MAGNETICALLY AUGMENTED CRYOGENIC REFRIGERATION

by

Gregory F. Nellis

B.S.M.E., University of Wisconsin
(1992)

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1995

Signature of Author_____
Department of Mechanical Engineering
December 15, 1994

Certified by_____
Joseph L. Smith, Jr.
Thesis Supervisor

Accepted by_____
Ain A. Sonin, Chairman
Departmental Graduate Committee
Department of Mechanical Engineering

MAGNETICALLY AUGMENTED CRYOGENIC REFRIGERATION

by

Gregory F. Nellis

Submitted to the Department of Mechanical Engineering
on October 15, 1994 in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

ABSTRACT

This thesis presents an analysis of a cryogenic refrigeration device. The device considered is a magnetically active regenerative refrigerator composed of a paramagnetic matrix (Gadolinium Gallium Garnet or GGG) and a compressible working fluid (Helium - 4). The device produces refrigeration in the 4.75° K to 12° K temperature range.

Magnetically active regenerative devices contain GGG and helium; both of which are thermodynamic working fluids. Currently, only the magnetic material is used as a refrigerant in these devices. The helium is typically maintained at constant pressure and used as a heat transfer medium. The purpose of this analysis is to investigate refrigeration cycles which allow both the magnetic material and the helium to undergo complete thermodynamic cycles.

By making several simplifying assumptions, an ideal model of the refrigeration device is presented. This model is used to demonstrate that a more effective refrigeration cycle can be obtained by thermodynamically utilizing helium and GGG.

A comprehensive numerical simulation is developed which models the major sources of irreversibility present in a magnetically active regenerator. This simulation is used to develop an effective refrigeration cycle and design an experimental prototype.

Thesis Supervisor : Professor Joseph L. Smith, Jr.
Title : Professor of Mechanical Engineering

ACKNOWLEDGMENTS

I would like to thank Professor Smith for his patient answers to all of my questions. Professor Brisson and Sangkwon Jeong also helped me on numerous occasions. The friendly members of the Cryogenic Engineering Lab made working there a rewarding experience, in particular Ed Ognibene has given me much good advice. I wish to thank my parents for their constant love and encouragement and my mother and father in-law for their continued support. Most of all, I would like to thank my wife Jill. She has been my strength and perspective throughout.

This work was sponsored by the Naval Surface Warfare Center, Carderock Division. Some of the calculations contained in this thesis were performed on the MIT CRAY-XMP.

TABLE OF CONTENTS

NOMENCLATURE

aa	: Specific Area (Surface Area per unit Volume)
A	: Magnetic Vector Potential
A	: Area
A_H	: Area Available for Heat Transfer
$\mathbb{B}$	: Magnetic Induction Vector
B	: Magnitude of the Magnetic Induction Vector
c	: Specific Heat Capacity
Cr	: Heat Capacity Ratio in Heat Exchanger Calculations
$c_{\mu oH}$	: Specific Heat Capacity at Constant Applied Field
c_p	: Specific Heat Capacity at Constant Pressure
D	: Tube Diameter
dmr	: Differential Mass Ratio (dM_{he}/M_{GGG})
d_p	: Diameter of Packed Bed Particles
dM_{he}	: Differential Mass of Helium
ds	: Differential Conductor Vector
dt	: Differential Time
dx	: Differential Length
E_a	: Applied Voltage
f	: Friction Factor
F	: Force Vector
ff	: Flow Fraction (Flow Process Applied Field Swing/Total Applied Field Swing)
h	: Specific Enthalpy
H	: Magnetic Field Intensity or Magnetic Excitation
hflux	: Cyclic Flux of Energy due to Bulk Flow of Helium
htc	: Heat Transfer Coefficient
i	: Current
k	: Thermal Conductivity
L	: Inductance
L	: Length
m	: Number of Axial Segments
M	: Mass
M	: Magnetization or Magnetic Moment per Unit Volume
mf	: Mass Flow Rate
$\overline{mf}$	: Average Mass Flow Rate
mr	: Mass Ratio (Mass of Helium/Mass of GGG)
n	: Number of Time Steps
N	: Number of Solenoid Turns
N_{tube}	: Number of Heat Exchanger Tubes
NTU	: Number of Transfer Units
Nu	: Nusselt Number
P	: Perimeter
P	: Pressure
Pr	: Prandtl Number

q : Heat Transfer Rate
Q : Heat Transfer
Q_{wi} : Loss of Refrigeration Caused by Inflow of Warm Helium
q' : Heat Transfer Rate per Unit Length
r : Ratio of Time Step to Length Step
r_{cr} : Critical Ratio of Time Step to Length Step to Ensure Numerical Stability
R : Gas Constant
Re : Reynolds Number
u : Specific Internal Energy
U : Total Internal Energy
s : Specific Entropy
S : Total Entropy
$sflux$: Cyclic Flux of Entropy due to Bulk Flow of Helium
$Sirr_p$: Irreversibility Generated by Mass Flow Through a Pressure Gradient
$Sirr_{ht}$: Irreversibility Generated by Heat Flow Through a Temperature Gradient
t : Time
T : Temperature
T_c : Temperature of Cold Side in Heat Exchanger Calculations
T_h : Temperature of Hot Side in Heat Exchanger Calculations
$T1$: Temperature Determined from First Law Equations
$T2$: Temperature Determined from Second Law Equations
T_{boil} : Temperature of Boiling
T_{exp} : Temperature of Helium After Adiabatic Expansion within the Displacer
T_{pc} : Cold Side Entrance Temperature Within Precoolant Regenerator
$\overline{T}_{superheat}$: Mass Average Temperature of Helium Leaving the Warm End
$\overline{T}_{subcool}$: Mass Average Temperature of Helium Leaving the Cold End
v : Specific Volume
v : Velocity
vf : Volumetric Flow Rate
V : Volume
V_o : Unobstructed Velocity (Volumetric Flow/Surface Area)
W : Work
wg : Nodal GGG Heat Transfer Temperature Weighting
$wgts$: Previous Time Step GGG Heat Transfer Temperature Weighting
wh : Nodal Helium Heat Transfer Temperature Weighting
$whts$: Previous Time Step Helium Heat Transfer Temperature Weighting
$whas$: Previous Axial Step Helium Heat Transfer Temperature Weighting
x : Axial Position
$\overline{x}$: Non-dimensional Axial Position
ΔS_{xs} : Excess Entropy Stored in Regenerator Relative to Steady State
Δt : Length of Time Step
Δx : Length of Axial Step
ε : Effectiveness
ε : Porosity (Void Volume/Total Volume)
γ : Pressure Ratio (High Imposed Pressure/Low Imposed Pressure)

λ : Field Constant of Superconducting Solenoid (Tesla/Amp)
μ : Absolute Viscosity
μ : Permeability
μ_o : Permeability of Free Space ($4\pi \times 10^{-7}$ H/m)
$\mu_o H$: Applied Magnetic Field (Tesla)
ν : Kinematic Viscosity
φ : Shape Factor (Specific Area/Specific Area of Sphere)
Φ : Magnetic Flux
ρ : Density
τ : Time
Θ : Non-dimensional Temperature

Subscripts

chx : Cold Heat Exchanger
cold : Refrigeration Space
com : Compression Process
cyc : Cycle
deg : Loss of Refrigeration
dewar : Cryogenic Dewar
dis : Displacer
exp : Expansion Process
he : Helium-4
high : Highest Value during a Cycle
inlet : Inlet Stroke
GGG : Gadolinium Gallium Garnet
low : Lowest Value during a Cycle
mag : Magnetic
mech : Mechanical
out : Outlet Process
warm : Warm Reservoir
whx : Warm Heat Exchanger
1 : Thermodynamic State after Compression/Adiabatic Demagnetization Process
2 : Thermodynamic State after Inlet Process
3 : Thermodynamic State after Expansion/Adiabatic Magnetization Process
4 : Thermodynamic State after Outlet Process

Superscripts

cyc : Summed over a Cycle
ht : Heat Transfer Interaction
i : Axial Node
j : Temporal Node
r : Representative Temperature (Temperature used to Evaluate Properties)

LIST OF FIGURES

Tables

I. INTRODUCTION

This chapter is divided into three sections. In the first section, the thermodynamics of a magnetic system are described and analogies between magnetic and compressible systems are explained. The second section details a brief history of magnetic refrigeration. The various types of magnetic coolers currently being developed are described. In the final section, magnetically active regenerative refrigeration is explained. The concept of a 'dual working fluid' or 'combined' cycle is described and contrasted with the simpler thermodynamic cycles currently being used to drive magnetically active devices.

1.1 The Magneto-Caloric Effect

A thermodynamic substance can change internal energy through both heat and work interactions. This is represented in a differential energy balance.

$$dU = T \cdot dS + dW \tag{1.1}$$

The first term corresponds to an inflow of heat and the second to an inflow of work. In general, work can flow in many forms (e.g. mechanical, electrical, magnetic, and chemical). Typically, only mechanical work is considered and Equation 1.1 is transformed into the familiar relation.

$$dU = T \cdot dS - P \cdot dV \tag{1.2}$$

For each independent, reversible interaction which may increase the energy contained in a system, a canonical conjugate property pair exists which completely describes that interaction.[1] Temperature and entropy form a canonical conjugate pair of properties that completely describes a reversible heat transfer interaction. Pressure and volume form a similar pair describing a reversible mechanical work transfer.

A paramagnetic material may increase in energy due to a reversible magnetic work interaction. The magnetic property pair required to describe a magnetic work transfer must be derived from electromagnetic principles.

The magnetic induction vector field (**B**) is defined by the force exerted on an element (ds) of linear conductor carrying a current (i).

$$i \cdot d\mathbf{s} \times \mathbf{B} = \mathbf{F} \tag{1.3}$$

In general, the magnetic induction vector field is determined by calculating the magnetic vector potential (**A**). In a vacuum, the magnetic vector potential created by a system of current carriers can be calculated by considering each individual conducting element[2].

$$\mathbf{A} = \frac{\mu_o}{4 \cdot \pi} \int_C \frac{i \cdot d\mathbf{s}}{r} \tag{1.4}$$

The magnetic induction vector field is the curl of the magnetic vector potential.

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{1.5}$$

These equations are sufficient to determine the magnetic induction generated in a vacuum by any configuration of current carriers. To simplify the application of Equations 1.4 and 1.5, a long uniform solenoid with no end effects is considered. The resulting magnetic induction is perpendicular to the axis of the solenoid with a magnitude that is proportional to the number of windings (N) and inversely proportional to the length (L).

$$B = \frac{\mu_o \cdot i \cdot N}{L} \tag{1.6}$$

When the solenoid is filled with a paramagnetic material, the magnetic induction induces a sympathetic magnetic moment within the material. This magnetic moment results from the alignment of individual dipoles. The magnetic moment per unit volume generated in this manner is called the magnetization (M). The magnetization enhances the magnetic moment of the solenoid.

$$\frac{B}{\mu_o} = \frac{N \cdot i \cdot A}{V} + M \tag{1.7}$$

The magnetic field intensity or magnetic excitation (H) is defined as:

$$H = \frac{B}{\mu_o} - M \tag{1.8}$$

At this point, an assumption is made regarding the interrelationship between the quantities H, B, and M. Equation 1.8 would suggest that two of these quantities are required in order to determine the remaining one. It is a fundamental assumption within this derivation that only one of these quantities is required in order to determine the other two (at a given temperature, pressure, and composition)[3]. This assumption implies that B is a single valued function of H. Therefore, this derivation of magnetic work must preclude consideration of the hysteresis phenomenon. This is equivalent to assuming that the magnetic material must remain at all times in a stable equilibrium state. Fortunately, magnetic materials typically used as magnetic refrigerants are 'soft' magnets (i.e. they exhibit negligible hysteresis).

The magnetic excitation can now be written as:

$$H = \frac{B}{\mu} \tag{1.9}$$

Provided the material is isotropic, the permeability of the magnetic material (μ) is a scalar value which depends on composition, pressure, temperature, and field strength. The temperature dependence of the permeability causes the paramagnetic material to behave as a coupled thermodynamic system, allowing it to be used as a refrigerant.

The differential work done on the solenoid is the applied voltage (E_s) multiplied by the differential current[4].

$$dW = i \cdot E_s \cdot dt \tag{1.10}$$

The applied voltage is obtained by applying Faraday's Law to the solenoid.

$$E_s = N \cdot \frac{d\Phi}{dt} \tag{1.11}$$

By substituting Equation 1.11 and the definition of magnetic flux (Φ) into Equation 1.10, the differential work term can be written in terms of magnetic quantities.

$$dW = V \cdot H \cdot dB \tag{1.12}$$

The differential change in magnetic inductance can be evaluated from Equation 1.8 and substituted into Equation 1.12.

$$dW = \mu_o \cdot H \cdot V \cdot dH + \mu_o \cdot H \cdot V \cdot dM \tag{1.13}$$

The first term in Equation 1.13 represents the work required to change the magnetic field intensity within a vacuum. The second term indicates the work which is done on the magnetic material.

$$dW_{mcg} = \mu_o \cdot H \cdot V \cdot dM \tag{1.14}$$

Since magnetic materials are nearly incompressible, the mechanical work transfer may be neglected. The magnetic counterpart to Equation 1.2 is therefore:

$$dU = T \cdot dS + \mu_o \cdot H \cdot V \cdot dM \tag{1.15}$$

Equations 1.15 and 1.2 are very similar. Each of these equations reflect the fact that the energy of the system may be changed by either a heat or a work transfer. Just as pressure and volume are an intensive and extensive pair of properties describing a mechanical work transfer, applied field ($\mu_o H$) and magnetic moment (MV) form a canonical conjugate pair of properties which describes a magnetic work transfer. The applied field is an intensive property, analogous to pressure in a mechanical system. The magnetic moment is extensive and analogous to negative volume. The negative sign appears because it requires work to increase the magnetic moment of a paramagnetic substance or to decrease the volume of a compressible substance. Temperature and entropy describe heat transfer interactions in both cases.

Using the analogies developed above, all of the thermodynamic results derived for substances which undergo only mechanical work interactions can be applied to paramagnetic materials which undergo only magnetic work interactions. The equation of state for a mechanical system is:

$$v = f(T, P) \tag{1.16}$$

A magnetic system has an analogous equation of state.

$$vM = f(T, \mu_o H) \tag{1.17}$$

The equation of state for Gadolinium Gallium Garnet (GGG) has been obtained experimentally and is illustrated in Figure 1-1. In contrast to a ferromagnetic material, the dipoles of a paramagnetic substance are sufficiently isolated that thermal agitation reduces the overall alignment. For this reason, the applied field must increase at higher temperature in order to maintain a constant magnetic moment. The paramagnetic substance is therefore a coupled thermodynamic system with a cross influence between its thermal and magnetic states.

The equation of state coupled with the constant field specific heat at zero field (or any value of constant applied field) is sufficient to yield the entropy of the magnetic substance as a function of temperature and applied field. This is the fundamental relation for a paramagnetic system, containing all of the thermodynamic information about the stable equilibrium states.[5] The fundamental relation for GGG is illustrated in Figure 1-2. Notice

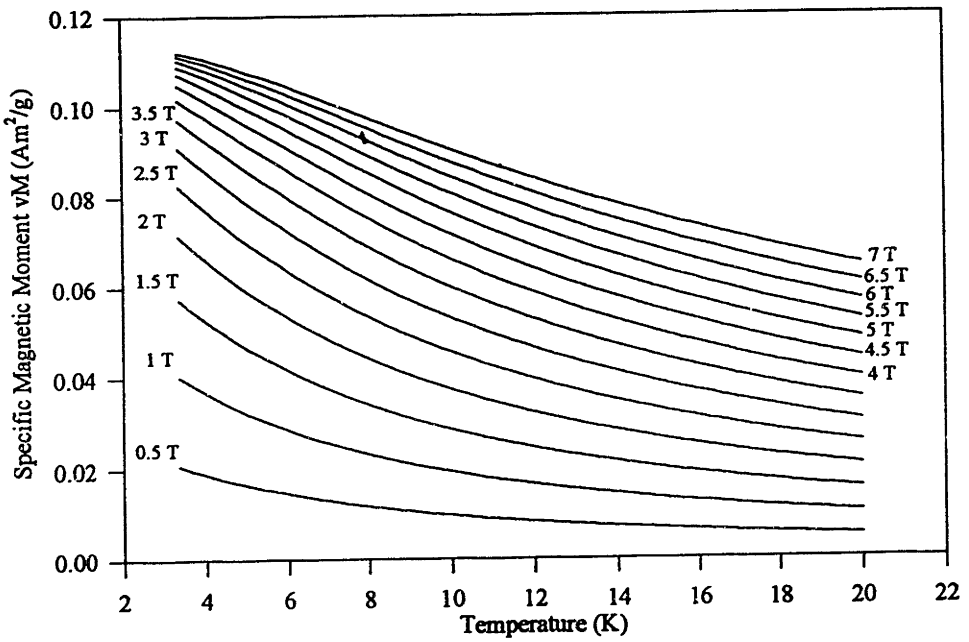

Figure 1-1. Equation of State for Gadolinium Gallium Garnet (GGG)

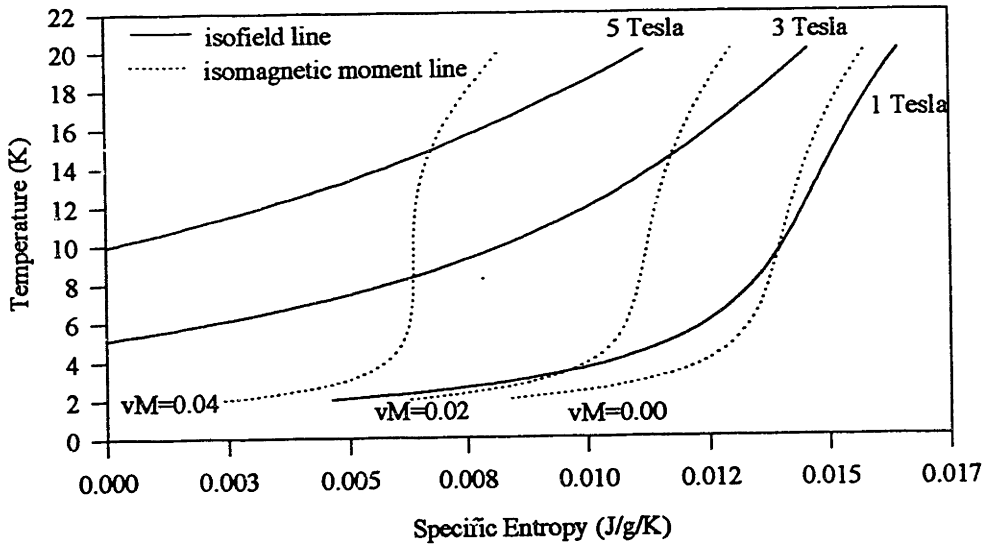

Figure 1-2. Fundamental Relation for Gadolinium Gallium Garnet (GGG)

that the function has the same qualitative shape as a typical mechanical substance. Isofield lines correspond to lines of constant pressure and isomagnetic moment lines are analogous to isochors.

1.2 Magnetic Refrigeration

The thermodynamic coupling which exists between the mechanical and thermal properties of a compressible fluid make it possible to force these fluids to pump heat with a mechanical work input.[6] In a paramagnetic material, the magnetic properties are coupled to the thermal properties as a consequence of the magneto-caloric effect. It follows that a paramagnetic material can be made to pump heat with a magnetic work input. This is the principle behind magnetic refrigeration.

In 1933, Giauque and Debye independently proposed the concept of using the magneto-caloric effect to reach temperatures below 1° K.[7] This technique broke the 'temperature barrier' which had previously been imposed by the inherent limitations of a vapor compression cycle. These systems are limited to temperatures above about 1° K by the liquefaction temperature of the working fluid[8].

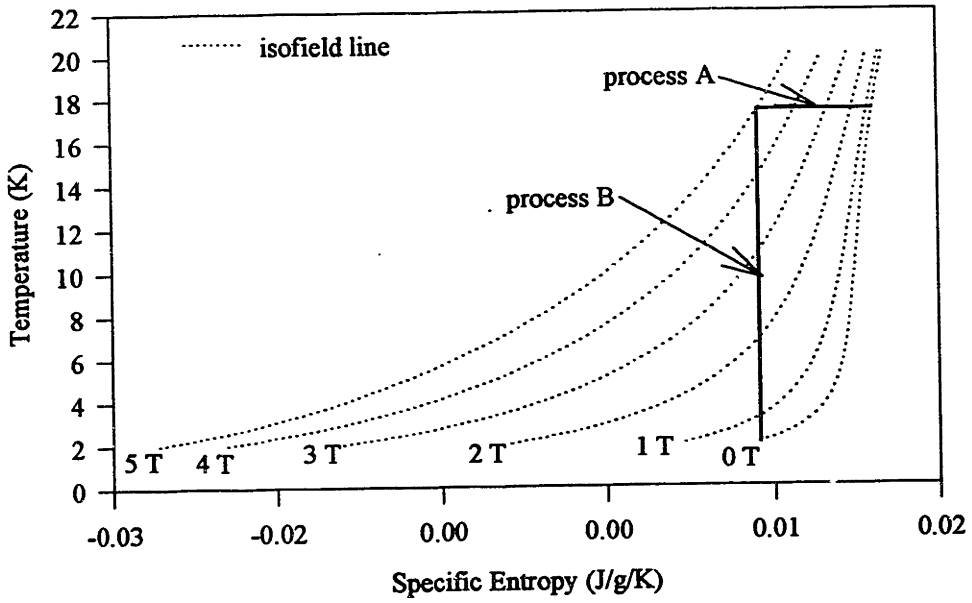

Figure 1-3. 'One-Shot' Adiabatic Demagnetization Cooling Cycle

Early magnetic cooling was a temporary or 'one-shot' technique used to reach very low temperatures. The refrigeration process is analogous to one half of a Carnot cycle as shown in Figure 1-3. The magnetic material is magnetized while in thermal contact with a temperature reservoir (process A). The material is then thermally isolated and demagnetized (process B). This adiabatic demagnetization is the magnetic counterpart to the adiabatic expansion of a gas.

Until recently, there has not been a need to develop refrigerators capable of continuously cooling at temperatures below 10° K. However, there are several engineering applications currently being developed which require a reliable, compact, and efficient refrigerator capable of supplying watts of cooling at 4° to 6° K. The most important of these include cooling superconducting devices and infrared detectors.[9] Although it is theoretically possible to meet this demand using helium gas as the working refrigerant, there are several advantages associated with a magnetic cooling system.[10] A magnetic refrigerator can be more compact and efficient than a conventional refrigerator in this temperature range. Magnetic refrigeration is particularly attractive as the lowest temperature stage of a Stirling type device.[11] Regenerative gas refrigerators typically cease operation near 15 °K because the specific heat capacity of regenerator materials drops off rapidly at these temperatures. This effect is illustrated in Figure 1-4.[12] These cryocoolers must have an added low temperature expander stage (e.g. a Joule-Thomson device) which inherently suffers from poor reliability.

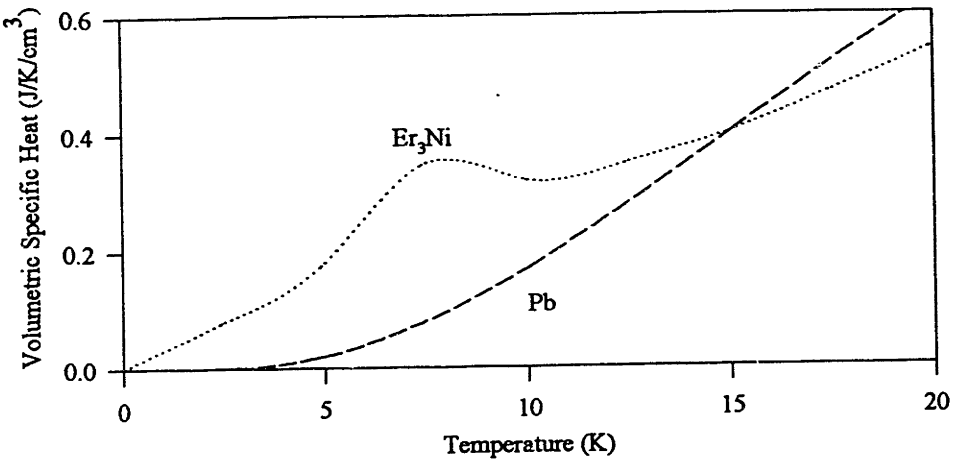

Figure 1-4. Heat Capacity of Typical Regenerator Materials

All continuously operating magnetic refrigerators require that a paramagnetic material be exposed to a time varying applied field. The manner in which this field variation is imposed defines the configuration of the refrigerator. There are three major types of configurations being developed.

In the first configuration, the magnetic material is mechanically rotated through a stationary magnetic field. This type of setup is called a 'rotating wheel' refrigerator and is illustrated schematically in Figure 1-5. Los Alamos National Labs has developed a

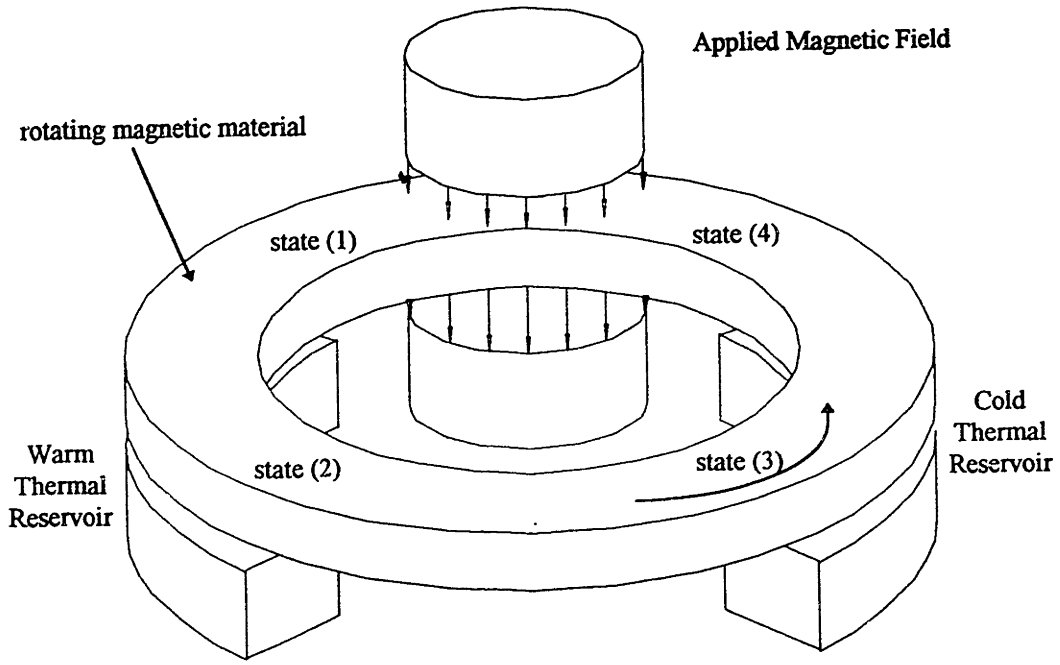

Figure 1-5. Schematic of a 'Rotating Wheel' Magnetic Cooler

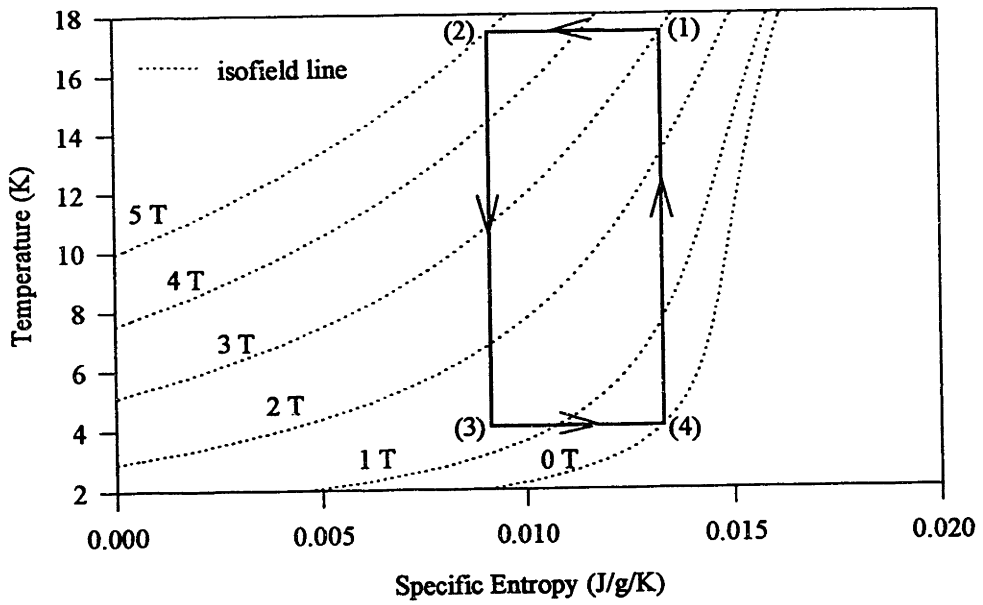

Figure 1-6. Ideal Cycle for 'Rotating Wheel' Magnetic Cooler

'rotating wheel' type device.[13] Work on this type of device has also been done at Hughes Aircraft Company and in Grenoble, France.

The applied field can be generated by a solenoid operating in steady state. The placement of the applied field relative to the heat exchangers allows simple control of the thermodynamic cycle. In theory, it is possible to induce this type of arrangement to undergo a Carnot cycle as shown in Figure 1-6. The adiabatic magnetization and demagnetization processes are performed while the magnetic material is thermally isolated. Isothermal processes occur while the magnetic material is in contact with the thermal reservoirs.

The heat exchange is typically accomplished by pumping helium through flow channels within the rotating magnetic material. This results in refrigeration losses associated with helium leakage between the warm and cold thermal reservoirs. There are also significant losses associated with conduction heat leaks within the magnetic material and frictional heating due to the mechanical motion.

The second configuration is the 'reciprocating' device. The magnetic material is moved translationally rather than rotationally through a stationary magnetic field. Work is being done on 'reciprocating' magnetic coolers at Los Alamos National Laboratories[14], Jet Propulsion Laboratories[15], the David Taylor Naval Research Center[16], and Grenoble[17].

The principle behind this type of device is identical to the 'rotating wheel' device described earlier. All of the magnetic material undergoes the same thermodynamic cycle and the heat transfer is usually accomplished by forcing a fluid through flow passages within the magnetic material.

The final configuration is the 'ramped field' device. In this case, the magnetic material is held stationary and the applied field is varied with an alternating current solenoid. 'Ramped field' devices fall into two groups, those which use thermal switches to control heat transfer[18-20] and those which use a fluid regenerator.

Devices using thermal switches operate on the same principle as both the 'rotating wheel' and the 'reciprocating' magnetic coolers. The capacity of these devices are inherently limited by the capacity of the thermal switches.

In all of the magnetic coolers described so far, the magnetic material has undergone one thermodynamic cycle (e.g. a Carnot cycle). The overall temperature range which can be spanned by any of these coolers is therefore limited by the adiabatic temperature swing of the magnetic working material.

In 'ramped field' devices which use fluid regeneration, the heat transfer is accomplished by forcing a fluid to flow through a regenerator matrix composed of the magnetic material. This configuration is referred to as a magnetically active regenerative refrigerator and is illustrated schematically in Figure 1-7. The principle of magnetically active regenerative refrigeration is the subject of Section 1.3.

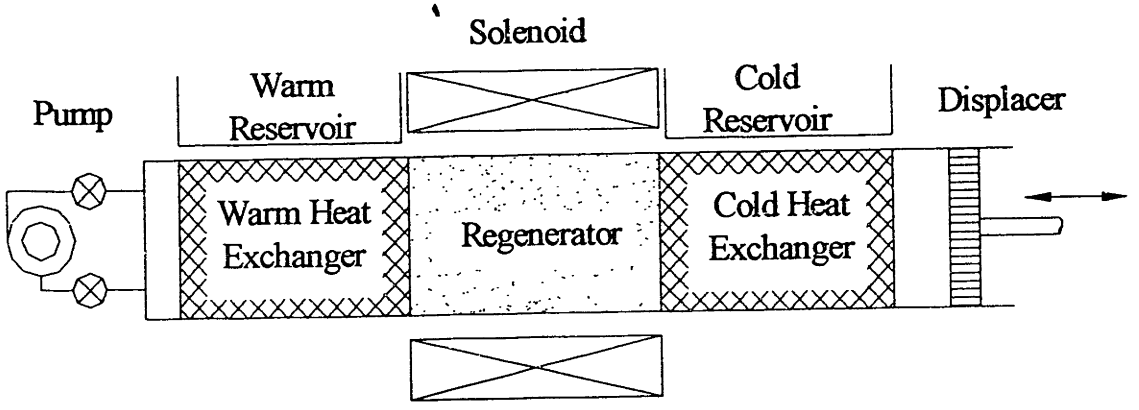

Figure 1-7. Schematic of a Magnetically Active Regenerative Cooler

1.3 Magnetically Active Regenerative Refrigeration

There are two fundamental methods of producing refrigeration with the magnetically active regenerative device illustrated in Figure 1-7 corresponding to the two thermodynamic fluids which are present. The magnetic material is a thermodynamic substance which couples heat transfer with magnetic work. The compressible fluid flowing through the regenerator is a thermodynamic fluid which couples heat transfer with mechanical work. In all of the magnetically active regenerative devices currently being developed, some variation of the following cycle is implemented.

Process 1 Adiabatic Demagnetization Process: the regenerator temperature is reduced by nonflow demagnetization. The temperature of the magnetic regenerator matrix decreases at each axial location according to the adiabatic demagnetization temperature swing of the magnetic material.

Process 2 Inflow Process: a heat transfer fluid (e.g. helium) is forced through the regenerator from the warm end to the cold end. Since the temperature of the regenerator was reduced during process 1, the heat transfer fluid reaches the cold heat exchanger at a temperature below the refrigeration temperature. This 'subcooling' effect produces the refrigeration.

Process 3 Adiabatic Magnetization Process: the regenerator temperature is increased by nonflow magnetization. The temperature of the magnetic regenerator matrix increases at each axial location according to the adiabatic magnetization temperature swing of the magnetic material.

Process 4 Outflow Process: the heat transfer fluid is forced to flow through the regenerator from the cold end to the warm end. Since the temperature of the regenerator was increased during process 3, the heat transfer fluid reaches the warm heat exchanger at a temperature above the warm reservoir temperature. This 'superheating' effect causes heat to be rejected to the warm reservoir.

The magnetic material will undergo a different thermodynamic process at each axial location. Consequently, this type of cryocooler can span a much larger temperature range than those described earlier.[21]

All of the refrigeration produced by the cycle described above is due to the thermodynamic coupling between heat transfer and magnetic work which exists within the magnetic material. Therefore, this type of cycle will subsequently be referred to as a 'magnetic cycle' in the context of magnetically active regenerative devices. Figure 1-8 illustrates the ideal form of a 'magnetic cycle' undergone by the magnetic material at some arbitrary axial position. The entire regenerator undergoes a series of these cycles cascaded from the warm end to the cold end as shown in Figure 1-9. The thermodynamic cycle associated with the compressible fluid is illustrated in Figure 1-10. Since the pressure is not varied, the entire cycle takes place on a single line, indicating that no net heat has been absorbed and no refrigeration produced by the compressible fluid.

