                        Time Step = 0.010000 sec
16.  Initial Guess at Mass Flowrate = 0.100000 kg/sec

CHANGE A PARAMETER? (1=Y or 0=N)0

CHANGE ANOTHER PARAMETER? (1=Y or 0=N)0
Data Sample Frequency?(print every xth data point..)
10
```

Figure E.5: Data Input Screen #4 for UPPOWER

It should be noted that, as a final input, UPPOWER asks for a sampling frequency. This determines the amount of data written to the data file. Calculations are still performed at the required timestep, however. For example, if every data point were to be written to the data file using a timestep of 1 ms for a simulated transient lasting 10 s, there would be 10,000 data points written to the output file (one for each millisecond). If a sample frequency of 100 were to be used for the same transient, only 100 data points would be written to the output file (one every 100 ms). This serves only to reduce the data file to a manageable size and will not eliminate any of the calculations.

D.3 Hydrogen Equations of State

To analyze fuel element transients, properties of hydrogen coolant (such as density, viscosity and enthalpy) must be determined for various temperatures and pressures. Relationships to do this are placed at the end of the source code as subroutines and are accessed during the main part of the program as needed.

The density of hydrogen is determined by using a known value of hydrogen at 300 K and 101.325 kPa (1 atmosphere) and using the ideal gas relationship to determine other values. The subroutine in the source codes determines the value of coolant density as a function of temperature and pressure.

Viscosity, enthalpy, coolant heat conductivity and Prantdl number, on the other hand are determined as a function of temperature only since the contribution due to a change in pressure is negligible (R-1). To illustrate this, the enthalpy change from 200 K to 1200 K at a pressure of 101.325 kPa is 16406.2 kJ/kg and at a pressure of 2 MPa is 16478.7 kJ/kg. The difference between the two values is approximately 0.4%. This is considered negligible for the simple model. The other properties change in a similar fashion.

D.4 SOURCE CODE FOR STEADY

```
/*  STEADY.C                                        */
/*  Copywrite WILLIAM E. CASEY, MAY 1990.           */
/*  The author hereby grants to MIT and to the US Government   */
/*  permission to reproduce and distribute copies of this code. */
/*  This source code was written using the Microsoft QUICK-C   */
/*  programming environment.                        */
/*  Set up include files:                           */
#include <c:\msc\include\stdio.h>
#include <c:\msc\include\conio.h>
#include <c:\msc\include\graph.h>
#include <c:\msc\include\stdlib.h>
#include <c:\msc\include\math.h>
#define PI 3.14159
/*  Define a structure for geometry related variables:      */
struct fuelelem
{
        float OUTRAD;      /*  Outer radius of Inlet Plenum      */
        float CFRAD;       /*  Outer radius of Cold Frit         */
        float CFTHK;       /*  Cold Frit Thickness               */
        float CFDp;        /*  Diameter of Particles in Cold Frit  */
        float CFPOR;       /*  Cold Frit Porosity                */
        float PBEDTHK;     /*  Particle Bed Thickness            */
        float PBEDDp;      /*  Diameter of Fuel Particles        */
        float PBEDVOID;    /*  Particle Bed Void Fraction        */
        float HFTHK;       /*  Hot Frit Thickness                */
        float HFPOR;       /*  Hot Frit Porosity                 */
        float MFIN;        /*  Manifold Factor for Inlet Plenum  */
        float MFOUT;       /*  Manifold Factor for Outlet Plenum  */
        float LENGTH;      /*  Overall Length of Element         */
};
/*  Define a structure for fuel particle composition:       */
struct fuelpart
{
        char FUELMAT[10];  /*  Material used for Fuel          */
        char L1MAT[10];    /*  Material for Layer 1            */
        char L2MAT[10];    /*  Material for Layer 2            */
        char L3MAT[10];    /*  Material for Layer 3            */
        float FUELRAD;     /*  Radius of Fuel Core             */
        float L1RAD;       /*  Radius of Layer 1               */
        float L2RAD;       /*  Radius of Layer 2               */
        float L3RAD;       /*  Radius of Layer 3               */
        float FUELDEN;     /*  Density of Fuel                 */
        float L1DEN;       /*  Density of Layer 1              */
        float L2DEN;       /*  Density of Layer 2              */
        float L3DEN;       /*  Density of Layer 3              */
        float FUELCp;      /*  Cp for Fuel                     */
        float L1Cp;        /*  Cp for Layer 1                  */
        float L2Cp;        /*  Cp for Layer 2                  */
        float L3Cp;        /*  Cp for Layer 3                  */
        float FUELK;       /*  Heat Transfer Coef for Fuel     */
        float L1K;         /*  Heat Transfer Coef for Layer 1  */
        float L2K;         /*  Heat Transfer Coef for Layer 2  */
        float L3K;         /*  Heat Transfer Coef for Layer 3  */
};
```

```
/*  Define a structure for initial and final conditions:      */
struct conditions
{
        float INIT_TIN;    /*  Initial Value of T in            */
        float INIT_PIN;    /*  Initial Value of P in            */
        float INIT_POUT;   /*  Initial Value of P out           */
        float INIT_PRDEN;  /*  Initial Value of Power Density   */
        float FIN_PRDEN;   /*  Final Value of Power Density     */
        float delta_t;     /*  Increment                        */
        float INIT_W;      /*  Initial Guess at Mass Flowrate   */
};
void showdefaults( struct fuelelem *e_ptr );
void fuelelemsum( struct fuelelem *e_ptr, struct fuelelem *f_ptr );
void showfuelpart ( struct fuelpart *g_ptr );
void showconditions ( struct conditions *h_ptr );

/*  Define H2 state relationships found at end of source code:  */
float h2density (float xx, float yy);
float h2viscosity (float xx);
float h2heatxfer (float xx);
float h2cp (float xx);
float h2prandle (float xx);
float h2H (float xx);

/* ** BEGINNING OF COMPUTATIONAL CODE ********************** */
main()
{
/*  Define variables for use in source code:                  */
        double W0, W1;
        double pinI, pinII, pinIII, pout;
        double pbarI, pbarII, pbarIII;
        double Tin, Tout, TbarII;
        float PinI, PinII, PinIII, Pout, PbarI, PbarII, PbarIII;
        float uin, uout, ubarII;
        float RI, RII, RIII, RTOTAL;
        float PDROPI1, PDROPII1, PDROPIII1;
        float PDROPI2, PDROPII2, PDROPIII2;
        float HinI1, HinII1, HinIII1, Hout1;
        float HinI2, HinII2, HinIII2, Hout2;
        float HbarI1, HbarI2, HbarII1, HbarII2, HbarIII1, HbarIII2;
        float Qbed;
        float AinI, AoutI, ACF, AinII, AoutII, AHF, AinIII, AoutIII;
        float AbarII;
        float CPin, CPout, CPtemp;
        float Werror, CPerr;
        float Deq_in, Deq_out;
        float RIf, RIa, RIk, RIm;
        float RCF, ReI, ReIII, RIIa;
        float RIIIf, RIIIa, RIIIk, RIIIm;
        float mfout;
        float powerdensity, vout;
        int ch, chr;
        float aa, alpha, cc;
        float Htemp, Herror;
```

```
        static struct fuelelem A =
        {   .04458, .02936, .00187, .0000027, .685, .012432, .0005,
              .4, .00076, .23, .95, 1.1, .265
        };
        static struct fuelelem B =
        {   .04458, .02936, .00187, .0000027, .685, .012432, .0005,
              .4, .00076, .23, .95, 1.1, .265
        };
        static struct fuelpart C =
        {   "UC2", "Low D Car", "High D Car", "ZrC", .000117, .00015, \
              .0002, .00025, 10500, 1000, 1900, 6300, 200, 3000, 3000, \
              200, 30, 1.5, 3, 40
        };
        static struct conditions D =
        {   300, 2000, 1915, 0, 2.5, .1, .1
        };
        FILE *out;
        out = fopen ( "c:steadyst.dat","w+" );
        ch = 1;
        while (ch == 1 )
        {
              showdefaults ( &A );
              printf("\nCHANGE ANOTHER PARAMETER FOR ELEMENT A? (1=Y or 0=N)");
              scanf("%i", &ch);
        }
        fuelelemsum ( &A, &B );
        ch = 1;
        while (ch == 1 )
        {
              showfuelpart ( &C );
              printf("\nCHANGE ANOTHER PARAMETER? (1=Y or 0=N)");
              scanf("%i", &ch);
        }
        ch = 1;
        while (ch == 1 )
        {
              showconditions ( &D );
              printf("\nCHANGE ANOTHER PARAMETER? (1=Y or 0=N)");
              scanf("%i", &ch);
        }

/*  Determination of Steady State Values-Initial Conditions    */

        fprintf( out,"STEADY STATE VALUES FOR THE FOLLOWING INITIAL CONDITIONS\n");
        fprintf( out,"\nInlet Temp = %.2f K\n", D.INIT_TIN);
        fprintf( out,"Inlet Press = %.2f kPa\n", D.INIT_PIN);
        fprintf( out,"Outlet Press = %.2f kPa\n\n", D.INIT_POUT);

fprintf(out,"PWRDN\tQbed\tW\tPDI\tPDII\tPDIII\tTout\tTbarII\tPinI\tPinII\tPinIII\tPout\tvout\n\n");
        W1 = D.INIT_W;
        W0 = W1;
        AinI = PI * (A.OUTRAD * A.OUTRAD - A.CFRAD * A.CFRAD);
        AoutI = 2 * PI * A.CFRAD * A.LENGTH;
        ACF = AoutI * A.CFPOR;
        AinII = 2 * PI * (A.CFRAD - A.CFTHK) * A.LENGTH * A.PBEDVOID;
        AoutII = 2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK) * A.LENGTH * \
                    A.PBEDVOID;
        AbarII = .5 * (AinII + AoutII);
```