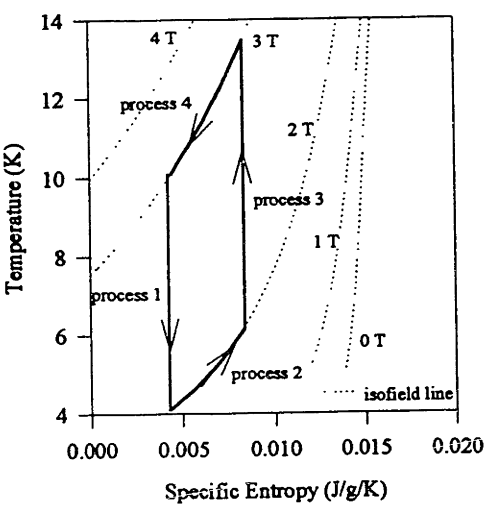

Figure 1-8. Ideal 'Magnetic' Cycle Undergone by Magnetic Material at an Arbitrary Axial Position

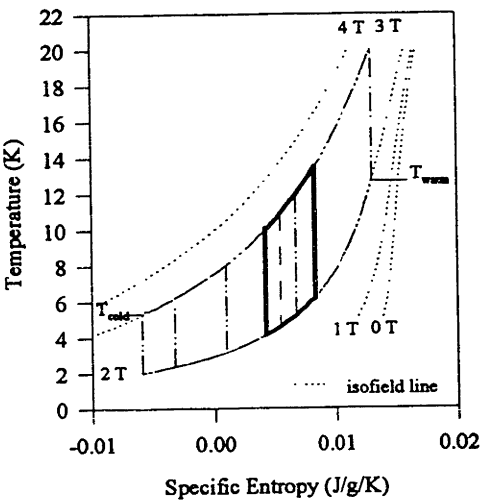

Figure 1-9. Cascaded 'Magnetic' Cycles

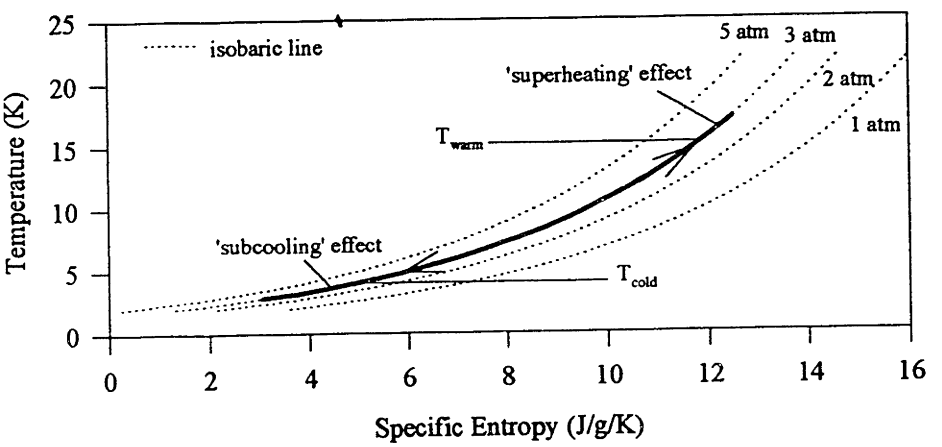

Figure 1-10. Ideal 'Magnetic' Cycle Undergone by Compressible Fluid

There have been several magnetically active regenerative refrigerators constructed. The Cryogenic Engineering Laboratory at the Massachusetts Institute of Technology has built two such devices[22-24]. Regenerative refrigerators are also being developed in Grenoble, France[25] and Tokyo[26]. All of these devices undergo some variation of a purely 'magnetic' cycle (i.e. all of the refrigeration is produced by 'subcooled' compressible fluid entering the refrigeration space). In some cases, the applied field is changed during both the flow and nonflow portions of the cycle, but in none of these devices is the imposed pressure of the working fluid varied.

The second method of producing refrigeration with a magnetically active regenerative device utilizes the thermodynamic coupling present within the compressible fluid. The magnetic material can be induced to behave as an ideal regenerator by changing the applied field only during the flow portions of the cycle[27]. If the applied field is varied correctly, a temporally constant temperature profile can be maintained within the magnetic material during the cycle. The compressible fluid is expanded in the refrigeration space to accept heat and compressed in the warm end to reject heat.

All of the refrigeration is produced by the thermodynamic coupling between heat transfer and mechanical work which exists within the compressible fluid. Therefore, this type of cycle will subsequently be referred to as a 'mechanical' cycle in the context of magnetically active regenerative devices. A 'mechanical' refrigeration cycle is described below.

Process 1 Compression Process: all of the compressible fluid initially resides within the warm space. The fluid is isothermally compressed, inducing a heat rejection to the warm reservoir.

Process 2 Inflow Process: the compressible fluid is forced to flow through the magnetic regenerator matrix while the applied field is decreased. The regenerator isothermally accepts heat as the flowing fluid is cooled from the warm reservoir temperature to the refrigeration temperature. The compressible fluid enters the refrigeration space at precisely the refrigeration temperature so that no 'subcooling' occurs. The magneto-caloric effect allows the regenerator to behave as if it had an infinite heat capacity.

Process 3 Expansion Process: the compressible fluid resides within the refrigeration space. The fluid is isothermally expanded, producing refrigeration.

Process 4 Outflow Process: the compressible fluid is forced to flow through the regenerator matrix while the applied field is increased to its original value. The regenerator isothermally rejects heat as the flowing fluid is heated from the refrigeration temperature to the warm reservoir temperature. The compressible fluid enters the warm space at precisely the warm reservoir temperature so that no 'superheating' occurs.

Figure 1-11 illustrates the ideal form of a 'mechanical' cycle undergone by the magnetic material at several arbitrary axial positions. The fact that each cycle contains no area indicates that no net heat has been absorbed and no refrigeration produced by the magnetic regenerator. Figure 1-12 illustrates the thermodynamic cycle associated with the compressible fluid.

The cycles illustrated in Figures 1-8 through 1-12 presume that four important conditions are satisfied.

1. No pressure gradients exist within the flowing compressible fluid
2. Perfect thermal contact exists between the flowing fluid and the magnetic material
3. No entrained compressible fluid is present within the regenerator matrix
4. The compressible fluid is behaving as a perfect gas

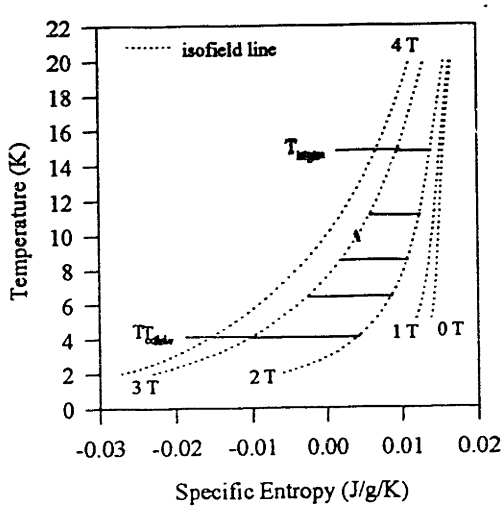

Figure 1-11. Ideal 'Mechanical' Cycles Undergone by Magnetic Material

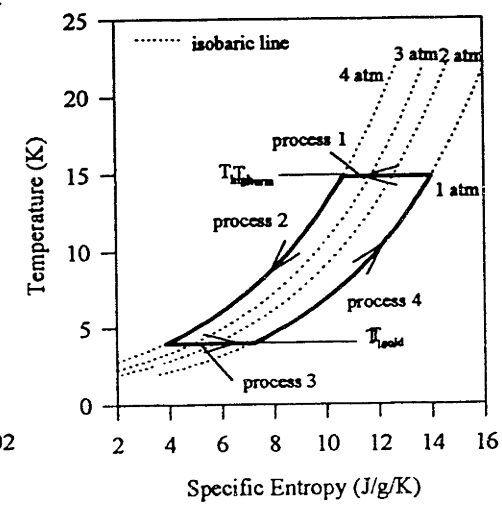

Figure 1-12. Ideal 'Mechanical' Cycle Undergone by Compressible Fluid

Several secondary assumptions are also implied.

- No axial conduction occurs within the regenerator
- Perfect thermal contact exists in the heat exchangers
- No void volume is present in the heat exchangers
- No eddy current heating is present
- No radial heat leaks are present
- The magnetic material is undergoing internally reversible processes

The 'mechanical' and the 'magnetic' cycles represent two fundamental methods of producing refrigeration in a magnetically active regenerative device. An infinite number of hybrid or 'combined' cycles also exist. 'Combined' cycles utilize the thermodynamic coupling which exists in both the compressible fluid and the magnetic material. The combined cycle considered throughout this thesis is implemented as described below.

Process 1 Compression/Adiabatic Demagnetization Process: all of the compressible fluid initially resides within the warm space. The fluid is isothermally compressed, inducing a heat rejection to the warm reservoir. Concurrently, the regenerator temperature is reduced by nonflow demagnetization. The temperature of the magnetic regenerator matrix decreases at each axial position according to the adiabatic demagnetization temperature swing of the magnetic material.

Process 2 Inflow Process: the compressible fluid is forced to flow through the magnetic regenerator matrix while the applied field is reduced to its lowest value. The regenerator accepts heat as the flowing fluid is cooled from the warm reservoir temperature to a temperature below the refrigeration temperature. The compressible fluid reaches the refrigeration space 'subcooled', producing the magnetic portion of the total refrigeration.

Process 3 Expansion/Adiabatic Magnetization Process: the compressible fluid resides within the refrigeration space. The fluid is isothermally expanded, producing the mechanical portion of the total refrigeration. Simultaneously, the regenerator temperature is increased by nonflow magnetization. The temperature of the magnetic regenerator matrix increases at each axial position according to the adiabatic magnetization temperature swing of the magnetic material.

Process 4 Outflow Process: the compressible fluid is forced to flow through the regenerator while the applied field is increased to its original value. The regenerator rejects heat as the flowing fluid is heated to some temperature above the warm reservoir temperature. The compressible fluid reaches the warm space 'superheated', causing heat rejection to the warm reservoir.

Figure 1-13 illustrates the ideal form of a 'combined' cycle undergone by the magnetic material at several arbitrary axial positions. Figure 1-14 illustrates the thermodynamic cycle associated with the compressible fluid. The assumptions made earlier guarantee that the magnetic material will undergo isentropic nonflow processes, but the shape of the flow processes are undetermined. Isothermal flow processes cannot, in general, be accomplished even with the simplifications made above[28]. The flow process path will be determined by the mass ratio (i.e. the ratio of compressible fluid mass to magnetic material mass) and will lie somewhere between an isofield line and an isothermal line.

The magnetically active regenerative coolers being developed use exclusively 'magnetic' cycles due to analytical and experimental convenience rather than any inherent advantage associated with this type of cycle. The interactions which occur between two thermodynamic working fluids become highly complex, particularly when the properties of both fluids are nonideal. No model has been developed to analyze a process in which both the compressible fluid and the magnetic material are allowed to undergo complete

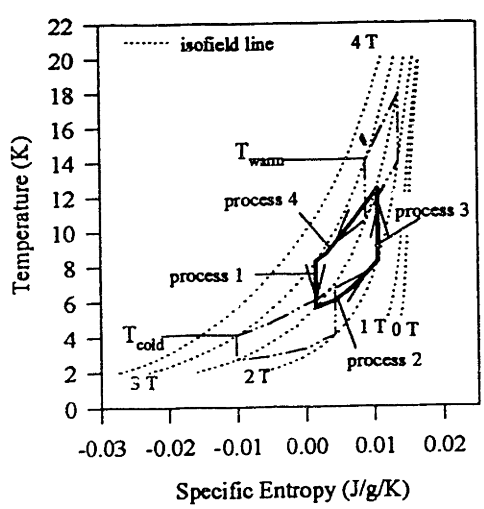

Figure 1-13. Ideal 'Combined' Cycles Undergone by Magnetic Material

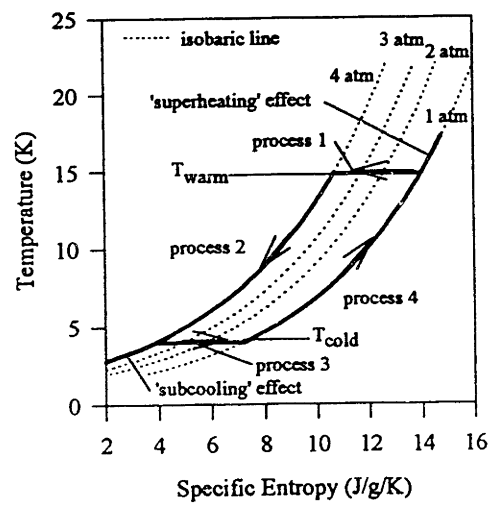

Figure 1-14. Ideal 'Combined' Cycle Undergone by Compressible Fluid

thermodynamic cycles. Moreover, it is difficult to construct an experimental device in which both the pressure and applied field can be cyclically varied.

In Chapter 2, the ideal cycles described here will be used to demonstrate that a 'combined' cycle is always more effective than a 'magnetic' cycle. Furthermore, magnetically active regenerative devices are typically used to extend the cooling range of a conventional Stirling cryocooler. The working fluid within a Stirling cycle undergoes a significant pressure variation. In order to maintain the isobaric compressible fluid required for a 'magnetic' cycle, the upper and lower temperature stages must be mechanically isolated. A magnetically active regenerative device which undergoes pressure variations identical to the upper stage Stirling device is a more effective method of extending the cooling range of a conventional cryocooler because no mechanical isolation is necessary.

II. ANALYSIS OF IDEAL CYCLES

In this chapter, the ideal cycles described in Section 1.3 are analyzed in order to demonstrate the advantages associated with a 'combined' cycle. The system considered throughout the remainder of this thesis is composed of a Gadolinium Gallium Garnet (GGG) matrix and a Helium-4 compressible fluid.

The major assumptions which are necessary for ideal cycles are restated here.

1. No pressure gradients exist within the flowing compressible fluid
2. Perfect thermal contact exists between the flowing fluid and the magnetic material
3. No entrained compressible fluid is present within the regenerator matrix
4. The compressible fluid is behaving as a perfect gas

This chapter is divided into six sections. The first section describes the procedure used to determine the thermodynamic properties of GGG. In the second section, the model of an ideal 'mechanical' cycle is developed. In the third section, the model of an ideal 'magnetic' cycle is developed. In the fourth section, the model of an ideal 'combined' cycle is developed. The results of these models are presented in the fifth section. In the final section, the consequences of not conforming to the four ideal conditions are separately examined.

2.1 Determination of Gadolinium Gallium Garnet Properties

The thermodynamic properties of GGG have been obtained by Gregory Gallagher (1986)[29]. An empirical magnetization function was developed based on experimental data from two sources (Figure 1-1). This function, coupled with the zero field specific heat, was used to determine the entropy of GGG as a function of temperature and applied field (Figure 1-2).

The entropy of a magnetic substance can be obtained by integrating the appropriate differential entropy equation along a line of constant applied field to the desired temperature and then integrating at constant temperature to the appropriate applied field.

$$s_{GGG}(T,\mu_o H) = s_{GGG_o} + \int_{T_o}^{T}\left(\frac{\partial s_{GGG}}{\partial T}\right)_{\mu_o H_o} dT + \int_{\mu_o H_o}^{\mu_o H}\left(\frac{\partial s_{GGG}}{\partial \mu_o H}\right)_{T} d\mu_o H \qquad (2.1)$$

The reference entropy was assigned a value of zero and arbitrary reference conditions were selected (0 Tesla and 1° K).

$$s_{GGG}(T,\mu_o H) = \int_{1^\circ K}^{T}\left(\frac{\partial s_{GGG}}{\partial T}\right)_{0T} dT + \int_{0T}^{\mu_o H}\left(\frac{\partial s_{GGG}}{\partial \mu_o H}\right)_{T} d\mu_o H \qquad (2.2)$$

The partial derivative of entropy with respect to temperature at zero applied field is determined from the zero field specific heat of GGG[30].

$$\left(\frac{\partial s_{GGG}}{\partial T}\right)_{0T} = \frac{c_{\mu_o H=0T}(T)}{T} = 6.900\cdot 10^{-7}\cdot T^2 + \frac{6.385\cdot 10^{-2}}{T^3} - \frac{5.096\cdot 10^{-2}}{T^4} \qquad (2.3)$$

In this way, the first integral in Equation 2.2 is reduced to a closed form analytical expression.

The partial derivative of entropy with respect to applied field at constant temperature is equivalent to the partial derivative of magnetic moment with respect to temperature at constant applied field. This relationship is the magnetic equivalent of a Maxwell Relation.

$$\left(\frac{\partial s_{GGG}}{\partial \mu_o H}\right)_{T} = \left(\frac{\partial \nu M}{\partial T}\right)_{\mu_o H} \qquad (2.4)$$

Gallagher's empirical correlation for the magnetic equation of state is of the following form.

$$\nu M = \frac{\mu_o H}{\sum_{i=0}^{4} c_i(T)\cdot \mu_o H^i} \qquad (2.5)$$

Each $c_i(T)$ in Equation 2.5 represents a third order polynomial in T. The second integral in Equation 2.2 cannot be evaluated in closed form. Instead, a Gauss-Legendre quadrature method is used to approximate this integration numerically[31].

The subroutine GGG.c is used in both the ideal and nonideal simulations to evaluate the properties of Gadolinium Gallium Garnet. The code is listed in Appendix A.1. The subroutine GGG.c was translated from the FORTRAN subroutine GGGPROP[32].

2.2 Simulation of an Ideal 'Mechanical' Cycle

The ideal 'mechanical' cycle is simulated by determining the temperature profile within the GGG regenerator that satisfies an energy balance at every axial position. The process is an iterative one in which successive temperature profiles are adjusted until they converge to the final solution.

The regenerator is divided into m differential axial segments. Each segment undergoes an isothermal process as illustrated in Figure 2-1. As helium flows from the warm to the cold end, it is gradually cooled by the demagnetization of the magnetic material. Therefore, the magnetic material contained within a particular node (e.g. node i) is responsible for changing the temperature of the flowing helium mass (M_{he}) by ΔT_i. This can be expressed in the form of an energy balance at each node.

$$M_{he} \cdot c_p \cdot \Delta T_i = \frac{M_{GGG} \cdot \Delta s_{GGGi} \cdot T_i}{m} \tag{2.6}$$

The mass ratio was defined earlier as the ratio of helium mass to GGG mass. The mass ratio (mr) which can be accommodated by each node for a given temperature profile can be determined from Equation 2.6.

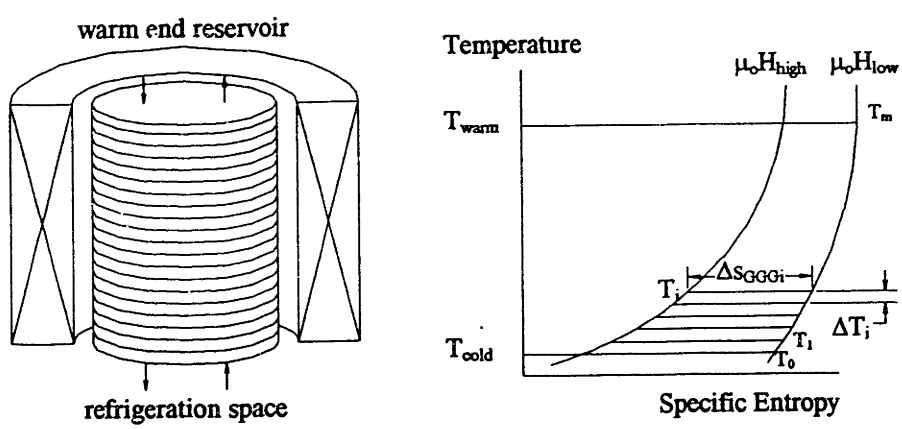

Figure 2-1. Schematic of an Ideal 'Mechanical' Cycle Model

$$mr_i = \frac{\Delta s_{GGGi} \cdot T_i}{m \cdot c_p \cdot \Delta T_i} \tag{2.7}$$

The average mass ratio which can be cooled by the regenerator is calculated.

$$\overline{mr} = \frac{\sum_{i=1}^{m} mr_i}{m} \tag{2.8}$$

The temperature change which can be induced in the flowing based on the average mass ratio is determined for each node.

$$\Delta T_i = \frac{\Delta s_{GGGi} \cdot T_i}{m \cdot \overline{mr} \cdot c_p} \tag{2.9}$$

The temperature profile is adjusted using the weighted sum of these differential temperature differences.

$$T_i = T_{i-1} + \frac{\Delta T_i \cdot (T_{warm} - T_{cold})}{\sum_{i=1}^{m} \Delta T_i} \tag{2.10}$$

The process is repeated until both the temperature profile and the mass ratio converge. The refrigeration produced during the cycle occurs during the isothermal expansion of the helium mass in the refrigeration space. The cyclic specific refrigeration (i.e. the refrigeration per mass of GGG per cycle) is determined by the pressure ratio (γ) and the final mass ratio.

$$\frac{Q_{ref}}{M_{GGG}} = T_{cold} \cdot R_{he} \cdot mr \cdot \ln\gamma \tag{2.11}$$

The temperature profiles predicted by the model for various refrigeration temperatures are illustrated in Figure 2-2. These profiles were generated based on an applied field swing between 3 and 5 Tesla and a warm reservoir temperature of 17° K. The helium was modelled with a constant pressure specific heat of 5.192 kJ/kg/K and an ideal gas constant of 2.078 kJ/kg/K. The code used to simulate an ideal 'mechanical' cycle is entitled MECH.c and is listed in Appendix A.2.

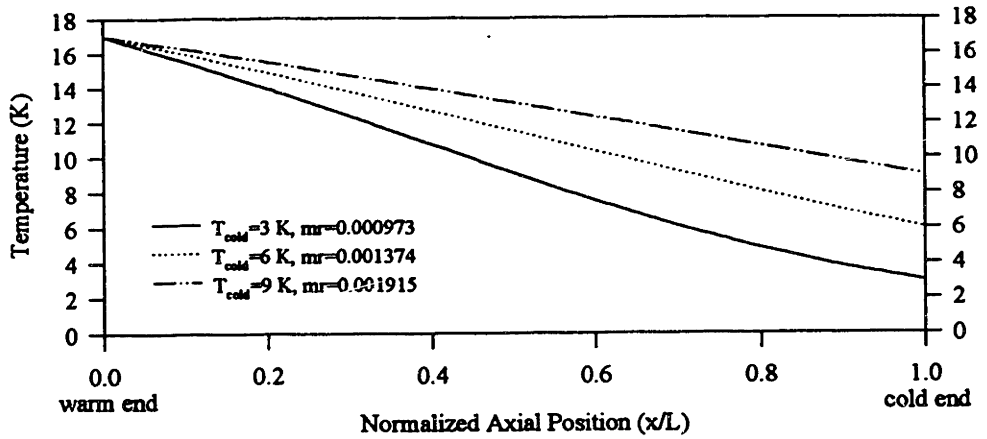

Figure 2-2. Temperature Profiles for an Ideal 'Mechanical' Cycle Simulation

2.3 Simulation of an Ideal 'Magnetic' Cycle

In order to simulate an ideal 'magnetic' cycle the regenerator is again split into m differential segments. However, the temperature profile within the regenerator changes continuously during this type of cycle. Therefore, the flow portions of the cycle must be split into a series of steps. During each step, a differential helium mass is sent down the axis of the regenerator. The temperature profile within the regenerator is adjusted after each step. The schematic of the model used to simulate an ideal 'magnetic' cycle is illustrated in Figure 2-3. The subscripts indicate the axial position of the node. The

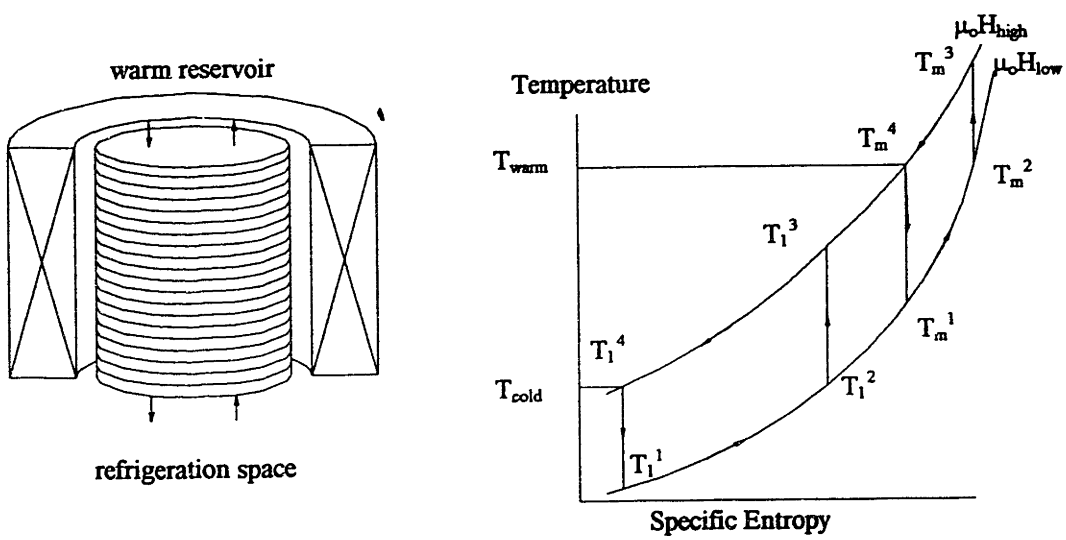

Figure 2-3. Schematic of an Ideal 'Magnetic' Cycle Model

superscripts indicate the position within the cycle (e.g. T_1^2 refers to the temperature of node 1 at state (2)).

The simulation is an iterative process. The mass ratio must be successively changed in order to ensure that the cycle closes at each node. The process begins with a linear temperature profile between T_{cold} and T_{warm}, corresponding to state (4) in Figure 2-3. The adiabatic demagnetization process is simulated by decreasing the temperature of each node according to the adiabatic demagnetization temperature swing of the GGG. The temperature profile within the regenerator is changed to state (1).

The inflow process is simulated by sending differential masses of helium down the axis of the regenerator. The temperature profile is adjusted at each node according to Equation 2.12. This equation represents an energy balance between the cooling differential helium mass (dM_{he}) and the warming regenerator matrix (M_{GGG}).

$$c_p \cdot dM_{he}(T_i - T_{i-1}) = \frac{c_{\mu_o H} \cdot M_{GGG} \cdot dT_i}{m} \tag{2.12}$$

Equation 2.12 can be rearranged to yield the required temperature change (dT_i) at each node as a function of the differential mass ratio (dmr) and the partial derivative of GGG entropy with respect to temperature at constant applied field.

$$dT_i = \frac{c_p \cdot dmr \cdot (T_i - T_{i-1}) \cdot m}{\left(\frac{\partial s_{GGGi}}{\partial T}\right)_{\mu_o H} T_i} \tag{2.13}$$

A differential portion of the total specific refrigeration is produced by each 'subcooled' differential helium mass which enters the refrigeration space.

$$\frac{dQ_{ref}}{M_{GGG}} = c_p \cdot dmr \cdot (T_{cold} - T_1) \tag{2.14}$$

The downflow process is complete when all of the helium has been passed down the axis of the regenerator.

The mass ratio required to close the cycle must be determined iteratively. The remainder of the refrigeration cycle is simulated in the same manner as the adiabatic demagnetization process and the inlet process. If the final temperature profile does not match the original temperature profile, then the mass ratio is adjusted and the process repeated.

Figure 2-4 illustrates the thermodynamic cycles undergone by the GGG at three nodes. These cycles were generated by the ideal 'magnetic' model based on a warm reservoir temperature of 17° K, a refrigeration temperature of 6°K, and an applied field swing between 3 and 5 Tesla. The mass ratio which closed the cycle was 0.001072.

The code used to simulate the ideal 'magnetic' cycle is call MAG.c and is listed in Appendix A.4. This code calls the function TGSUOH.c in order to determine the temperature of GGG associated with a given entropy and applied field. TGSUOH.c is listed in Appendix A.3.

The function TGSUOH.c generates a vector of temperatures and the associated entropy at a given value of applied field. The cubic spline technique of interpolation is then used to interpolate the temperature corresponding to the desired entropy. The cubic spline method of interpolation is discussed in Appendix B.

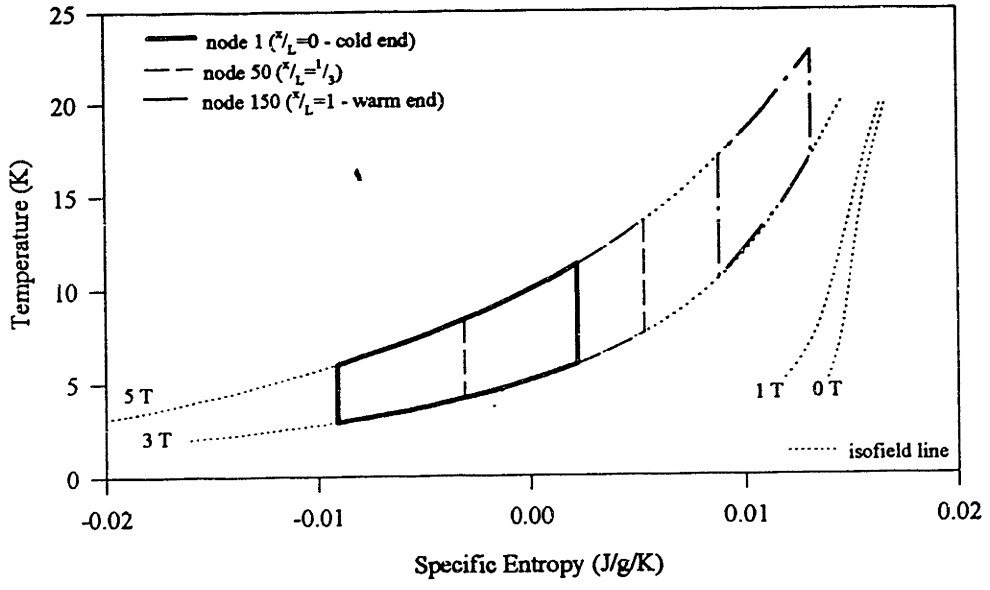

Figure 2-4. GGG Thermodynamic Cycles for an Ideal 'Magnetic' Cycle Simulation

2.4 Simulation of an Ideal 'Combined' Cycle

The 'combined' cycle simulation differs from the 'magnetic' cycle simulation only in the variation of the applied field during the nonflow processes. The applied field variation during an ideal 'mechanical' cycle takes place entirely during the flow processes. The applied field variation takes place during the nonflow processes of an ideal 'magnetic' cycle. In an ideal 'combined' cycle, the total applied field change is divided into two components, a flow variation and a nonflow variation. An additional parameter is therefore required to specify the operating characteristics of an ideal 'combined' cycle. This parameter is called the flow fraction (ff) and is defined as the ratio of applied field change during the flow portion of the cycle to the total applied field variation.

$$ff = \frac{\Delta \mu_o H_{flow}}{(\mu_o H_{high} - \mu_o H_{low})} \tag{2.15}$$

Figure 2-5 illustrates the schematic of the model used to simulate an ideal 'combined' cycle. As the flow fraction goes to zero, the cycle approaches an ideal 'magnetic' cycle. A flow fraction of unity corresponds to an ideal 'mechanical' cycles. If the pressure variation were identical, the specific refrigeration of the ideal 'combined' cycle would limit to these two extreme cases. The motivating question behind this analysis is whether the variation of flow fraction from zero to unity produces a maximum or a minimum in specific refrigeration.

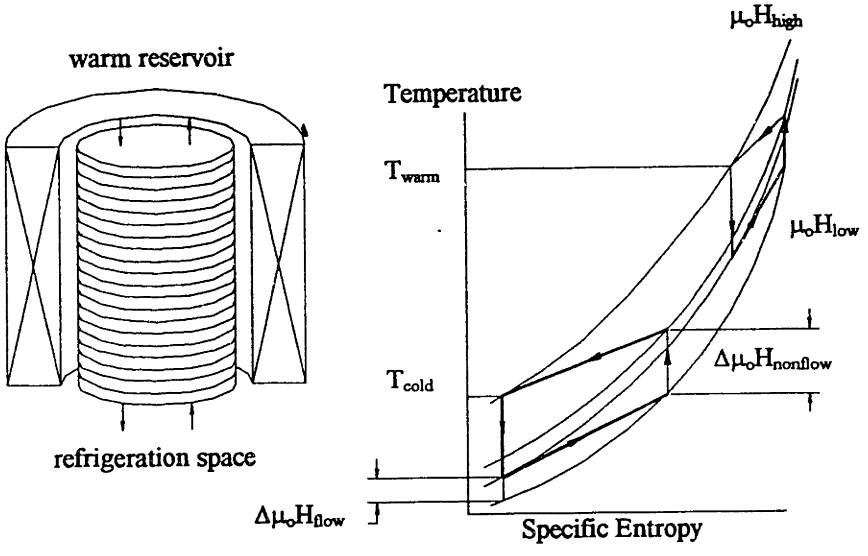

Figure 2-5. Schematic of an Ideal 'Combined' Cycle Model

The flow process applied field variation must be accounted for in the nodal energy balance. In the ideal 'magnetic' cycle, the regenerator absorbed heat at constant applied field. The resulting temperature change within each differential GGG mass was determined in Section 2.3 by Equation 2.13.

$$dT_i = \frac{c_p \cdot dmr \cdot (T_i - T_{i-1}) \cdot m}{\left(\frac{\partial s_{GGGi}}{\partial T}\right)_{\mu_o H} T_i} \tag{2.13}$$

Equation 2.13 may be altered to reflect the applied field variation during the flow process.

$$dT_i = \frac{c_p \cdot dmr \cdot (T_i - T_{i-1}) \cdot m}{\left(\frac{\partial s_{GGGi}}{\partial T}\right)_{\mu_o H} T_i} - \frac{\left(\frac{\partial s_{GGGi}}{\partial \mu_o H}\right)_T \cdot d\mu_o H}{\left(\frac{\partial s_{GGGi}}{\partial T}\right)_{\mu_o H}} \tag{2.16}$$

Equation 2.16 reflects the fact that the differential applied field change ($d\mu_oH$) decreases the amount of temperature change required to cool the flowing helium. The simulation procedure is otherwise identical to the one described in Section 2.3.

The thermodynamic cycle undergone by a typical GGG node during an ideal 'combined' cycle is illustrated in Figure 2-6. This cycle was generated based on a warm reservoir temperature of 17 °K, a refrigeration temperature of 6 °K, a pressure ratio of 3.5, a high applied field of 5 Tesla, a low applied field of 3 Tesla, and a flow fraction of 0.5 (i.e. half of the applied field swing was performed during the flow portion of the cycle). The mass ratio required to close the cycle was 0.0009812.

The code used to simulate an ideal 'combined' cycle is called COM.c and is listed in Appendix A.5.

2.5 Results of Ideal Simulations

In Section 1.3, the different methods of producing refrigeration within a magnetically active regenerator are explained. All of the magnetically active regenerative refrigerators being developed use the 'magnetic' method of producing refrigeration. That is to say that none of the refrigeration is produced by the compressible fluid. The models developed in

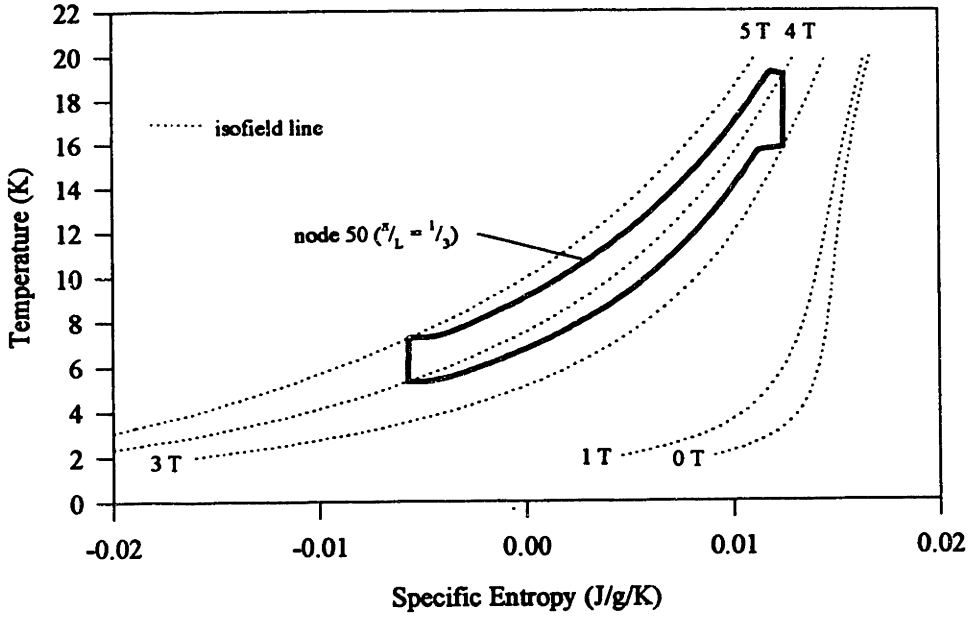

Figure 2-6. GGG Thermodynamic Cycle at an Arbitrary Axial Position for an Ideal 'Combined' Cycle Simulation

Sections 2.2 through 2.4 make it possible to determine whether the 'magnetic' cycle is, in fact, the most effective method of producing refrigeration under ideal conditions.

Figure 2-7 illustrates the specific refrigeration potential of the ideal cycles as a function of refrigeration temperature. The warm reservoir temperature was set at 15 °K for all of the

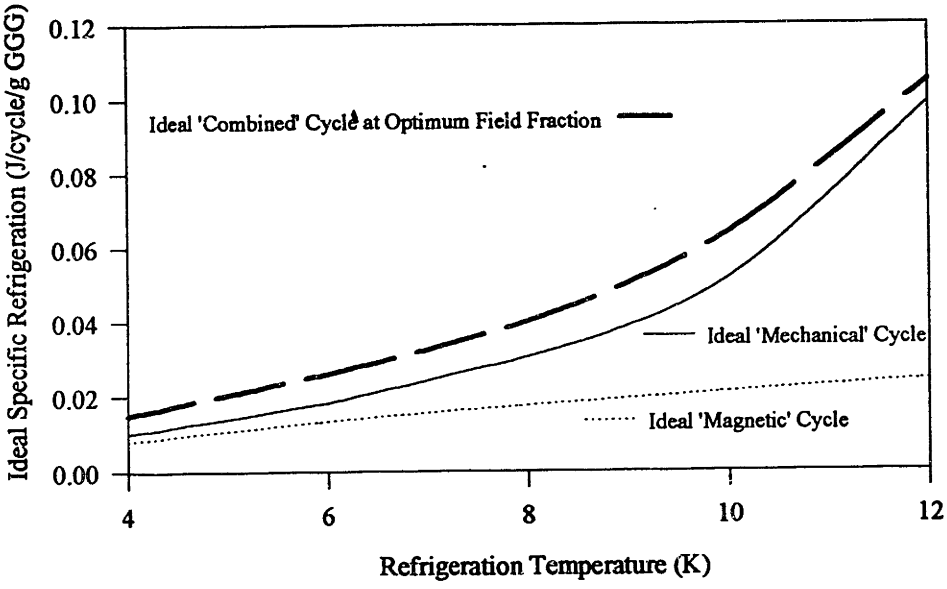

Figure 2-7. Variation of Ideal Cycle Refrigeration with Refrigeration Temperature

cycles. Figure 2-7 was generated based on a high applied field of 3 Tesla, a low applied field of 1 Tesla, and a pressure ratio of 3.5. The flow fraction of the ideal 'combined' cycle was optimized at each value of refrigeration temperature.

The refrigeration potential of an ideal 'combined' refrigeration cycle is the largest. The ideal 'magnetic' and ideal 'mechanical' cycles are extreme values in the coordinate of flow fraction. Therefore, variation of the flow fraction between zero and unity results in a maximum value of refrigeration potential.

The 'combined' cycle begins to approach the 'mechanical' cycle in the upper temperature regions where mechanical expansion of the helium gas is a more effective method of producing refrigeration. In the lower temperature regions, the 'combined' cycle shifts towards a magnetic cycle as the magneto-caloric effect increases. It is particularly interesting to note that in the 4° to 6° K range, the 'combined' cycle yields almost twice the refrigeration of a 'magnetic' cycle.

2.6 Examination of Ideal Assumptions

The ideal cycles simulated in this chapter only occur if the four conditions listed in Section 1.3 are met. To varying degrees, none of these conditions will exist in a real device. This section will qualitatively examine how these nonidealities will affect the shape of the thermodynamic cycles undergone by the working fluids and therefore the performance of the refrigerator. These effects are examined in a more quantitative fashion in Chapter 4 using the nonideal model developed in Chapter 3.

This section is divided into four subsections. Each subsection addresses the effect of one nonideal condition.

2.6.1 Nonzero Pressure Gradients

In order to qualitatively understand the effect of finite pressure gradients within the regenerator, the thermodynamic cycle of the helium working fluid must be examined.

Figure 2-8 illustrates an ideal 'combined' cycle undergone by the flowing helium (shown in solid lines) and the deviation which results from pressure gradients in the flow direction (shown in dashed lines).

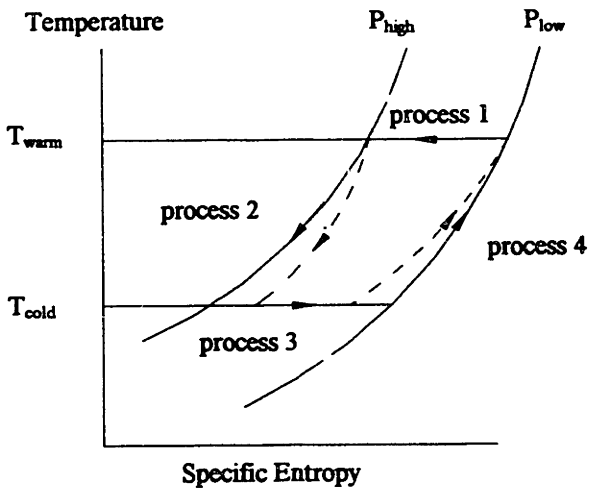

Figure 2-8. Pressure Gradient Induced Deviation from Ideal Cycle

The mechanical component of the total refrigeration is produced because the helium is forced to act as a heat pump. The helium transfers heat from a low temperature to a high temperature. The cyclic heat rejection is equivalent to the area of Figure 2-8. This is a qualitative measure of how effectively the helium provides refrigeration. As the pressure gradients increase, the area of the temperature-entropy diagram is diminished.

2.6.2 Imperfect Heat Transfer

During the inlet process (i.e. process 2), the magnetic material accepts heat from the flowing helium. If nonzero temperature gradients in the direction of heat flow are present within the regenerator, then the helium must assume a higher temperature in order to achieve the same heat transfer. The result is that the helium enters the cold space at a slightly higher temperature than during the ideal cycle. This constitutes a degradation of the magnetic component of refrigeration.

Alternatively, the degradation associated with imperfect heat transfer can be understood by realizing that a finite amount of irreversibility is produced when heat is transferred through a nonzero temperature gradient. This irreversibility must be convected out of the device within the flowing helium. Since all of the entropy associated with the refrigeration load must also be removed in this manner, the total refrigeration capacity of the device is reduced.

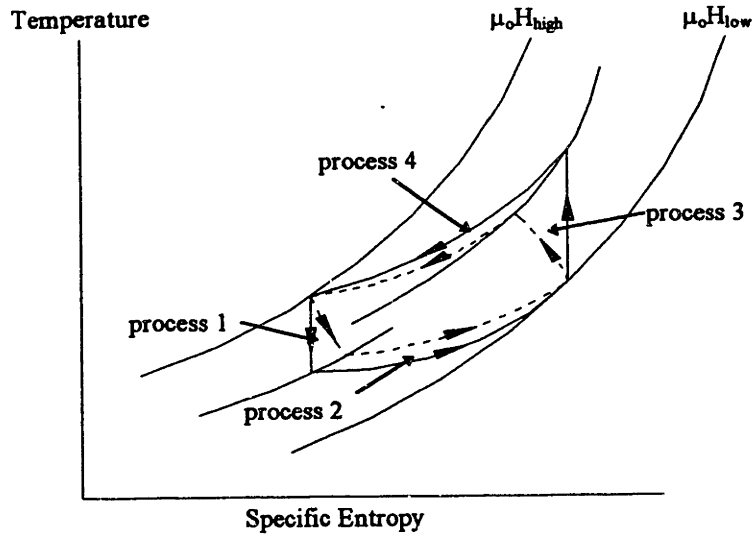

Figure 2-9. Entrained Helium Induced Deviation from Ideal Cycle

2.6.3 Nonzero Regenerator Porosity

Any entrained helium within the regenerator matrix will prevent the magnetic material from undergoing adiabatic nonflow processes (i.e. process 1 and process 3 will deviate from isentropic). In general, the pressure and applied field are simultaneously varied during a nonflow process. The assumption of zero porosity made it possible to examine these effects separately during the simulation of the ideal cycles. However, when helium is in contact with the GGG, the two effects are coupled.

Figure 2-9 indicates the thermodynamic cycle undergone by an arbitrary mass of GGG. The ideal cycle is indicated in solid lines. The qualitative form of the nonideal cycle which will result from entrained helium is indicated in dashed lines.

In order to understand the effects associated with the presence of entrained helium, the compression/adiabatic demagnetization process (i.e. process 1) is examined. During this process, the pressure of the helium is raised and the applied field is lowered.

The compression process is an exothermic process inducing the helium initially entrained within the regenerator to transfer heat to the GGG. A secondary effect occurs as the pressure driven increase in density draws warm helium into the regenerator from the warm heat exchanger. This flow of relatively hot helium also transfers heat to the GGG.

Also, the adiabatic demagnetization process is endothermic, inducing the regenerator material to decrease in temperature and accept heat from the helium. The helium may experience a sympathetic temperature drop. A secondary effect occurs as the temperature driven increase in density draws warm helium into the regenerator from the warm heat exchanger.

All of the effects described above cause the entrained helium to transfer heat (and entropy) to the GGG during the compression/adiabatic demagnetization process. This corresponds to pushing the endpoint of process 1 to the right (i.e. towards greater entropy) on the temperature-entropy diagram, as shown in Figure 2-9. Since the process must end at the imposed value of applied field, the endpoint of process 1 is also pushed upwards (i.e. towards higher temperature). Similar effects drive the endpoint of the expansion/adiabatic magnetization process (i.e. process 3) to the left and down. The area of the thermodynamic cycle is consequently reduced, indicating that the GGG is generating less refrigeration per cycle due to the presence of entrained helium within the regenerator.

2.6.4 Nonideal Helium Behavior

The properties of helium at low temperature depart drastically from the perfect gas model. This results in a specific heat mismatch during the flow processes which diminishes the amount of refrigeration produced by the GGG.

We can analyze this effect by assuming that all of the ideal conditions are present except for helium perfect gas behavior. A cyclic entropy balance around a differential segment of regenerator, as shown in Figure 2-10, leads to the following expression.

$$M_{he} \cdot s_2(x) + M_{he} \cdot s_4(x) = M_{he} \cdot s_2(x) + \frac{d}{dx}\left[M_{he} \cdot s_2(x)\right] + M_{he} \cdot s_4(x) - \frac{d}{dx}\left[M_{he} \cdot s_4(x)\right] \quad \textbf{(2.17)}$$

The subscript 2 refers to the high pressure inflow (i.e. process 2) and the subscript 4 refers to the low pressure outflow (i.e. process 4). Equation 2.17 may be simplified by utilizing the assumption of zero regenerator porosity to remove the helium mass from the derivative. Also, the axial derivative of entropy may be expanded using the chain rule. The result is shown in Equation 2.18.

$$\left(\frac{\partial s_2}{\partial T}\right)_P (x)\left(\frac{dT_2(x)}{dx}\right) = \left(\frac{\partial s_4}{\partial T}\right)_P (x)\left(\frac{dT_4(x)}{dx}\right) \quad \textbf{(2.18)}$$

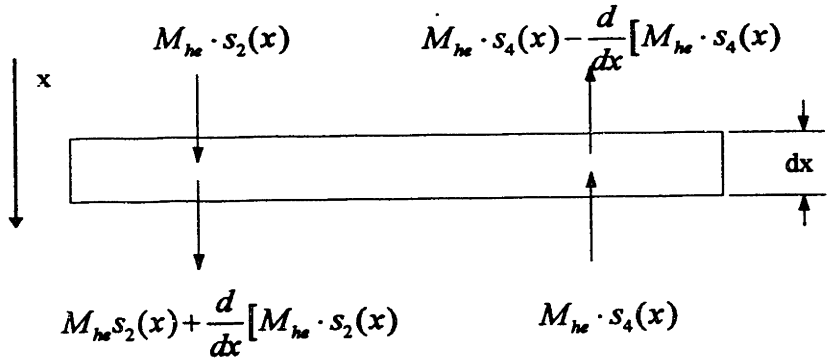

Figure 2-10. Differential Regenerator Segment for Effect of Nonideal Helium Property Analysis

The partial derivative of entropy with respect to temperature at constant pressure is equivalent to the constant pressure specific heat divided by temperature. The final differential equation which governs the temperature profiles within the regenerator is expressed in Equation 2.19.

$$\frac{c_p(P_{high}, T_2(x))}{T_2(x)} \cdot \frac{dT_2}{dx} = \frac{c_p(P_{low}, T_4(x))}{T_4(x)} \cdot \frac{dT_4}{dx} \tag{2.19}$$

The inlet and outlet temperature profiles are referred to as $T_2(x)$ and $T_4(x)$, respectively. These profiles adjust themselves according to the specific heat mismatch at each location.

To understand what effect the specific heat variation has on overall cycle performance, an ideal 'mechanical' cycle is examined. All of the refrigeration produced by a 'mechanical' cycle is due to helium expansion. Therefore, no 'subcooled' helium enters the cold heat exchanger and no 'superheated' helium enters the warm heat exchanger. The endpoints of both the inlet and outlet temperature profiles are fixed. Theoretically, Equation 2.19 can be solved to yield the inlet and outlet temperature profiles subject to a given set of operating conditions.

The behavior of the constant pressure specific heat of helium at low temperature is illustrated in Figure 2-11. The specific heat of helium at low pressure and temperature is significantly larger than at higher pressures. If the temperature differentials in Equation 2.19 are neglected, then a ratio of the outlet to the inlet temperature at each position is obtained in terms of the constant pressure specific heats.

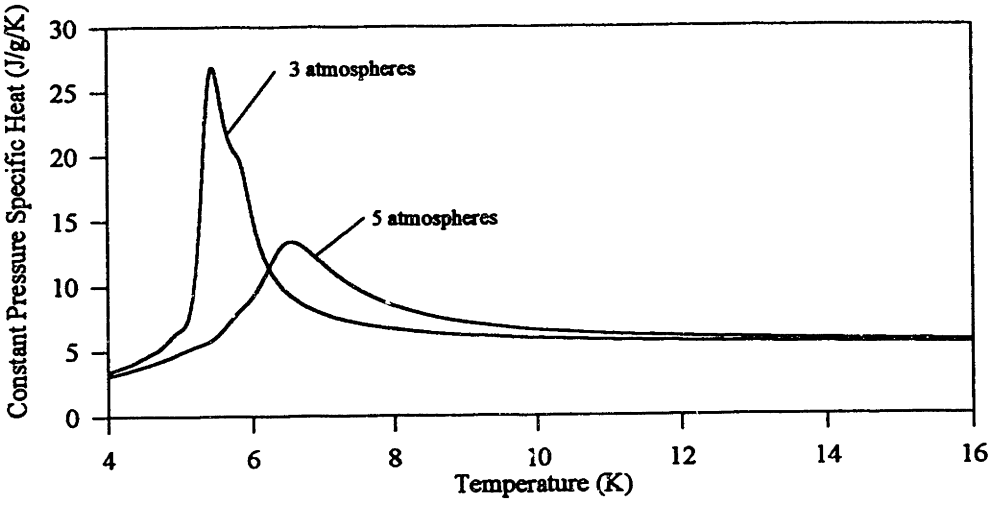

Figure 2-11. Low Temperature Behavior of Helium Constant Pressure Specific Heat

$$\frac{T_2(x)}{T_4(x)} \approx \frac{c_p(P_{high}, T_2(x))}{c_p(P_{low}, T_4(x))} \leq 1 \qquad (2.20)$$

Equation 2.20 indicates that the temperature during the high pressure inflow process will be less than the temperature during the low pressure outflow process. The magnetic material accepts heat from the low temperature flowing helium during the inlet process and rejects heat to the helium at a higher temperature during the outlet process. Since the magnetic material is forced to act as a heat pump by the specific heat mismatch, a net magnetic work input is required. This magnetic work input does not contribute to the total refrigeration produced and constitutes a degradation of cycle performance.

A similar effect occurs regardless of the type of cycle being considered. An approximate solution to Equation 2.19 is obtained in Appendix C and illustrated in Figure 2-12. The solution is generated based on a refrigeration temperature of 4.75° K, a warm reservoir temperature of 12° K, a high pressure of 500 kPa, and a low pressure of 300 kPa.

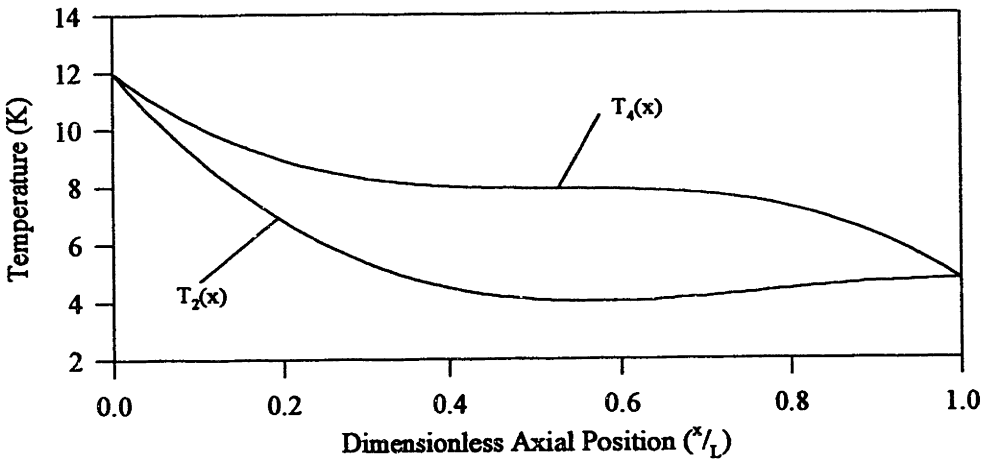

Figure 2-12. 'Mechanical' Cycle Temperature Profiles with Nonideal Helium Properties

III. DEVELOPMENT OF THE NONIDEAL MODEL

In Chapter 2, the mechanisms of refrigeration within a magnetically active regenerative refrigerator were examined. The refrigeration cycles were simulated under ideal conditions in order to determine the advantages associated with allowing a pressure variation. The results of this analysis indicated that significantly more refrigeration can be obtained when the helium pressure is cyclically changed.

The analysis carried out in Chapter 2 is only valid under a restrictive set of conditions. In this chapter, a model is developed which relaxes these assumptions. This model is a computational simulation which divides the regenerator into a series of axial segments (from 1 to m) and divides the cycle into a series of time steps (from 1 to n). The solution technique is numerically intensive and the program code requires the computational resources of a supercomputer platform in order to run effectively.

This chapter is divided into six sections. In the first section, the procedure used to determine the properties of Helium-4 is described. The second section details the process associated with moving forward an axial step. In the third section, the method of simulating a single time step is explained. The fourth section examines the numerical techniques used. The fifth section investigates the validity and effect of the assumptions made during the development of the nonideal model. The final section shows how the results of the model are used to determine cycle characteristics.

3.1 Determination of Helium Properties

The properties of Helium-4 at low temperature are highly nonideal. Representing these properties realistically is critical to the accuracy of the model.

The properties of Helium-4 are obtained through bicubic spline interpolation on tables of properties in the dimensions of temperature and pressure. These tables are constructed using the FORTRAN program *NBS Standard Reference Database* 12[33]. This program uses a series of empirically based equations to generate the properties of Helium-4. The program is based on the PvT surface described by McCarty[34]. The helium property tables constructed in this fashion include: density, enthalpy, entropy, viscosity, and thermal conductivity. Any other properties or property derivatives which are required can be obtained from this set of properties or their derivatives.

The theory of cubic and bicubic spline interpolation is described in Appendix B. The cubic spline technique yields interpolated results with a continuous second derivative[35]. This is in contrast to linear interpolation, where the second derivative is undefined at each abscissa and zero within each interval.

Thermodynamically, this feature is very important. The higher order derivatives of the thermodynamic properties contain the second law consistency. In particular, the higher order derivatives guarantee the concavity of the fundamental relation with respect to energy and the convexity of the energy relation with respect to entropy[36]. When numerical models based on thermodynamic principles employ a set of properties which is not self-consistent, they may converge to the point of greatest second law inconsistency. For this reason, the extra computational time required to implement cubic interpolation is justified.

3.2 Simulation of an Axial Step

It was difficult to develop a finite difference technique which could simulate an irreversible process involving two thermodynamic fluids. Techniques based purely on first law equations yielded results which were inconsistent from a second law standpoint (due to the fluctuating nature of the helium properties). Moreover, it was impossible to solve the problem entirely from a second law standpoint due to the presence of inherently irreversible processes.

The problem was solved using an iterative technique based on both first and second law equations. This technique ensures that the solution will be thermodynamically consistent and also accurately characterize the irreversible processes.

This section is divided into four subsections. The first subsection describes the overall solution procedure used to move forward one axial step. The section subsection describes how the heat and momentum transport processes are modelled. In the third subsection, the first law equations are developed. The final subsection shows how second law balances are used to check and adjust the first law results.

3.2.1 Axial Step Simulation Procedure

A differential regenerator segment is modelled as shown in Figure 3-1. The pressure gradient is assumed to be concentrated in adiabatic fluid resistances between adjacent

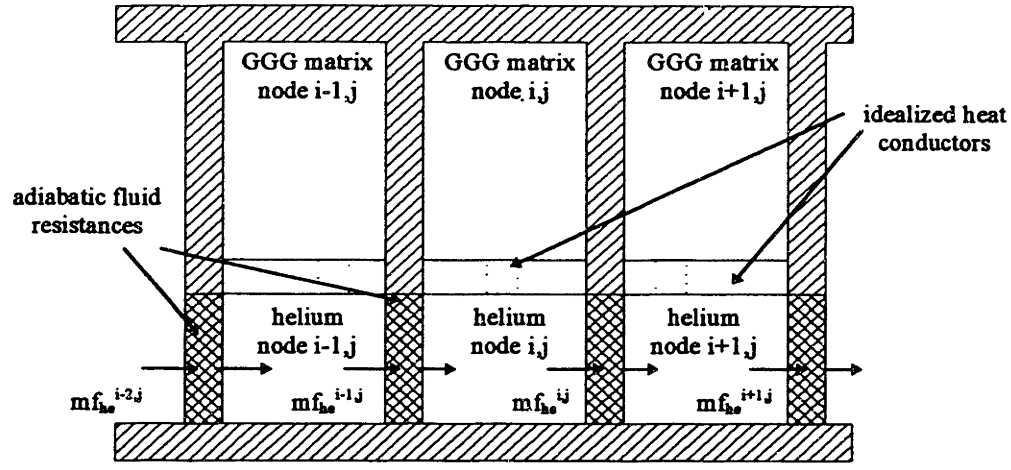

Figure 3-1. Schematic of Differential Regenerator Model

nodes. The heat transfer between the GGG and the helium is assumed to pass through ideal conductors. Since axial heat conduction is neglected, the only energy or entropy flow between adjacent nodes occurs due to the bulk motion of the helium.

Subscripts will be used to refer to the axial and temporal position of a node. The first subscript (i) will refer to the axial position within the solution grid. The second subscript (j) will refer to the temporal position (e.g. $T^{i,j}$ refers to the i'th node from the warm end and the j'th timestep from the beginning of a cycle; cycles are assumed to start with the compression/adiabatic demagnetization process). The solution proceeds in the direction of flow. This is necessary in order to have the number of axial boundary conditions required to solve the finite difference equations at each node.

The solution procedure is illustrated in Figure 3-2. The characteristics of the helium flow entering the node are used to model the transport processes (i.e. the heat transfer and pressure drop). A first order finite difference solution is developed in Subsection 3.2.3 based on continuity and energy equations. The partial derivatives of helium and GGG properties used to linearize the differential equations are evaluated at a representative temperature (i.e. T_{he}^{r} and T_{GGG}^{r}). All of the numerical uncertainty associated with the finite difference technique is generated by the property fluctuation over the axial and temporal space contained within the node. Most of this uncertainty is contained in the temperature induced variations (rather than pressure or applied induced variations). Theoretically, there is a set of representative temperatures which will give the correct

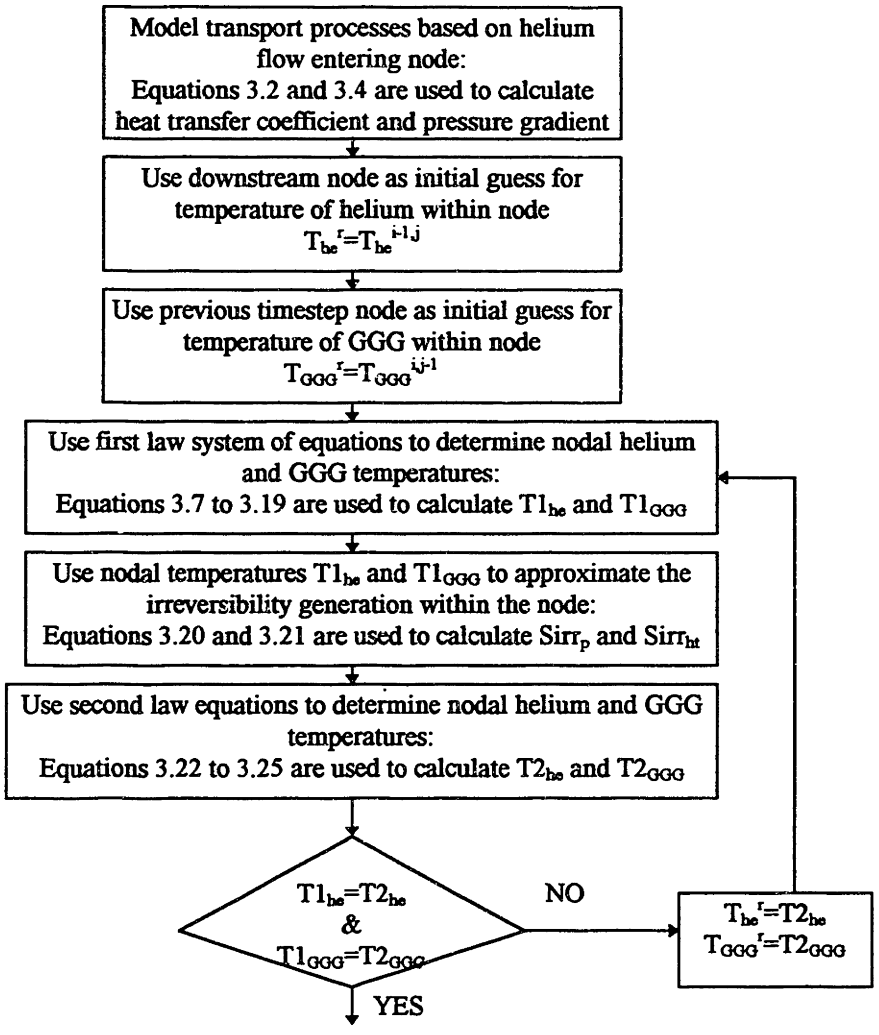

Figure 3-2. Flowchart of Axial Step Simulation Procedure

average property and property derivative values during an axial step simulation. This set of temperatures can only be obtained through iteration.

The nodal temperatures obtained with first law equations (i.e. $T1_{he}$ and $T1_{GGG}$) are used to approximate the irreversibility generated by pressure and temperature gradients. The second law balances can then be solved for new nodal temperatures (i.e. $T2_{he}$ and $T2_{GGG}$). These temperatures are used to adjust the representative temperatures within the first law balances. This process is repeated until the nodal temperatures converge to final values which must satisfy both the first and second laws.

3.2.2 Transport Modelling

The transport processes are modelled with empirical correlations obtained from the literature. These processes are dependent upon the helium flow regime. The flow within the regenerator at a given node depends upon the regenerator geometry, helium flow rate, and helium properties.

The regenerator geometry considered is a packed bed composed of spherical particles. The helium flow can be characterized with the Reynolds Number based on the packed bed particle diameter (d_p) and the unobstructed helium velocity (V_o). The unobstructed velocity is the velocity required to give the same mass flow in an empty regenerator. The Reynolds Number is calculated using the mass flow, temperature, and pressure of the helium which enters the node.

$$\mathrm{Re}_{d_p} = \frac{mf_{he}^{\,i-1,j} \cdot d_p}{A \cdot \mu_{he}(T_{he}^{\,i-1,j}, P^{i-1,j})} \tag{3.1}$$

The heat transfer process is modelled with a heat transfer coefficient. This coefficient is calculated from a semi-empirical correlation for packed beds[37].

$$Nu_{d_p} = \frac{d_p \cdot htc}{k_{he}(T_{he}^{\,i-1,j}, P^{i-1,j})} = 2 + \mathrm{Pr}(T_{he}^{\,i-1,j}, P^{i-1,j})^{1/3} \cdot \mathrm{Re}_{d_p}^{\,1/2} \tag{3.2}$$

The heat transfer occurs over the entire surface area of the packed bed particles. For spherical particles, the area available for heat transfer per volume of regenerator (i.e. the specific area or aa) is inversely proportional to the particle diameter.

$$aa = \frac{6}{d_p} \tag{3.3}$$

Another semi-empirical correlation is used to determine the pressure gradient in the axial direction[38].

$$\left(\frac{\partial P}{\partial x}\right)_t = \frac{150 \cdot (1-\varepsilon)^2 \cdot \nu_{he}(T_{he}^{\,i-1,j}, P^{i-1,j}) \cdot mf_{he}^{\,i-1,j}}{\varepsilon^3 \cdot d_p^{\,2} \cdot A} + \frac{1.75 \cdot (1-\varepsilon) \cdot \left(mf_{he}^{\,i-1,j}\right)^2}{\varepsilon^3 \cdot d_p \cdot A^2 \cdot \rho_{he}(T_{he}^{\,i-1,j}, P^{i-1,j})} \tag{3.4}$$

Equation 3.4 allows the pressure in the node and the partial derivative of pressure with respect to time to be calculated immediately.

$$P^{i,j} = P^{i-1,j} + \left(\frac{\partial P}{\partial x}\right)_t \cdot dx \tag{3.5}$$

$$\left(\frac{\partial P}{\partial t}\right)_x = \frac{(P^{i,j} - P^{i,j-1})}{dt} \tag{3.6}$$

The transport processes are not iterated because they are fairly insensitive to small temperature fluctuations.

3.2.3 First Law Equations

The governing equations are obtained from a mass balance and two energy balances. A differential mass balance around the entrained helium yields:

$$\left[\left(\frac{\partial mf_{he}}{\partial x}\right)_t\right] + A \cdot \varepsilon \cdot \left[\left(\frac{\partial \rho_{he}}{\partial t}\right)_x\right] = 0 \tag{3.7}$$

An energy balance around the entrained helium must include heat transfer to the GGG, energy flow due to bulk helium motion, and energy storage. After expanding the partial derivatives and substituting Equation 3.7, the helium energy balance can be expressed as:

$$\frac{P^{i-1,j}}{\rho_{he}(T_{he}^{\,r}, P^{i-1,j})}\left[\left(\frac{\partial mf_{he}}{\partial x}\right)_t\right] + mf_{he}^{\,i-1,j} \cdot \left[\left(\frac{\partial h_{he}}{\partial x}\right)_t\right]$$
$$+ A \cdot \varepsilon \cdot \rho_{he}(T_{he}^{\,r}, P^{i-1,j})\left[\left(\frac{\partial u_{he}}{\partial t}\right)_x\right] + [q'] = 0 \tag{3.8}$$

The internal process undergone by the GGG is assumed to be reversible.

$$[q'] - \rho_{GGG} \cdot (1-\varepsilon) \cdot T_{GGG}^{\,i,j-1} \cdot A \cdot \left[\left(\frac{\partial s_{GGG}}{\partial t}\right)_x\right] = 0 \tag{3.9}$$

The condition of reversible heat storage within the GGG is a computational assumption. The validity is investigated in Subsection 3.4.2.

The axial and temporal property derivatives are expanded by the chain rule into their temperature and pressure components.

$$\left[\left(\frac{\partial \rho_{he}}{\partial t}\right)_x\right] = \left(\frac{\partial \rho_{he}}{\partial P}\right)_T (T_{he}^{\,r}, P^{i,j}) \cdot \left(\frac{\partial P}{\partial t}\right)_x + \left(\frac{\partial \rho_{he}}{\partial T}\right)_P (T_{he}^{\,r}, P^{i,j}) \cdot \left[\left(\frac{\partial T_{he}}{\partial t}\right)_x\right] \quad \textbf{(3.10)}$$

$$\left[\left(\frac{\partial h_{he}}{\partial x}\right)_t\right] = \left(\frac{\partial h_{he}}{\partial P}\right)_T (T_{he}^{\,r}, P^{i,j}) \cdot \left(\frac{\partial P}{\partial x}\right)_t + c_p (T_{he}^{\,r}, P^{i,j}) \cdot \left[\left(\frac{\partial T_{he}}{\partial x}\right)_t\right] \quad \textbf{(3.11)}$$

$$\left[\left(\frac{\partial u_{he}}{\partial t}\right)_x\right] = \left(\frac{\partial u_{he}}{\partial P}\right)_T (T_{he}^{\,r}, P^{i,j}) \cdot \left(\frac{\partial P}{\partial t}\right)_x + \left(\frac{\partial u_{he}}{\partial T}\right)_P (T_{he}^{\,r}, P^{i,j}) \cdot \left[\left(\frac{\partial T_{he}}{\partial t}\right)_x\right] \quad \textbf{(3.12)}$$

$$\left[\left(\frac{\partial s_{GGG}}{\partial t}\right)_x\right] = \left(\frac{\partial s_{GGG}}{\partial \mu_o H}\right)_T (T_{GGG}^{\,r}, \mu_o H) \cdot \frac{d\mu_o H}{dt} + \left(\frac{\partial s_{GGG}}{\partial T}\right)_{\mu_o H} (T_{GGG}^{\,r}, \mu_o H) \cdot \left[\left(\frac{\partial T_{GGG}}{\partial t}\right)_x\right] \quad \textbf{(3.13)}$$

The heat transfer process is modelled with the heat transfer coefficient calculated with Equation 3.2.

$$[q'] = aa \cdot A \cdot htc \cdot \left(\left[T_{he}^{\,ht}\right] - \left[T_{GGG}^{\,ht}\right]\right) \quad \textbf{(3.14)}$$

The heat transfer is assumed to occur through an ideal conductor between the entrained helium and the GGG. The temperature at which the helium transfers heat is assumed to be a weighted average of the temperature of the entering helium, the final temperature of the helium, and the temperature of the original entrained helium. Similarly, the temperature at which the GGG transfers heat is assumed to be a weighted average of the final and initial temperature of the GGG contained in the node. The effect and validity of this heat transfer representation is investigated in Subsection 3.4.3. In Subsection 3.4.3, it is shown that the exact weightings become unimportant provided the grid size is sufficiently fine. The final weightings are shown in Equations 3.15 and 3.16.

$$\left[T_{he}^{\,ht}\right] = 0.8 \cdot \left[T_{he}^{\,i,j}\right] + 0.1 \cdot T_{he}^{\,i-1,j} + 0.1 \cdot T_{he}^{\,i,j-1} \quad \textbf{(3.15)}$$

$$\left[T_{GGG}^{\,ht}\right] = 0.9 \cdot \left[T_{GGG}^{\,i,j}\right] + 0.1 \cdot T_{GGG}^{\,i,j-1} \quad \textbf{(3.16)}$$

The final three equations are obtained by neglecting the higher order axial and temporal derivatives. The resulting linearized equations relate the nodal temperatures to the axial and temporal boundary conditions.

$$\left[T_{he}^{\,i,j}\right] = T_{he}^{\,i-1,j} + \left[\left(\frac{\partial T_{he}}{\partial x}\right)_t\right] \cdot dx \tag{3.17}$$

$$\left[T_{he}^{\,i,j}\right] = T_{he}^{\,i,j-1} + \left[\left(\frac{\partial T_{he}}{\partial t}\right)_x\right] \cdot dt \tag{3.18}$$

$$\left[T_{GGG}^{\,i,j}\right] = T_{GGG}^{\,i,j-1} + \left[\left(\frac{\partial T_{GGG}}{\partial t}\right)_x\right] \cdot dt \tag{3.19}$$

Equations 3.7 through 3.19 form a system of thirteen linear equations in thirteen unknowns. The unknowns are indicated in square brackets. This system of equations is placed in matrix form and solved with lower upper decomposition and back substitution[39]. The subroutines used to solve Equations 3.7 through 3.19 are called LUDCMP.c and LUBKSB.c. These subroutines are listed in Appendix A.6.

The solution of these equations yields the first law projection of the nodal temperatures. These projections are referred to as $T1_{he}$ and $T1_{GGG}$.

3.2.4 Second Law Equations

The results of the first law balances are used to estimate the irreversibility generated due to pressure and temperature gradients within the node.

The pressure gradient induced irreversibility is determined using an entropy balance around the adiabatic fluid resistance through which the incoming helium flows.

$$Sirr_p = mf_{he}^{\,i-1,j} \cdot dt \cdot (s_{he}(h_{he}^{\,i-1,j}, P^{i,j}) - s_{he}(T^{i-1,j}, P^{i-1,j})) \tag{3.20}$$

All of this irreversibility is assumed to be concentrated in the ideal fluid resistance. The entropy corresponding to the enthalpy of the incoming flow and the reduced pressure is obtained with the function SHHP.c. This function uses the same interpolation technique as TGSUOH.c and is therefore not listed.

The temperature gradient induced irreversibility is assumed to be concentrated within the ideal conductor.

$$Sirr_{ht} = q' \cdot dx \cdot dt \cdot \left(\frac{1}{T_{he}^{\ ht}} - \frac{1}{T_{GGG}^{\ ht}} \right) \tag{3.21}$$

Modelling the irreversibility generation in discrete resistances and conductors is an assumption investigated in Subsection 3.5.1.