```
        AinIII = 2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK) * \
                A.LENGTH;
        AoutIII = PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK) * \
                (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK);
        AHF = AinIII * A.HFPOR;
        Deq_in = 4 * AinI /(2 * PI * (A.OUTRAD + A.CFRAD));
        Deq_out = 4 * AoutIII / (2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - \
                A.HFTHK));
        for (powerdensity = D.INIT_PRDEN; powerdensity < D.FIN_PRDEN+.1;\
                powerdensity += .1)
        {
                Qbed = powerdensity * 1000000000 * PI * ((A.CFRAD - A.CFTHK) * \
                        (A.CFRAD - A.CFTHK) - (A.CFRAD - A.CFTHK - A.PBEDTHK) * \
                        (A.CFRAD - A.CFTHK - A.PBEDTHK)) * A.LENGTH;
                Tin = D.INIT_TIN;
                HinI1 = h2H(Tin);
                PinI = D.INIT_PIN;
                PinII = D.INIT_PIN;
                PinIII = D.INIT_POUT;
                Pout = D.INIT_POUT;
                Tout = Tin;
         do
         {
                W0 = W1;
                Hout1 = Qbed/W0 + HinI1;
                        do
                        {
                                Htemp = h2H(Tout);
                                Herror = Hout1 - Htemp;
                                Tout = Tout + Herror/100000;
                        }   while ( Herror/1000000 > .000001 );
/*      Set up all variables                                    */
                TbarII = .5*(Tin + Tout);
                PbarI = .5*(PinI + PinII);
                PbarII = .5*(PinII + PinIII);
                PbarIII = .5*(PinIII + Pout);
                pbarI = h2density(Tin, PbarI);
                pbarII = h2density(TbarII, PbarII);
                pbarIII= h2density(Tout, PbarIII);
                uin = h2viscosity(Tin);
                ubarII = h2viscosity(TbarII);
                uout = h2viscosity(Tout);
                pinII = h2density(Tin, PinII);
                pinIII = h2density(Tout, PinIII);
                pout = h2density(Tout,Pout);
/*      Inlet Resistance Calculations                           */
                ReI = W0*Deq_in / (uin*AinI);
                RIf = .138*pow(ReI, -.151)*(A.LENGTH/Deq_in)/(2*pbarI*AinI*AinI);
                RIa = (1/(AinII*AinII) - 1/(AinI*AinI))/(2*pbarI);
                RIk = (1-(AinI/AoutI))*(1-(AinI/AoutI))/(2*pbarI*AinI*AinI)    \
                        + .5 * (1-(ACF/AoutI))*(1-(ACF/AoutI)) / (2*pinII*ACF*ACF) \
                        + .5 * (1-(AinII/ACF))*(1-(AinII/ACF)) / (2*pinII*AinII*AinII);
                RIm = A.MFIN / (pbarI*AinI*AinI);
                RIm = RIm + A.MFIN/(pbarI*AoutI*AoutI);
                RCF = 150*uin*(1-A.CFPOR)*(1-A.CFPOR)/(A.CFDp*A.CFDp*pinII*AoutI*W0* \
                        A.CFPOR*A.CFPOR*A.CFPOR)\
```

```
                    + 1.75*(1-A.CFPOR)/(A.CFDp*pinII*AoutI*AoutI*A.CFPOR*A.CFPOR* \
                    A.CFPOR);
            RCF = RCF * A.CFTHK;
            RI = RIf/2 + RIa + RIk + RIm + RCF;
/*      Resistance Calculations for the Particle Bed            */
            RII = 150*ubarII*(1-A.PBEDVOID)*(1-A.PBEDVOID)/(A.PBEDDp*A.PBEDDp* \
                    pbarII*AbarII*W0*A.PBEDVOID*A.PBEDVOID)                 \
                    + 1.75*(1-A.PBEDVOID)/(A.PBEDDp*pbarII*AbarII*AbarII*A.PBEDVOID);
            RII = RII * A.PBEDTHK;
            RIIa = ( 1/(pinIII*AinIII) - 1/(pinII*AinII) )*((AinII+AinIII)/ \
                    (2*AinII*AinIII));
            RII = RII + RIIa;
/*      Resistance for Exit Plenum                      */
            ReIII = W0*Deq_out / (uout*AoutIII);
            RIIIf = .138*pow(ReIII, -.151)*(A.LENGTH/Deq_out)/(2*pbarIII*AoutIII \
                    *AoutIII);
            RIIIa = (1/(AoutIII*AoutIII)-1/(AoutII*AoutII))/(2*pbarIII);
            RIIIk = .5*(1-AHF/AoutII)*(1-AHF/AoutII)/(2*pinIII*AHF*AHF) \
                    + (1-AHF/AinIII)*(1-AHF/AinIII)/(2*pinIII*AHF*AHF)  \
                    + (1-AoutIII/AinIII)*(1-AoutIII/AinIII)/(2*pbarIII* \
                    AoutIII*AoutIII);
            mfout = A.MFOUT;
            RIIIm = mfout / (pbarIII*AoutIII*AoutIII) + mfout / (pbarIII \
                    *AinIII*AinIII);
            RIII = RIIIf/2 + RIIIa + RIIIk + RIIIm;
/*      Final Assembly                          */
            RTOTAL = RI + RII + RIII;
            W1 = (PinI - Pout)*1000/RTOTAL;
            W1 = sqrt(W1);
            Werror = W0 - W1;
            Werror = fabs(Werror);
            PDROPI1 = RI*W1*W1/1000;
            PDROPII1 = RII*W1*W1/1000;
            PDROPIII1 = RIII*W1*W1/1000;
            PinII = PinI-PDROPI1;
            PinIII = Pout+PDROPIII1;
            vout = W1/(pout*AoutIII);
        } while (Werror > .0000001);
            W1=W1*1000;
            Qbed = Qbed/1000;

fprintf(out,"%.2f\t%.1f\t%.5f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n", \
                    powerdensity, Qbed, W1,PDROPI1,PDROPII1,PDROPIII1,Tout,TbarII,Pi-
nI,PinII,PinIII,Pout,vout);
            printf ("Pwrden = %f\tW = %f\n", powerdensity, W1);
            Qbed = Qbed*1000;
            W1=W1/1000;
            if (Tout > 2700)
                goto meltdown;
    }
    meltdown:
    if (Tout>2700)
    {   printf("Tout > 2700");
            fprintf(out,"Temp exceeds 2700");
    }
```

```
}
/* ********************************************************************** */
void showdefaults( struct fuelelem *e_ptr )
{
        int num;
        float entry;
        int c;
        _clearscreen(0);
        printf("THE FOLLOWING ARE PARAMETERS FOR PPBR FUEL ELEMENT:\n\n");
        printf("1.              Outer Radius = %f m\n", e_ptr->OUTRAD);
        printf("2.            Cold Frit Radius = %f m\n", e_ptr->CFRAD);
        printf("3.         Cold Frit Thickness = %f m\n", e_ptr->CFTHK);
        printf("4.    Dia of Cold Frit Particle = %.8f m\n", e_ptr->CFDp);
        printf("5.          Cold Frit Porosity = %f\n", e_ptr->CFPOR);
        printf("6.       Particle Bed Thickness = %f m\n", e_ptr->PBEDTHK);
        printf("7.       Fuel Particle Diameter = %f m\n", e_ptr->PBEDDp);
        printf("8.   Particle Bed Void Fraction = %f\n", e_ptr->PBEDVOID);
        printf("9.          Hot Frit Thickness = %f m\n", e_ptr->HFTHK);
        printf("10.          Hot Frit Porosity = %f\n", e_ptr->HFPOR);
        printf("11.       Inlet Manifold Factor = %f\n", e_ptr->MFIN);
        printf("12.       Exit Manifold Factor = %f\n", e_ptr->MFOUT);
        printf("13.        Fuel Element Length = %f m\n", e_ptr->LENGTH);
        printf("\n\n\nCHANGE A PARAMETER? (1=Y or 0=N)");
        scanf( "%i", &c );
        if ( c == 0 )
                return;
        printf("\nPLEASE ENTER THE PARAMETER NUMBER AND VALUE (eg 12, 1.705):\n");
        scanf( "%i, %f", &num, &entry );
        switch (num)
        {
                case 1:
                        e_ptr->OUTRAD = entry;
                        break;
                case 2:
                        e_ptr->CFRAD = entry;
                        break;
                case 3:
                        e_ptr->CFTHK = entry;
                        break;
                case 4:
                        e_ptr->CFDp = entry;
                        break;
                case 5:
                        e_ptr->CFPOR = entry;
                        break;
                case 6:
                        e_ptr->PBEDTHK = entry;
                        break;
                case 7:
                        e_ptr->PBEDDp = entry;
                        break;
                case 8:
                        e_ptr->PBEDVOID = entry;
                        break;
                case 9:
                        e_ptr->HFTHK = entry;
                        break;
                case 10:
```

```
                e_ptr->HFPOR = entry;
                break;
        case 11:
                e_ptr->MFIN = entry;
                break;
        case 12:
                e_ptr->MFOUT = entry;
                break;
        case 13:
                e_ptr->LENGTH = entry;
                break;
        default:
                printf("\n\nTRY ANOTHER PARAMETER\n");
                _clearscreen(0);
                break;
    }
    return;
}
/* ********************************************************************** */
void fuelelemsum ( struct fuelelem *e_ptr, struct fuelelem *f_ptr )
{
    _clearscreen (0);
    printf ("PARAMETER\t\t\tELEMENT A\t\tBASELINE\n\n");
    printf("1. Outer Radius\t\t\t%f\t\t%f m\n", e_ptr->OUTRAD, f_ptr->OUTRAD);
    printf("2. Cold Frit Radius\t\t%f\t\t%f m\n", e_ptr->CFRAD, f_ptr->CFRAD);
    printf("3. Cold Frit Thickness\t\t%f\t\t%f m\n", e_ptr->CFTHK, f_ptr->CFTHK);
    printf("4. Dia of Cold Frit Particle\t%.8f\t\t%.8f m\n", e_ptr->CFDp, f_ptr->CFDp);
    printf("5. Cold Frit Porosity\t\t%f\t\t%f\n", e_ptr->CFPOR, f_ptr->CFPOR);
    printf("6. Particle Bed thickness\t%f\t\t%f m\n", e_ptr->PBEDTHK, f_ptr->PBEDTHK);
    printf("7. Fuel Particle Diameter\t%f\t\t%f m\n", e_ptr->PBEDDp, f_ptr->PBEDDp);
    printf("8. Particle Bed Void Fraction\t%f\t\t%f\n", e_ptr->PBEDVOID, f_ptr->PBEDVOID);
    printf("9. Hot Frit Thickness\t\t%f\t\t%f m\n", e_ptr->HFTHK, f_ptr->HFTHK);
    printf("10.Hot Frit Porosity\t\t%f\t\t%f\n", e_ptr->HFPOR, f_ptr->HFPOR);
    printf("11.Inlet Manifold Factor\t%f\t\t%f\n", e_ptr->MFIN, f_ptr->MFIN);
    printf("12.Exit Manifold Factor\t\t%f\t\t%f\n", e_ptr->MFOUT, f_ptr->MFOUT);
    printf("13.Length of Element\t\t%f m\t\t%f\n", e_ptr->LENGTH, f_ptr->LENGTH);
    printf("\n\nHIT ANY KEY TO CONTINUE");
    getch();

}
/* ********************************************************************** */
void showfuelpart ( struct fuelpart *g_ptr )
{
    int num;
    float entry;
    int c;
    _clearscreen(0);
    printf("THE FOLLOWING ARE PARAMETERS FOR THE FUEL PARTICLE\n");
    printf("1.     Fuel Material = %s\n", g_ptr->FUELMAT);
    printf("2.  Layer 1 Material = %s\n", g_ptr->L1MAT);
    printf("3.  Layer 2 Material = %s\n", g_ptr->L2MAT);
    printf("4.  Layer 3 Material = %s\n", g_ptr->L3MAT);
    printf("5.    Radius of Fuel = %f m\n", g_ptr->FUELRAD);
    printf("6.  Radius of Layer 1 = %f m\n", g_ptr->L1RAD);
    printf("7.  Radius of Layer 2 = %f m\n", g_ptr->L2RAD);
    printf("8.  Radius of Layer 3 = %f m\n", g_ptr->L3RAD);
    printf("9.   Density of Fuel = %f kg/m3\n", g_ptr->FUELDEN);
    printf("10.Density of Layer 1 = %f kg/m3\n", g_ptr->L1DEN);
```

```
printf("11.Density of Layer 2 = %f kg/m3\n", g_ptr->L2DEN);
printf("12.Density of Layer 3 = %f kg/m3\n", g_ptr->L3DEN);
printf("13.       Cp of Fuel = %f J/kgK\n", g_ptr->FUELCp);
printf("14.    Cp of Layer 1 = %f J/kgK\n", g_ptr->L1Cp);
printf("15.    Cp of Layer 2 = %f J/kgK\n", g_ptr->L2Cp);
printf("16.    Cp of Layer 3 = %f J/kgK\n", g_ptr->L3Cp);
printf("17.       k for Fuel = %f W/m2K\n", g_ptr->FUELK);
printf("18.    k for Layer 1 = %f W/m2K\n", g_ptr->L1K);
printf("19.    k for Layer 2 = %f W/m2K\n", g_ptr->L2K);
printf("20.    k for Layer 3 = %f W/m2K\n", g_ptr->L3K);
printf("\nCHANGE A PARAMETER? (1=Y or 0=N)");
scanf( "%i", &c );
if ( c == 0 )
        return;
printf("\nENTER PARAMETER NUMBER ( NOT 1-4 ), VALUE (eg 20, 40)\n");
scanf( "%i, %f", &num, &entry );
switch (num)
{
        case 5:
                g_ptr->FUELRAD = entry;
                break;
        case 6:
                g_ptr->L1RAD = entry;
                break;
        case 7:
                g_ptr->L2RAD = entry;
                break;
        case 8:
                g_ptr->L3RAD = entry;
                break;
        case 9:
                g_ptr->FUELDEN = entry;
                break;
        case 10:
                g_ptr->L1DEN = entry;
                break;
        case 11:
                g_ptr->L2DEN = entry;
                break;
        case 12:
                g_ptr->L3DEN = entry;
                break;
        case 13:
                g_ptr->FUELCp = entry;
                break;
        case 14:
                g_ptr->L1Cp = entry;
                break;
        case 15:
                g_ptr->L2Cp = entry;
                break;
        case 16:
                g_ptr->L3Cp = entry;
                break;
        case 17:
                g_ptr->FUELK = entry;
                break;
        case 18:
```

```
                g_ptr->L1K = entry;
                break;
        case 19:
                g_ptr->L2K = entry;
                break;
        case 20:
                g_ptr->L3K = entry;
                break;
        default:
                printf("CANNOT CHANGE THIS PARAMETER...SORRY.\n");
                getch();
                break;
    }
    return;
}
/* ********************************************************************** */
void showconditions ( struct conditions *h_ptr )
{
    int num;
    float entry;
    int c;
    _clearscreen(0);
    printf("THE FOLLOWING INITIAL AND FINAL CONDITIONS ARE SET\n\n");
    printf("1.      Initial Inlet Temperature = %f K\n", h_ptr->INIT_TIN);
    printf("2.        Initial Inlet Pressure = %f KPa\n", h_ptr->INIT_PIN);
    printf("3.       Initial Outlet Pressure = %f KPa\n", h_ptr->INIT_POUT);
    printf("4.         Initial Power Density = %f GW/m3\n", h_ptr->INIT_PRDEN);
    printf("5.           Final Power Density = %f GW/m3\n", h_ptr->FIN_PRDEN);
    printf("6.       Power Density Increment = %f sec\n", h_ptr->delta_t);
    printf("7.  Initial Guess at Mass Flowrate = %f kg/sec\n", h_ptr->INIT_W);
    printf("\nCHANGE A PARAMETER? (1=Y or 0=N)");
    scanf( "%i", &c );
    if ( c == 0 )
            return;
    printf("\nENTER PARAMETER NUMBER, VALUE (eg 20, 40)\n");
    scanf( "%i, %f", &num, &entry );
    switch (num)
    {
            case 1:
                    h_ptr->INIT_TIN = entry;
                    break;
            case 2:
                    h_ptr->INIT_PIN = entry;
                    break;
            case 3:
                    h_ptr->INIT_POUT = entry;
                    break;
            case 4:
                    h_ptr->INIT_PRDEN = entry;
                    break;
            case 5:
                    h_ptr->FIN_PRDEN = entry;
                    break;
            case 6:
                    h_ptr->delta_t = entry;
                    break;
            case 7:
                    h_ptr->INIT_W = entry;
```

```
                break;
        default:
                printf("CANNOT CHANGE THIS PARAMETER...SORRY.\n");
                getch();
                break;
    }
    return;
}
/* ********************************************************************** */
float h2density(float xx, float yy)
{
    float dens;
    dens = .08185 * 300 / xx * yy / 101.325;
    return dens;
}
/* ********************************************************************** */
float h2viscosity(float xx)
{
    float visc;
    visc = .000008963 * pow( xx/300, .6733);
    return visc;
}
/* ********************************************************************** */
float h2heatxfer(float xx)
{
    float tem1;
    int i;
    float tem2;
    float heat[45] = {0,.0362, .0665, .0981, .1282, .1561, .182, .206, .228,\
                      .251, .272, .292, .315, .333, .351, .3665, .384, .398,\
                      .412, .426, .44, .452, .464, .476, .488, .500, .512, \
                      .524, .536, .548, .560, .572, .584, .596, .608, .62, \
                      .632, .644, .656, .668, .680, .692, .704, .716, .728 };
    tem2 = xx / 50.0;
      i = floor(tem2);
    tem1 = heat[i] + (heat[i+1] - heat[i]) * ((xx - (i * 50)) / 50 );
    return tem1;
}
/* ********************************************************************** */
float h2cp(float xx)
{
    float temcp;
    if ( xx <= 420 )
        temcp = 4.1868 * 1000 * (1.5395 + .0150825 * xx - 4.02449e-5 * \
                xx * xx + 3.63544e-8 * xx * xx * xx);
    else
        temcp = 4.1868 * 1000 * (3.58927 - 5.55096e-4 * xx + 6.94235e-7 * xx \
                * xx - 1.45155e-10 * xx * xx * xx);
    return temcp;
}
/* ********************************************************************** */
float h2H(float xx)
{
    float temH;
    float temH1;
    temH=0;
    temH1=0;
```

```
        if (xx <= 420)
        {
        temH=4.1868*1000*(1.5395*xx + .0150825*xx*xx/2 - 4.02449e-5*        \
                xx*xx*xx/3 + 3.63544e-8*xx*xx*xx*xx/4);
        temH=temH - 4.1868*1000*(1.5395*50 + .0150825*50*50/2 - 4.02449e-5*  \
                50*50*50/3 + 3.63544e-8*50*50*50*50/4);
        }
        else
        {
        temH1=4.1868*1000*(3.58927*xx - 5.55096e-4*xx*xx/2 + 6.94235e-7*     \
                 xx*xx*xx/3 - 1.45155e-10*xx*xx*xx*xx/4);
        temH1=temH1 - 4.1868*1000*(3.58927*420 - 5.55096e-4*420*420/2 +      \
                 6.94235e-7*420*420*420/3 - 1.45155e-10*420*420*420*420/4);
        temH=temH1 + 4.1868*1000*(1.5395*420+.0150825*420*420/2-4.02449e-5*  \
                420*420*420/3 + 3.63544e-8*420*420*420*420/4);
        temH=temH - 4.1868*1000*(1.5395*50 + .0150825*50*50/2 - 4.02449e-5*  \
                50*50*50/3 + 3.63544e-8*50*50*50*50/4);

        }


        return temH;
}
/* ********************************************************************** */
float h2prandle(float xx)
{
        float tem1;
        int i;
        float tem2;
        float pran[23] =    {.000, .712, .719, .706, .690, .675, .664,   \
                                    .659, .664, .676, .686, .703, .715, .733,   \
                                    .748, .763, .778, .793, .808, .823, .838,   \
                                    .853, .868};
        tem2 = xx / 100.0;
          i = floor(tem2);
        tem1 = pran[i] + (pran[i+1] - pran[i]) * ((xx - (i * 100)) / 100 );
        return tem1;
}
/* ********************************************************************** */
```

E.5 SOURCE CODE FOR UPPOWER