The final helium entropy can be determined with an entropy balance around the entrained helium and the adiabatic fluid resistance by assuming that the temperature of the helium leaving the node is the same as the final temperature of the entrained helium.

$$s_{he} = \frac{Sirr_p + s_{he}(T_{he}^{\ i-1,j}, P^{i-1,j}) \cdot \left[mf_{he}^{\ i-1,j} \cdot dt + A \cdot \varepsilon \cdot dx \cdot \rho_{he}(T_{he}^{\ i,j-1}, P^{i,j-1}) \right] - \frac{q' \cdot dt \cdot dx}{T_{he}^{\ ht}}}{mf_{he}^{\ i-1,j} \cdot dt + A \cdot \varepsilon \cdot dx \cdot \rho_{he}(T_{he}^{\ i,j-1}, P^{i,j-1})} \tag{3.22}$$

An entropy balance around the GGG and the conductor yields the final entropy of the magnetic material.

$$s_{GGG} = s_{GGG}(T_{GGG}^{\ i,j-1}, \mu_o H^{i,j-1}) + \frac{Sirr_{ht} + \frac{q' \cdot dt \cdot dx}{T_{he}^{\ ht}}}{\rho_{GGG} \cdot A \cdot (1-\varepsilon) \cdot dx} \tag{3.23}$$

The helium entropy and pressure are sufficient to determine the final temperature of the entrained helium.

$$T2_{he} = T_{he}(s_{he}, P^{i,j}) \tag{3.24}$$

The GGG entropy and applied field are sufficient to determine the final temperature of the GGG.

$$T2_{GGG} = T_{GGG}(s_{GGG}, \mu_o H^{i,j}) \tag{3.25}$$

3.3 Simulation of a Time Step

This section describes the procedure used to simulate a timestep. The procedure differs depending on whether the time step is a downflow (i.e. flow from the warm to the cold end) or an upflow (i.e. from the cold to the warm end) situation.

The method of solution is dictated by the appropriate axial boundary conditions. The boundary conditions required can be determined by examining the governing equations developed in Section 3.2. Each partial derivative requires a boundary condition in order to determine a unique solution to the associated differential equation.

The temporal derivatives all require an initial condition (i.e. the previous timestep). The axial derivatives require a boundary value at some point along the axis. Examination of the governing equation reveals axial derivatives of pressure, mass flow rate, and helium temperature. Different boundary conditions are imposed during a downflow process and an upflow process. The location of the boundary conditions dictates the solution technique.

This section is divided into two subsections. The first subsection explains the procedure used to simulate a downflow timestep. In the second section, the procedure for an upflow timestep is detailed. Two other flow conditions can also occur. In some situtations, helium can exit or enter at both ends of the regenerator (i.e. 'mixed flow' processes). This model cannot simulate these situations. Fortunately, the pressure gradients associated with a spherical packed bed are not extreme enough to create 'mixed flow' situations.

During the development of an experimental apparatus in Chapter 5, a crushed bed regenerator is considered. The extreme pressure gradients associated with this type of geometry causes 'mixed flow' situations to arise. These situations are handled one timestep at a time by manually varying the location of the 'zero velocity' point and the 'zero velocity' pressure until the boundary conditions are satisfied. The program used for this purpose is not listed.

3.3.1 Downflow Process

A downflow process occurs when helium enters at the warm end and exits at the cold end. Therefore, the helium temperature boundary condition is set at the warm end by the warm heat exchanger. Since the warm heat exchanger is assumed to be perfect, the helium enters the regenerator at precisely the warm reservoir temperature. The pressure boundary condition

is also set at the warm end since the pressure variation is assumed to be imposed upon the warm end of the device (as shown in Figure 1-8) and pressure drop through the warm heat exchanger is neglected.

The mass flow rate boundary condition is obtained by applying mass conservation to the cold heat exchanger and displacer. These devices are assumed to be isothermal and axially isobaric.

$$mf_{cold} = (V_{dis} + V_{chx}) \cdot \left(\frac{\partial \rho_{he}}{\partial P}\right)_T (T_{cold}, P^{m,j}) \cdot \frac{(P^{m,j} - P^{m,j-1})}{dt} + \rho_{he}(T_{cold}, P^{m,j}) \cdot A_{dis} \cdot v_{dis} \quad \textbf{(3.26)}$$

The mass flow rate boundary condition is a function of the unknown cold end pressure ($P^{m,j}$). The solution proceeds by using a shooting technique for a two point boundary value problem[40]. This method attempts to match the boundary conditions at the endpoint of integration by varying the freely specifiable (i.e. unconstrained) starting values. In this case, the mass flow rate at the warm end is varied until the mass flow rate at the cold end matches Equation 3.26.

The procedure for moving an axial step developed in Section 3.2 is applied repeatedly in order to move from the warm to the cold end based on a warm end mass flow which is not directly constrained. The resulting cold end mass flow rate is compared to Equation 3.26 and the discrepancy is used to adjust the warm end mass flow rate.

The procedure used to adjust the input mass flow rate is based on linear interpolation. The warm end mass flow rate is rapidly varied until the desired cold end mass flow rate has been bracketed. Linear interpolation quickly zeroes in on the correct mass flow rate. The iterative process continues until the mass flow rate at the cold end matches Equation 3.26 within the desired tolerance (i.e. 0.02 g/s).

3.3.2 Upflow Process

During an upflow process, mass enters the regenerator at the cold end and exits at the warm end. Therefore, the axial helium temperature boundary condition is set at the cold end. The mass flow rate boundary condition is fixed at the cold end by Equation 3.26 (which again involves the unknown cold end pressure). The pressure boundary condition is specified at the warm end.

The procedure for solving an upflow process is similar to a downflow process. The unconstrained cold end pressure is varied and the equations are integrated from the cold to the warm end (i.e. in the direction of flow). The process is complete when the warm end pressure matches the imposed pressure within the desired tolerance (i.e 0.05 kPa).

The model is entirely specified at this point. The program is called REGEN.c and is listed in Appendix 7.

3.4 Numerical Methodology

The solution technique described in Sections 3.2 and 3.3 was chosen only after an extensive period of trial and error involving other numerical methods. This section is divided into two subsections. The first subsection lists some of the numerical methods which were tried and discarded during the trial and error period. The second subsection is a stability analysis based on the first law difference equations.

3.4.1 Solution Techniques Attempted

Several numerical methods were tried during the development of this computational model. These techniques are listed in Table 3-1.

The initial techniques used only first law equations. The Runge-Kutta Fourth Order[41] model gave results which seemed physically reasonable until they were checked with the second law over a steady state cycle. In order to obtain greater accuracy at reduced computational speed, an Adams-Bashforth four step[42] model was developed. This model also failed to satisfy the second law.

The next model used an implicit method of solution axially and an explicit technique spatially. All of the required derivatives were evaluated based on the previous timestep

Numerical Technique	Reason for Failure
Runge Kutta Fourth Order Single Step Method	second law discrepancies
Adams Bashforth Fourth Order Multistep Method	second law discrepancies
Newton Raphson Method over a timestep	instability, second law discrepancies
Newton Raphson Method over a cycle	instability, second law discrepancies

Table 3-1. Solution Techniques Tried and Discarded

and a system of linear equations was generated for the next timestep. These equations were solved by matrix decomposition. The results were then used to recalculate the derivatives. The process was iterated until the profiles all converged. This process is referred to in Table 3-1 as a Newton Raphson Method over a timestep. The model was unstable and failed to satisfy the second law. The Newton Raphson Method was attempted over an entire cycle with equally discouraging results.

3.4.2 Stability Analysis based on the First Law Difference Equations

The partial differential equations which govern this problem are formally derived in Subsection 3.3.3 (Equations 3-7 to 3-9). These equations are combined to give:

$$
\begin{aligned}
& mf_{he} \cdot c_p \cdot \left(\frac{\partial T_{he}}{\partial x}\right)_t + mf_{he}\left(\frac{\partial h_{he}}{\partial P}\right)_T \cdot \left(\frac{\partial P}{\partial x}\right)_t + \\
& \left[A \cdot \varepsilon\left[\rho_{he} \cdot \left(\frac{\partial u_{he}}{\partial T}\right)_P - \frac{P}{\rho_{he}} \cdot \left(\frac{\partial \rho_{he}}{\partial T}\right)_P\right]\right] \cdot \left(\frac{\partial T_{he}}{\partial t}\right)_x + \\
& A \cdot (1-\varepsilon) \cdot \rho_{GGG} \cdot T_{GGG} \cdot \left(\frac{\partial s_{GGG}}{\partial T}\right)_{\mu_o H} \cdot \left(\frac{\partial T_{GGG}}{\partial t}\right)_x + \\
& A \cdot (1-\varepsilon) \cdot \rho_{GGG} \cdot T_{GGG} \cdot \left(\frac{\partial s_{GGG}}{\partial \mu_o H}\right)_T \cdot \left(\frac{d\mu_o H}{dt}\right) + \\
& \left[A \cdot \varepsilon\left[\rho_{he} \cdot \left(\frac{\partial u_{he}}{\partial P}\right)_T - \frac{P}{\rho_{he}} \cdot \left(\frac{\partial \rho_{he}}{\partial P}\right)_T\right]\right] \cdot \left(\frac{\partial P_{he}}{\partial t}\right)_x = 0
\end{aligned}
\qquad \textbf{(3.27)}
$$

The nonlinear nature of this equation is evident in the temperature, pressure, and applied field dependency of the thermodynamic properties; as well as in the mass flow which multiplies the helium axial derivatives and the temperature which multiplies the GGG entropy derivatives. As a result, the stability criterion constantly changes during the simulation. This analysis approximates the stability criterion based on several simplifications.

Equation 3.27 can be examined in order to determine the relative size of the time and axial steps required in order to ensure stability. The pressure effects are generally small relative to the temperature effects and can be neglected in this analysis. The time derivative of the helium temperature is approximately equal to that of the GGG temperature. These simplifications reduce the complexity of the governing partial differential equation.

$$\left[A\cdot(1-\varepsilon)\cdot\rho_{GGG}\cdot T_{GGG}\cdot\left(\frac{\partial s_{GGG}}{\partial T}\right)_{\mu_o H}+A\cdot\varepsilon\cdot\left[\rho_{he}\cdot\left(\frac{\partial u_{he}}{\partial T}\right)_P-\frac{P}{\rho_{he}}\cdot\left(\frac{\partial \rho_{he}}{\partial T}\right)_P\right]\right]\cdot\left(\frac{\partial T}{\partial t}\right)+$$

$$mf_{he}\cdot c_p\cdot\left(\frac{\partial T}{\partial x}\right)=\text{constant} \qquad \textbf{(3.28)}$$

Equation 3.28 indicates that the problem can be viewed as a balance between the temporal energy change caused by heat absorption within the GGG and helium entrained in the node, the enthalpy change in the flowing helium, and some constant which represents the driving magneto-caloric effect. The critical ratio of the time step to the length step can be obtained from these assumptions[43].

$$r_{cr}=\frac{\Delta t}{\Delta x}=\frac{\left[A\cdot(1-\varepsilon)\cdot\rho_{GGG}\cdot T_{GGG}\cdot\left(\frac{\partial s_{GGG}}{\partial T}\right)_{\mu_o H}+A\cdot\varepsilon\cdot\left[\rho_{he}\cdot\left(\frac{\partial u_{he}}{\partial T}\right)_P-\frac{P}{\rho_{he}}\cdot\left(\frac{\partial \rho_{he}}{\partial T}\right)_P\right]\right]}{mf_{he}\cdot c_p} \qquad \textbf{(3.29)}$$

The numerator of this ratio is composed of two parts. The first term represents the absorption of heat by the magnetic material and the second represents the absorption of heat by the entrained helium. Typically, the first term dominates and the problem reduces to a balance between GGG regeneration and helium flow cooling. However, at temperatures below approximately 6° K, the density and specific heat of helium increase dramatically. In this temperature range, the second term is of the same order of magnitude as the first.

Figure 3-3 illustrates the relative magnitude of the two terms in the numerator of Equation 3.29. The critical ratio of time step size to axial step size is plotted in Figure 3-4 for unity mass flow rate. The critical ratio must be less than one half in order to ensure stability and convergence of the numerical technique. By reducing the critical ratio to less than one fourth, the errors generated by the finite difference approximation will not oscillate[44]. Figure 3-4 shows that the correct axial length step spacing is much smaller in the low temperature region than in the higher temperature regions. The simulations presented in Chapter 4 use a ratio of 2.49. This is well below one half of the critical ratio over the entire temperature range.

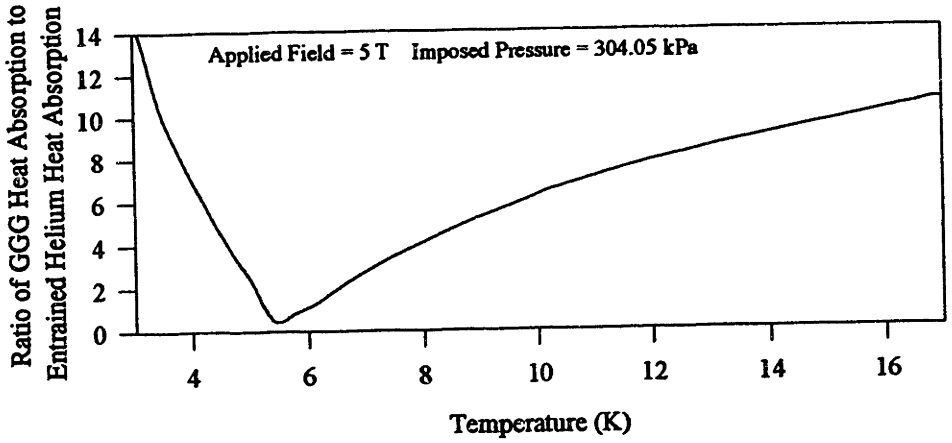

Figure 3-3. Relative Magnitude of Time Step Terms

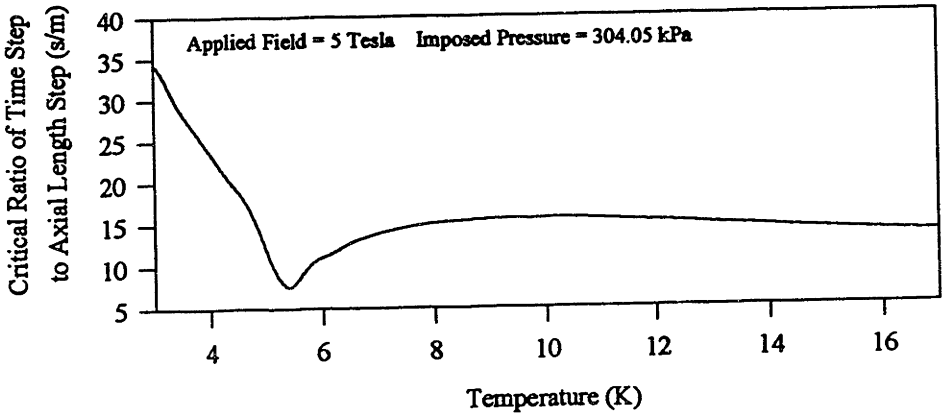

Figure 3-4. Critical Ratio of Time to Axial Step Size

3.5 Validity of Modelling Assumptions

In Section 3.2, a differential segment of a magnetically active regenerator was modelled. The three assumptions made during the development of this model are listed below.

1. The entropy generation associated with flow through a pressure gradient and heat transfer through a temperature gradient can be represented by discrete fluid resistances and heat conductors.
2. The GGG undergoes an internally reversible process.
3. The heat transfer interaction can be modeled using representative heat tranfer temperatures which a weighted average of the temperatures existing within the axial and temporal space occupied by the node.

This section is divided into three subsections. In the first subsection, the effect of modelling the irreversible processes discretely rather than as a distributed phenomenon is examined. In the second section, the reversibility of the internal GGG heat transfer process is examined. The final section investigates the validity of modelling the heat transfer process with average nodal temperatures.

3.5.1 Transport Simulation

Irreversibility generation due to imperfect heat and momentum transport are expected to be important effects within any real regenerative device. These processes are included in the simulation as discrete sources of irreversibility within each node. In order to model these processes in discrete devices rather than as a distributed phenomenon, the grid size must be sufficiently fine.

The entropy generated by the heat transfer process typically dominates the pressure drop irreversibility within a packed bed composed of spherical particles. Most of the entropy generation occurs during the nonflow processes due to the reduced heat transfer coefficient. The irreversibility generated over the course of a compression process is examined in Figure 3-5. The same process is simulated with several grid sizes (the ratio of the time step length to the axial step length was 2.5, as determined in Section 3.4.2). Figure 3-5 indicates that the assumption of discrete entropy generation is valid only when the axial step size is less than one centimeter.

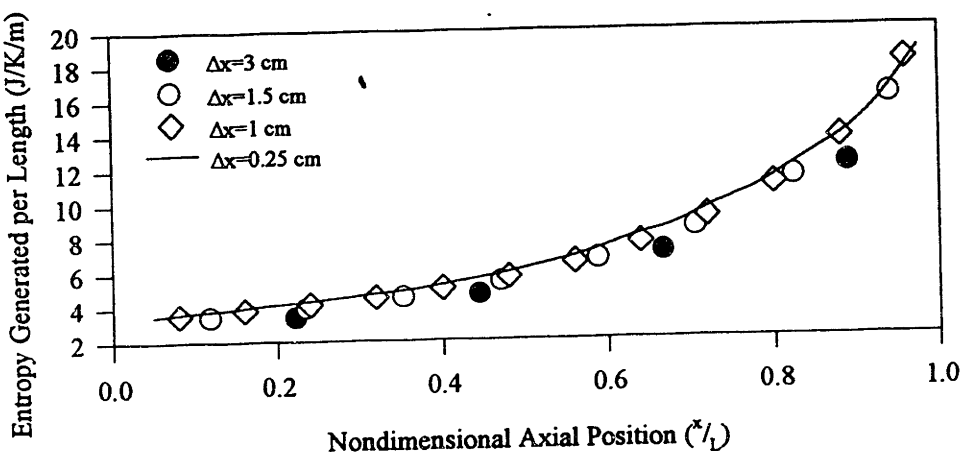

Figure 3-5. Convergence of Discrete Entropy Generation with Grid Size

3.5.2 Reversible GGG Process

The heat transfer process which occurs from the surface of the GGG to the helium is modelled by Equation 3.2. A separate heat transfer process transports heat from within the GGG matrix to the surface. This process is assumed to occur reversibly.

In order to determine the degree of irreversibility associated with the internal heat transfer process, the uniformity of the temperature within the GGG must be examined. The nondimensional parameter which characterizes the relative magnitude of the internal temperature gradient to the external temperature gradient is the Biot Number. The appropriate Biot Number for a packed bed is given by Equation 3.30.

$$Bi = \frac{htc \cdot d_p}{k_{GGG}} \tag{3.30}$$

The thermal conductivity of GGG in the range of 4° to 10° K has been empirically correlated[45].

$$k_{GGG} = 0.2423 \cdot T^{1.2381} \qquad (W / cm / K) \tag{3.31}$$

The Biot Number is largest during the flow processes since the external heat transfer process is enhanced by the helium motion. The variation of Biot Number at several axial locations during a typical inlet process is illustrated in Figure 3-6. This figure indicates that the internal temperature gradients are at most 33% of the external gradients. Typically, the internal temperature gradients are less than 10% of the external gradients.

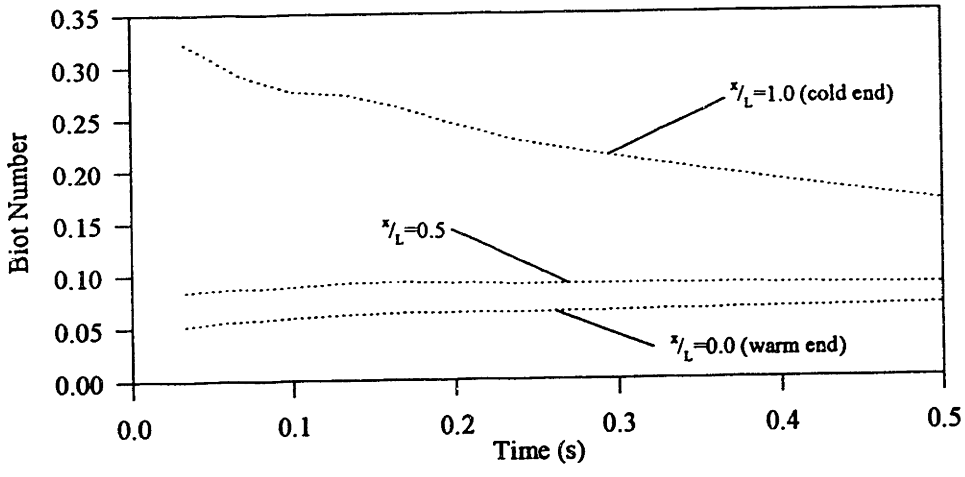

Figure 3-6. Biot Number for a Typical Flow Process

3.5.3 Heat Transfer Temperature Representation

The heat transfer process between the surface of the GGG and the helium is represented by a heat transfer coefficient. The interaction is assumed to occur between helium at some representative temperature and GGG at a different temperature. The temperatures between which the heat transfer occurs is not immediately obvious.

Several temperatures exist within the temporal and axial space occupied by any given node. The helium flows in at the temperature of the upstream node and leaves at the final temperature of the node. The entrained helium within the node is initially at the final temperature obtained during the previous timestep. Therefore, three helium temperatures are present within the space occupied by the node and some heat will necessarily be transferred at each of these temperatures. Specification of the temperature at which the GGG transfers heat is similarly unclear.

The representative temperatures for the heat transfer process must appropriately combine all of the temperatures present within the node. Equations 3.32 and 3.33 use a series of weightings in order to determine the final heat transfer temperatures. The weighting given to the final temperature of the nodes (i.e. wh and wg) are referred to as the nodal weightings.

$$T_{he}^{\ ht} = wh \cdot T_{he}^{\ i,j} + whts \cdot T_{he}^{\ i,j-1} + whas \cdot T_{he}^{\ i-1,j} \tag{3.32}$$

$$T_{GGG}^{\ ht} = wg \cdot T_{GGG}^{\ i,j} + wgts \cdot T_{GGG}^{\ i,j-1} \tag{3.33}$$

Fortunately, these temperatures vary at most by a few tenths of a degree. Also, the iterative process used to simulate an axial step reduces the effect of this intermediate calculation on the overall results. In Figure 3-7, the GGG and helium temperature profiles after a compression process are plotted. The same process was simulated with two different nodal weightings. The final profiles differ negligibly, indicating that the results are relatively insensitive to the representation scheme.

When the weight of the final nodal temperature is reduced below 25%, the solution becomes unstable causing the temperatures to oscillate in time. The weightings given in Equations 3.15 and 3.16 were used throughout Chapter 4.

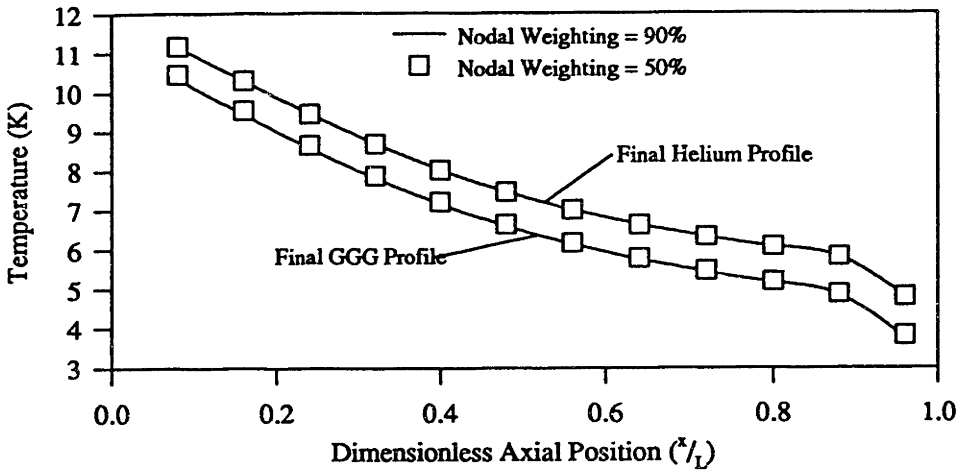

Figure 3-7. Temperature Profiles with Varying Heat Transfer Temperature Representation Schemes

3.6 Interpretation of Simulation Output

A differential segment of a magnetically active regenerative refrigerator is modelled in Section 3.2 after making three simplifying assumptions. These assumptions are checked in Section 3.5 and found to be reasonable. The simulation is propogated through time using the methods described in Section 3.3. The model calculates the helium temperature, GGG temperature, helium mass flow rate, and pressure at every axial and temporal node. In order for the simulation to be useful, it must converge to a steady state cycle and techniques must be developed in order to translate the output data into meaningful results (e.g. refrigeration).

This section is divided into five subsections. In the first subsection, the behavior of the model as it converges to steady state is examined. The second subsection describes how the output data can be interpreted to give the energy and entropy fluxes at any cross section. The third subsection explains how the output data may be used to calculate the magnetic work input. The fourth subsection defines the concept of mechanical and magnetic refrigeration components and describes how each component is calculated. In the final subsection, the methods used to determine the refrigeration characteristics of interest are explained. The techniques developed in Subsections 3.6.2 through 3.6.5 are illustrated using the model output for an example steady state cycle. The input parameters used to generate this example simulation correspond to those listed in Tables A-1 through A-3.

3.6.1 Convergence to Steady State

The simulation results are only useful when applied to a steady state refrigeration cycle. The extent to which steady state had been achieved is measured by the net entropy change of the helium contained in the regenerator over the course of a cycle. As this value approaches zero, the cycle approaches steady state.

The computational model exhibitd three types of convergence behavior: oscillation, divergence, and convergence. When oscillatory behavior occurs, the model alternatively displays positive and negative entropy storage over consecutive cycles. This behavior is exhibited when the computational tolerances are too large. These tolerances include the degree to which the first and second law temperatures must converge during an axial step simulation and the degree to which the unconstrained boundary condition must be met during a time step simulation.

Divergent behavior is characterized by entropy storage which continually increases, quickly moving the model out of the temperature range of interest. This behavior is exhibited when the initial temperature profiles (i.e. the temporal boundary condition) differ significantly from the steady state profiles corresponding to the input conditions. This behavior may be avoided by using the final temperature profiles from a steady state cycle with similar input parameters as the initial temperature profiles for a new simulation.

Convergent behavior is illustrated in Figure 3-8. The regenerator initially exhibits large

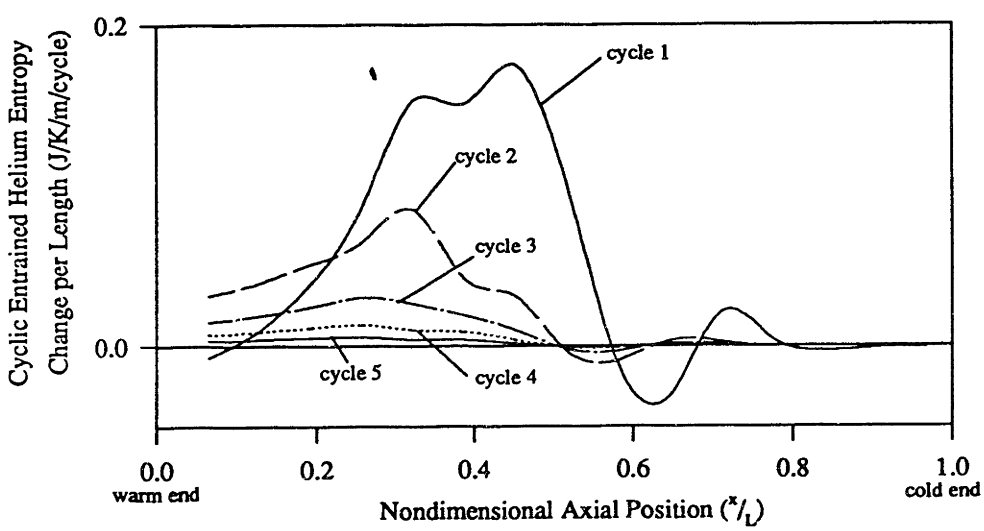

Figure 3-8. Cyclic Entropy Storage During Convergence to Steady State

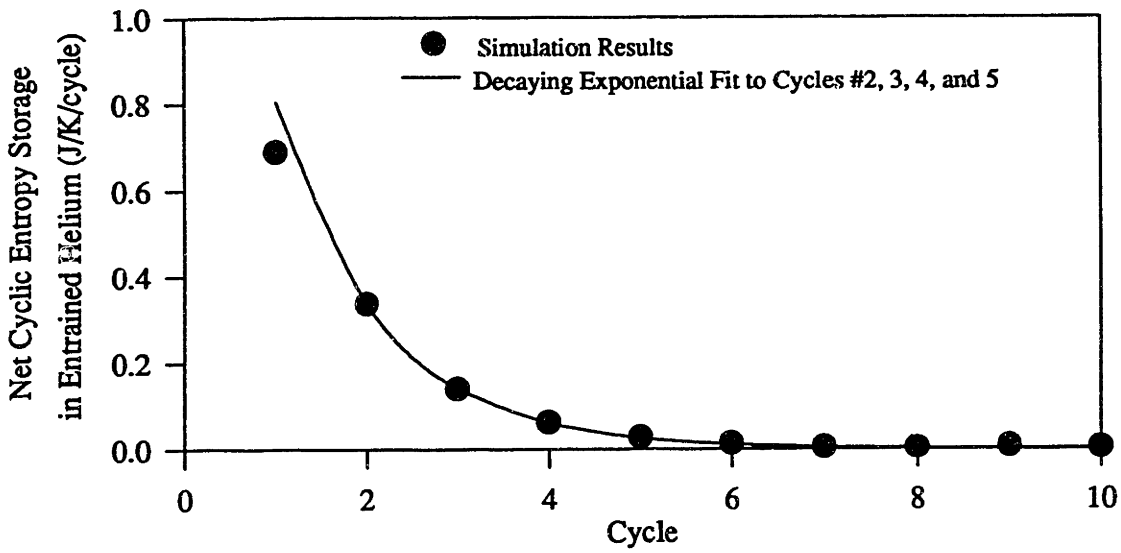

Figure 3-9. Decaying Exponential Extrapolation to Steady State

transients which are damped out from the cold end to the warm end. The regenerator then approaches steady state very gradually as shown in Figure 3-9.

The regenerator may be viewed as a dynamic system subject to a nonzero initial condition (i.e. the initial temperature profile). Figure 3-9 suggests that the response of the device resembles a first order system. There is a physical explanation for this behavior. The system is driven towards steady state by the excess entropy which is initially contained in the regenerator. The entropy can only leave the system through bulk motion of the helium. The rate at which entropy is convected is proportional to the axial temperature gradient and corresponds to the rate at which the system approaches steady state. The degree of departure from steady state (ΔS_{xs}) is also proportional to the axial temperature gradient, resulting in a first order system.

$$\frac{d\Delta S_{xs}}{dt} \approx \Delta S_{xs} \approx \frac{\partial T}{\partial x} \tag{3.34}$$

Because of limited supercomputer time, the cycle was not allowed to reach a 'true' steady state. Instead, the results of the second through the fifth cycles were curve fit with a decaying exponential. The extrapolated steady state was then used to calculate the refrigeration characteristics for the corresponding set of input parameters. Figure 3-9 illustrates that the extrapolated steady state matches the 'true' steady state.

3.6.2 Cross Sectional Energy and Entropy Fluxes

Because axial conduction has been neglected, energy or entropy can only be transported through a regenerator cross-section by the bulk motion of the flowing helium. The net flux of energy or entropy through a given cross-section is the sum of the differential helium masses which flow through that cross section multiplied by their associated enthalpy or entropy.

$$hflux^{i,cyc} = \sum_{j=1}^{n} mf_{he}^{\,i,j} \cdot h_{he}(T_{he}^{\,i,j}, P^{i,j}) \cdot dt \qquad (3.35)$$

$$sflux^{i,cyc} = \sum_{j=1}^{n} mf_{he}^{\,i,j} \cdot s_{he}(T_{he}^{\,i,j}, P^{i,j}) \cdot dt \qquad (3.36)$$

In Equations 3.35 and 3.36, the mass flow rate of helium is taken to be positive if helium is flowing from the warm to the cold end. The mass flow is equal to the rate of mass storage in the cold space and in the regenerator void volume between the axial position and the cold heat exchanger (i.e. in the working space 'below' the axial position). Therefore, Equations 3.35 and 3.36 approximate the integration of helium enthalpy or

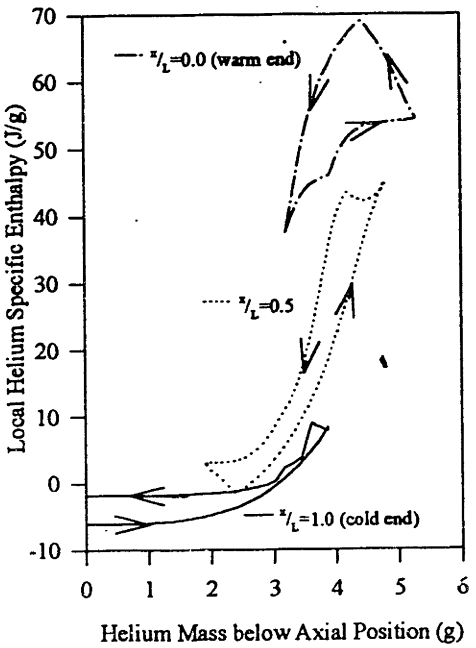

Figure 3-10. Helium Enthalpy as a Function of Mass Contained below a Particular Axial Position

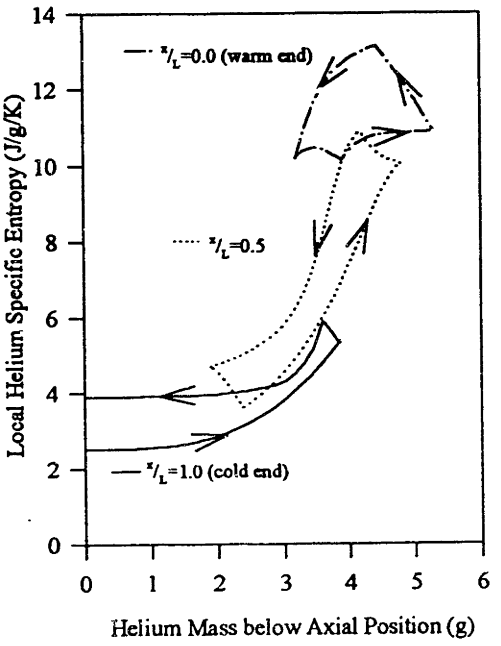

Figure 3-11. Helium Entropy as a Function of Mass Contained below a Particular Axial Position

entropy as a function of helium mass contained in the system below the appropriate axial position. These plots are shown in Figures 3-10 and 3-11 for several axial nodes during a steady state cycle.

The negative area of these figures indicates that the cyclic flux is directed towards the warm end (i.e. the system is producing refrigeration). The energy flux and entropy flux as function of axial position are illustrated in Figure 3-12.

The entropy flux as a function of axial position has a negative slope, indicating that the model is consistent from a second law standpoint. Furthermore, the derivative of the energy flux corresponds to the magnetic work input which is calculated in Subsection 3.6.3 and illustrated in Figure 3-14. This indicates that the model is consistent from a first law standpoint. These are two independent checks on the validity of the simulation.

3.6.3 Magnetic Work

The magnetic work input during a steady state cycle can be determined by using the assumption of internally reversible GGG heat absorption. The heat absorbed during a reversible thermodynamic process is the integration of temperature multiplied by entropy change. This quantity varies along the axis of the regenerator. Figure 3-13 illustrates the temperature-entropy cycles undergone by the GGG at several axial locations. The path direction indicates that the net heat absorbed by the GGG is negative. The GGG at each location is acting as a differential heat pump, removing heat from a low temperature inlet helium stream and rejecting it to a higher temperature outlet stream.