```
/*  UPPOWER.C                                                    */
/*  Copywrite WILLIAM E. CASEY, MAY 1990.                        */
/*  The author hereby grants to MIT and to the US Government     */
/*  permission to reproduce and distribute copies of this code. */
/*  This source code was written using the Microsoft QUICK-C     */
/*  programming environment.                                     */
/*  Set up include files:                                        */
#include <c:\msc\include\stdio.h>
#include <c:\msc\include\conio.h>
#include <c:\msc\include\graph.h>
#include <c:\msc\include\stdlib.h>
#include <c:\msc\include\math.h>
#define PI 3.14159265442
struct fuelelem
{
        float OUTRAD;      /*  Outer radius of Inlet Plenum       */
        float CFRAD;       /*  Outer radius of Cold Frit          */
        float CFTHK;       /*  Cold Frit Thickness                */
        float CFDp;        /*  Diameter of Particles in Cold Frit */
        float CFPOR;       /*  Cold Frit Porosity                 */
        float PBEDTHK;     /*  Particle Bed Thickness             */
        float PBEDDp;      /*  Diameter of Fuel Particles         */
        float PBEDVOID;    /*  Particle Bed Void Fraction         */
        float HFTHK;       /*  Hot Frit Thickness                 */
        float HFPOR;       /*  Hot Frit Porosity                  */
        float MFIN;        /*  Manifold Factor for Inlet Plenum   */
        float MFOUT;       /*  Manifold Factor for Outlet Plenum  */
        float LENGTH;      /*  Overall Length of Element          */
};
struct fuelpart
{
        char FUELMAT[10];  /*  Material used for Fuel             */
        char L1MAT[10];    /*  Material for Layer 1               */
        char L2MAT[10];    /*  Material for Layer 2               */
        char L3MAT[10];    /*  Material for Layer 3               */
        float FUELRAD;     /*  Radius of Fuel Core                */
        float L1RAD;       /*  Radius of Layer 1                  */
        float L2RAD;       /*  Radius of Layer 2                  */
        float L3RAD;       /*  Radius of Layer 3                  */
        float FUELDEN;     /*  Density of Fuel                    */
        float L1DEN;       /*  Density of Layer 1                 */
        float L2DEN;       /*  Density of Layer 2                 */
        float L3DEN;       /*  Density of Layer 3                 */
        float FUELCp;      /*  Cp for Fuel                        */
        float L1Cp;        /*  Cp for Layer 1                     */
        float L2Cp;        /*  Cp for Layer 2                     */
        float L3Cp;        /*  Cp for Layer 3                     */
        float FUELK;       /*  Heat Transfer Coef for Fuel        */
        float L1K;         /*  Heat Transfer Coef for Layer 1     */
        float L2K;         /*  Heat Transfer Coef for Layer 2     */
        float L3K;         /*  Heat Transfer Coef for Layer 3     */
};
struct conditions
{
```

```
        float INIT_TIN;    /*  Initial Value of T in            */
        float FIN_TIN;     /*  Final Value of T in              */
        float DUR_TIN;     /*  Duration of T in Transient       */
        float INIT_PIN;    /*  Initial Value of P in            */
        float FIN_PIN;     /*  Final Value of P in              */
        float DUR_PIN;     /*  Duration of P in Transient       */
        float INIT_POUT;   /*  Initial Value of P out           */
        float FIN_POUT;    /*  Final Value of P out             */
        float DUR_POUT;    /*  Duration of P out Transient      */
        float INIT_PRDEN;  /*  Initial Value of Power Density   */
        float FIN_PRDEN;   /*  Final Value of Power Density     */
        float DUR_PRDEN;   /*  Duration of Power Density Transient */
        float t_DELAY;     /*  Time Delay prior to Transient    */
        float t_AFTER;     /*  Window of Time After Transient   */
        float delta_t;     /*  Time Step                        */
        float INIT_W;      /*  Initial Guess at Mass Flowrate   */
};
void showdefaults( struct fuelelem *e_ptr );
void fuelelemsum( struct fuelelem *e_ptr, struct fuelelem *f_ptr );
void showfuelpart ( struct fuelpart *g_ptr );
void showconditions ( struct conditions *h_ptr );
double h2density(double xx, double yy);
double h2viscosity(double xx);
double h2heatxfer(double xx);
double h2cp(double xx);
float h2prandle(float xx);
float h2H(float xx);
/*  *** BEGINNING OF COMPUTATIONAL CODE  ******************  */
main()
{
/*  Define variables:                                        */
        int    ch, chr;
        float  W1, W2;
        float  W;
        double Werror;
        float  pinI, pinII, pinIII, pout;
        float  pbarI, pbarII, pbarIII;
        float  uin, uout, ubarII;
        float  Tin, TinI, TinII, TinIII, Tout, TbarI, TbarII, TbarIII;
        float  PinI, PinII, PinIII, Pout, PbarI, PbarII, PbarIII;
        float  PinI1, PinII1, PinIII1, Pout1, PbarI1, PbarII1, PbarIII1;
        float  PDROPI1, PDROPII1, PDROPIII1, PDROPI2, PDROPII2, PDROPIII2;
        float  Hin1, HinI1, HinII1, HinIII1, Hout1;
        float  Hin2, HinI2, HinII2, HinIII2, Hout2, Htemp;
        double Herror;
        float  Deq_in, Deq_out;
        float  AinI, AoutI, ACF, AinII, AoutII, AHF, AinIII, AoutIII;
        float  AbarII;
        float  volI, volII, volIII;
        float  RI, RII, RIII, RTOTAL;
        float  RIf, RIa, RIk, RIm, RIIa;
        float  RCF, ReI, ReII, ReIII;
        float  RIIIf, RIIIa, RIIIk, RIIIm, mfout;
        float  Qbed, powerdensity, vout, power, powerout;
        double time, transtime, deltat;
        float  ma1, ma2, ma3, ma4, mat;
        float  U1, U2, U3, U4, UT, UBAR, Cpbar;
        float  f1, f2, f3, fbar;
```

```
float  Nu, Pr, k, h;
float  I1, I2, I3, M1, M2, M3;
float  AA, BB, DD, EE, N;
float  Vcv, Av;
float  Tf2, Tf1, Qs;
float WA,pfuelave;
static struct fuelelem A =
{   .04458, .02936, .00187, .0000027, .685, .012432, .0005,
        .4, .00076, .23, .95, 1.1, .265
};
static struct fuelelem B =
{   .04458, .02936, .00187, .0000027, .685, .012432, .0005,
        .4, .00076, .23, .95, 1.1, .265
};
static struct fuelpart C =
{   "UC2", "Low D Car", "High D Car", "ZrC", .000117, .00015,\
        .0002, .00025, 10500, 1000, 1900, 6300, 200, 3000, 3000, \
        200, 30, 1.5, 3, 40
};
static struct conditions D =
{   300, 300, 1, 2000, 2000, 1, 1915, 1915, 1, 0, 2, 1, 1, 8, .01, .1
};
FILE *out;
out = fopen ( "c:power.dat","w+" );

ch = 1;
while (ch == 1 )
{
        showdefaults ( &A );
        printf("\nCHANGE ANOTHER PARAMETER FOR ELEMENT A? (1=Y or 0=N)");
        scanf("%i", &ch);
}
fuelelemsum ( &A, &B );
ch = 1;
while (ch == 1 )
 {
        showfuelpart ( &C );
        printf("\nCHANGE ANOTHER PARAMETER? (1=Y or 0=N)");
        scanf("%i", &ch);
 }
ch = 1;
while (ch == 1 )
{
        showconditions ( &D );
        printf("\nCHANGE ANOTHER PARAMETER? (1=Y or 0=N)");
        scanf("%i", &ch);
}
printf("Data Sample Frequency?(print every xth data point..)\n");
scanf("%f", &EE);
fprintf( out,"VALUES FOR THE FOLLOWING INITIAL CONDITIONS\n");
fprintf( out,"\n    Initial Inlet Temp = %.2f K\n", D.INIT_TIN);
fprintf( out,"   Initial Inlet Press = %.2f kPa\n", D.INIT_PIN);
fprintf( out,"  Initial Outlet Press = %.2f kPa\n\n", D.INIT_POUT);
fprintf(out, "time\ttr t\tHI2\t\tHII2\t\tHIII2\t\tHout2\tpden\tW2\n");
```

```
        W1 = D.INIT_W;
        AinI = PI * (A.OUTRAD * A.OUTRAD - A.CFRAD * A.CFRAD);
        AoutI = 2 * PI * A.CFRAD * A.LENGTH;
        ACF = AoutI * A.CFPOR;
        AinII = 2 * PI * (A.CFRAD - A.CFTHK) * A.LENGTH * A.PBEDVOID;
        AoutII = 2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK) * A.LENGTH * \
                A.PBEDVOID;
        AbarII = .5 * (AinII + AoutII);
        AinIII = 2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK) * \
                A.LENGTH;
        AoutIII = PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK) * \
                    (A.CFRAD - A.CFTHK - A.PBEDTHK - A.HFTHK);
        AHF = AinIII * A.HFPOR;
        Deq_in = 4 * AinI /(2 * PI * (A.OUTRAD + A.CFRAD));
        Deq_out = 4 * AoutIII / (2 * PI * (A.CFRAD - A.CFTHK - A.PBEDTHK - \
                    A.HFTHK));
        volI = AinI*A.LENGTH+ACF*A.CFTHK;
        volII = A.PBEDVOID*PI*((A.CFRAD-A.CFTHK)*(A.CFRAD-A.CFTHK)-(A.CFRAD- \
                    A.CFTHK-A.PBEDTHK)*(A.CFRAD-A.CFTHK-A.PBEDTHK))*A.LENGTH;
        volIII = AoutIII*A.LENGTH + AHF*A.HFTHK;
        transtime=0;
        N = 0;
 for (time = 0; time <= D.t_DELAY+D.delta_t; time += D.delta_t)
 {
        powerdensity = D.INIT_PRDEN;
        W = W1;
        Qbed = powerdensity * 1000000000 * PI * ((A.CFRAD - A.CFTHK) *   \
                    (A.CFRAD - A.CFTHK) - (A.CFRAD - A.CFTHK - A.PBEDTHK) * \
                    (A.CFRAD - A.CFTHK - A.PBEDTHK)) * A.LENGTH;
        HinI1 = h2H(D.INIT_TIN);
        PinI = D.INIT_PIN;
        PinII = D.INIT_PIN;
        PinIII = D.INIT_POUT;
        Pout = D.INIT_POUT;
        Tin = D.INIT_TIN;
        Tout = Tin;
        do
        {
                W = W1;
                Hout1 = Qbed/W + HinI1;
                do
                {
                        Htemp = h2H(Tout);
                        Herror = Hout1 - Htemp;
                        Tout = Tout + Herror/100000;
                }   while ( Herror/10000 > .001 );
/*  Set up all variables */
        TbarII = .5 * (Tin + Tout);
        PbarI = .5 * (PinI + PinII);
        PbarII = .5 * (PinII + PinIII);
        PbarIII = .5 * (PinIII + Pout);
        pbarI = h2density(Tin, PbarI);
        pbarII = h2density(TbarII, PbarII);
        pbarIII= h2density(Tout, PbarIII);
        uin = h2viscosity(Tin);
        ubarII = h2viscosity(TbarII);
        uout = h2viscosity(Tout);
        pinI = h2density(Tin, PinI);
```

```
        pinII = h2density(Tin, PinII);
        pinIII = h2density(Tout, PinIII);
        pout = h2density(Tout, Pout);
/*  Inlet Resistance Calculations */
        ReI = W*Deq_in / (uin*AinI);
        RIf = .138*pow(ReI, -.151)*(A.LENGTH/Deq_in) / (2*pbarI*AinI*AinI);
        RIa = (1/(AinII*AinII) - 1/(AinI*AinI)) / (2*pbarI);
        RIk = (1-(AinI/AoutI))*(1-(AinI/AoutI)) / (2*pbarI*AinI*AinI)    \
                + .5 * (1-(ACF/AoutI))*(1-(ACF/AoutI)) / (2*pinII*ACF*ACF)  \
                + .5 * (1-(AinII/ACF))*(1-(AinII/ACF)) / (2*pinII*AinII*AinII);
        RIm = A.MFIN / (pbarI*AinI*AinI);
        RIm = RIm + A.MFIN/(pbarI*AoutI*AoutI);
        RCF = 150*uin*(1-A.CFPOR)*(1-A.CFPOR)/(A.CFDp*A.CFDp*pinII*AoutI*W* \
                A.CFPOR*A.CFPOR*A.CFPOR)\
                + 1.75*(1-A.CFPOR)/(A.CFDp*pinII*AoutI*AoutI*A.CFPOR*A.CFPOR* \
                A.CFPOR);
        RCF = RCF * A.CFTHK;
        RI = RIf/2 + RIa + RIk + RIm + RCF;
/*  Resistance Calculations for the Particle Bed                */
        RII = 150*ubarII*(1-A.PBEDVOID)*(1-A.PBEDVOID)/(A.PBEDDp*A.PBEDDp* \
                pbarII*AbarII*W*A.PBEDVOID*A.PBEDVOID)\
                + 1.75*(1-A.PBEDVOID)/(A.PBEDDp*pbarII*AbarII*AbarII*A.PBEDVOID);
        RII = RII * A.PBEDTHK;
        RIIa = ( 1/(pinIII*AinIII) - 1/(pinII*AinII) )*((AinII+AinIII)/ \
                        (2*AinII*AinIII));
        RII = RII + RIIa;
/*  Resistance for Exit Plenum                                  */
        ReIII = W*Deq_out / (uout*AoutIII);
        RIIIf = .138*pow(ReIII, -.151)*(A.LENGTH/Deq_out)/(2*pbarIII*AoutIII\
                *AoutIII);
        RIIIa = (1/(AoutIII*AoutIII)-1/(AoutII*AoutII))/(2*pbarIII);
        RIIIk = .5*(1-AHF/AoutII)*(1-AHF/AoutII)/(2*pinIII*AHF*AHF)\
                + (1-AHF/AinIII)*(1-AHF/AinIII)/(2*pinIII*AHF*AHF)  \
                + (1-AoutIII/AinIII)*(1-AoutIII/AinIII)/(2*pbarIII*\
                AoutIII*AoutIII);
        mfout = A.MFOUT;
        RIIIm = mfout / (pbarIII*AoutIII*AoutIII) + mfout / (pbarIII\
                *AinIII*AinIII);
        RIII = RIIIf/2 + RIIIa + RIIIk + RIIIm;
        RTOTAL = RI + RII + RIII;
        W1 = (PinI - Pout)*1000/RTOTAL;
        W1 = sqrt(W1);
        Werror = W - W1;
        Werror = fabs(Werror);
        PDROPI1=RI*W1*W1/1000;
        PDROPII1=RII*W1*W1/1000;
        PDROPIII1=RIII*W1*W1/1000;
        PinII=PinI-PDROPI1;
        PinIII=Pout+PDROPIII1;
        PbarI=.5*(PinI+PinII);
        PbarII=.5*(PinII+PinIII);
        PbarIII=.5*(PinIII+Pout);
        vout = W1/(pout*AoutIII);
        power = powerdensity;
        } while(Werror > .0000001);
```

```
        HinI2=HinI1;
        HinII1=HinI1;
        HinII2=HinII1;
        HinIII1=Hout1;
        HinIII2=HinIII1;
        Hout2=Hout1;
        powerout = W*(Hout2-HinI2);
        DD = fmod(N, EE);
        if (DD == 0)
        {
                W1=W1*1000;
                fprintf(out,"%.4f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n", \
                        time, power, W1, Tin, TbarII, Tout, powerout,       \
                        PinI, PinII, PinIII, Pout, vout);
                printf("time=%f\tTin=%f\tTout=%f\tW=%f\n", time, Tin, Tout, W1);
                W1=W1/1000;
        }
        DD = 0;
        N += 1.0;
        if (Tout > 2700)
                goto meltdown;

  }
        PinI1 = PinI;
        PinII1 = PinII;
        PinIII1 = PinIII;
        Pout1 = Pout;
        PbarI1 = PbarI;
        PbarII1 = PbarII;
        PbarIII1 = PbarIII;
        ma1 = ((C.L3RAD*C.L3RAD*C.L3RAD)-(C.L2RAD*C.L2RAD*C.L2RAD))*C.L3DEN/\
                (3*C.L3RAD*C.L3RAD);
        ma2 = ((C.L2RAD*C.L2RAD*C.L2RAD)-(C.L1RAD*C.L1RAD*C.L1RAD))*C.L2DEN/\
                (3*C.L3RAD*C.L3RAD);
        ma3 = ((C.L1RAD*C.L1RAD*C.L1RAD)-
(C.FUELRAD*C.FUELRAD*C.FUELRAD))*C.L1DEN/\
                (3*C.L3RAD*C.L3RAD);
        ma4 = (C.FUELRAD*C.FUELRAD*C.FUELRAD)*C.FUELDEN / (3*C.L3RAD*C.L3RAD);
        mat = ma1+ma2+ma3+ma4;
        pfuelave=(1/( 1.3333*PI*pow(C.L3RAD,3) ) )*( C.FUELDEN*pow(C.FUELRAD,3)\
                + C.L1DEN*( pow(C.L1RAD,3)-pow(C.FUELRAD,3) ) + C.L2DEN*( pow(C.L2RAD,3)\
                -pow(C.L1RAD,3) ) + C.L3DEN*( pow(C.L3RAD,3)-pow(C.L2RAD,3) ) );
        Cpbar = (ma1*C.L3Cp + ma2*C.L2Cp + ma3*C.L1Cp + ma4*C.FUELCp)/mat;
        U1 = C.L3RAD*C.L2RAD*C.L3K/(C.L3RAD*C.L3RAD*(C.L3RAD-C.L2RAD));
        U2 = C.L2RAD*C.L1RAD*C.L2K/(C.L3RAD*C.L3RAD*(C.L2RAD-C.L1RAD));
        U3 = C.L1RAD*C.FUELRAD*C.L1K/(C.L3RAD*C.L3RAD*(C.L1RAD-C.FUELRAD));
        U4 = 2*C.FUELK/(3*C.L3RAD);
        UT = 1/(1/U1 + 1/U2 + 1/U3 + 1/U4);
        f1 = UT/U1;
        f2 = UT/U2 + f1;
        f3 = UT/U3 + f2;
        fbar = (ma1*C.L3Cp*f1 + ma2*C.L2Cp*(f1+f2) + ma3*C.L1Cp*(f2+f3) + \
                ma4*C.FUELCp*(f3+1))/(2*mat*Cpbar);
        Vcv=volII/A.PBEDVOID;
        Av=6*(1-A.PBEDVOID)/A.PBEDDp;
        TinI=Tin;
        Tf1 = TbarII;
        TinII=TinI;
```

```
        TinIII=Tout;
        deltat=D.delta_t;
        N = 0.0;
        for (time=D.t_DELAY; time < (D.t_DELAY+D.DUR_PRDEN+D.t_AFTER); \
                time += deltat)
        {
                powerdensity = D.INIT_PRDEN + ((D.FIN_PRDEN - D.INIT_PRDEN) \
                                * transtime / D.DUR_PRDEN);
                if (powerdensity >= D.FIN_PRDEN)
                        powerdensity = D.FIN_PRDEN;
                k = h2heatxfer(TbarII);
                Pr = h2prandle(TbarII);
                ReII = (W*6)/(AbarII*A.PBEDVOID*ubarII*Av);
                Nu = .8*pow(ReII, .7)*pow(Pr, .33);
                h = Nu*k*2/A.PBEDTHK;
                UBAR = UT*h/(fbar*h+UT);
                power = powerdensity;
                Qs = power*1000000000/Av;
                BB=mat*Cpbar*Vcv*Av;
                AA=UBAR*Vcv*Av;
                if (transtime == 0);
                        Tf1=Qs*Vcv*Av/(AA) + TbarII;
                if (transtime > 1)
                {
                        if (D.DUR_TIN > 0)
                        {
                        TinI=D.INIT_TIN+(D.FIN_TIN - D.INIT_TIN)*(transtime/D.DUR_TIN);
                        HinI2=h2H(TinI);
                        }
                        if (transtime >= D.DUR_TIN)
                        {
                        TinI = D.FIN_TIN;
                        HinI2 = h2H(TinI);
                        }
                }
                if (D.DUR_PIN > 0)
                        PinI1=D.INIT_PIN+(D.FIN_PIN-D.INIT_PIN)*(transtime/D.DUR_PIN);
                if (transtime >= D.DUR_PIN)
                        PinI1 = D.FIN_PIN;
                if (D.DUR_POUT > 0)
                        Pout1=D.INIT_POUT+(D.FIN_POUT-D.INIT_POUT)*(transtime/D.DUR_POUT);
                if (transtime >= D.DUR_POUT)
                        Pout1 = D.FIN_POUT;
/*  Control Volume I */
                M1 = pinI*volI;
                I1 = .5*A.LENGTH/AinI+ (A.OUTRAD-A.CFRAD)/AoutI + A.CFTHK/ACF;
                W2=(W*I1+deltat*(PinI1-PinII1)*1000)/(I1+deltat*RI*W);
                HinII2 = (HinII1 + (deltat/M1)*((volI*(PbarI1-PbarI)*1000/deltat) + \
                                W*HinI2 + 0.00*Qbed))/(1+deltat*W/M1);
/*  Control Volume II */
                Tf2 = (Tf1 + deltat*(Qs*Vcv*Av + AA*TbarII)/BB)/(1 + deltat*AA/BB);
                Qbed = AA*(Tf2-TbarII);
                Tf1 = Tf2;
                M2 = (pbarII*volII + pfuelave*volII*(1-A.PBEDVOID))/2;
                I2 = A.PBEDTHK/AbarII;
                W2=(W*I2+deltat*(PinII1-PinIII1)*1000)/(I2+deltat*RII*W);
                HinIII2 = (HinIII1 + (deltat/M2)*((Vcv*(PbarII1-PbarII)*1000/deltat) \
                                + W*HinII2 + 1.00*Qbed ))/(1+deltat*W/M2);
```