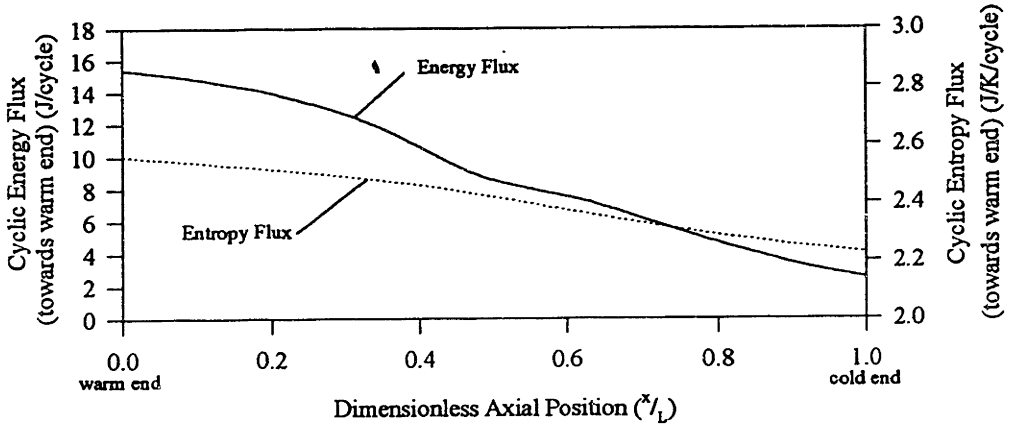

Figure 3-12. Cyclic Energy and Entropy Fluxes within Regenerator

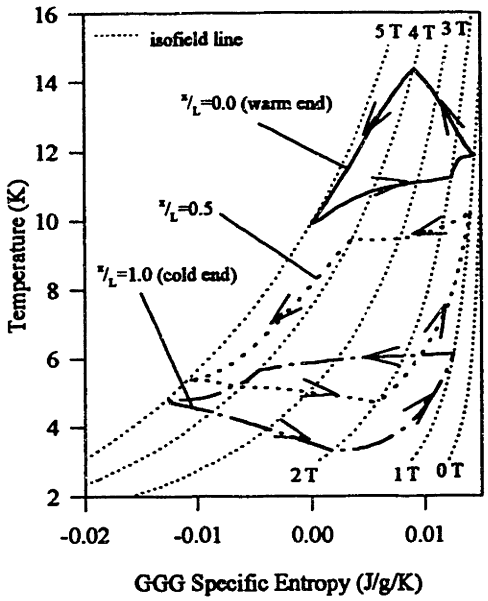

Figure 3-13. Temperature Entropy Diagram for GGG Contained at Particular Axial Positions

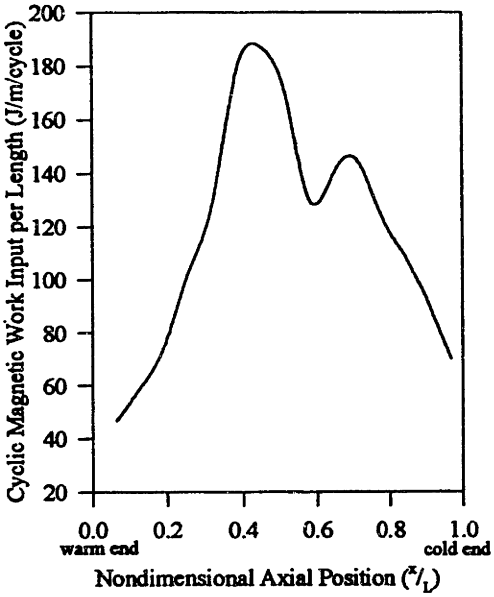

Figure 3-14. Magnetic Work Input per Length as a Function of Axial Position

Since the cycle is in steady state, the magnetic work input must be equal to the heat absorbed and of opposite sign (i.e. positive).

$$W_{mag} = \sum_{j=1}^{n} A \cdot (1-\varepsilon) \cdot \rho_{GGG} \cdot {T_{GGG}}^{i,j} \cdot dx \cdot \left[s_{GGG}({T_{GGG}}^{i,j}, \mu_o H^{i,j}) - s_{GGG}({T_{GGG}}^{i,j-1}, \mu_o H^{i,j-1}) \right] \quad \textbf{(3.37)}$$

The magnetic work as a function of axial position is plotted in Figure 3-14.

3.6.4 Magnetic and Mechanical Refrigeration Components

In Chapter 2, the two refrigeration mechanisms available in a magnetically active regenerative refrigerator were described (i.e. mechanical and magnetic). The ideal 'combined' cycle described in Section 2.4 uses both mechanisms to produce more refrigeration than could otherwise be obtained.

In general, the nonideal cycles produce some refrigeration by both 'subcooling' and expanding the helium (i.e. due to both the magnetic and mechanical effects). In order to examine their relative contribution, the total refrigeration is artificially split into a magnetic and a mechanical component.

The magnetic component of refrigeration is produced by the 'subcooling' effect which results from the thermodynamic cycling in the GGG. It is formally defined as the energy required to isobarically heat the helium entering the cold heat exchanger to the refrigeration temperature. Because the heat exchangers are assumed to be perfect, this process occurs instantaneously.

$$Q_{mag} = \sum_{j=1}^{n} mf_{he}^{m,j} \cdot \left[h_{he}(T_{cold}, P^{m,j}) - h_{he}(T_{he}^{m,j}, P^{m,j}) \right] \cdot dt \qquad \textbf{(3.38)}$$

The mechanical component of refrigeration is produced by the helium expansion in the refrigeration space. If perfect thermal contact exists between the helium and the cold reservoir then the expansion process must be isothermal. The mechanical component of refrigeration is formally defined as the energy required to isothermally expand the helium entering the constant temperature section of the refrigeration space.

$$Q_{mech} = \sum \left[P^{m,j} \cdot (V_{dis}^{m,j} - V_{dis}^{m,j-1}) - mf_{he}^{m,j} \cdot h_{he}(T_{cold}, P^{m,j}) \cdot dt \right] \qquad \textbf{(3.39)}$$

The sum of the magnetic and mechanical refrigeration components is equal to the total refrigeration. The magnetic and mechanical refrigeration components are illustrated in Figure 3-15.

3.6.5 Refrigeration Characteristics

The total refrigeration is the sum of the heat transfer distributed over the isothermal mechanical space and the concentrated heat transfer at the junction between the isothermal space and the magnetic regenerator space, as illustrated in Figure 3-15. This is equivalent to adding Equations 3.38 and 3.39 (i.e. the magnetic and mechanical refrigeration components). The refrigeration load is simply the refrigeration divided by the cycle time. The specific refrigeration is defined as the refrigeration per gram of magnetic material.

A reversible pump attached to the warm end of the refrigeration device will require the minimum work in order to cyclically compress the helium. The magnitude of this work transfer can be calculated with an availability balance.

$$W_{mech} = \sum_{j=1}^{n} mf_{he}^{1,j} \cdot \left[h_{he}(T_{warm}, P^{1,n}) - T_{warm} \cdot s_{he}(T_{warm}, P^{1,n}) \right] \cdot dt \qquad \textbf{(3.40)}$$

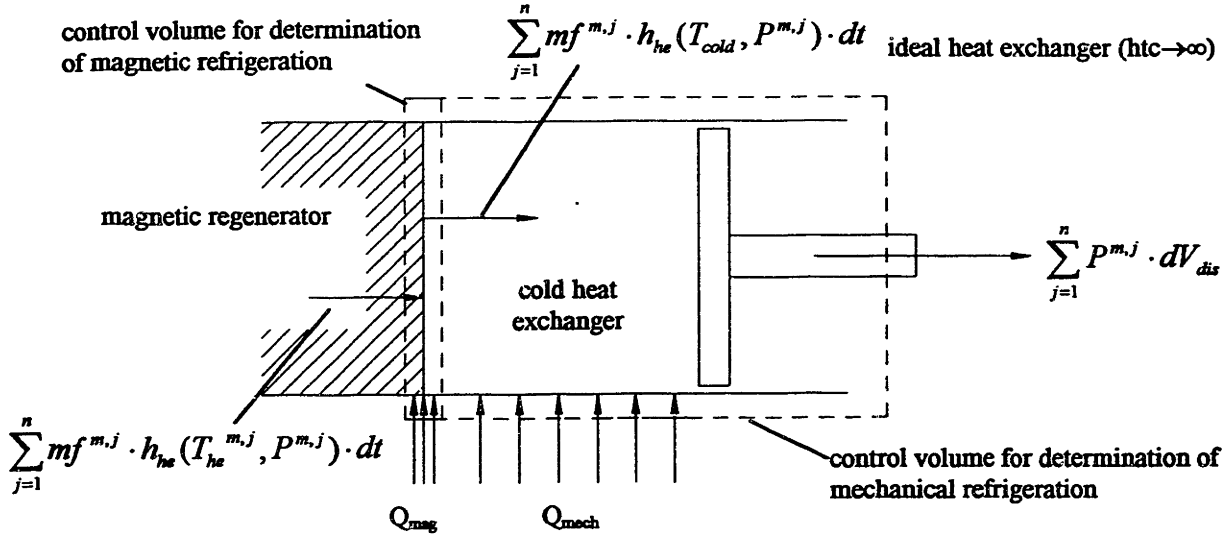

Figure 3-15. Control Volumes for Magnetic and Mechanical Refrigeration Calculation

The coefficient of performance for the refrigeration cycle is the refrigeration divided by the magnetic and mechanical work input.

Other parameters of interest include the irreversibility generated in the cold heat exchanger, the regenerator, and the warm heat exchanger. Irreversibility is produced in the 'perfect' heat exchangers due to the mixing which occurs between the 'subcooled' or 'superheated' helium and the isothermal helium contained within the heat exchanger. These quantities may be calculated from an entropy balance around the appropriate component.

$$Sirr_{chx} = -\sum_{j=1}^{n} mf_{he}^{m,j} \cdot s_{he}(T_{he}^{m,j}, P^{m,j}) \cdot dt - \frac{(Q_{mag} + Q_{mech})}{T_{cold}} \quad (3.41)$$

$$Sirr_{reg} = \sum_{j=1}^{n} \left[mf_{he}^{m,j} \cdot s_{he}(T_{he}^{m,j}, P^{m,j}) - mf_{he}^{1,j} \cdot s_{he}(T_{he}^{1,j}, P^{1,j}) \right] \cdot dt \quad (3.42)$$

$$Sirr_{whx} = W_{mech} - \sum mf_{he}^{1,j} \cdot \left[h_{he}(T_{he}^{1,j}, P^{1,j}) - T_{he}^{1,j} \cdot s_{he}(T_{he}^{1,j}, P^{1,j}) \right] \cdot dt \quad (3.43)$$

IV. CYCLE OPTIMIZATION

In this chapter, the model developed in Chapter 3 is used to determine the operating parameters which generate the most effective refrigeration cycle. The results of the model are also used to gain insight into the physical effects associated with varying the operating parameters.

This chapter is divided into two sections. In the first section, the operating parameters are characterized in nondimensional form. The second section presents the results of the cycle optimization analysis.

4.1 Operating Parameter Characterization

The operating parameters are the input parameters which may be varied in an existing refrigeration device. These parameters include the imposed pressure variation, the applied field variation, and the displacer volume variation. These imposed time variations may be generated in an infinite number of ways, all of which correspond to a unique refrigeration cycle.

The goal of this chapter is to generate a refrigeration surface which explores the effect of varying these parameters. In order to reduce the complexity of the problem, the operating profiles must be simplified to the point where a single nondimensional parameter may be used to characterize the entire time variation.

The shape of the simplified operating profiles are shown in Figure 4-1. All of the

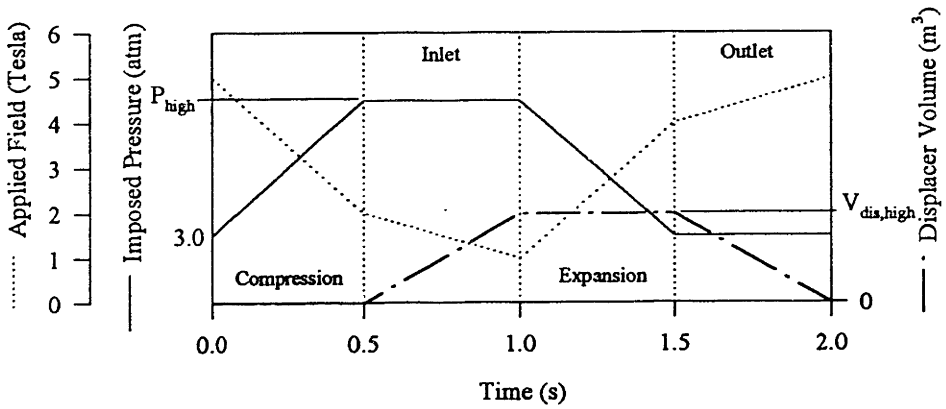

Figure 4-1. Operating Parameter Profile Simplification

refrigeration cycles have a frequency of 0.5 Hz (i.e. one cycle every two seconds). The refrigeration cycle is divided evenly into the four required processes.

The pressure variation begins at three atmospheres and varies linearly during the compression and expansion processes. The pressure is held constant during the flow processes. With these simplifications, the pressure profile may be characterized by a single parameter. The pressure ratio (γ) is defined as the ratio of the high imposed pressure to the low pressure.

The displacer volume profile begins at zero and varies linearly during the flow processes. The displacer profile could be characterized by the maximum displacer volume, but the mass of helium shuttled through the regenerator is a more important parameter. It is therefore more appropriate to characterize the displacer volume profile with a mass ratio. The mass ratio is defined as the ratio of the maximum mass of helium contained in the displacer during the cycle to the mass of GGG contained in the regenerator.

$$mr = \frac{V_{dis,high} \cdot \rho_{he}(T_{cold}, P^{m,max})}{A \cdot (1-\varepsilon) \cdot L \cdot \rho_{GGG}} \tag{4.1}$$

The applied field profile used to generate the results presented in Subsection 4.2.1 is not changed. The applied field variation was initially optimized at reasonable levels of pressure ratio and mass ratio. This optimization resulted in the applied field profile illustrated in Figure 4-1.

Holding the applied field profile constant while varying the pressure ratio and mass ratio implies that the effect of applied field variation is decoupled from the other effects. In Subsection 5.4.2, the applied field variation is reoptimized during the design of an experimental prototype. The resulting profile is only slightly different from the one shown in Figure 4-1, indicating that the effect of applied field variations are relatively insensitive to mass ratio and pressure ratio.

4.2 Simulation Results

The program REGEN.c is used to examine the effect of changing the operating parameters. The code was run on the MIT Cray X-MP computing platform. The design parameters

were held constant during these simulations. The design parameters used throughout this section are listed in Table A-2 in Appendix A.7.

This section is divided into three subsections. The first subsection varies the pressure ratio and mass ratio in order to generate a refrigeration surface in these dimensions. The second subsection examines the effect of varying the pressure ratio. In the final subsection, the effect of varying the mass ratio is investigated

4.2.1 Refrigeration Surface

The pressure ratio was varied from 1.25 to 4.93 and the mass ratio was varied from 4.5e-3 to 10.0e-3 in order to generate the refrigeration curves shown in Figure 4-2. The cycle performance is a strong function of mass ratio and a weaker function of pressure ratio. The specific refrigeration gradually increases with pressure ratio up to a critical pressure ratio (i.e. 4.93). Further increase in the pressure ratio results in decreasing specific refrigeration. The optimal cycle parameters, in terms of overall specific refrigeration, correspond to a pressure ratio of 4.9 and a mass ratio of 8e-3. Only slightly less refrigeration is obtained at the more conventional pressure ratio of 2.5 and an adjusted mass ratio of 7.5e-3.

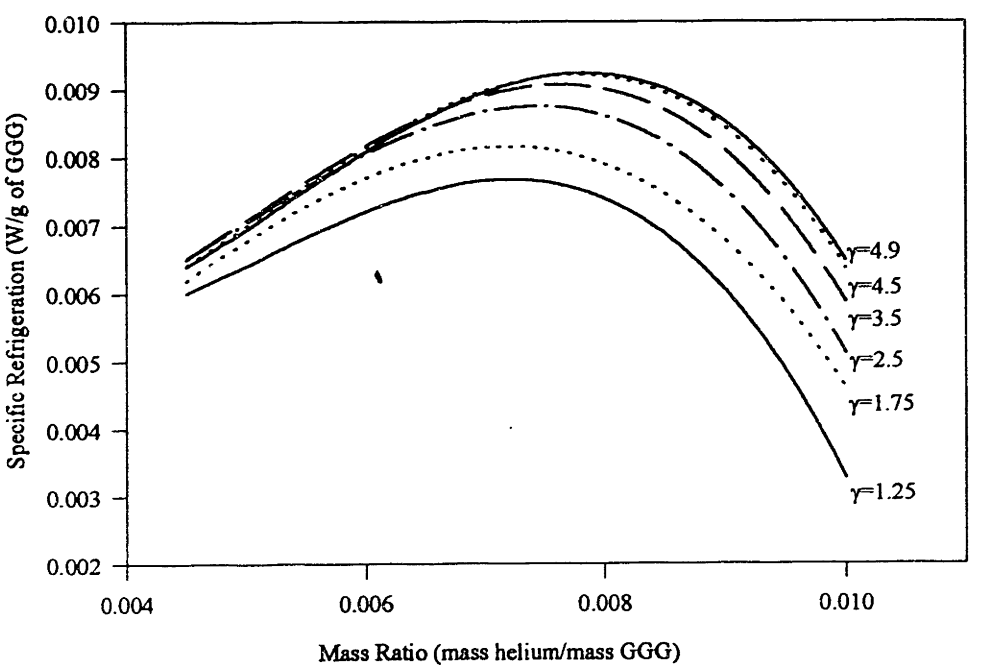

Figure 4-2. Specific Refrigeration Surface

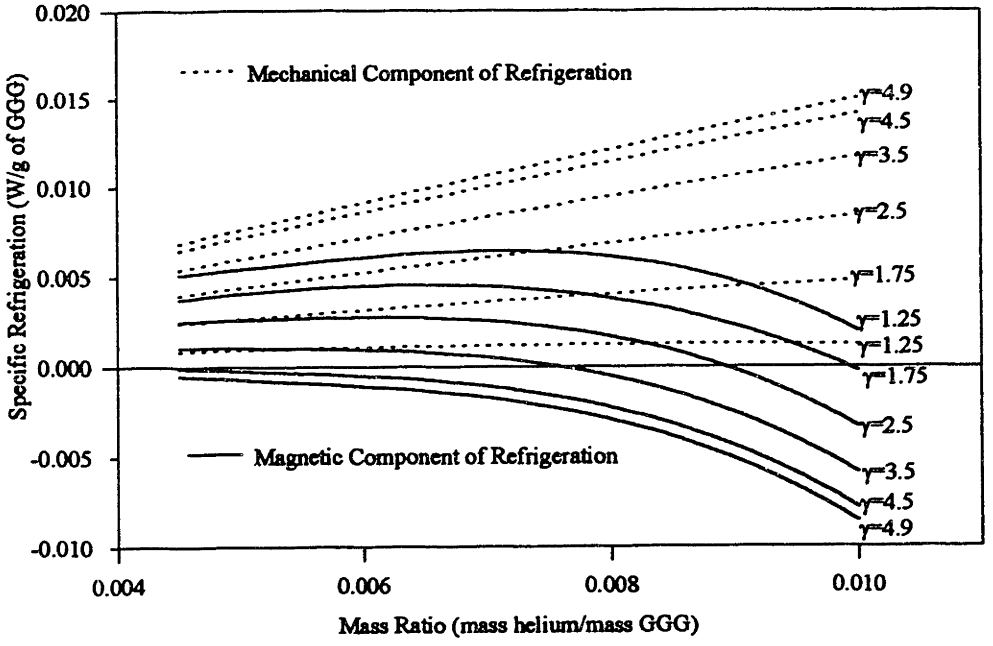

Figure 4-3. Mechanical and Magnetic Components of Refrigeration

The total refrigeration is divided into its magnetic and mechanical components and plotted in Figure 4-3. The specific mechanical refrigeration increases with pressure ratio and mass ratio since both of these parameters tend to increase the work transfer on the displacer. The specific magnetic refrigeration decreases at high mass ratios because the heat capacity of the flowing helium overwhelms the magneto-caloric effect within the GGG, reducing the 'subcooling' effect. The specific magnetic refrigeration decreases with pressure ratio since the compression and expansion of the entrained helium degrades the nonflow portions of the cycle.

The coefficient of performance is illustrated in Figure 4-4. The efficiency of the cycle decreases with both pressure ratio and mass ratio. Figure 4-4 shows that a more efficient cycle is generated at the chosen operating parameters (i.e. γ=2.5 and mr=7.5e-3) than at the optimal operating parameters (i.e. γ=4.9 and mr=8.0e-3).

The dimensionless irreversibility generation within the heat exchangers is plotted in Figure 4-5 and the dimensionless irreversibility generation within the regenerator is shown in Figure 4-6. These plots can be interpreted as the refrigeration lost due to entropy generation (e.g. a dimensionless entropy generation of 0.25 indicates that 25% more refrigeration would be produced by eliminating this source of irreversibility).

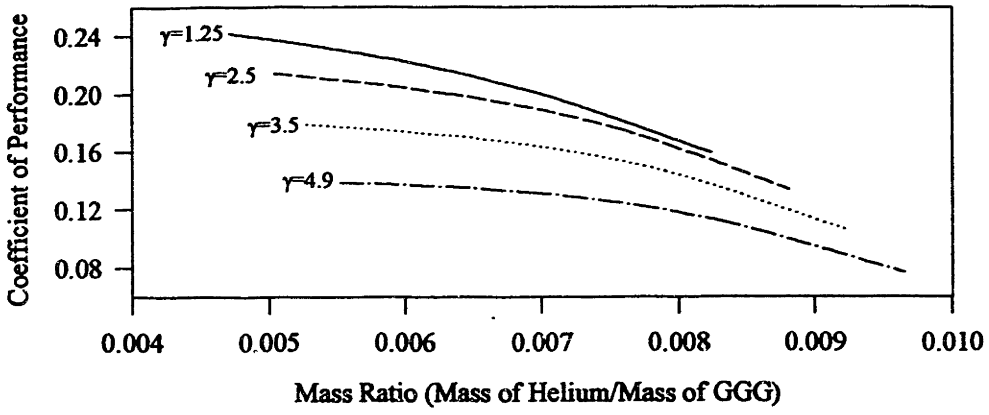

Figure 4-4. Coefficient of Performance

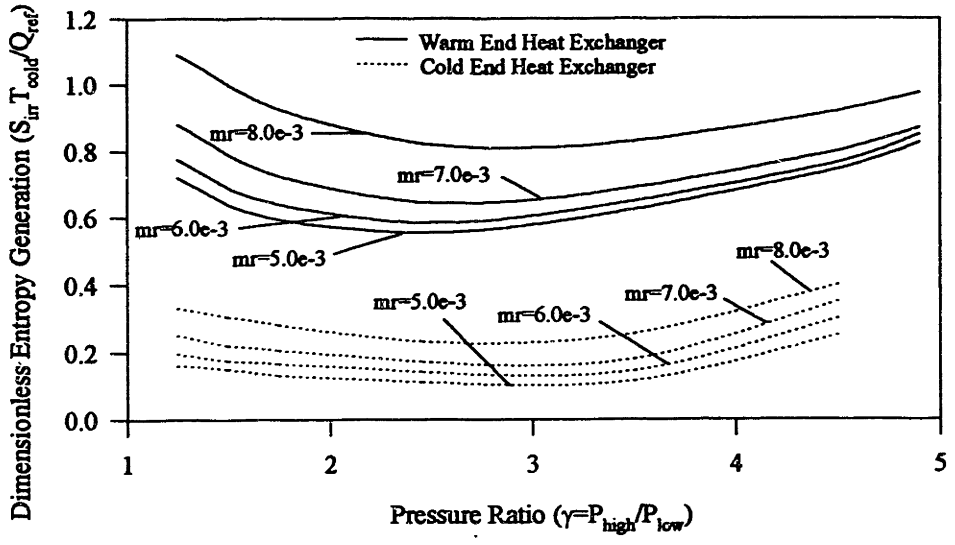

Figure 4-5. Dimensionless Entropy Generation due to 'Mixing Effect' in Heat Exchangers

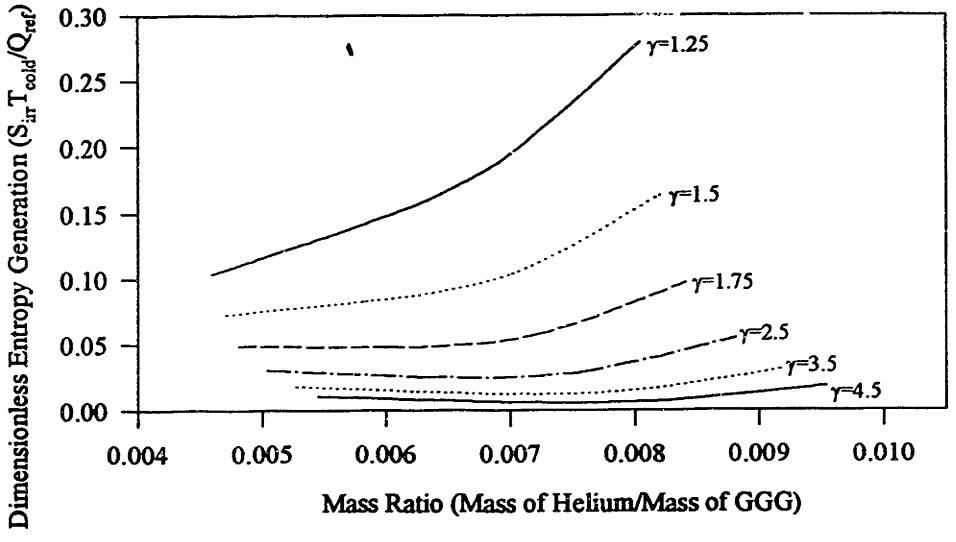

Figure 4-6. Dimensionless Entropy Generation within the Regenerator

4.2.2 Effect of Pressure Ratio

The amount of mechanical refrigeration produced by expanding the helium within the cold space is proportional to the natural log of the pressure ratio. This can be seen by examining the mechanical component of refrigeration shown in Figure 4-3. Unfortunately, the increase in mechanical refrigeration is offset by a decrease in magnetic refrigeration. This decrease is caused by compression and expansion of the entrained helium and by an increasing specific heat mismatch within the flowing helium.

The dominant effect is associated with the entrained helium. The specific heat mismatch is a secondary effect which was examined in Subsection 2.6.4.

The qualitative effect of pressure variations within the entrained helium was examined in Subsection 2.6.3. The nonideal simulation can be used to examine this phenomenon more quantitatively.

During the nonflow processes, the pressure and applied field are varied. The ideal 'combined' cycle assumes that these processes are uncoupled which allows the regenerator to undergo an adiabatic applied field swing (as pictured in Figure 2-5). The simultaneous compression of the entrained helium within the regenerator causes the demagnetization and magnetization processes to deviate from adiabatic. Figure 4-7 indicates the thermodynamic cycle undergone by the GGG contained at the midpoint of the regenerator for several pressure ratios and a constant mass ratio (i.e. mr=7.5e-3).

Even at the lowest pressure ratio (i.e. 1.25), the nonflow processes are more nearly isothermal than isentropic. As the pressure ratio is increased, the heat absorbed by the GGG during the compression process (and rejected during the expansion process) increases. This effect drives the end point of the compression process towards greater specific entropy and therefore higher temperature. Similarly, the end point of the expansion process is driven towards smaller specific entropy and lower temperatures. The area enclosed by the thermodynamic cycle is decreased, indicating that the effectiveness of the GGG as a refrigerant has been reduced.

The loss of magnetic refrigeration is reflected in the decrease in magnetic refrigeration with increasing pressure ratio shown in Figure 4-3. At some point, the magnetic component of refrigeration becomes negative. This indicates that the helium is entering

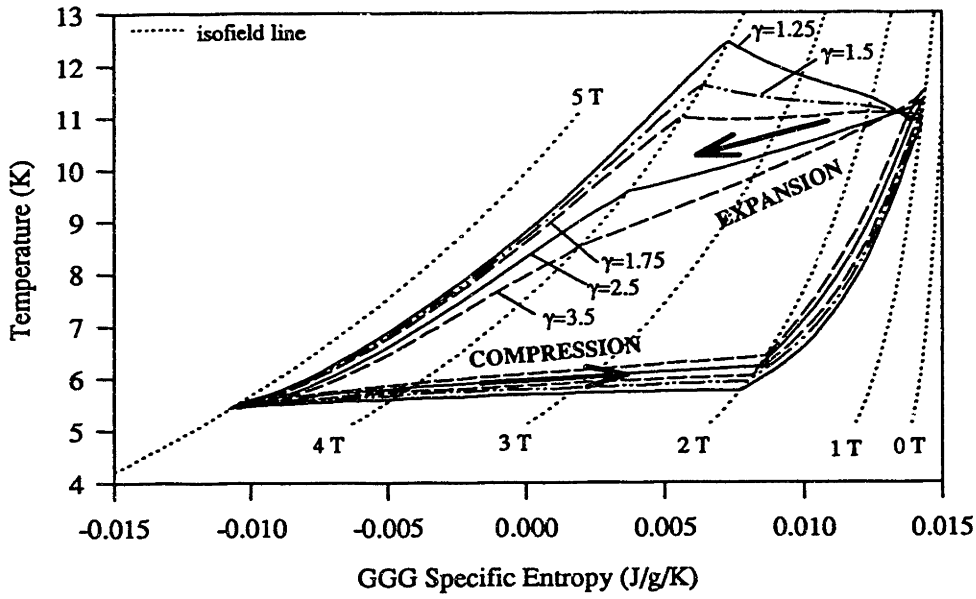

Figure 4-7. Thermodynamic Cycles Undergone by the Magnetic Material at the Regenerator Midpoint for Varying Pressure Ratios

the refrigeration space 'superheated' and rejecting heat to the cold reservoir. This 'superheating' effect eventually overwhelms the increase in mechanical refrigeration associated with larger pressure variations, resulting in an optimum pressure ratio.

The concepts described above explain the shape of the dimensionless entropy generation curves within the heat exchangers (shown in Figure 4-5). This irreversibility is generated when 'subcooled' or 'superheated' helium mixes with the isothermal fluid contained in the heat exchangers. This effect increases with mass ratio because more helium is being shuttled through the regenerator. However, as the pressure ratio is increased, the helium entering the cold heat exchanger becomes less 'subcooled' and the helium entering the warm heat exchanger becomes less 'superheated'. The irreversibility generated by these effects is therefore initially reduced with increasing pressure ratio. At some point, the temperature differential at the regenerator inlet and outlet switches sign and begins to grow again. This results in the minimum entropy generation as a function of pressure ratio.

4.2.3 Effect of Mass Ratio

As the mass ratio is increased, the amount of refrigeration generated by expansion of the helium in the cold end (i.e. the mechanical component of refrigeration) is increased. However, the increased mass ratio reduces the magnetic refrigeration. The nature of this

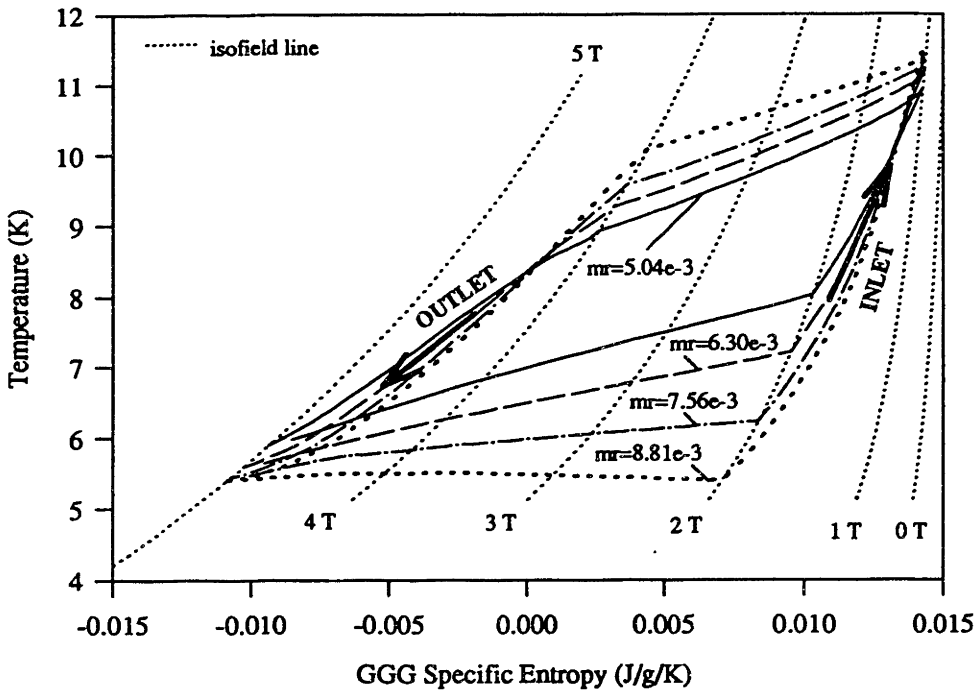

Figure 4-8. Thermodynamic Cycles Undergone by the Magnetic Material at the Regenerator Midpoint for Varying Mass Ratios

degradation is more complex than the pressure ratio effect studied in Subsection 4.2.2.

As the mass ratio is increased, the GGG actually becomes a more effective refrigerant. Figure 4-8 shows the thermodynamic cycle undergone by the GGG at the midpoint of the regenerator for several different mass ratios and a constant pressure ratio (i.e. γ=2.5). The effect of the mass ratio shows up primarily during the flow processes (i.e. the inlet and outlet processes). During the inlet process, the increasing mass ratio causes the temperature of the GGG to rise more drastically. Similarly, the temperature of the GGG at the endpoint of the outlet process is reduced as more cold helium is shuttled through the regenerator. The total area enclosed by the temperature entropy traces increases with mass ratio, indicating that the GGG becomes a more effective refrigerator as the mass ratio is raised. The increase in the amount of refrigeration produced by the GGG is eventually overwhelmed by the increase in the amount of helium which must be refrigerated, causing the magnetic refrigeration to ultimately decrease.

V. EXPERIMENTAL APPARATUS DESIGN

The analysis detailed in Chapters 2 through 4 indicate that the 'combined' refrigeration cycle has the potential to effectively and reliably produce refrigeration at very low temperatures. In this chapter, the preliminary design of an experimental apparatus will be presented. This apparatus will be used to prove that the analytical results presented in Chapter 4 are accurate. The nonideal computational model developed in Chapter 3 is used as a design tool.

This chapter is divided into five sections. In the first section, the proposed experimental system is qualitatively described. The second section describes the system which will be used to generate the magnetic field. The third section presents the experimental work which has been done regarding packed bed regenerator construction techniques and presents the proposed magnetic regenerator matrix design. The fourth section reoptimizes the operating conditions with the design constraints arrived at in Sections 5.2 and 5.3. These results are then used during the design of the heat exchangers and the precoolant regenerator. The final section considers the displacer system.

5.1 System Description

The experimental apparatus will be a magnetically active regenerative system which supplies refrigeration at 4.2° K and rejects heat at 12° K. A schematic of the overall system is illustrated in Figure 5-1.

The experimental system may be subdivided into upper stage components and magnetically active regenerative refrigerator components. The upper stage components are used to generate the warm reservoir temperature and to impose the pressure variations on the helium working fluid. The upper stage components include the liquid helium bath, the precoolant regenerator, and the room temperature compressor. In a working device, the upper stage components will be replaced by a conventional Gifford McMahon Cryocooler.

The magnetically active regenerative refrigerator components are used to span the 12° K to 4.2° K temperature range. These components include the superconducting solenoid, the displacer system, the magnetic regenerator, and the warm and cold heat exchangers.

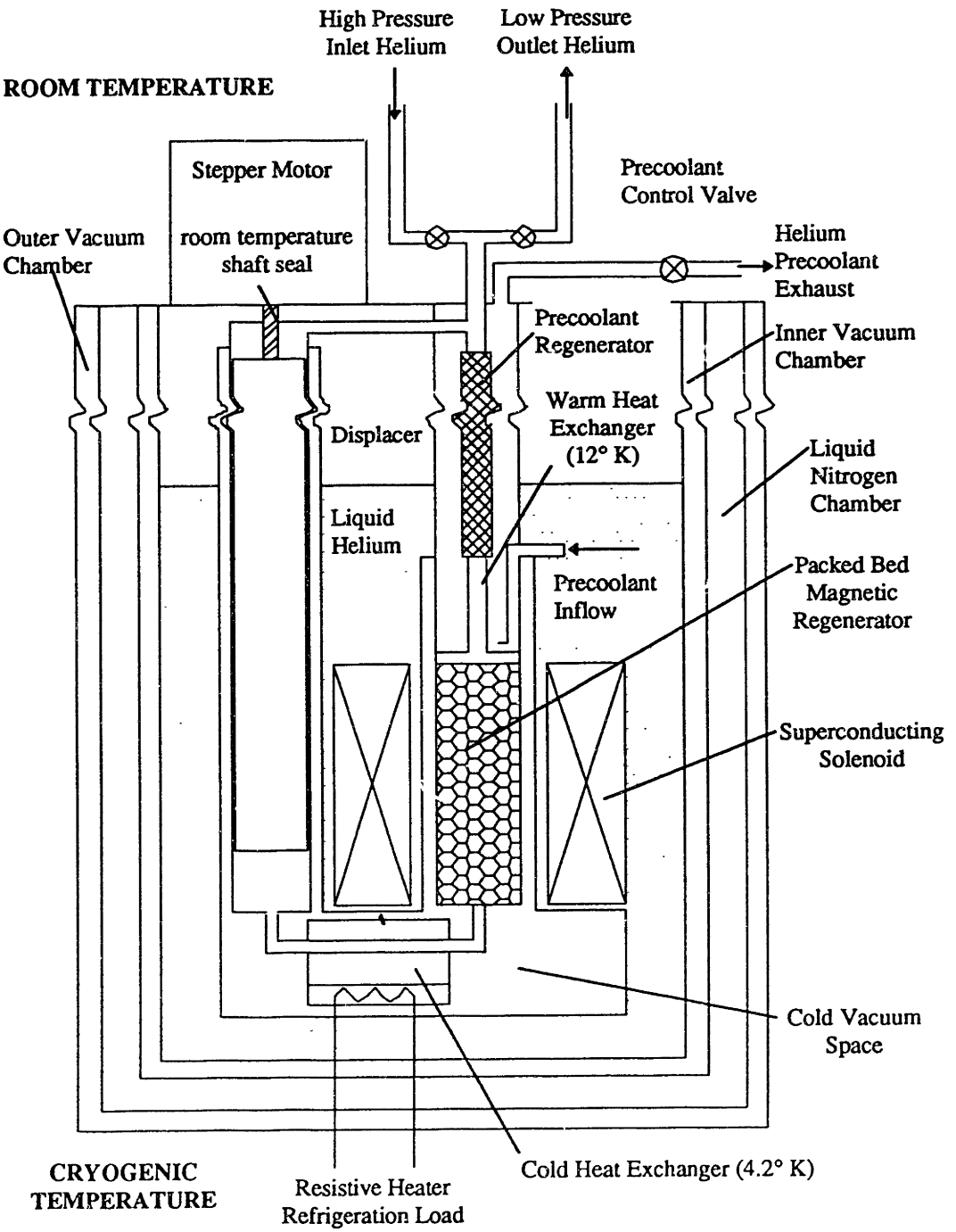

Figure 5-1. Schematic of Proposed Experimental Apparatus

This section is divided into two subsections. In the first subsection, the magnetically active regenerative refrigerator system is described. The second subsection explains the upper stage system.