```
/*  Control Volume III */
            M3 = pinIII*volIII;
            I3 = A.HFTHK/AoutII + (A.CFRAD-A.CFTHK-A.PBEDTHK \
                  -A.HFTHK)/AinIII + .5*A.LENGTH/AoutIII;
            W2=(W*I3+deltat*(PinIII1-Pout1)*1000)/(I3+deltat*RIII*W);
            Hout2 = (Hout1 + (deltat/M3)*( (volIII*(PbarIII1-PbarIII)*1000/deltat) \
                  + W*HinIII2 + 0.00*Qbed ))/(1+deltat*W/M3);


/* ********************************************************************** */
            PbarI = PbarI1;
            PbarII = PbarII1;
            PbarIII = PbarIII1;
            PinI = PinI1;
            Pout = Pout1;
            PinII = PinII1;
            PinIII = PinIII1;
      do
       {
            Htemp = h2H(TinI);
            Herror = HinI2 - Htemp;
            TinI = TinI + Herror/100000;
       } while ( Herror/100000 > .001 );

      do
       {
            Htemp = h2H(Tout);
            Herror = Hout2 - Htemp;
            Tout = Tout + Herror/100000;
       } while ( Herror/100000 > .001 );
      do
       {
            Htemp = h2H(TinII);
            Herror = HinII2 - Htemp;
            TinII = TinII + Herror/100000;
       } while ( Herror/100000 > .001 );
      do
       {
            Htemp = h2H(TinIII);
            Herror = HinIII2 - Htemp;
            TinIII = TinIII + Herror/100000;
       } while ( Herror/100000 > .001 );


      TbarI=.5*(TinI+TinII);
      TbarII=.5*(TinII+TinIII);
      TbarIII=.5*(TinIII+Tout);
      HinI1 = HinI2;
      HinII1 = HinII2;
      HinIII1 = HinIII2;
      Hout1 = Hout2;
      pinI = h2density(TinI, PinI);
      pinII = h2density(TinII, PinII);
      pinIII = h2density(TinIII, PinIII);
      pout = h2density(Tout, Pout);
      pbarI = h2density(TbarI,PbarI);
      pbarII = h2density(TbarII, PbarII);
      pbarIII = h2density(Tout, PbarIII);
```

```
uin = h2viscosity(TbarI);
ubarII = h2viscosity(TbarII);
uout = h2viscosity(TbarIII);
/*  Inlet Resistance Calculations      */
ReI = (W*Deq_in)/(uin*AinI);
RIf = .138*pow(ReI, -.151)*(A.LENGTH/Deq_in)/(2*pbarI*AinI*AinI);
RIa = (1/(AinII*AinII) - 1/(AinI*AinI))/(2*pbarI);
RIk = (1-(AinI/AoutI))*(1-(AinI/AoutI)) / (2*pbarI*AinI*AinI)    \
        + .5 * (1-(ACF/AoutI))*(1-(ACF/AoutI)) / (2*pinII*ACF*ACF)  \
        + .5 * (1-(AinII/ACF))*(1-(AinII/ACF)) / (2*pinII*AinII*AinII);
RIm = A.MFIN / (pbarI*AinI*AinI);
RIm = RIm + A.MFIN/(pbarI*AoutI*AoutI);
RCF = 150*uin*(1-A.CFPOR)*(1-A.CFPOR)/(A.CFDp*A.CFDp*pinII*AoutI*W* \
        A.CFPOR*A.CFPOR*A.CFPOR)\
        + 1.75*(1-A.CFPOR)/(A.CFDp*pinII*AoutI*AoutI*A.CFPOR*A.CFPOR* \
        A.CFPOR);
RCF = RCF * A.CFTHK;
RI = RIf/2 + RIa + RIk + RIm + RCF;
/*  Resistance Calculations for the Particle Bed          */
RII = 150*ubarII*(1-A.PBEDVOID)*(1-A.PBEDVOID)/(A.PBEDDp*A.PBEDDp* \
        pbarII*AbarII*W*A.PBEDVOID*A.PBEDVOID)\
        + 1.75*(1-A.PBEDVOID)/(A.PBEDDp*pbarII*AbarII*AbarII*A.PBEDVOID);
RII = RII * A.PBEDTHK;
RIIa = ( 1/(pinIII*AinIII) - 1/(pinII*AinII) )*((AinII+AinIII)/ \
            (2*AinII*AinIII));
RII = RII + RIIa;
/*  Resistance for Exit Plenum                          */
ReIII = W*Deq_out / (uout*AoutIII);
RIIIf = .138*pow(ReIII, -.151)*(A.LENGTH/Deq_out)/(2*pbarIII*AoutIII\
        *AoutIII);
RIIIa = (1/(AoutIII*AoutIII)-1/(AoutII*AoutII))/(2*pbarIII);
RIIIk = .5*(1-AHF/AoutII)*(1-AHF/AoutII)/(2*pinIII*AHF*AHF)  \
        + (1-AHF/AinIII)*(1-AHF/AinIII)/(2*pinIII*AHF*AHF)  \
        + (1-AoutIII/AinIII)*(1-AoutIII/AinIII)/(2*pbarIII* \
        AoutIII*AoutIII);
mfout = A.MFOUT;
RIIIm = mfout / (pbarIII*AoutIII*AoutIII) + mfout / (pbarIII\
        *AinIII*AinIII);
RIII = RIIIf/2 + RIIIa + RIIIk + RIIIm;
/*  TOTALS                                      */
RTOTAL = RI + RII + RIII;
W1 = (PinI - Pout)*1000/RTOTAL;
W1 = sqrt(W1);
W = W1;
PDROPI1 = RI*W*W/1000;
PDROPII1 = RII*W*W/1000;
PDROPIII1 = RIII*W*W/1000;
PinII1 = PinI1 - PDROPI1;
PinIII1 = Pout1 + PDROPIII1;
PbarI1 = .5*(PinI1+PinII1);
PbarII1 = .5*(PinII1+PinIII1);
PbarIII1 = .5*(PinIII1+Pout1);
vout = W/(pout*AoutIII);
powerout = W*(Hout2-HinI2);
DD = fmod(N, EE);
if (DD == 0)
{
```

```
            W = W*1000;
            fprintf(out,"%.4f\t %.4f\t %f\t %.3f\t %.3f\t %.3f\t %.2f\t %.2f\t %.2f\t \
                        %.2f\t %.2f\t %.2f\t %.3f\n",                    \
                        time, power, W, TinI, TbarII, Tout, powerout,       \
                        PinI1, PinII1, PinIII1, Pout1, vout, Tf1);
            printf("time=%f\tTinI=%f\tTout=%f\tW=%f\n", time, TinI, Tout, W);
            W = W/1000;
        }
        DD = 0;
        N += 1.0;
        if (Tout > 2700)
            goto meltdown;
        transtime += deltat;
        }
        meltdown:
            if (Tout > 2700)
            {   printf("Tout > 2700");
                fprintf(out,"Temp exceeds 2700");
            }
}
/* ********************************************************************** */
void showdefaults( struct fuelelem *e_ptr )
{
        int num;
        float entry;
        int c;
        _clearscreen(0);
        printf("THE FOLLOWING ARE PARAMETERS FOR PPBR FUEL ELEMENT:\n\n");
        printf("1.              Outer Radius = %f m\n", e_ptr->OUTRAD);
        printf("2.            Cold Frit Radius = %f m\n", e_ptr->CFRAD);
        printf("3.         Cold Frit Thickness = %f m\n", e_ptr->CFTHK);
        printf("4.    Dia of Cold Frit Particle = %.8f m\n", e_ptr->CFDp);
        printf("5.          Cold Frit Porosity = %f\n", e_ptr->CFPOR);
        printf("6.       Particle Bed Thickness = %f m\n", e_ptr->PBEDTHK);
        printf("7.       Fuel Particle Diameter = %f m\n", e_ptr->PBEDDp);
        printf("8.   Particle Bed Void Fraction = %f\n", e_ptr->PBEDVOID);
        printf("9.          Hot Frit Thickness = %f m\n", e_ptr->HFTHK);
        printf("10.          Hot Frit Porosity = %f\n", e_ptr->HFPOR);
        printf("11.       Inlet Manifold Factor = %f\n", e_ptr->MFIN);
        printf("12.        Exit Manifold Factor = %f\n", e_ptr->MFOUT);
        printf("13.        Fuel Element Length = %f m\n", e_ptr->LENGTH);
        printf("\n\n\nCHANGE A PARAMETER? (1=Y or 0=N)");
        scanf( "%i", &c );
        if ( c == 0 )
            return;
        printf("\nPLEASE ENTER THE PARAMETER NUMBER AND VALUE (eg 12, 1.705):\n");
        scanf( "%i, %f", &num, &entry );
        switch (num)
        {
            case 1:
                e_ptr->OUTRAD = entry;
                break;
            case 2:
                e_ptr->CFRAD = entry;
                break;
            case 3:
                e_ptr->CFTHK = entry;
                break;
```

```
                case 4:
                        e_ptr->CFDp = entry;
                        break;
                case 5:
                        e_ptr->CFPOR = entry;
                        break;
                case 6:
                        e_ptr->PBEDTHK = entry;
                        break;
                case 7:
                        e_ptr->PBEDDp = entry;
                        break;
                case 8:
                        e_ptr->PBEDVOID = entry;
                        break;
                case 9:
                        e_ptr->HFTHK = entry;
                        break;
                case 10:
                        e_ptr->HFPOR = entry;
                        break;
                case 11:
                        e_ptr->MFIN = entry;
                        break;
                case 12:
                        e_ptr->MFOUT = entry;
                        break;
                case 13:
                        e_ptr->LENGTH = entry;
                        break;
                default:
                        printf("\n\nTRY ANOTHER PARAMETER\n");
                        _clearscreen(0);
                        break;
        }
        return;
}
/* ********************************************************************** */
void fuelelemsum ( struct fuelelem *e_ptr, struct fuelelem *f_ptr )
{
        _clearscreen (0);
        printf ("PARAMETER\t\t\tELEMENT A\t\tBASELINE\n\n");
        printf("1. Outer Radius\t\t\t%f\t\t%f m\n", e_ptr->OUTRAD, f_ptr->OUTRAD);
        printf("2. Cold Frit Radius\t\t%f\t\t%f m\n", e_ptr->CFRAD, f_ptr->CFRAD);
        printf("3. Cold Frit Thickness\t\t%f\t\t%f m\n", e_ptr->CFTHK, f_ptr->CFTHK);
        printf("4. Dia of Cold Frit Particle\t%.8f\t\t%.8f m\n", e_ptr->CFDp, f_ptr->CFDp);
        printf("5. Cold Frit Porosity\t\t%f\t\t%f\n", e_ptr->CFPOR, f_ptr->CFPOR);
        printf("6. Particle Bed thickness\t%f\t\t%f m\n", e_ptr->PBEDTHK, f_ptr->PBEDTHK);
        printf("7. Fuel Particle Diameter\t%f\t\t%f m\n", e_ptr->PBEDDp, f_ptr->PBEDDp);
        printf("8. Particle Bed Void Fraction\t%f\t\t%f\n", e_ptr->PBEDVOID, f_ptr->PBEDVOID);
        printf("9. Hot Frit Thickness\t\t%f\t\t%f m\n", e_ptr->HFTHK, f_ptr->HFTHK);
        printf("10.Hot Frit Porosity\t\t%f\t\t%f\n", e_ptr->HFPOR, f_ptr->HFPOR);
        printf("11.Inlet Manifold Factor\t%f\t\t%f\n", e_ptr->MFIN, f_ptr->MFIN);
        printf("12.Exit Manifold Factor\t\t%f\t\t%f\n", e_ptr->MFOUT, f_ptr->MFOUT);
        printf("13.Length of Element\t\t%f m\t\t%f\n", e_ptr->LENGTH, f_ptr->LENGTH);
        printf("\n\nHIT ANY KEY TO CONTINUE");
        getch();
```

```
}
/* ********************************************************************** */
void showfuelpart ( struct fuelpart *g_ptr )
{
        int num;
        float entry;
        int c;
        _clearscreen(0);
        printf("THE FOLLOWING ARE PARAMETERS FOR THE FUEL PARTICLE\n");
        printf("1.     Fuel Material = %s\n", g_ptr->FUELMAT);
        printf("2.  Layer 1 Material = %s\n", g_ptr->L1MAT);
        printf("3.  Layer 2 Material = %s\n", g_ptr->L2MAT);
        printf("4.  Layer 3 Material = %s\n", g_ptr->L3MAT);
        printf("5.    Radius of Fuel = %f m\n", g_ptr->FUELRAD);
        printf("6. Radius of Layer 1 = %f m\n", g_ptr->L1RAD);
        printf("7. Radius of Layer 2 = %f m\n", g_ptr->L2RAD);
        printf("8. Radius of Layer 3 = %f m\n", g_ptr->L3RAD);
        printf("9.   Density of Fuel = %f kg/m3\n", g_ptr->FUELDEN);
        printf("10.Density of Layer 1 = %f kg/m3\n", g_ptr->L1DEN);
        printf("11.Density of Layer 2 = %f kg/m3\n", g_ptr->L2DEN);
        printf("12.Density of Layer 3 = %f kg/m3\n", g_ptr->L3DEN);
        printf("13.       Cp of Fuel = %f J/kgK\n", g_ptr->FUELCp);
        printf("14.    Cp of Layer 1 = %f J/kgK\n", g_ptr->L1Cp);
        printf("15.    Cp of Layer 2 = %f J/kgK\n", g_ptr->L2Cp);
        printf("16.    Cp of Layer 3 = %f J/kgK\n", g_ptr->L3Cp);
        printf("17.       k for Fuel = %f W/m2K\n", g_ptr->FUELK);
        printf("18.    k for Layer 1 = %f W/m2K\n", g_ptr->L1K);
        printf("19.    k for Layer 2 = %f W/m2K\n", g_ptr->L2K);
        printf("20.    k for Layer 3 = %f W/m2K\n", g_ptr->L3K);
        printf("\nCHANGE A PARAMETER? (1=Y or 0=N)");
        scanf( "%i", &c );
        if ( c == 0 )
                return;
        printf("\nENTER PARAMETER NUMBER ( NOT 1-4 ), VALUE (eg 20, 40)\n");
        scanf( "%i, %f", &num, &entry );
        switch (num)
        {
                case 5:
                        g_ptr->FUELRAD = entry;
                        break;
                case 6:
                        g_ptr->L1RAD = entry;
                        break;
                case 7:
                        g_ptr->L2RAD = entry;
                        break;
                case 8:
                        g_ptr->L3RAD = entry;
                        break;
                case 9:
                        g_ptr->FUELDEN = entry;
                        break;
                case 10:
                        g_ptr->L1DEN = entry;
                        break;
                case 11:
                        g_ptr->L2DEN = entry;
                        break;
```

```
            case 12:
                    g_ptr->L3DEN = entry;
                    break;
            case 13:
                    g_ptr->FUELCp = entry;
                    break;
            case 14:
                    g_ptr->L1Cp = entry;
                    break;
            case 15:
                    g_ptr->L2Cp = entry;
                    break;
            case 16:
                    g_ptr->L3Cp = entry;
                    break;
            case 17:
                    g_ptr->FUELK = entry;
                    break;
            case 18:
                    g_ptr->L1K = entry;
                    break;
            case 19:
                    g_ptr->L2K = entry;
                    break;
            case 20:
                    g_ptr->L3K = entry;
                    break;
            default:
                    printf("CANNOT CHANGE THIS PARAMETER...SORRY.\n");
                    getch();
                    break;
    }
    return;
}
/* ********************************************************************** */
void showconditions ( struct conditions *h_ptr )
{
    int num;
    float entry;
    int c;
    _clearscreen(0);
    printf("THE FOLLOWING INITIAL AND FINAL CONDITIONS ARE SET\n\n");
    printf("1.       Initial Inlet Temperature = %f K\n", h_ptr->INIT_TIN);
    printf("2.         Final Inlet Temperature = %f K\n", h_ptr->FIN_TIN);
    printf("3.  Duration of Temperature Change = %f sec\n", h_ptr->DUR_TIN);
    printf("4.          Initial Inlet Pressure = %f KPa\n", h_ptr->INIT_PIN);
    printf("5.            Final Inlet Pressure = %f KPa\n", h_ptr->FIN_PIN);
    printf("6.     Duration of Pressure Change = %f sec\n", h_ptr->DUR_PIN);
    printf("7.        Initial Outlet Pressure = %f KPa\n", h_ptr->INIT_POUT);
    printf("8.          Final Outlet Pressure = %f KPa\n", h_ptr->FIN_POUT);
    printf("9.     Duration of Pressure Change = %f sec\n", h_ptr->DUR_POUT);
    printf("10.         Initial Power Density = %f GW/m3\n", h_ptr->INIT_PRDEN);
    printf("11.           Final Power Density = %f GW/m3\n", h_ptr->FIN_PRDEN);
    printf("12.Duration of Power Density Change = %f sec\n", h_ptr->DUR_PRDEN);
    printf("13.        Time Delay to Transient = %f sec\n", h_ptr->t_DELAY);
    printf("14.          Time After Transient = %f sec\n", h_ptr->t_AFTER);
    printf("15.                   Time Step = %f sec\n", h_ptr->delta_t);
    printf("16.  Initial Guess at Mass Flowrate = %f kg/sec\n", h_ptr->INIT_W);
```

```
printf("\nCHANGE A PARAMETER? (1=Y or 0=N)");
scanf( "%i", &c );
if ( c == 0 )
        return;
printf("\nENTER PARAMETER NUMBER, VALUE (eg 20, 40)\n");
scanf( "%i, %f", &num, &entry );
switch (num)
{
        case 1:
                h_ptr->INIT_TIN = entry;
                break;
        case 2:
                h_ptr->FIN_TIN = entry;
                break;
        case 3:
                h_ptr->DUR_TIN = entry;
                break;
        case 4:
                h_ptr->INIT_PIN = entry;
                break;
        case 5:
                h_ptr->FIN_PIN = entry;
                break;
        case 6:
                h_ptr->DUR_PIN = entry;
                break;
        case 7:
                h_ptr->INIT_POUT = entry;
                break;
        case 8:
                h_ptr->FIN_POUT = entry;
                break;
        case 9:
                h_ptr->DUR_POUT = entry;
                break;
        case 10:
                h_ptr->INIT_PRDEN = entry;
                break;
        case 11:
                h_ptr->FIN_PRDEN = entry;
                break;
        case 12:
                h_ptr->DUR_PRDEN = entry;
                break;
        case 13:
                h_ptr->t_DELAY = entry;
                break;
        case 14:
                h_ptr->t_AFTER = entry;
                break;
        case 15:
                h_ptr->delta_t = entry;
                break;
        case 16:
                h_ptr->INIT_W = entry;
                break;
        default:
                printf("CANNOT CHANGE THIS PARAMETER...SORRY.\n");
```