5.1.1 Magnetically Active Regenerative Refrigerator System

The entire cryogenic assembly will be immersed in a slightly pressurized liquid helium bath. The liquid helium will be in direct contact with the superconducting solenoid thus supplying the necessary cooling. The magnetically active regenerative refrigerator will be thermally isolated from the liquid helium bath by placing it in a vacuum space (i.e. the cold vacuum space). The vacuum insulation ensures that the refrigeration absorbed in the cold heat exchanger is the only heat flow into the device. In this way, the performance of the device can be monitored.

The displacer system will be powered with a motor at room temperature. The displacer will span the length of the system. Helium gas from the inlet stream will be vented to the back of the displacer in order to reduce the force on the motor. This eliminates the need for cold seals.

The cold heat exchanger will be maintained at 4.2° K by condensing helium. The refrigeration load will be simulated with a resistive heater placed in a liquid helium chamber. The helium vapor condenses on the flow passages containing the working fluid. This type of system eliminates the need for temperature control and has excellent heat transfer characteristics due to the condensation process. A pressure tranducer in the boiling helium chamber can be used to regulate the refrigeration load.

The superconducting solenoid is an existing piece of hardware which was salvaged from an earlier magnetic refrigerator. The magnetic field generated by this device and the corresponding power requirements have been thoroughly examined. An appropriate power supply must be purchased or fabricated.

The packed bed magnetic regenerator will be sized so that the entire length of the solenoid which is exposed to a uniform applied field is filled with magnetic material. The regenerator will be constructed by crushing GGG particles into a metal cylinder. The desired porosity can be obtained by controlling the packing pressure.

In the warm heat exchanger, the 'superheated' helium exiting the magnetic regenerator transfers heat to the precoolant stream. The precoolant stream will be obtained directly from the liquid helium bath. The boiling process which occurs when the precoolant stream comes into contact with the warm flow passages is characterized by excellent heat transfer. The temperature of the working fluid at the interface between the conventional regenerator and the warm heat exchanger will be maintained at 12° K by controlling the precoolant flow rate with a room temperature valve.

5.1.2 Upper Stage System

The pressure variation within the working fluid will be imposed at room temperature by an existing compressor system. A continuous flow of makeup helium will be required by the magnetic refrigeration cycle as well as the pressure driven density gradients within the working volume of the entire system.

The makeup helium must be cooled to the warm reservoir temperature. This will be accomplished with a conventional regenerator system (i.e. the precoolant regenerator). The regenerator will store the heat of the high pressure, warm inflow and return it to the low pressure, cool outflow. The regenerative process will be augmented by allowing the precoolant to flow in thermal contact with the regenerator matrix.

The entire system will be placed within an existing cryogenic dewar designed to reduce liquid helium boiloff. This dewar is composed of a liquid nitrogen filled chamber sandwiched between two vacuum chambers.

5.2 Magnetic Field Generation

This section is divided into two subsections. The first subsection describes the existing superconducting solenoid which will be used within the experimental apparatus. The second subsection examines the associated power supply requirements.

5.2.1 Superconducting Solenoid

The superconducting solenoid which will be used in the experimental apparatus was constructed by Woodrow Chin[46] at M.I.T. for use within the first M.I.T. magnetic

refrigerator[47]. It was later modified during construction of the second M.I.T. magnetic refrigerator[48].

The superconducting solenoid is shown in Figure 5-2. The solenoid is composed of two circuits and each circuit is composed of one main and two compensating coils. The primary circuit generates a uniform magnetic field within the solenoid bore. The primary main coil is uniformly wrapped around a 1.5 mm Micarta coil form. The two primary compensating coils offset the end effects within the solenoid bore and create an applied field which is more nearly axially uniform. The secondary circuit may be used to axially vary the applied field. Only the primary circuit will be used in the experimental apparatus.

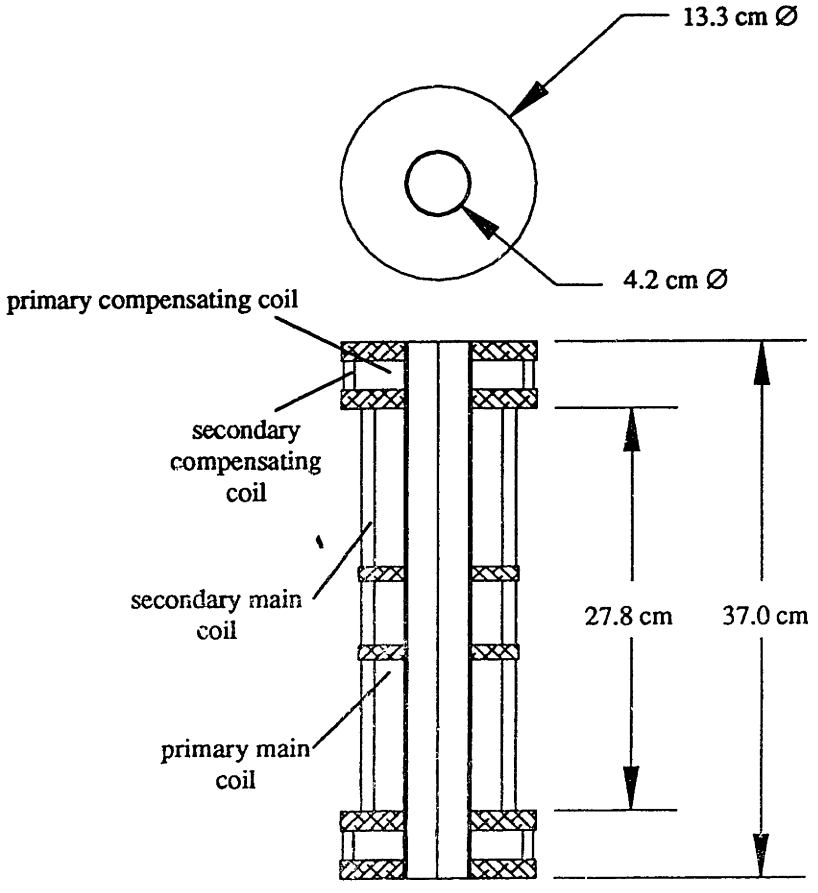

Figure 5-2. Superconducting Solenoid

The structure of the solenoid is created with G10 disks and rods. These disks have been constructed with numerous flow openings which allow the liquid helium coolant to penetrate into the internal windings.

The axial applied field generated by the primary circuit has been measured[49]. The results are shown in Figure 5-3. The applied field is approximately uniform over a length of twenty eight centimeters. This is the design length of the magnetic regenerator matrix.

5.2.2 Power Supply

The field constant (λ) and inductance (L) associated with the primary circuit of the superconducting solenoid have been determined to be 0.039 Tesla/Amp and 1.2 Henry, respectively.

The field constant of the superconducting solenoid specifies the current required in order to generate a given applied field. The applied field within the experimental apparatus will be varied between 1 Tesla and 4 Tesla. Therefore, the current within the superconducting solenoid must be cyclically changed from 26 Amp and 103 Amp. At cryogenic temperatures, the resistance of the superconducting solenoid will be negligible and the winding can be modelled as a pure inductance. The supply voltage can be determined from the constitutive relation for a linear inductance.

$$E_s = L \cdot \frac{di}{dt} = \frac{L \cdot (\mu_o H_{high} - \mu_o H_{low}) \cdot 2}{\lambda \cdot \tau_{cyc}} \tag{5.1}$$

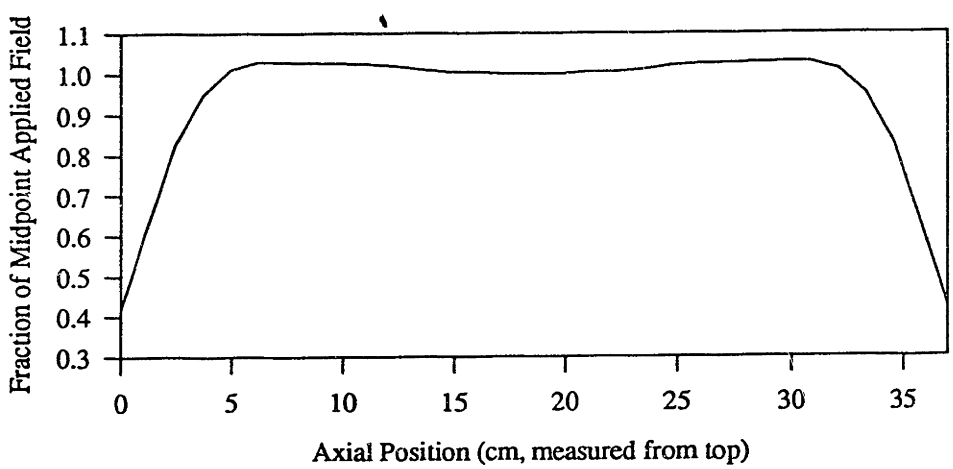

Figure 5-3. Axial Variation in Applied Field

Design Parameter	Specified Value
Length of Magnetic Regenerator	28.0 cm
Bore of Magnetic Regenerator	3.01 cm
High Applied Field	4 Tesla
Low Applied Field	1 Tesla
Total Cycle Time	6 seconds
Power Supply Voltage Requirement	15 volt (bipolar)
Power Supply Current Requirement	103 amp

Table 5-1. Design Parameters Specified by Superconducting Solenoid

Equation 5.1 indicates that the supply voltage is inversely proportional to the cycle time (τ_{cyc}). Because the superconducting solenoid quenches at approximately 16 V, the applied voltage is specified at 15 V. The corresponding cycle time is 12 seconds. The design parameters specified by the superconducting solenoid system are summarized in Table 5-1.

5.3 Magnetic Regenerator

The external geometry of the magnetic regenerator matrix is completely specified by the bore of the superconducting magnet and the axial length over which the applied field is uniform.

The internal geometry of the packed bed is obtained by balancing the refrigeration loss associated with pressure gradients against those associated with entrained helium. The porosity of the regenerator matrix constitutes a major loss mechanism within any type of regenerative refrigeration cycle. The magnitude of this loss can be demonstrated by examining Figure 4-3 which shows the magnetic and mechanical components of refrigeration. As the pressure ratio is increased, the magnetic component of refrigeration is drastically reduced. This reduction is almost entirely due to the compression and expansion of the helium entrained within the regenerator matrix. If the porosity of the regenerator could be reduced to zero, the magnetic component of refrigeration would be completely unaffected by increasing the pressure ratio as was the case in the ideal cycles examined in Chapter 2.

At very low porosities, the pressure drop through the regenerator matrix will become a major loss mechanism. However, the cycle time specified in Section 5.2 is twelve seconds due to constraints imposed by the superconducting solenoid. This large cycle time

will reduce the mass flow rate through the regenerator to the point where pressure gradients are not a dominant loss mechanism.

The regenerator matrix will be a packed bed composed of GGG particles. The GGG material will be crushed to the correct particle diameter and then packed into place. This type of geometry was chosen because it has good heat transfer characteristics and low construction costs.

This section is divided into two subsections. The first subsection presents experimental results which express the regenerator porosity as a function of packing pressure. The second subsection specifies the geometry of the packed bed regenerator.

5.3.1 Packed Bed Experiments

In order to reduce the porosity of the packed bed regenerator matrix, each layer of precrushed particles will be packed into the regenerator cylinder. Several experiments were conducted with glass particles in order to determine the appropriate packing pressure. The results of these tests are illustrated in Figure 5-4. The individual glass particles were compared to GGG particles of the same size under magnification and found to be morphologically similar.

Figure 5-4 indicates that the porosity of the regenerator can be controlled by the packing pressure. The internal character of the packed beds was examined by determining the associated friction factors. The Ergun correlation for pressure gradients within a packed

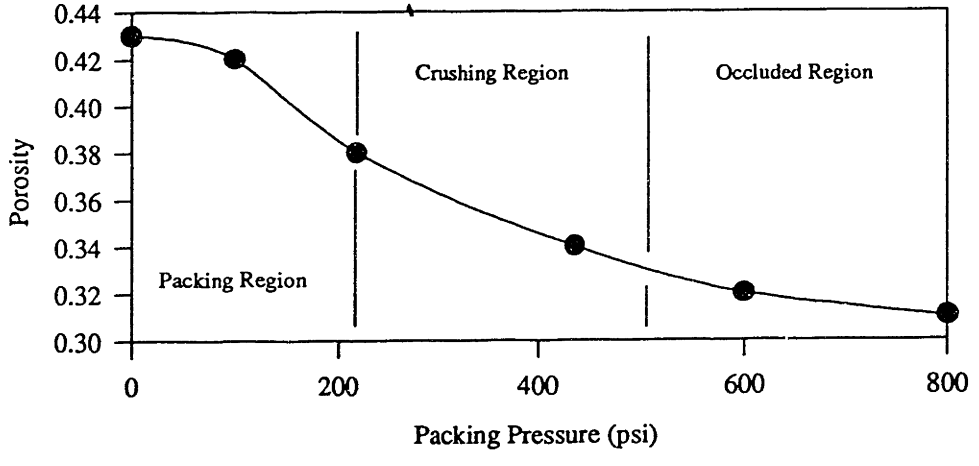

Figure 5-4. Regenerator Porosity at Varying Packing Pressure

bed composed of spherical particles was used within the nonideal regenerator model (Equation 3.4). This relation can be generalized to a packed bed composed of any type of particle by adding a correction factor[50]. The shape factor (ϕ) characterizes the 'irregularity' of the packed bed particles. The shape factor is defined as the ratio of the specific area of a spherical bed to the specific area of the packed bed in question.

$$\frac{dP}{dx} = \frac{150 \cdot (1-\varepsilon)^2 \cdot \mu \cdot V_o}{\varepsilon^3 \cdot \phi^2 \cdot d_p^{\,2}} + \frac{1.75 \cdot (1-\varepsilon) \cdot \rho \cdot V_o^{\,2}}{\varepsilon^3 \cdot \phi \cdot d_p} \tag{5.2}$$

Equation 5.2 can be rearranged in the form of a friction factor as a function of Reynolds Number and geometry.

$$f_{d_p} = \frac{\left(\frac{dP}{dx}\right) \cdot d_p}{\frac{1}{2} \cdot \rho \cdot V_o^{\,2}} = \frac{300 \cdot (1-\varepsilon)^2}{\varepsilon^3 \cdot \phi^2 \cdot \mathrm{Re}_{d_p}} + \frac{3.75 \cdot (1-\varepsilon)}{\varepsilon^3 \cdot \phi} \tag{5.3}$$

The shape factor of the precrushed material was experimentally determined to be 0.55. The particles used to construct the packed beds were precrushed to a particle diameter between 0.42 mm and 0.6 mm. The measured friction factor based on the precrushed particle diameter is illustrated in Figure 5-5 for several regenerator beds. The corresponding curves generated with Equation 5.3 based on the mean particle diameter of the precrushed material are also shown.

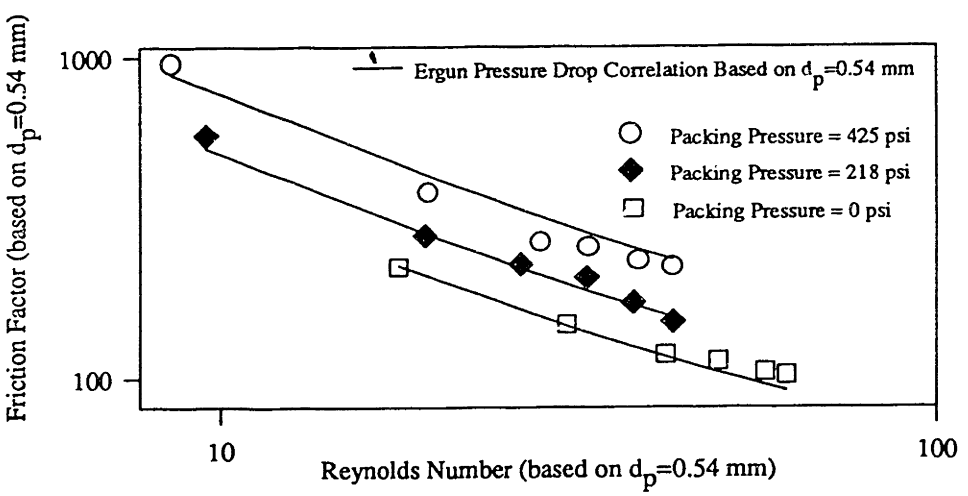

Figure 5-5. Experimental Friction Factor Measurements

The friction factor calculated with the Ergun correlation based on the precrushed mean particle diameter correlates well with the experimental data until the packing pressure reaches approximately 200 psi. At that point, the packing pressure is sufficient to further crush the particles contained in the regenerator bed. The region below 200 psi is denoted in Figure 5-4 as the 'Packing Region' since no further particle crushing occurs during construction of the regenerator.

Above 200 psi, the experimental data begins to deviate from the Ergun correlation. This is denoted in Figure 5-4 as the 'Crushing Region'. When the packing pressure is increased above 500 psi, the bed blocks all flow at pressures below 5 atmospheres. This region is labelled as the 'Occluded Region' in Figure 5-4.

The results of the friction factor experiments are used to modify the nonideal model in order to more realistically simulate the pressure gradients within the packed bed. The nonideal model results presented in Section 5.4 are generated with the modified pressure gradient correlation (i.e. Equation 5.2).

5.3.2 Packed Bed Design

The desire for reduced porosity would indicate that the packing pressure should be made as large as possible. However, the packing pressure must not be increased to the point where the bed becomes occluded. This situation indicates that favorable flow passages are present within the regenerator, thermally isolating segments of the magnetic material.

In order to avoid the occluded region while still reducing the porosity, the packed bed will be constructed with a packing force of 375 psi. This should reduce the porosity to approximately 35% without creating favorable flow passages. The design parameters corresponding to the packed bed internal geometry are listed in Table 5-2.

Design Parameter	Specified Value
Mean Precrushed Particle Diameter	0.56 mm
Packing Force	375 psi
Shape Factor	0.55
Regenerator Porosity	0.35

Table 5-2. Design Parameters Specified by Packed Bed Internal Geometry

5.4 Heat Exchangers and Precoolant Regenerator

A preliminary estimate of the type of flow which will be encountered within the two heat exchangers and the precoolant regenerator is obtained using the nonideal regenerator model. These results are then used in the design of the heat transfer devices.

This section is divided into five subsections. In the first subsection, the mass ratio is reoptimized based on the design specifications obtained in Sections 5.2 and 5.3. The second subsection determines the appropriate applied field profile. The final three subsections present preliminary calculations related to the size of each heat transfer device. These calculations are meant to give order of magnitude estimates only. More detailed analysis will be required for a final design.

5.4.1 Mass Ratio Optimization

The operating mass ratio obtained in Section 4.2 was based on slightly different conditions than are specified for the experimental apparatus. The discrepancies are listed in Table 5-3. The lower refrigeration temperature and decreased total applied field variation are the major changes. The change in cycle time and regenerator geometry should not significantly affect the optimum mass ratio.

Because of the changes listed in Table 5-3, the nonideal model was used to reoptimize the mass ratio at the design parameters. The pressure variation was specified as three to five atmospheres. The results of this optimization are shown in Figure 5-6. Both the specific refrigeration and the optimal mass ratio have been substantially reduced. These results

Input Parameter	Section 4.2 Value	Design Value
High Applied Field	5.0 Tesla	4.0 Tesla
Total Cycle Time	2.0 seconds	12.0 seconds
High Pressure	varied from 3.75 atmospheres to 15 atmospheres	5 atmospheres
Refrigeration Temperature	4.75° K	4.2° K
Particle Shape Factor	1.0 (i.e. spherical)	0.55 (i.e. irregular)
Regenerator Length	12.1 cm	28.0 cm
Regenerator Diameter	3.5 cm	3.01 cm

Table 5-3. Discrepancies Between Design Parameters and Chapter 4 Input Parameters

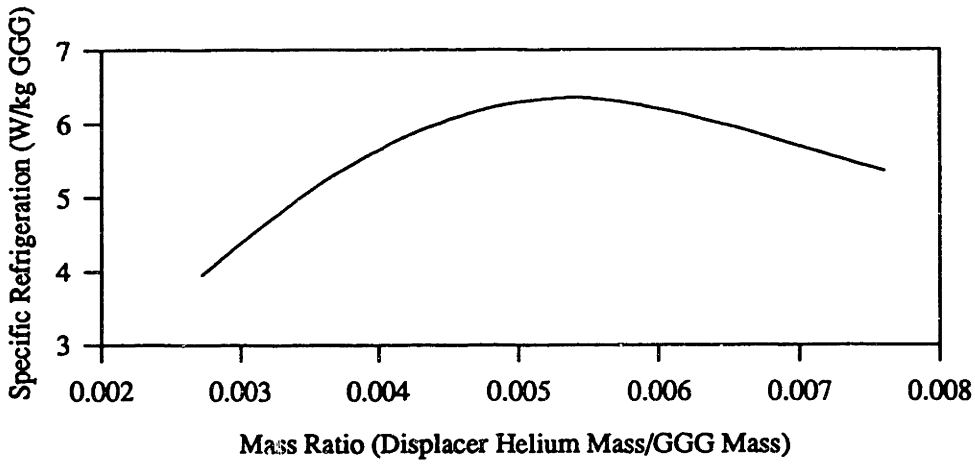

Figure 5-6. Specific Refrigeration as a Function of Mass Ratio

can be compared to the refrigeration surface generated in Chapter 4 by normalizing the refrigeration with respect to cycle time and by defining a more appropriate mass ratio. The total mass ratio is defined as the ratio of the mass of helium which enters the magnetic regenerator to the mass of GGG contained in the system. The total mass ratio accounts for the increased heat capacity of the helium entrained within the regenerator that results when the refrigeration temperature is reduced.

The normalized results are compared in Figure 5-7. The refrigeration produced per cycle

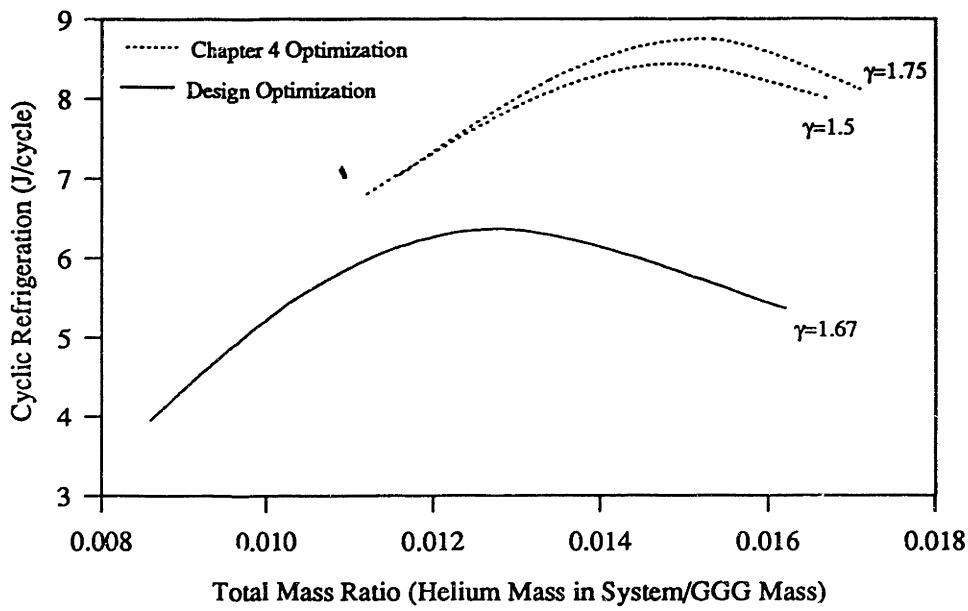

Figure 5-7. Normalized Refrigeration as a Function of Total Mass Ratio

has been reduced by lowering the refrigeration temperature and the total applied field swing. The qualitative shape of the curves are similar, indicating that the results from Chapter 4 are applicable to magnetically active regenerative devices of various sizes.

5.4.2 Field Phasing Optimization

The results obtained in Chapter 4 were based on the applied field profile shown in Figure 4-1. It is not clear how the appropriate applied field variation changes as other parameters are varied. For this reason, the effect of applied field variation is studied with input parameters that correspond to the design specifications. The mass ratio used during these simulations corresponds to the optimal mass ratio determined in Subsection 5.4.1 (i.e. 5.4e-3).

The applied field variation during the cycle must be linear if the power supply voltage is maintained constant. The magnitude of the applied field swing during the compression process can be changed by varying the time of the compression process (e.g. if the applied field variation during the compression process is 1.5 Tesla, then the compression process is 3 seconds and the inflow process is 3 seconds).

The results of varying the applied field phasing are illustrated in Figure 5-8. The refrigerator is more effective when the bulk of the applied field variation occurs during the compression process. The rapid degradation of the cyclic refrigeration which occurs as the compression applied field swing approaches the total applied field swing (i.e. 3 Tesla) is due to the extremely large pressure gradients created as the inflow process is forced to occur in a shorter period of time.

The explanation for the improvement in performance which occurs when the applied field swing is imposed concurrently with the pressure variation can be seen by examining the thermodynamic cycles undergone by the GGG. The thermodynamic cycles undergone by the GGG at the midpoint of the regenerator are plotted in Figure 5-9 for a compression applied field swing of 2.0 Tesla and 2.5 Tesla.

The amount of heat generated by the compression process is relatively unaffected by the applied field phasing. Therefore, the amount of heat which the GGG must absorb during the compression process does not change. If we neglect the entropy generation during the

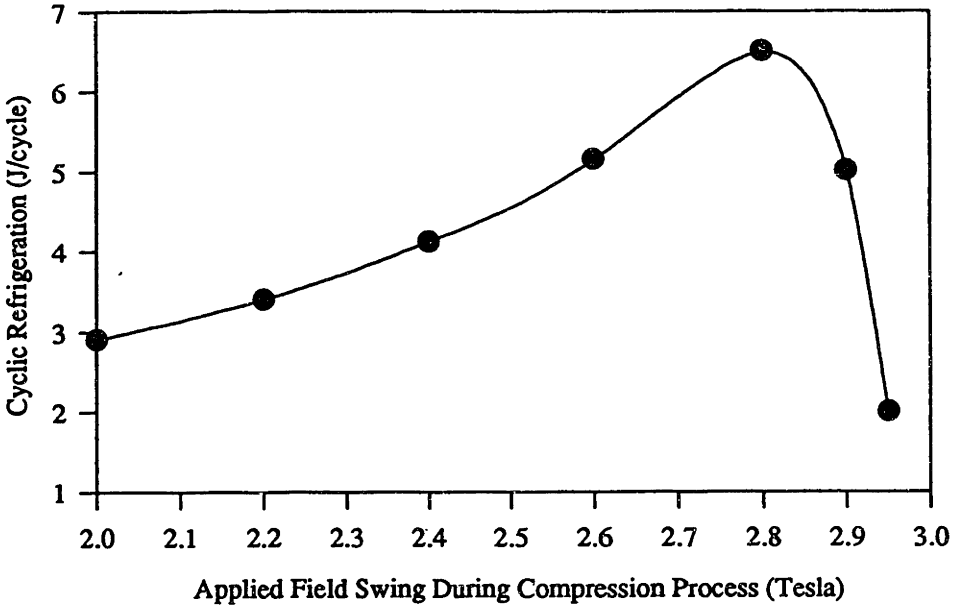

Figure 5-8. Cyclic Refrigeration as a Function of Applied Field Swing During Compression

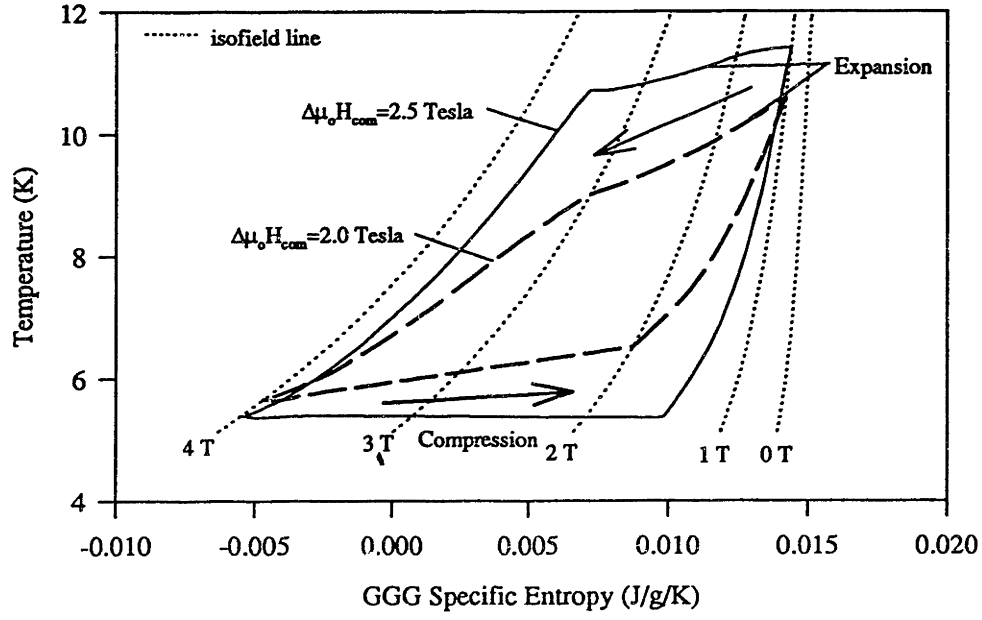

Figure 5-9. Thermodynamic Cycles Undergone by the Magnetic Material at the Regenerator Midpoint for Varying Applied Field Phasing

compression process, then the heat transfer and the applied field variation completely determine the state of the GGG after it undergoes the compression process. Figure 5-9 shows that by reducing the applied field swing, the final position is forced upwards (i.e. towards higher temperature). A similar effect is observed during the expansion process. The net result is that the thermodynamic cycle undergone by the GGG sweeps out less area when the applied

field swing during the compression process is reduced. This constitutes a degradation in the magnetic component of refrigeration without any associated improvement in the mechanical component of refrigeration.

The design parameters which are specified based on the results of this section are listed in Table 5-4.

5.4.3 Warm Heat Exchanger

The function of the warm heat exchanger is to cool the 'superheated' working fluid down to 12° K. The results of the nonideal model obtained in Section 5.4.1 are used to characterize the flow through the warm heat exchanger. In particular, the mass average temperature of the flow leaving the warm end ($\overline{T}_{superheat}$) is 12.7° K and the average mass flow rate ($\overline{mf}$) is 1.64 g/s.

A preliminary analysis of the warm end heat exchanger is carried out using the effectiveness-NTU method[51]. The necessary effectiveness in the warm heat exchanger can be expressed in terms of various flow temperatures (T_{boil} is the temperature at which the precoolant flow boils).

$$\varepsilon = \frac{(\overline{T}_{superheat} - T_{warm})}{(\overline{T}_{superheat} - T_{boil})} = 0.0824 \qquad (5.4)$$

The precoolant flow is assumed to be large enough to promote boiling throughout the cold side of the heat exchanger. Therefore, the heat capacity ratio is zero and the number of transfer units required is not a function of the precoolant mass flow rate (mf_c)[52].

Design Parameter	Specified Value
High Imposed Pressure	500 kPa
Compression Time	5.0 seconds
Inlet Time	1.0 seconds
Expansion Time	5.0 seconds
Outlet Time	1.0 seconds
Displacer Volume	35.3 cm^3

Table 5-4. Design Parameters Specified by Cycle Optimization

$$NTU = -\ln(1-\varepsilon) = 0.086 \tag{5.5}$$

The number of transfer units is defined in Equation 5.6.

$$NTU = \frac{htc \cdot A_H}{\overline{mf} \cdot c_p} \tag{5.6}$$

The warm heat exchanger will have a shell and tube geometry. The working fluid will flow inside the tube bundle while the boiling precoolant flows through the shell. The outer diameter of the shell is constrained by the solenoid bore. Therefore, the number of tubes depends only on the diameter of the tube and the tube spacing. This relation has been determined graphically for a 1.25 by 1.25 staggered matrix.

The resistance to heat transfer inside the tube bundles will dominate the resistance in the boiling precoolant. The Reynolds Number based on the average mass flow rate can be determined as a function of tube diameter.

$$\mathrm{Re}_D = \frac{\overline{mf} \cdot 4}{N_{tube}(D) \cdot \pi \cdot D \cdot \mu} \tag{5.7}$$

The Reynolds Number varies between 3817 for a 0.035 inch diameter tube to 3460 for a 0.065 inch diameter tube. These values are above the accepted transition value (i.e. 2300) and turbulent flow is assumed to be present within the tubes. The friction factor (f) is estimated as 0.042 based on the smooth pipe curve of the Moody chart. The Gnielinski Formula[53] is used to estimate the appropriate Nusselt Number.

$$Nu_D = \frac{(f/8) \cdot (\mathrm{Re}_D - 1000) \cdot \mathrm{Pr}}{1 + 12.7 \cdot (f/8)^{1/2} \cdot (\mathrm{Pr}^{2/3} - 1)} \tag{5.8}$$

Equations 5.6 through 5.8 and the number of tubes as a function of tube diameter are combined in order to determine the required heat exchanger as a function of the tube diameter. This relation is plotted in Figure 5-10. The heat exchanger volume and associated pressure drop are also indicated. The heat exchanger volume represents the amount of warm helium which is swept back into the regenerator during the downflow.

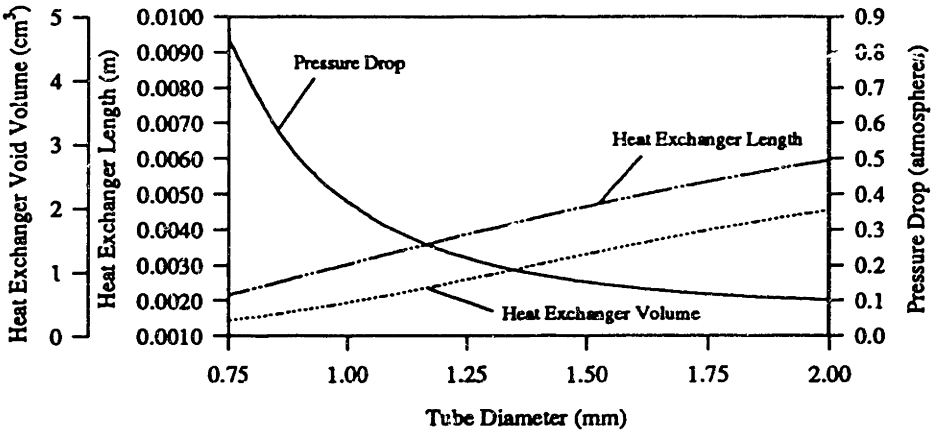

Figure 5-10. Warm Heat Exchanger Characteristics as a Function of Tube Diameter

This constitutes a direct loss of refrigeration. On the other hand, the pressure drop across the heat exchanger at very low tube diameters becomes unacceptable. A good balance is achieved for tube diameters between 0.035 inch and 0.065 inch (i.e. between 0.89 mm and 1.65 mm).

The mass flow rate of precoolant which is required in order for the initial assumption of boiling heat transfer on the entire cold side to be valid is calculated.

$$mf_c \approx \frac{\overline{mf} \cdot c_p \cdot (\overline{T}_{superheat} - T_{warm})}{h_{fg}} = 0.319 \text{ g/s} \qquad (5.9)$$

The additional heat load in the warm heat exchanger due to the entrained helium compression (q_{com}) can be approximated and compared with the design heat transfer rate (q_{whx}) for a 0.065 inch diameter tube geometry.

$$\frac{q_{com}}{q_{whx}} \approx \frac{V_{whx} \cdot T_{warm} \cdot (s_{he}(T_{warm}, P_{low}) - s_{he}(T_{warm}, P_{high})) \cdot \rho_{he}}{\tau \cdot \overline{mf} \cdot c_p \cdot (\overline{T}_{superheat} - T_{warm})} \approx 0.0039 \qquad (5.10)$$

Finally, the refrigeration loss associated with the initial inflow of warm helium entrained in the heat exchanger into the regenerator during the inflow process can be compared to the predicted refrigeration.

$$\frac{Q_{wi}}{Q_{ref}} \approx \frac{V_{whx} \cdot c_p \cdot \rho_{he} \cdot (\overline{T}_{superheat} - T_{warm})}{Q_{ref}} \approx 0.014 \qquad (5.11)$$

The preliminary design specifications associated with the warm heat exchanger are listed in Table 5-5.

5.4.4 Precoolant Regenerator

The initial sizing of the precoolant regenerator is accomplished by modelling the system as an unbalanced, counterflow, steady state heat exchanger. The cold side heat capacity is composed of the precoolant stream and the average mass flow rate of the working fluid. This fluid is modelled as entering the system at the mass average temperature of the two components.

$$T_{pc} = \frac{(\overline{mf} \cdot T_{warm} + mf_c \cdot T_{boil})}{\overline{mf} + mf_c} \qquad (5.12)$$

The boiling within the precoolant is neglected because the amount of heat which is accepted by the boiling process is negligible relative to the total heat transfer within the precoolant regenerator. The hot side is composed of the average mass flow rate of the working fluid. This stream enters at room temperature (i.e. T_{room}=293° K) and must be cooled to at least the mass average 'superheat' temperature determined in Subsection 5.4.3 (i.e 12.7° K).

The effectiveness required for these boundary conditions can be determined as a function of the precoolant mass flow rate[54].

$$\varepsilon = \frac{(T_{room} - \overline{T}_{superheat})}{(T_{room} - T_{pc}(mf_c))} \qquad (5.13)$$

Design Parameter	Specified Value
Tube Diameter	0.065 inch
Number of Tubes	90 tubes
Minimum Precoolant Mass Flow Rate	0.319 g/s
Length	5.5 cm

Table 5-5. Design Parameters Specified by Warm Heat Exchanger

The required number of transfer units can also be expressed as a function of the precoolant mass flow rate. This relation is illustrated in Figure 5-11.

$$NTU = \frac{\ln\left[\dfrac{\varepsilon - 1}{C_r(mf_c)\cdot\varepsilon - 1}\right]}{C_r(mf_c) - 1} \qquad C_r(mf_c) = \frac{\overline{mf}}{\overline{mf} + mf_c} \tag{5.14}$$

The precoolant regenerator will have a shell and tube geometry, the regenerative heat capacity will be contained within the tube bundles. Therefore, the heat transfer resistance in the hot and the cold side will be of the same order of magnitude. The flow regime within the tube bundles will change with tube diameter, number of tubes, and the temperature inside the precoolant regenerator.

If the entrainment of helium within the regenerator void volume is neglected, then the mass flow remains constant along the length of the regenerator and the Reynolds Number may be written in the following form.

$$\mathrm{Re}_D = \frac{4\cdot mf}{\pi\cdot D\cdot N\cdot \mu_{he}(T)} \tag{5.15}$$

The only axially dependent variable in Equation 5.15 is the helium viscosity. The viscosity of helium increases by an order of magnitude as it is heated from the warm reservoir temperature to room temperature. Therefore, the Reynolds Number will decrease by an order of magnitude. This places most of the flow regime in the laminar region. The hot

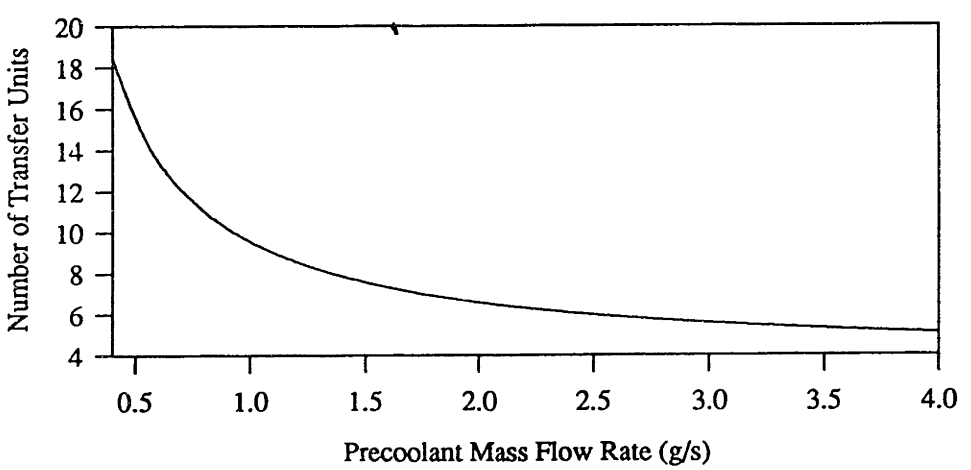

Figure 5-11. Number of Transfer Units Required in Precoolant Regenerator

and cold side heat transfer coefficients are approximated based on the constant Nusselt number corresponding to laminar, fully developed pipe flow subject to a constant heat flux. The overall heat transfer coefficient is taken as one half of the heat transfer coefficient associated with each side.

$$htc_{tot}(x) = \frac{Nu_D \cdot k_{he}(T(x))}{2 \cdot D} \tag{5.16}$$

The thermal conductivity of helium increases by an order of magnitude as the temperature is raised. This causes an equivalent increase in the heat transfer coefficient. The definition of a transfer unit was given by Equation 5.6 assuming a constant heat transfer coefficient.

$$NTU = \frac{htc \cdot A_H}{\overline{mf} \cdot c_p} \tag{5.6}$$

A more general definition accounts for variations in the heat transfer coefficient.

$$NTU = \frac{\int_0^L htc(x) \cdot P \cdot dx}{\overline{mf} \cdot c_p} \tag{5.17}$$

By substituting Equation 5.16 into Equation 5.17, the number of transfer units may be expressed in terms of temperature and regenerator length.

$$NTU = \frac{N \cdot \pi \cdot Nu_D}{2 \cdot c_p \cdot \overline{mf}} \int_0^L k_{he}(T(x)) \cdot dx \tag{5.18}$$

The assumptions implied by Equation 5.18 are listed below.

1. The constant pressure specific heat of helium does not vary significantly between 12° K and 293° K.
2. The temperature profile in the hot and cold sides are nearly identical.
3. The flow regime is laminar throughout the precoolant regenerator.
4. The effect of the entrained helium within the precoolant regenerator is negligible.

The first assumption is easily validated by examining the variation in helium constant pressure specific heat as a function of temperature. This plot is illustrated in Figure 5-12. The remaining assumptions are examined after the regenerator geometry is specified.

By obtaining an analytical expression for the thermal conductivity of helium and assuming a linear temperature profile, Equation 5.18 may be combined with Equations 5.12 through 5.14 in order to yield the length of precoolant regenerator required as a function of the precoolant mass flow rate and the number of regenerator tubes. These calculations assume a 0.065 inch tube diameter. These results are obtained in Appendix D and are presented in Figure 5-13.

The cryogenic dewar in which the experimental apparatus will operate limits the overall precoolant regenerator length to approximately one half of a meter. A length of 0.57 meters is taken to be the design constraint. Increasing the number of tubes will decrease the mass flow rate of precoolant required. The experiment can continue until the liquid helium bath recedes below the level of the superconducting solenoid. In the absence of other heat leaks and neglecting dissipation within the solenoid, the precoolant mass flow rate can be converted directly into experiment time ($\tau_{experiment}$) based on the geometry of the cryogenic dewar. The results are presented in Figure 5-14.

$$\tau_{experiment} = \frac{L_{dewar} \cdot A_{dewar} \cdot \rho_{he,liquid}}{\dot{m} f_c} \tag{5.19}$$

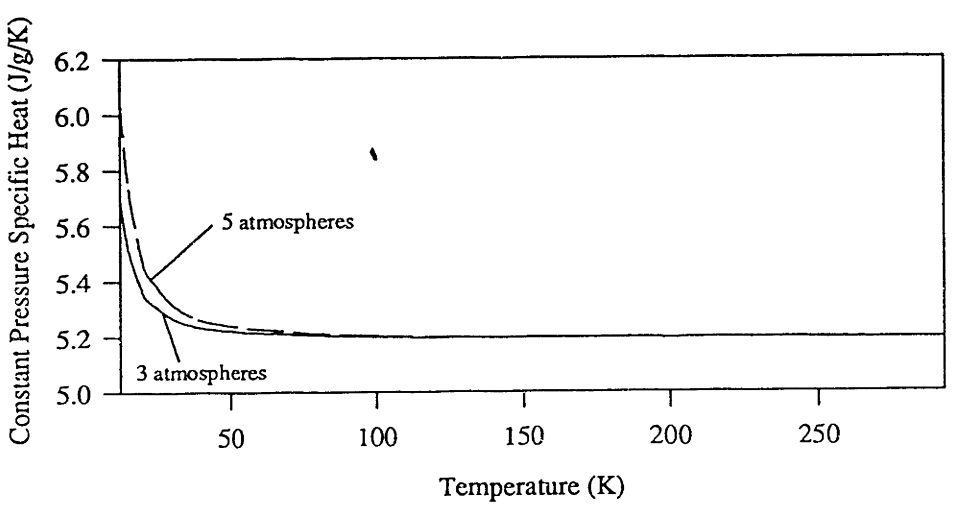

Figure 5-12. Helium Constant Pressure Specific Heat Variation Between Warm Reservoir Temperature and Room Temperature

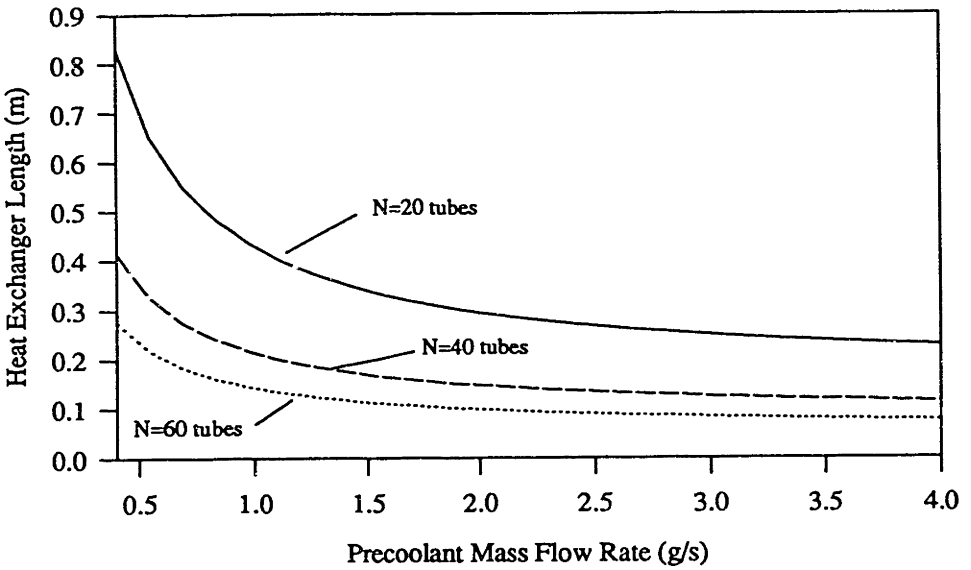

Figure 5-13. Precoolant Regenerator Length

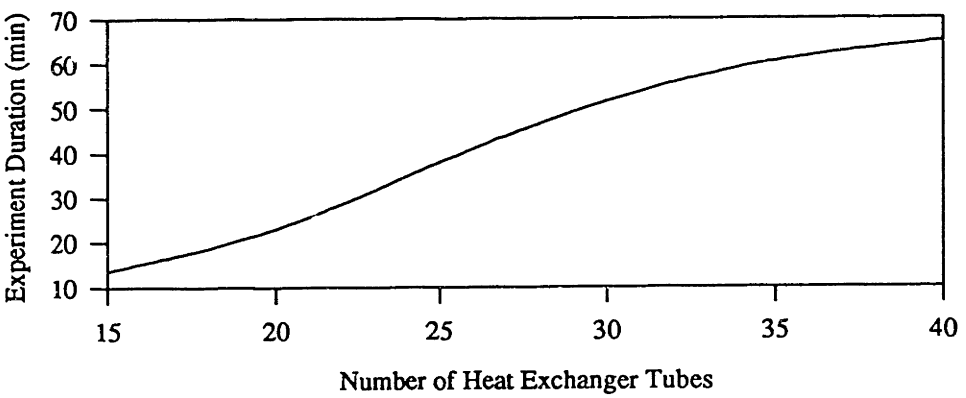

Figure 5-14. Duration of Experiment as a Function of Number of Precoolant Regenerator Tubes

In order to obtain the longest experiment duration, the number of tubes should be made as large as possible.

The assumptions regarding the temperature profiles are checked by numerically integrating the governing equations within the regenerator. The resulting temperature profiles indicate that the assumption of a linear temperature variation significantly overestimates the performance of the regenerator. The thermal conductivity variation tends to pull the temperature profiles down. As a result, the majority of the regenerator operates in the low temperature region which is characterized by inferior heat transfer. A set of appropriate

Design Parameter	Specified Value
Tube Diameter	0.065 inch
Number of Tubes	60 tubes
Length	57 cm
Required Precoolant Mass Flow Rate	0.5 g/s

Table 5-6. Design Parameters Specified by the Precoolant Regenerator

design specifications are listed in Table 5-6. The temperature profiles corresponding to this geometry are illustrated in Figure 5-15.

Although satisfactory heat transfer performance can be obtained with a smaller tube diameter, the resulting pressure drop across the regenerator becomes prohibitively high at the warm end due to the comparatively large viscosity at higher temperatures. The pressure drop for a given mass flow is proportional to tube diameter to the fifth power. By using 0.065 inch diameter tubes rather than 0.035 inch diameter tubes, the pressure drop is reduced to an acceptable level.

The Reynolds Number within the regenerator tubes for these design parameters is determined in Appendix E and illustrated in Figure E-2. The flow regime over most of the length of the regenerator is in the laminar or transition region, validating the assumption of a constant Nusselt Number.

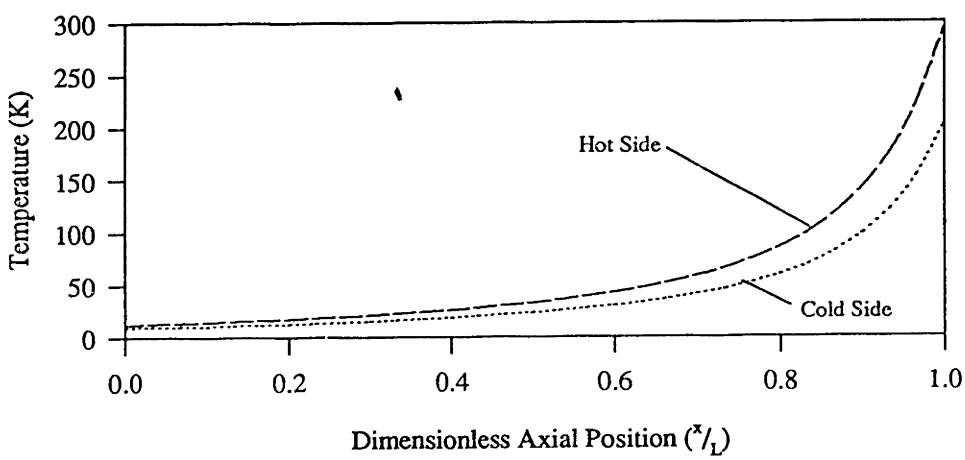

Figure 5-15. Temperature Profiles within Precoolant Regenerator

The compression of the entrained helium will add to the heat load within the regenerator. The relative effect of this heat load can be approximated by comparing it to the total heat transfer within the regenerator.

$$\frac{Q_{com}}{Q_{pc}} \approx \frac{N \cdot \pi \cdot D^2 \cdot L \cdot \int_{T_{warm}}^{T_{room}} \rho_{he} \cdot T \cdot (s_{he}(T, P_{low}) - s_{he}(T, P_{high})) \cdot dT}{4 \cdot \tau \cdot (T_{room} - T_{warm})^2 \cdot \overline{mf} \cdot c_p} \tag{5.20}$$

By modelling helium as a perfect gas within this temperature range, Equation 5.20 can be simplified.

$$\frac{Q_{com}}{Q_{pc}} \cong \frac{\overline{P} \cdot \ln(\gamma) \cdot N \cdot \pi \cdot D^2 \cdot L}{\tau \cdot (T_{room} - T_{warm}) \cdot \overline{mf} \cdot c_p} \approx 0.0025 \tag{5.21}$$

Equation 5.21 indicates that the heat transfer associated with the compression of the helium entrained within the precoolant regenerator can safely be ignored. However, the void volume of the precoolant regenerator is a significant fraction of the total dead volume in the system and must be accounted for when the total volumetric flow rate is estimated for the room temperature compressor.

The total volumetric flow rate can be approximated by determining the volumetric flow rate at low pressure and room temperature corresponding to the average mass flow rate entering the magnetic refrigerator and then adding the volumetric flow rate required by the density variation in the precoolant regenerator void volume.

$$vf \approx \frac{\overline{mf}}{\rho_{he}(T_{room}, P_{low})} + \frac{N \cdot \pi \cdot L \cdot D^2 \cdot \int_{T_{warm}}^{T_{room}} (\rho_{he}(T, P_{high}) - \rho_{he}(T, P_{low})) \cdot dT}{4 \cdot \tau \cdot (T_{room} - T_{warm}) \cdot \rho_{he}(T_{room}, P_{low})} \tag{5.22}$$

By modelling the helium as a perfect gas, Equation 5.22 can be simplified.

$$vf \approx \frac{\overline{mf}}{\rho_{he}(T_{room}, P_{low})} + \frac{N \cdot \pi \cdot L \cdot D^2 \cdot (P_{high} - P_{low}) \cdot T_{room} \cdot \ln\left[\frac{T_{room}}{T_{warm}}\right]}{4 \cdot \tau \cdot (T_{room} - T_{warm}) \cdot P_{low}} \approx 7.8 \text{ cfm} \tag{5.23}$$

5.4.5 Cold Heat Exchanger

The cold heat exchanger is very similar to the warm heat exchanger. The 'subcooled' helium which leaves the regenerator during the inlet stroke is warmed to the refrigeration temperature as it flows through the cold heat exchanger. The heat is obtained from helium vapor condensing on the flow passages within a shell and tube geometry. The helium is expanded within the displacer and therefore enters the cold heat exchanger 'subcooled' during the outlet stroke as well.

The cyclic flow patterns within the cold heat exchanger are quite different from those in the warm heat exchanger. The mass flow rate within the warm heat exchanger could be modelled as approximately constant since the highly porous magnetic regenerator bed caused significant flow to occur during the compression and expansion processes. The mass flow rate at the ends of the regenerator are pictured in Figure 5-16.

It is more appropriate to model only the inlet and outlet strokes within the cold heat exchanger, since negligible flow occurs during the compression and expansion processes. The inlet and outlet strokes are characterized by a nearly constant mass flow rate of 4.9 g/s. The 'subcool' temperature (i.e. $\overline{T}_{subcool}$) is the mass average temperature of the helium entering the cold heat exchanger during the inlet stroke. This temperature is determined to be 3.7° K. The temperature at which the helium enters the cold heat exchanger as it

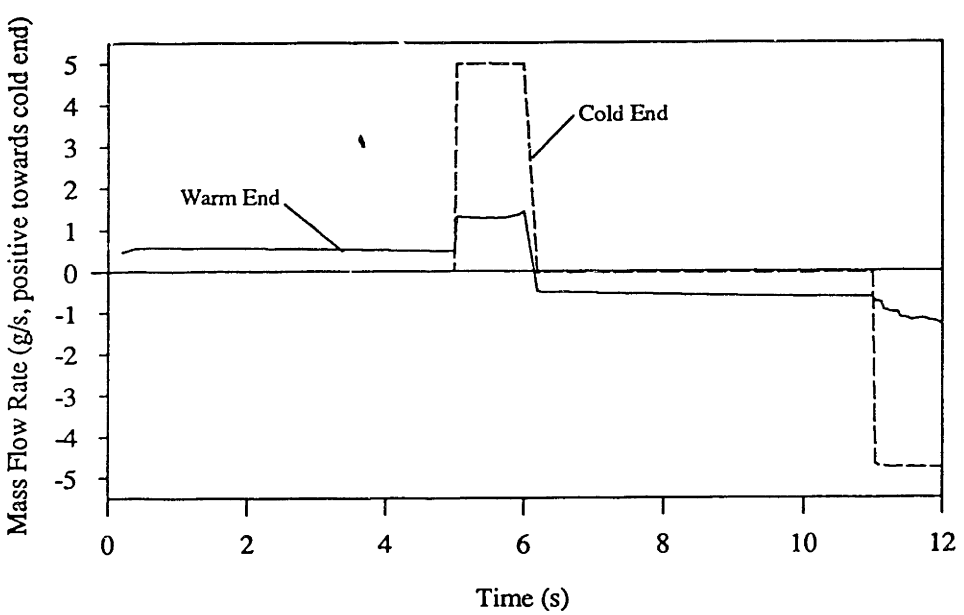

Figure 5-16. Mass Flow Rate Entering and Leaving the Regenerator

travels from the displacer to the regenerator during the outlet stroke is called the expansion temperature (T_{exp}). This temperature can be estimated by assuming that the displacer is adiabatic.

$$T_{exp} \approx T_{he}(s_{he}(4.2K,500kPa),300kPa) \approx 4.01\text{K} \tag{5.24}$$

The inlet process has the most dramatic temperature difference. Therefore, this is the process which should be considered during the design. The effectiveness of the cold heat exchanger during the inlet stroke is calculated with Equation 5.25.

$$\varepsilon = \frac{(T_{exit} - \overline{T}_{subcool})}{(T_{boil} - \overline{T}_{subcool})} \tag{5.25}$$

The number of transfer units required for a desired effectivenes is given by Equation 5.26.

$$NTU = -\ln(1-\varepsilon) \tag{5.26}$$

Complete effectiveness cannot be achieved (i.e. the helium working fluid will never leave the cold heat exchanger at precisely the refrigeration temperature). It is possible to obtain an effectiveness arbitrarily close to unity by increasing the number of transfer units.

As the number of transfer units is increased, both the pressure drop across the heat exchanger and the void volume within the heat exchanger are increased. These two effects tend to offset the increase in refrigeration produced by greater effectiveness. These processes can be compared by calculating the lost refrigeration associated with each effect.

The refrigeration degradation associated with a thermally imperfect heat exchanger ($Q_{deg,chx}$) is simply the relative enthalpy (with respect to the refrigeration temperature) that flows out of the heat exchanger.

$$Q_{deg,chx} = \overline{mf} \cdot \tau_{inlet} \cdot (h_{he}(T_{boil}, P_{high}) - h_{he}(T_{exit}, P_{high})) \tag{5.27}$$

The very small temperature differences within the cold heat exchanger allow the constant pressure specific heat to be considered constant. Therefore, Equation 5.27 can be rewritten in terms of the number of transfer units.

$$Q_{deg,chx} = c_p \cdot \overline{mf} \cdot \tau_{inlet} \cdot (T_{boil} - \overline{T}_{subcool}) \cdot \exp[-NTU] \qquad (5.28)$$

The pressure drop across the cold heat exchanger degrades the refrigeration by reducing the mechanical expansion which occurs within the displacer. This loss (i.e. $Q_{deg,mech}$) can be quantified in terms of the lost work transfer on the displacer.

$$Q_{deg,mech} = 2 \cdot V_{dis} \cdot \Delta P_{chx} \qquad (5.29)$$

In order to relate this loss to the number of transfer units, the geometry of the cold end heat exchanger must be specified. Tentatively, this heat exchanger will be specified as a tube bank consisting of 0.035 inch diameter tubes in a 1.25x1.25 staggered matrix. The smaller tube diameter is acceptable since the helium viscosity at these temperatures is very low.

The heat exchanger will operate most effectively if it is placed directly beneath the magnetic regenerator. The external diameter of the heat exchanger is therefore constrained by the bore of the solenoid. The number of 0.035 inch diameter tubes which may be placed within this area was determined in Section 5.4.3 to be 230.

The Reynolds Number corresponding to the average mass flow rate and this geometry is approximately 7500. This is well above the transition region and the flow may be considered turbulent. The friction factor corresponding to fully developed, turbulent, smooth pipe flow is obtained from the Moody Diagram and the Gnielinski Formula (Equation 5.8) is used to determine the approximate heat transfer coefficient within the tubes. The resistance to heat transfer within the condensing helium is negligible. The overall heat transfer coefficient (i.e. htc_{tot}) is found to be approximately 580 W/m^2/K.

The number of transfer units under these conditions is completely determined by the length of the heat exchanger. By combining the definition of the number of transfer units, the friction factor, and Equation 5.29, the refrigeration degradation due to the pressure drop can be expressed in terms of the number of transfer units[55].

$$Q_{deg,mech} = \frac{4 \cdot f \cdot \overline{mf}^3 \cdot V_{dis} \cdot NTU \cdot c_p}{\pi^3 \cdot \rho \cdot D^6 \cdot N^3 \cdot htc_{tot}} \qquad (5.30)$$

The void volume within the cold heat exchanger degrades the refrigeration by contributing a heat transfer during the compression process. This degradation can be approximated by modelling the compression within the heat exchanger as an isothermal process.

$$Q_{deg,com} = V_{chx} \cdot \rho_{he} \cdot (s_{he}(T_{cold}, P_{low}) - s_{he}(T_{cold}, P_{high})) \tag{5.31}$$

With the geometry constraints described above, Equation 5.31 may be expressed in terms of the number of transfer units.

$$Q_{deg,com} = \frac{D \cdot \rho \cdot (s_{he}(T_{cold}, P_{low}) - s_{he}(T_{cold}, P_{high})) \cdot NTU \cdot \overline{mf} \cdot c_p}{4 \cdot htc_{tot}} \tag{5.32}$$

The refrigeration lost due to compression of the helium entrained within the cold heat exchanger is found to be negligible.

The refrigeration degradation due to imperfect heat transfer and pressure drop are compared in Figure 5-17. The cycle degradation is less than 0.75 J/cycle when the number of transfer units is greater than 2.8. This corresponds to a length of 10 cm. The design specifications for the cold heat exchanger are listed in Table 5-7.

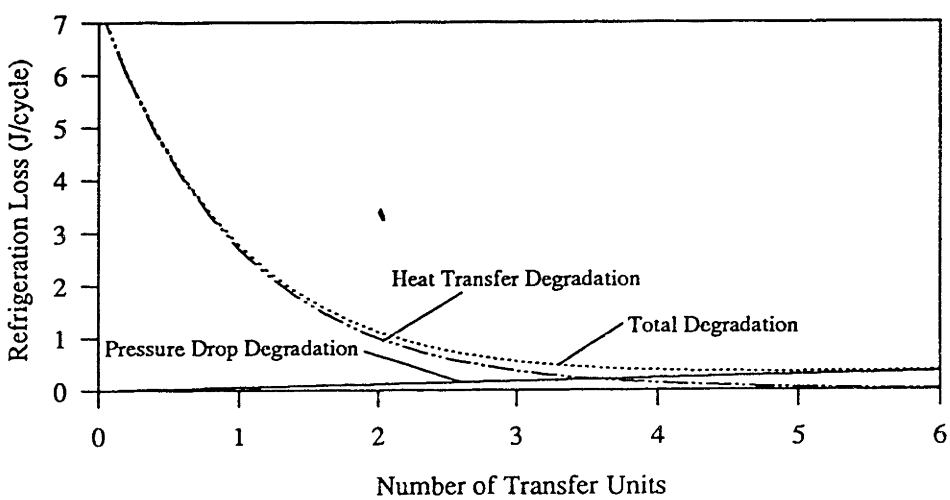

Figure 5-17. Degradation of Refrigeration as a Function of Cold Heat Exchanger Transfer Units

Design Parameter	Specified Value
Length	10.0 cm
Tube Diameter	0.035 inch
Number of Tubes	230 tubes
External Diameter	3.01 cm

Table 5-7. Design Parameters Specified by Cold Heat Exchanger

5.5 Displacer System

The displacer moves within a cylinder which extends from room temperature to refrigeration temperature. Warm, pressurized helium will be vented into the back end of the piston. A warm seal will prevent helium leakage around the motor shaft.

The most important design criterion is that helium flow through the displacer-cylinder gap must be negligible. The flow resistance of the displacer and cylinder must be much higher than the flow resistance of the precoolant regenerator, heat exchangers, and magnetic regenerator in series.

The largest pressure differentials are produced across the displacer during the inlet and outlet strokes. The pressure drop across the precoolant regenerator during the inlet stroke is estimated in Appendix E by numerically integrating the friction factor corresponding to fully developed flow within a smooth pipe. This calculation is complicated by the temperature dependence of the helium viscosity and the transition from laminar to turbulent flow. The total pressure drop across the precoolant regenerator for the specified geometry is determined to be approximately 37 kPa.

The pressure drop across the heat exchangers may be estimated more easily since the helium properties can be considered constant over the relatively small temperature changes spanned by these devices. The pressure drop across the warm heat exchanger was determined in Section 5.4.3 to be approximately 10 kPa. The pressure drop across the cold heat exchanger is negligible. The average pressure drop across the magnetic regenerator during a flow process is 3.95 kPa. The total pressure drop across the displacer is approximately 51 kPa during the flow process.

The flow in the displacer-cylinder gap during a flow process is quite complicated. Fortunately, the temperature gradients tend to create a buoyantly stable situation, allowing

the temperature driven density forces to be ignored. The flow situation is illustrated in Figure 5-18. The flow may be modelled by superposing the velocity profiles generated by the displacer motion and the pressure gradients. The displacer and cylinder are modelled as a pair of parallel plates since the gap (h) is much smaller than the displacer diameter (D_{dis}).

$$v_x(y) = -\frac{h^2}{2 \cdot \mu} \cdot \frac{dP}{dx} \cdot \left(1 - \left(\frac{y}{h}\right)^2\right) + v_{dis} \frac{y}{h} \tag{5.33}$$

As a first approximation, the pressure gradient and helium properties are assumed to be constant. The bulk velocity is obtained by integrating Equation 5.33 across the gap.

$$v_b = \frac{-h^2}{3 \cdot \mu} \cdot \frac{dP}{dx} + \frac{v_{dis}}{2} \tag{5.34}$$

The total mass flow rate into the cold space through the displacer gap (i.e. mf_{dis}) is given by Equation 5.35.

$$mf_{dis} = \left[-\frac{h^2}{3 \cdot \mu} \cdot \frac{dP}{dx} + \frac{v_{dis}}{2} \right] \cdot h \cdot \pi \cdot D_{dis} \cdot \rho_{cold} \tag{5.35}$$

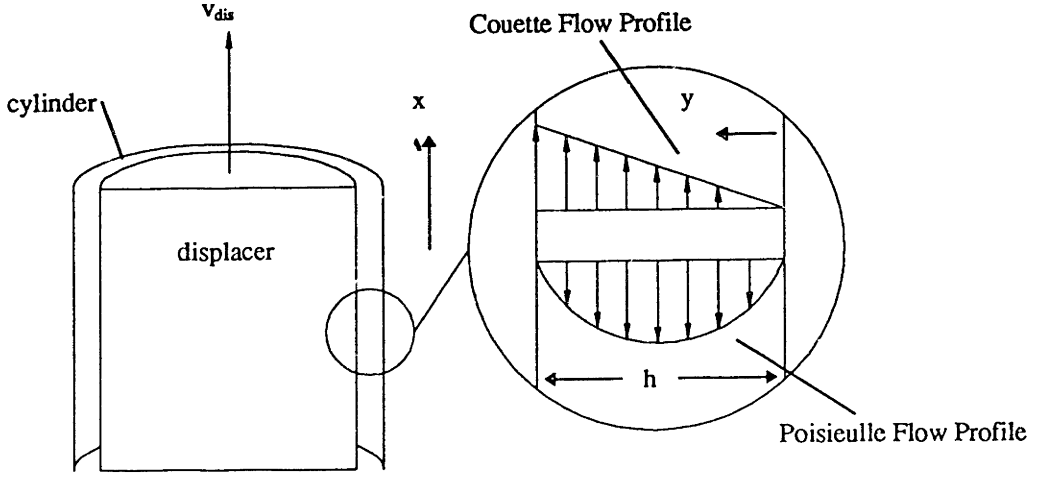

Figure 5-18. Flow Situation Between Displacer and Cylinder

The displacer volume (i.e. V_{dis}) is completely specified by the optimal mass ratio determined in Subsection 5.4.1. The time period of the inlet stroke (i.e. τ_{inlet}) has been specified at 1 second. Therefore, the diameter and velocity of the displacer are related by Equation 5.36.

$$v_{dis} = \frac{4 \cdot V_{dis}}{\pi \cdot D_{dis}^{\ 2} \cdot \tau_{inlet}} \tag{5.36}$$

The total length of the displacer cylinder (L_{cyl}) is constrained by the cryogenic dewar. This length is approximately 1.0 m. The pressure driven mass flow rate through the gap will be largest when the viscosity and density are small (i.e. at lower temperatures). In order to gain a conservative estimate of this flow, the properties at the lowest temperature are used. The ratio of the mass flow through the displacer gap to the mass flow through the refrigeration system (mf_{sys}) during an inlet stroke can be expressed as a function of gap clearance and displacer diameter.

$$\frac{mf_{dis}}{mf_{sys}} = \frac{-h^3 \cdot \Delta P_{dis} \cdot D_{dis} \cdot \pi \cdot \rho_{cold}}{3 \cdot \mu_{cold} \cdot L_{cyl} \cdot mf_{sys}} + \frac{2 \cdot V_{dis} \cdot h \cdot \rho_{cold}}{D_{dis} \cdot \tau_{inlet} \cdot mf_{sys}} \tag{5.37}$$

This relation is illustrated in Figure 5-19. The leakage flow must be small compared to the system flow. Figure 5-19 indicates that a small diameter and a small clearance are required.

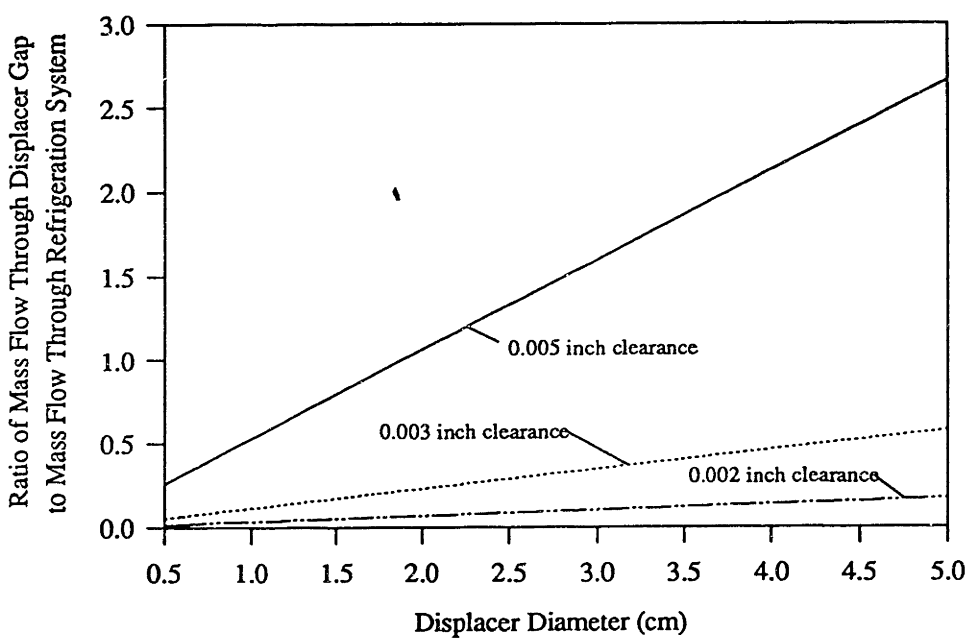

Figure 5-19. Ratio of Leakage Flow to System Flow

Increasing the displacer diameter will increase the force which must be overcome by the displacer drive motor while decreasing the required displacer stroke. The final design specifications must be based on both the characteristics of the displacer drive motor and the leakage flow ratio illustrated in Figure 5-19.

VI. CONCLUSION

Conventional Stirling cycle cryocoolers cease operation at temperatures below 10° K because the lattice specific heat of their regenerators becomes negligible. Devices which use a low temperature expansion process can supply refrigeration below 10° K but suffer from inherently poor reliability due to their low temperature valves.