```
                getch();
                break;
        }
        return;
}
/* ********************************************************************** */
double h2density(double xx, double yy)
{
        float dens;
        dens = .08185 * 300 / xx * yy / 101.325;
        return dens;
}
/* ********************************************************************** */
double h2viscosity(double xx)
{
        float visc;
        visc = .000008963 * pow( xx/300, .6733);
        return visc;
}
/* ********************************************************************** */
double h2heatxfer(double xx)
{
        float tem1;
        int i;
        float tem2;
        float heat[45] = {0,.0362, .0665, .0981, .1282, .1561, .182, .206, .228,\
                        .251, .272, .292, .315, .333, .351, .3665, .384, .398,\
                        .412, .426, .44, .452, .464, .476, .488, .500, .512, \
                        .524, .536, .548, .560, .572, .584, .596, .608, .62, \
                        .632, .644, .656, .668, .680, .692, .704, .716, .728 };
        tem2 = xx / 50.0;
        i = floor(tem2);
        tem1 = heat[i] + (heat[i+1] - heat[i]) * ((xx - (i * 50)) / 50 );
        return tem1;
}
/* ********************************************************************** */
double h2cp(double xx)
{
        float temcp;
        if ( xx <= 420 )
                temcp = 4.1868 * 1000 * (1.5395 + .0150825 * xx - 4.02449e-5 * \
                        xx * xx + 3.63544e-8 * xx * xx * xx);
        else
                temcp = 4.1868 * 1000 * (3.58927 - 5.55096e-4 * xx + 6.94235e-7 * xx \
                        * xx - 1.45155e-10 * xx * xx * xx);
        return temcp;
}
/* ********************************************************************** */
float h2H(float xx)
{
        float temH;
        float temH1;
        temH=0;
        temH1=0;
        if (xx <= 420)
        {
        temH=4.1868*1000*(1.5395*xx + .0150825*xx*xx/2 - 4.02449e-5*        \
                xx*xx*xx/3 + 3.63544e-8*xx*xx*xx*xx/4);
```

```
        temH=temH - 4.1868*1000*(1.5395*50 + .0150825*50*50/2 - 4.02449e-5*  \
                50*50*50/3 + 3.63544e-8*50*50*50*50/4);
        }
        else
        {
        temH1=4.1868*1000*(3.58927*xx - 5.55096e-4*xx*xx/2 + 6.94235e-7*     \
                xx*xx*xx/3 - 1.45155e-10*xx*xx*xx*xx/4);
        temH1=temH1 - 4.1868*1000*(3.58927*420 - 5.55096e-4*420*420/2 +      \
                6.94235e-7*420*420*420/3 - 1.45155e-10*420*420*420*420/4);
        temH=temH1 + 4.1868*1000*(1.5395*420+.0150825*420*420/2-4.02449e-5*  \
                420*420*420/3 + 3.63544e-8*420*420*420*420/4);
        temH=temH - 4.1868*1000*(1.5395*50 + .0150825*50*50/2 - 4.02449e-5*  \
                50*50*50/3 + 3.63544e-8*50*50*50*50/4);
        }
        return temH;
}
/* ********************************************************************** */
float h2prandle(float xx)
{
        float tem1;
        int i;
        float tem2;
        float pran[23] =    {.000, .712, .719, .706, .690, .675, .664,   \
                                        .659, .664, .676, .686, .703, .715, .733,   \
                                        .748, .763, .778, .793, .808, .823, .838,   \
                                        .853, .868};
        tem2 = xx / 100.0;
        i = floor(tem2);
        tem1 = pran[i] + (pran[i+1] - pran[i]) * ((xx - (i * 100)) / 100 );
        return tem1;
}
/* ********************************************************************** */
```

APPENDIX E:

TABLE OF SYMBOLS AND ACRONYMS

A cross sectional area perpendicular to flow

A_V particle surface area per unit volume

A_{small} smaller areas involved in expansion/contraction

$\overline{C}_p$ average particle specific heat

d_p effective particle diameter

D_P particle diameter

D_{eq} equivalent channel diameter

ε void fraction or porosity

F_a equivalent pressure drop due to acceleration

$(F_a)_{PBED}$ spatial acceleration losses in the particle bed

F_{eq} equivalent resistive pressure drop

F_f equivalent pressure drop due to friction

F_M equivalent pressure drop due to the manifold effect

F_{Pbed} equivalent pressure drop due particle bed effects

F_k equivalent loss due to sudden expansion/contraction

f_f friction factor

h heat transfer coefficient

H specific enthalpy

I_{CV} control volume inertia

k coolant heat conductivity

K_k, K_e, K_c expansion/contraction coefficients

L_b average length of travel through particle bed

L_p average length of travel through plenums

m_a particle mass per unit surface area

M mass within the control volume

Nu_p particle bed Nusselt number

P_{wet} wetted perimeter

P average pressure within the control volume

ΔP_{CV} pressure drop across control volume due to the resistance

q heat energy

q'' heat generation per surface area

Q_I interphase heat transfer

Re Reynolds Number

R_{Total} total equivalent resistance

S_p particle surface area per unit volume

$\overline{T}$ effective fuel particle temperature

T_G bulk temperature of coolant

T_S fuel particle surface temperature

$\overline{U}$ effective over all heat transfer coefficient

μ viscosity

$\vec{v}_o$ superficial velocity

V_{CV} size of the control volume

W mass flowrate

ACRONYMS

ALMCR Advanced Liquid Metal Cooled Reactor

BNL Brookhaven National Laboratory

HTGR High Temperature Gas Cooled Reactor

IMEO Initial Mass in Earth Orbit

INEL Idaho National Engineering Laboratory

LEO Low Earth Orbit

NERVA Nuclear Engine for Rocket Vehicle Application

NTR Nuclear Thermal Rocket

PBR Particle Bed Reactor

PFNTR Pressure Fed Nuclear Thermal Rocket

SNAP Space Nuclear Applications Program

SNL Sandia National Laboratory

REFERENCES

A-1 J. A. Angelo, Jr. and David Buden, Space Nuclear Power, Orbit Book Co., Malabar, FL. 1985.

B-1 R. Benenati, K. J. Araj, F. L. Horn, "Thermal-Hydraulic Considerations for Particle Bed Reactors," Space Nuclear Power Systems 1987, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1988.

B-2 T. E. Bolts, J. R. Powell, F. L. Horn, O. W. Lazareth, "A Bimodal Reactor for High and Low Space Power," Space Nuclear Power Systems 1984, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1985.

B-3 R. A. Bajura, "A Model for Distribution in Manifolds," Journal of Engineering for Power, Trans. ASME, Vol. 93, No. 1, Jan 1971.

B-4 R. A. Bajura, V. F. LeRose, L. E. Williams, "Fluid Distribution in Combining, Dividing and Reverse Flow Manifolds," ASME paper 73-Pwr-1, 1972.

B-5 R. A. Bajura, E. H. Jones, "Flow Distribution Manifolds," Journal of Fluids Engineering, Trans. ASME, Vol. 98, Dec 1976.

B-6 R. W. Bird, W. E. Stewart, E. N. Lightfoot, Transport Phenomena, John Wiley and Sons, New York, 1960.

C-1 S. H. Crandall, Engineering Analysis, A Survey of Numerical Procedures, Robert E. Krieger Publishing Co., Malabar, FL, 1986.

D-1 A. B. Datta, A. K. Majumdar, "Flow Distribution in Parallel and Reverse Flow Manifolds," Int. J. of Heat & Fluid Flow Manifolds,Vol. 2, No. 4, 1980.

E-1 E. R. G. Eckert, R. M. Drake, Jr., Analysis of Heat and Mass Transfer, Hemisphere Publishing Corp., New York, 1987.

E-2 S. Ergun, "Fluid Flow Through Packed Columns," Chem. Eng. Prog, Vol. 48, pg 89-94, 1952.

G-1 D. R. Gallup, M. W. Edenburn, "Comparison of Open and Closed, Burst Mode, Multimegawatt Space Power Systems," Space Nuclear Power Systems 1987, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1988.

H-1 F. L. Horn, J. R. Powell, J. M. Savino, "Particulate Fuel Bed Tests," Space Nuclear Power Systems 1985, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1986.

H-2 F. L. Horn, J. R. Powell, O. W. Lazareth, "Particle Bed Reactor Propulsion Vehicle Performance and Characteristics as an Orbital Transfer Rocket," Space Nuclear Power Systems 1986, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1987.

L-1 O. W. Lazareth, K. J. Araj, F. L. Horn, H. Ludewig, J. R. Powell, "Analysis of the Start-up and Control of a Particle Bed Reactor," Space Nuclear Power Systems 1987, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1988.

L-2 C. F. Leyse, W. W. Madsen, J. H. Ransthaler, B. G. Schntzler, "Pressure Fed Nuclear Thermal Rockets for Space Missions," Idaho National Engineering Memo for US Department of Energy, Aug 1989.

L-3 H. Ludewig, O. Lazareth, S. Mughabghab, K. Perkins, J. R. Powell, "Small Propulsion Reactor Design Based on Particle Bed Reactor," Transactions of the Sixth Symposium on Space Nuclear Power Systems (Summaries), M. S. El-Genk and M. D. Hoover eds., University of New Mexico, Albuquerque, NM. 1989.

M-1 J. D. Malloy, R. P. Potekhen, "A Conceptual Study of the Use of a Particle Bed Reactor Nuclear Propulsion Module for the Orbital Maneuvering Vehicle," Babcock and Wilcox Technical Paper NPD-TP-1544, Aug 1989.

M-2 J. E. Meyer, "Some Physical and Numerical Considerations for the SSC-S Code," Brookhaven National Laboratory Report BNL-NUREG-50913, Sep 1978.

M-3 A. K. Majumdar, "Mathematical Modelling of Flows in Dividing and Combining Manifolds," Appl. Math Modelling, Vol. 4, Dec 1980.

P-1 B. L. Pierce, "Gas Cooled Reactor Concepts for Space Applications," Space Nuclear Power Systems 1984, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1985.

P-2 J. R. Powell, F. L. Horn, "High Power Density Reactors Based on Direct Cooled Particle Beds," Space Nuclear Power Systems 1985, M. S. El-Genk and M. D. Hoover eds., Orbit Book Co., Malabar, FL. 1986.

R-1 W. C. Reynolds, "Thermodynamic Properties in SI; Graphs, Tables, and Computational Equations for Forty Substances," Mechanical Engineering Dept., Stanford Univ., 1979.

T-1 R. S. Tuddenham, Thermal Hydraulic Analysis of a Packed Bed Reactor Fuel Element, MIT, Department of Nuclear Engineering, MS and Nuclear Engineer Thesis, May 1989.

T-2 N. E. Todreas and M. S. Kazimi, **Nuclear Systems I Thermal Hydraulic Fundamentals**, Hemisphere Publishing, New York, 1990

V-1 M. E. Vernon, "PIPE Series Experiment Plan," Sandia National Laboratory, Albuquerque, NM, 1988.

V-2 G. J. Van Tuyle, T. C. Nepsee, J. G. Guppy, MINET Code Documentation, Brookhaven National Laboratory Report NUREG/CR-3668, BNL-NUREG-51742, February 1984.

Thesis
C27474 Casey
c.1 Thermal-hydraulic
transient analysis of a
packed particle bed
reactor fuel element.