Magnetically active regenerative cooling is based on the magneto-caloric effect which is present in paramagnetic materials. This effect becomes more pronounced at precisely the temperatures where the specific heat of conventional materials drops off. Because these devices do not use low temperature valves, they can operate at least as reliably as conventional Stirling cryocoolers. A magnetically active regenerative device is particularly suited to extend the cooling range of conventional cryocoolers.

This thesis examines a magnetically active device in which both the pressure and applied field are cyclically varied. Cyclic variation of the working fluid pressure has not previously been explored due to analytical and experimental difficulties. The analytical difficulties arise from the complex interactions that occur between two thermodynamic working fluids. The experimental difficulties are present on a prototype basis only. A magnetically active device which undergoes cyclic pressure variations would be an ideal low temperature stage for a conventional cryocooler.

The computational model developed in this thesis shows that properly phased pressure variation within a magnetically active device can increase its refrigeration potential dramatically. These results motivate the design of an experimental prototype which will prove the concept of magnetically active regeneration in the presence of cyclic pressure variations.

REFERENCES

1. E. G. Cravalho, and J. L. Smith, Jr., *Engineering Thermodynamics*, Pitman, Boston (1981), p. 218.

2. Zahn, M., *Electromagnetic Field Theory : A Problem Solving Approach*, Krieger, Malabar, (1979), p. 338.

3. E. A. Guggenheim, *Thermodynamics, an Advanced Treatment for Chemists and Physicists*, North-Holland, Amsterdam, (1967), p. 339.

4. F. W. Sears, *An Introduction to Thermodynamics, the Kinetic Theory of Gases, and Statistical Mechanics*, Addison-Wesley, Reading (1953), p. 174.

5. E. P. Gyftopoulos, and G. P. Beretta, *Thermodynamics: Foundations and Applications*, Macmillan, New York (1991), p. 121.

6. J. L. Smith, Jr., "Entropy flow and generation in energy conversion systems", submitted for presentation at the *ASME Winter Annual Meeting*, New Orleans, LA, Nov. 28 - Dec. 3, (1993).

7. W. F. Giauque, and D.P. MacDougall, "Attainment of temperatures below 1° absolute by demagnetization of $Gd_2(SO_4)_3$ $8H_2O$", *Phys. Rev.*, **43**:7768, (1933).

8. R. Barron, *Cryogenic Systems*, McGraw-Hill, Inc., New York (1966), p. 5.

9. J. A. Barclay, "Magnetic refrigeration for low-temperature applications", proceedings of the *Third Cryocooler Conference*, Boulder, CO, September 17 - 18, **1**: 20, (1984).

10. W. P. Pratt, Jr., S. S. Rosenblum, W. A. Steyert, and J. A. Barclay, "A continuous demagnetization refrigerator operating near 2° K and a study of magnetic refrigerants", *Crygenics*, **17**:689, (1977).

11. J. L. Smith, Jr, "Cryogenic refrigeration for space exploration - a challenge for the future", proceedings of the *XVII' th International Congress of Refrigeration*, Montreal, Quebec, Canada, August 10 - 17, **1**: 2, (1991).

12. T. Kuriyama, M. Takahashi, and H, Nakagome, "Regenerator performance and refrigeration mechanism for 4 ° K GM refrigerator using rare earth compound regenerator materials", proceeding of the *Seventh International Cryocooler Conference*, Santa Fe, NM, November 17 - 19, **2**: 429, (1992).

13. W. A. Steyart, "Rotating Carnot-cycle magnetic refrigerators for use near 2 ° K", *Journal of Applied Physics*, **49**: 1227, (1978).

14. J. A. Barclay, O, Moze, and L. Paterson, "A reciprocating magnetic refrigerator for 2° - 4° K operation - initial results", *Journal of Applied Physics*, **50**:5870, (1979).

15. D. L. Johnson, "Reciprocating Magnetic Refrigerator", proceedings of the *Third Cryocooler Conference*, Boulder, CO, September 17 - 18, **1**:33, (1984).

16. G. Patton, G. Green, J. Stevens, and J. Humphrey, "Reciprocating Magnetic Refrigerator", proceedings of the *Fourth International Cryocoolers Conference*, Easton, MD, September 25 - 26, **1**:65, (1986).

17. A. F. Lacaze, A. A. Lacaze, R. Beranger, and G. Bon Mardion, "Thermodynamic analysis of a double acting reciprocating magnetic refrigerator", proceedings of the *Ninth International Cryogenic Engineering Conference*, Kobe, Japan, May 11 - 14, (1982), p. 14.

18. J. E. Zimmerman, J. D. McNutt, and H. V. Bohm, "A magnetic refrigerator employing superconducting solenoids", *Cryogenics*, **2**:3, (1962).

19. Y. Hakuraku, and H. Ogata, "Thermodynamic analysis of a magnetic refrigerator with static heat switches", *Cryogenics*, **26**:171, (1986).

20. T. Numazawa, H. Kimura, M. Sato, and H. Maeda, "Analysis of a magnetic refrigerator operating temperature between 10° K and 1.4° K", proceedings of the *Sixth International Cryocoolers Conference*, Plymouth, MA, October 25 - 26, **1**:199, (1990).

21. C. P. Taussig, G. R. Gallagher, J. L. Smith, Jr., and Y. Iwasa, "Magnetic refrigeration based on magnetically active regeneration", proceedings of the *Fourth International Cryocoolers Conference*, Easton, MD, September 25-26, **1**:79, (1986).

22. C. P. Taussig, *Magnetically Active Regeneration*, Massachusetts Institute of Technology Ph.D. Thesis, (1986).

23. F. J. Cogswell, *Cycle Control of a Regenerative Magnetic Refrigerator Operating From 4.2° K to 15° K*, Massachusetts Institute of Technology Ph.D. Thesis, (1989).

24. S. Jeong, *Development of the Regenerative Magnetic Refrigerator Operating Between 4.2° K and 1.8° K*, Massachusetts Institute of Technology Ph.D. Thesis, (1992).

25. P. Seyfert, P. Brédy, and G. Claudet, "Construction and testing of a magnetic refrigeration device for the temperature range of 5° to 15° K", proceedings of the *Twelfth International Cryogenic Engineering Conference*, Southampton, U.K., July 12 - 15, (1988), p. 607.

26. K. Matsumoto, and T. Hashimoto, "Thermodynamic analysis of magnetically active regenerator", proceeding of the *International Congress of Cryogenics and Refrigeration*, Hangzhou, China, May 22-26, (1989), p. 110.

27. S. Jeong, and J. L. Smith, Jr., "Magnetically augmented regeneration in Stirling Cryocooler", *Advances in Cryogenic Engineering*, **39B**:1399, (1994).

28. F. J. Cogswell, J. L. Smith, Jr., and Y. Iwasa, "Regenerative magnetic refrigeration over the temperature range of 4.2° to 15° K", proceedings of the *Fifth International Cryocoolers Conference*, Monterey, CA, August 18 - 19, **1**: 81, (1988).

29. G. R. Ghallagher, *Analysis of a Magnetically Active Regenerator*, Massachusetts Institute of Technology M.S. Thesis, (1986).

30. R. A. Fisher, G. E. Brodale, E. W. Hornung, and W. F. Giauque, "Magnetothermodynamics of gadolinium gallium garnet. I. Heat capacity, entropy, magnetic moment from 0.5° to 4.2° K, with fields to 90 kG along the [100] axis", *Journal of Chemistry and Physics*, **59**:4652, (1972).

31. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, Second Edition*, Cambridge University Press, Cambridge, (1992), p. 147.

32. G. R. Ghallagher, *et. al.*

33. R. D. McCarty, *NBS Standard Reference Database 12*, National Bureau of Standards Center for Chemical Engineering, Gaithersburg, (1986), Interactive FORTRAN Program.

34. R. D. McCarty, "Thermophysical properties of Helium-4 from 2° - 1500° K with pressures to 1000 atmospheres", *NBS-TN-631*, (1972).

35. G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, (1986), p. 177.

36. E. P. Gyftopoulos, *et al.*

37. W. M. Rohsenow, J. P. Hartnett, and E. N. Ganic, *Handbook of Heat Transfer Applications, Second Edition*, McGraw-Hill, New York, (1973), p. 6:5-11.

38. S. Ergun, "Fluid flow through packed columns", *Chem Eng Prog*, **48**:89-94, (1952).

39. W. H. Press *et al.*, p. 41.

40. W. H. Press *et. al.*, p. 757.

41. P. V. O'Neil, *Advanced Engineering Mathematics*, Wadsworth, Belmont, (1991), p352.

42. P. V. O'Neil *et. al.*, p. 358.

43. K. F. Gustafson, *Partial Differential Equation and Hilbert Space Methods*, John Wiley & Sons, New York, (1987), p. 305.

44. S. C. Chapra, and R. P. Canale, *Numerical Methods for Engineering*, McGraw-Hill, New York, (1985), p.738.

45. B. Daudin, R. Lagnier, and B. Salce, "Thermodynamic properties of the gadolinium gallium garnet, $Gd_3Ga_5O_{12}$ between 0.05° and 25° K", *Journal of Magnetism and Magnetic Materials*, **27**:315, (1982).

46. W. W. Chin, *Construction and Testing of an AC Superconducting Magnet for a Magnetic Refrigerator*, Massachusetts Institute of Technology B. S. Thesis, (1985).

47. C. P. Taussig *et al.*

48. F. J. Cogswell *et al.*

49. W. W. Chin, *et al.*

50. W. M. Rohsenow *et al.*

51. F. P. Incropera, D. P. DeWitt, *Fundamentals of Heat and Mass Transfer*, John Wiley & Sons, New York, (1990), p. 658.

52. F. P. Incropera, *et al.*

53. A. F. Mills, *Heat Transfer*, Irwin, Boston, (1992), p. 270.

54. F. P. Incropera, *et al.*

55. J. L. Smith, Jr, "The Presentation of Heat-Transfer and Friction-Factor Data for Heat Exchanger Design", *American Society of Mechanical Engineers Publication* HT-59, (1966).

56. A. F. Mills, *et al.*

57. W. H. Press *et al.*

Appendix A. Computer Code

This appendix contains the listing of all of the code used to simulate an ideal and nonideal magnetically active regenerative refrigerator. The appendix is divided into seven sections. In the first section, the subroutine used to generate the thermodynamic properties of GGG is listed. In the second section, the program used to simulate an ideal 'mechanical' cycle is listed. In the third section, the subroutine used to calculate the temperature of GGG corresponding to a given entropy and applied field is listed. The fourth section lists the program used to simulate an ideal 'magnetic' cycle. The fifth section lists the program used to simulate an ideal 'combined' cycle. In the sixth section, the subroutines used to solve the system of linear equations are listed. The final section contains the program used to simulate a nonideal magnetically active regenerative refrigerator.

A.1 Subroutine GGG.c

This is the C subroutine GGG.c discussed in section 2.1. This subroutine determines the entropy and partial derivatives of entropy with respect to both temperature and applied field at an input value of temperature (T) and applied field ($\mu_o H$). The function returns these values in the input array gggprop. The function is translated from a FORTRAN program written in 1986 by Gregory R. Ghallagher for his Master of Science Degree Thesis.

```
void ggg(float T,float uoH,float *gggprop)

{
   float C0,C1,C2,C3,C4;
   float dC0dT,dC1dT,dC2dT,dC3dT,dC4dT;
   float d2C0dT2,d2C1dT2,d2C2dT2,d2C3dT2,d2C4dT2;
   float DEN,dDENdT,d2DENdT2;
   float s0,ds0dT,sSUM,dsdTSUM;
   float x1,x2,vd2MdT2,vdMdT1,vdMdT2,vd2MdT21,vd2MdT22,vM,dvMduoH;
   float eta[7];
   float W[7];
   int   i,g;

   eta[1]=0.1252334085;
   eta[2]=0.3678314990;
   eta[3]=0.5873179543;
   eta[4]=0.7699026742;
   eta[5]=0.9041172564;
   eta[6]=0.9815606342;
   W[1]=0.2491470458;
   W[2]=0.2334925365;
   W[3]=0.2031674267;
   W[4]=0.1600783285;
   W[5]=0.1069393260;
   W[6]=0.0471753364;
   C0= 13.67388+2.718523*T+0.1278728*T*T-0.0025072373*T*T*T;
   C1= -2.890799+0.9144735*T0.073461875*T*T+0.0011088194*T*T*T+0.0000098684995*T*T*T*T;
   C2= 3.166695-0.6364962*T+0.048334930*T*T-0.0010940200*T*T*T+0.0000043787923*T*T*T*T;
   C3= -0.3461423+0.090944469*T-0.0069979527*T*T+0.00013868879*T*T*T;
```

```
C4= 0.013560383-0.0041968897*T+0.00033840450*T*T-0.0000068436516*T*T*T;
vM= uoH/(C0+uoH*C1+uoH*uoH*C2+uoH*uoH*uoH*C3+uoH*uoH*uoH*uoH*C4);
dvMduoH= (C0-C2*(uoH*uoH)-2*C3*(uoH*uoH*uoH)
          -3*C4*(uoH*uoH*uoH*uoH))/((C0+C1*uoH+C2*(uoH*uoH)
         +C3*(uoH*uoH*uoH)+C4*(uoH*uoH*uoH*uoH))*(C0+C1*uoH+C2*(uoH*uoH)
         +C3*(uoH*uoH*uoH)+C4*(uoH*uoH*uoH*uoH)));
dC0dT= 2.718523+2*0.1278728*T-3*0.0025072373*T*T;
dC1dT= 0.9144735-2*0.073461875*T+3*0.0011088194*T*T+4*0.0000098684995*T*T*T;
dC2dT= -0.6364962+2*0.04833493*T-3*0.0010840200*T*T+4*0.0000043787923*T*T*T;
dC3dT= 0.090944469-2*0.0069979527*T+3*0.00013868879*T*T;
dC4dT= -0.0041968897+2*0.00033840450*T-3*0.0000068436516*T*T;
d2C0dT2= 2*0.12787728-6*0.0025072373*T;
d2C1dT2= -2*0.073461875+6*0.0011088194*T+12*0.0000098684995*T*T;
d2C2dT2= 2*0.048334930-6*0.0010840200*T+12*0.0000043787923*T*T;
d2C3dT2= -2*0.0069979527+6*0.00013868879*T;
d2C4dT2= 2*0.00033840450-6*0.0000068436516*T;
DEN= C0+C1*uoH+C2*uoH*uoH+C3*uoH*uoH*uoH+C4*uoH*uoH*uoH*uoH;
dDENdT= dC0dT+dC1dT*uoH+dC2dT*uoH*uoH+dC3dT*uoH*uoH*uoH+dC4dT*uoH*uoH*uoH*uoH;
d2DENdT2= d2C0dT2+d2C1dT2*uoH+d2C2dT2*uoH*uoH+d2C3dT2*uoH*uoH*uoH
          +d2C4dT2*uoH*uoH*uoH*uoH;
gggprop[3] =  -uoH*dDENdT/(DEN*DEN);
vd2MdT2= uoH*(2*dDENdT*dDENdT-DEN*d2DENdT2)/(DEN*DEN*DEN);
s0= 0.00000023*T*T*T-0.031925/(T*T)+0.01699/(T*T*T)-(0.00000023-0.031925+0.01699);
ds0dT= 0.00000069*T*T+0.06385/(T*T*T)-0.05096/(T*T*T*T);
sSUM= 0.0;
dsdTSUM= 0.0;
for(i=1;i<=6;i++)
      {
      x1= uoH/2*(1+eta[i]);
      x2= uoH/2*(1-eta[i]);
      DEN= C0+C1*x1+C2*x1*x1+C3*x1*x1*x1+C4*x1*x1*x1*x1;
      dDENdT= dC0dT+dC1dT*x1+dC2dT*x1*x1+dC3dT*x1*x1*x1+dC4dT*x1*x1*x1*x1;
      d2DENdT2= d2C0dT2+d2C1dT2*x1+d2C2dT2*x1*x1+d2C3dT2*x1*x1*x1+d2C4dT2*x1*x1*x1*x1;
      vdMdT1= -x1*dDENdT/(DEN*DEN);
      vd2MdT21= x1*(2*dDENdT*dDENdT-DEN*d2DENdT2)/(DEN*DEN*DEN);
      DEN= C0+C1*x2+C2*x2*x2+C3*x2*x2*x2+C4*x2*x2*x2*x2;
      dDENdT= dC0dT+dC1dT*x2+dC2dT*x2*x2+dC3dT*x2*x2*x2+dC4dT*x2*x2*x2*x2;
      d2DENdT2= d2C0dT2+d2C1dT2*x2+d2C2dT2*x2*x2+d2C3dT2*x2*x2*x2+d2C4dT2*x2*x2*x2*x2;
      vdMdT2= -x2*dDENdT/(DEN*DEN);
      vd2MdT22= x2*(2*dDENdT*dDENdT-DEN*d2DENdT2)/(DEN*DEN*DEN);
      sSUM= sSUM+W[i]*(vdMdT1+vdMdT2);
      dsdTSUM= dsdTSUM+W[i]*(vd2MdT21+vd2MdT22);
      }
gggprop[1] = s0+uoH/2*sSUM;
gggprop[2] = ds0dT+uoH/2*dsdTSUM;
}
```

A.2 Program MECH.c

This is the C program MECH.c discussed in Section 2.2. This program simulates an ideal 'mechanical' cycle within a magnetically active regenerative refrigerator. The program requires a warm reservoir temperature (Thigh), a refrigeration temperature (Tlow), a pressure ratio (Prat), a high applied field (uoHhigh), and a low applied field (uoHlow). The program iteratively improves the temperature profile until the user is satisfied that convergence has been reached. At that point, the model outputs the specific refrigeration (Qref) and the mass ratio (mravg). The model also outputs two data files. The file TMCHPRF.dat contains the final temperature profile calculated by the model. The file MCHCYC.dat contains the temperature entropy cycles corresponding to selected nodes along the axis of the regenerator.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
#define TINY 1.0e-20

void ggg(float T,float uoH,float *gggprop);
float *vector(long nl, long nh);
void free_vector(float *v, long nl, long nh);

main()

{
        FILE *f_cyc;
        FILE *f_prof;
        float *gggprop,*T,*dT,*dSg,*mr;
        float cphe,Rhe,Thigh,Tlow,uoHhigh,uoHlow,count,ncount,Qref,Qrefold;
        float mravg,mravgold,dTsum;
        float sg1,sg2,Prat;
        int  i,j,n,m,g,donemech,choice;

        /*generate an array used only to store the results of the GGG property calculations*/
        gggprop=vector(1,3);
        /*perfect gas characteristics of helium*/
        cphe=5.192;
        Rhe=2.077;
        /*number of regenerator segments*/
        n=150;
        ncount=n+0.0;
        /*cycle parameters*/
        printf("enter Thigh\n");
        scanf("%f",&Thigh);
        printf("enter Tlow\n");
        scanf("%f",&Tlow);
        printf("enter uoHhigh\n");
        scanf("%f",&uoHhigh);
        printf("enter uoHlow\n");
```

```
scanf("%f",&uoHlow);
printf("enter pressure ratio\n");
scanf("%f",&Prat);
f_cyc=fopen("MCHCYC.dat","w");
f_prof=fopen("TMCHPRF.dat","w");
/*allocate vectors*/
T=vector(0,n);
dT=vector(1,n);
dSg=vector(1,n);
mr=vector(1,n);
T[0]=Tlow;
/*create an initial temperature profile*/
/*fill in the entropy change vector*/
for(i=1;i<=n;i++)
        {
        count=i+0.0;
        T[i]=Tlow+(Thigh-Tlow)*count/ncount;
        dT[i]=T[i]-T[i-1];
        ggg(T[i],uoHhigh,gggprop);
        sg2=gggprop[1];
        ggg(T[i],uoHlow,gggprop);
        sg1=gggprop[1];
        dSg[i]=sg1-sg2;
        }
donemech=0;
mravgold=0.0;
Qrefold=0.0;
do
        {
        mravg=0.0;
        /*calculate the mass ratio necessary for each node via an energy balance*/
        for(i=1;i<=n;i++)
                {
                mr[i]=dSg[i]*T[i]/(n*cphe*dT[i]);
                mravg=mravg+mr[i];
                }
        /*determine the average mass ratio*/
        mravg=mravg/n;
        dTsum=0.0;
        /*calculate the helium temperature change which each node can accomodate*/
        for(i=1;i<=n;i++)
                {
                dT[i]=dSg[i]*T[i]/(n*mravg*cphe);
                dTsum=dTsum+dT[i];
                }
        for(i=1;i<=n;i++)
                {
                T[i]=T[i-1]+dT[i]*(Thigh-Tlow)/dTsum;
                dT[i]=T[i]-T[i-1];
                ggg(T[i],uoHlow,gggprop);
                sg2=gggprop[1];
                ggg(T[i],uoHhigh,gggprop);
                sg1=gggprop[1];
                dSg[i]=sg2-sg1;
```

```
                }
            Qref=mravg*Tlow*Rhe*log(Prat);
            printf("Qref=%f Qrefold=%f mravg=%f mravgold=%f iterate again? (1)=yes
                (0)=no\n",Qref,Qrefold,mravg,mravgold);
            scanf("%i",&choice);
            if(choice==1)
                {
                Qrefold=Qref;
                mravgold=mravg;
                }
            else
                {
                ggg(T[0],uoHhigh,gggprop);
                fprintf(f_cyc,"%f %f ",T[0],gggprop[1]);
                ggg(T[50],uoHhigh,gggprop);
                fprintf(f_cyc,"%f %f ",T[50],gggprop[1]);
                ggg(T[100],uoHhigh,gggprop);
                fprintf(f_cyc,"%f %f ",T[100],gggprop[1]);
                ggg(T[150],uoHhigh,gggprop);
                fprintf(f_cyc,"%f %f\n",T[150],gggprop[1]);
                ggg(T[0],uoHlow,gggprop);
                fprintf(f_cyc,"%f %f ",T[0],gggprop[1]);
                ggg(T[50],uoHlow,gggprop);
                fprintf(f_cyc,"%f %f ",T[50],gggprop[1]);
                ggg(T[100],uoHlow,gggprop);
                fprintf(f_cyc,"%f %f ",T[100],gggprop[1]);
                ggg(T[150],uoHlow,gggprop);
                fprintf(f_cyc,"%f %f ",T[150],gggprop[1]);
                donemech=1;
                for(i=0;i<=n;i++)
                    {
                    count=i+0.0;
                    fprintf(f_prof,"%f %f\n",count/150,T[150-i]);
                    }
                fclose(f_prof);
                }
            }while(donemech==0);
        /*free all vectors used*/
        free_vector(T,0,n);
        free_vector(dSg,1,n);
        free_vector(dT,1,n);
        free_vector(mr,1,n);
}

float *vector(long nl, long nh)
/* This function allocates a float vector with subscript range v[nl..nh]
The code was adapted from Numerical Recipes*/

{
        float *v;

        v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
        return v-nl+NR_END;
}
```

```
void free_vector(float *v, long nl, long nh)
/*This function frees a float vector allocated with vector()
This code was adapted from Numerical Recipes*/

{
        free((FREE_ARG) (v+nl-NR_END));
}
```

A.3 Function TGSUOH.c

This is the subroutine TGSUOH.c discussed in Section 2-3. This function returns the temperature of GGG at input values of specific entropy (sg) and applied field (uoH).

```
float TgsuoH(float sg, float uoH)

{
         void ggg(float T,float uoH,float *gggprop);
         void free_vector(float *v, long nl, long nh);
         float *vector(long nl, long nh);
         void spline(float x[], float y[], int n, float yp1, float ypn, float y2[]);
         void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);

        float *sgv,*Tg,*Tg2,*gggprop;
        float count,Tres,gg1;
        int i,g,j,n;

        n=70;
        sgv=vector(1,n);
        Tg=vector(1,n);
        Tg2=vector(1,n);
        gggprop=vector(1,3);
        for(i=1;i<=n;i++)
                {
                count=i+0.0;
                Tg[i]=1+24*count/n;
                ggg(Tg[i],uoH,gggprop);
                sgv[i]=gggprop[1];
                 }
        if((sgv[1]>sg)||(sgv[n]<sg))
                {
                printf("ENTROPY OUT OF RANGE OF GGG PROPERTY VECTOR IN TGSUOH.c\n")
                scanf("%i",&g);
                }
        spline(sgv,Tg,n,1e30,1e30,Tg2);
        splint(sgv,Tg,Tg2,n,sg,&Tres);
        free_vector(sgv,1,n);
        free_vector(Tg,1,n);
        free_vector(Tg2,1,n);
        free_vector(gggprop,1,3);
        return Tres;
}
```

A.4 Program MAG.c

This is the C program MAG.c discussed in Section 2.3. This program simulates an ideal 'magnetic' cycle within a magnetically active regenerative refrigerator. This program requires the warm reservoir temperature (Thigh), the refrigeration temperature (Tcold), the high applied field (uoHhigh), and the low applied field (uoHlow). The initial mass ratio is determined by allowing the cold end temperature to reach the refrigeration temperature at the end of the inlet process.

The program returns the value of specific refrigeration associated with a given mass ratio. The degree of temperature profile convergence is also displayed. The user manually adjusts the mass ratio until the required degree of convergence is met. When the iteration process is complete, the program returns the data file MAGCYC.dat which contains the thermodynamic cycle associated with selected nodes.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
#define TINY 1.0e-20

void ggg(float T,float uoH,float *gggprop);
float *vector(long nl, long nh);
void free_vector(float *v, long nl, long nh);
void spline(float x[],float y[],int n,float yp1,float ypn,float y2[]);
void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);
float TgsuoH(float sg, float uoH);

main()

{
        FILE *f_cyc;
        float *gggprop,*dT;
        float cphe,Rhe,Thigh,Tlow,uoHhigh,uoHlow,count,ncount,mcount,Qref,Qrefold;
        float mrc,mrcavg,dThe;
        float sg1,sg2,mrcavgold,T1,T2;
        float *Tc,*Th,dmr,*dsgdTuoH,*dTf;
        int   i,j,n,m,g,choice,donemag;

        /*generate an array used only to store the results of the GGG property calculations*/
        gggprop=vector(1,3);
        /*zeroes and weightings for Gauss-Legendre Quadrature used to generate the GGG properties*/
        /*perfect gas characteristics of helium*/
        cphe=5.192;
        Rhe=2.077;
        /*number of regenerator segments*/
        n=150;
        ncount=n+0.0;
```

```
/*number of cycle steps-for magnetic cycle*/
m=300;
mcount=m+0.0;
/*cycle parameters*/
printf("enter Thigh\n");
scanf("%f",&Thigh);
printf("enter Tlow\n");
scanf("%f",&Tlow);
printf("enter uoHhigh\n");
scanf("%f",&uoHhigh);
printf("enter uoHlow\n");
scanf("%f",&uoHlow);
f_cyc=fopen("magcyc.dat","w");
/*allocate the necessary vectors*/
Tc=vector(0,n);
Th=vector(0,n);
dT=vector(0,n);
dTf=vector(0,n);
dsgdTuoH=vector(0,n);
/*set up the initial temperature limits according to the cycle definition*/
ggg(Tlow,uoHhigh,gggprop);
sg1=gggprop[1];
Tc[0]=TgsuoH(sg1,uoHlow);
ggg(Thigh,uoHhigh,gggprop);
sg2=gggprop[1];
Tc[n]=TgsuoH(sg2,uoHlow);
dThe=(Tc[n]-Tc[0]);
/*setup the the low field temperature profile*/
for(i=1;i<=n;i++)
        {
        count=i+0.0;
        Tc[i]=Tc[i-1]+dThe/ncount;
        }
/*choose an arbitrary differential mass ratio to send down the axis for*/
/*the first iteration*/
dmr=0.000008;
mrc=0.0;
Qref=0.0;
donemag=0;
ggg(Tc[0],uoHlow,gggprop);
fprintf(f_cyc,"%f %f ",TgsuoH(gggprop[1],uoHhigh),gggprop[1]);
ggg(Tc[50],uoHlow,gggprop);
fprintf(f_cyc,"%f %f ",TgsuoH(gggprop[1],uoHhigh),gggprop[1]);
ggg(Tc[100],uoHlow,gggprop);
fprintf(f_cyc,"%f %f ",TgsuoH(gggprop[1],uoHhigh),gggprop[1]);
ggg(Tc[150],uoHlow,gggprop);
fprintf(f_cyc,"%f %f\n",TgsuoH(gggprop[1],uoHhigh),gggprop[1]);
do
        {
        ggg(Tc[0],uoHlow,gggprop);
        fprintf(f_cyc,"%f %f ",Tc[0],gggprop[1]);
        ggg(Tc[50],uoHlow,gggprop);
        fprintf(f_cyc,"%f %f ",Tc[50],gggprop[1]);
        ggg(Tc[100],uoHlow,gggprop);
```

```
        fprintf(f_cyc,"%f %f ",Tc[100],gggprop[1]);
        ggg(Tc[150],uoHlow,gggprop);
        fprintf(f_cyc,"%f %f\n",Tc[150],gggprop[1]);
        /*determine the temperature difference which exists and the constant field specific*/
        /*heat corresponding to the top node*/
        dT[n]=(Thigh-Tc[n]);
        ggg(Tc[n],uoHlow,gggprop);
        dsgdTuoH[n]=gggprop[2];
        /*determine the temperature difference and constant field specific heat for each node*/
        for(i=n-1;i>=0;i--)
                {
                ggg(Tc[i],uoHlow,gggprop);
                dsgdTuoH[i]=gggprop[2];
                dT[i]=(Tc[i+1]-Tc[i]);
                }
        /*use an energy balance to compute the temperature change which occurs as the*/
        /*differential mass ratio is sent down the axis*/
        for(i=0;i<=n;i++)
                {
                dTf[i]=n*cphe*dmr*dT[i]/(Tc[i]*dsgdTuoH[i]);
                }
        /*recalculate the temperature profile based on the calculated temperature changes*/
        for(i=0;i<=n;i++)
                {
                Tc[i]=Tc[i]+dTf[i];
                if(Tc[i]>Thigh) Tc[i]=Thigh;
                }
        /*the differential refrigeration produced by the differential mass ratio is*/
        /*due entirely to the subcooling which occurs when the helium reaches the cold end*/
        Qref=Qref+cphe*dmr*(Tlow-Tc[0]);
        /*all of the refrigeration has been obtained if the cold end of the regenerator*/
        /*has reached the refrigeration temperature*/
        if(Tc[0]>Tlow) donemag=1;
        printf("%f\n",Tc[0]);
        mrc=mrc+dmr;
        }while(donemag==0);
/*the cycle has to close*/
/*using the final temperature profile, determine the high field profile by */
/*simulating an adiabatic magnetization process*/
for(i=0;i<=n;i++)
        {
        ggg(Tc[i],uoHlow,gggprop);
        sg1=gggprop[1];
        Th[i]=TgsuoH(sg1,uoHhigh);
        }
/*divide the process into m timesteps.  During each timestep, we send some fraction of*/
/*the total mass ratio calculated during the low field process*/
dmr=mrc/mcount;
for(j=1;j<=m;j++)
        {
        ggg(Th[0],uoHhigh,gggprop);
        fprintf(f_cyc,"%f %f ",Th[0],gggprop[1]);
        ggg(Th[50],uoHhigh,gggprop);
        fprintf(f_cyc,"%f %f ",Th[50],gggprop[1]);
```

```
        ggg(Th[100],uoHhigh,gggprop);
        fprintf(f_cyc,"%f %f ",Th[100],gggprop[1]);
        ggg(Th[150],uoHhigh,gggprop);
        fprintf(f_cyc,"%f %f\n",Th[150],gggprop[1]);
        /*carry out the same process as before only this time we stop when all of the*/
        /*precalculated helium mass has been sent down the axis*/
        dT[0]=(Th[0]-Tlow);
        ggg(Th[0],uoHhigh,gggprop);
        dsgdTuoH[0]=gggprop[2];
        for(i=1;i<=n;i++)
                {
                ggg(Th[i],uoHhigh,gggprop);
                dsgdTuoH[i]=gggprop[2];
                dT[i]=(Th[i]-Tc[i-1]);
                }
        for(i=0;i<=n;i++)
                {
                dTf[i]=n*cphe*dmr*dT[i]/(Th[i]*dsgdTuoH[i]);
                }
        for(i=0;i<=n;i++)
                {
                Th[i]=Th[i]-dTf[i];
                if(Th[i]<Tlow) Tc[i]=Tlow;
                }
        printf("%i\n",j);
        }
donemag=0;
/*we examine the extreme temperatures, the refrigeration, and the mass ratio*/
/*to determine if the overall cycle has closed*/
/*if the cycle does not close then we adjust the mass ratio and iterate*/
do
        {
        printf("Th[0]=%f Tlow=%f Th[%i]=%f Thigh=%f\n",Th[0],Tlow,n,Th[n],Thigh);
        printf("mr=%f Qref=%f\n",mrc,Qref);
        printf("adjust the mass ratio and iterate? (1)=yes (0)=no\n");
        scanf("%i",&choice);
        if(choice==1)
                {
                /*adjust the mass ratio if necessary*/
                printf("enter new mass ratio\n");
                scanf("%f",&mrc);
                ggg(Tlow,uoHhigh,gggprop);
                sg1=gggprop[1];
                Tc[0]=TgsuoH(sg1,uoHlow);
                ggg(Thigh,uoHhigh,gggprop);
                sg2=gggprop[1];
                Tc[n]=TgsuoH(sg2,uoHlow);
                dThe=(Tc[n]-Tc[0]);
                for(i=1;i<=n;i++)
                        {
                        count=i+0.0;
                        Tc[i]=Tc[i-1]+dThe/ncount;
                        }
                Qref=0.0;
```

```
                dmr=mrc/mcount;
                for(j=1;j<=m;j++)
                        {
                        dT[n]=(Thigh-Tc[n]);
                        ggg(Tc[n],uoHlow,gggprop);
                        dsgdTuoH[n]=gggprop[2];
                        for(i=n-1;i>=0;i--)
                                {
                                ggg(Tc[i],uoHlow,gggprop);
                                dsgdTuoH[i]=gggprop[2];
                                dT[i]=(Tc[i+1]-Tc[i]);
                                }
                        for(i=0;i<=n;i++)
                                {
                                dTf[i]=n*cphe*dmr*dT[i]/(Tc[i]*dsgdTuoH[i]);
                                }
                        for(i=0;i<=n;i++)
                                {
                                Tc[i]=Tc[i]+dTf[i];
                                if(Tc[i]>Thigh) Tc[i]=Thigh;
                                }
                        Qref=Qref+cphe*dmr*(Tlow-Tc[0]);
                        }
                for(i=0;i<=n;i++)
                        {
                        ggg(Tc[i],uoHlow,gggprop);
                        sg1=gggprop[1];
                        Th[i]=TgsuoH(sg1,uoHhigh);
                        }
                for(j=1;j<=m;j++)
                        {
                        dT[0]=(Th[0]-Tlow);
                        ggg(Th[0],uoHhigh,gggprop);
                        dsgdTuoH[0]=gggprop[2];
                        for(i=1;i<=n;i++)
                                {
                                ggg(Th[i],uoHhigh,gggprop);
                                dsgdTuoH[i]=gggprop[2];
                                dT[i]=(Th[i]-Tc[i-1]);
                                }
                        for(i=0;i<=n;i++)
                                {
                                dTf[i]=n*cphe*dmr*dT[i]/(Th[i]*dsgdTuoH[i]);
                                }
                        for(i=0;i<=n;i++)
                                {
                                Th[i]=Th[i]-dTf[i];
                                if(Th[i]<Tlow) Tc[i]=Tlow;
                                }
                        }
                }
        else donemag=1;
        }while(donemag==0);
free_vector(Tc,0,n);
```

```
        free_vector(Th,0,n);
        free_vector(dsgdTuoH,0,n);
        free_vector(dT,0,n);
        free_vector(dTf,0,n);
}

float *vector(long nl, long nh)
/* This function allocates a float vector with subscript range v[nl..nh]
The code was adapted from Numerical Recipes*/

{
        float *v;

        v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
        return v-nl+NR_END;
}


void free_vector(float *v, long nl, long nh)
/*This function frees a float vector allocated with vector()
This code was adapted from Numerical Recipes*/

{
        free((FREE_ARG) (v+nl-NR_END));
}
```

A.5 Program COM.c

This is the C program COM.c discussed in Section 2.4. This program simulates an ideal 'combined' cycle within a magnetically active regenerative refrigerator. This program requires the warm reservoir temperature (Thigh), the refrigeration temperature (Tlow), the pressure ratio (Prat), the high applied field (uoHhigh), the low applied field (uoHlow), and the flow fraction (ff).

The program returns the specific refrigeration corresponding to an input mass ratio (mr). The program also displays the degree of temperature profile convergence. The user may adjust the mass ratio until the desired degree of convergence is obtained. When the iteration process is complete, the program outputs the data file COMCYC.dat which contains the thermodynamic cycles associated with selected nodes.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
#define TINY 1.0e-20

void ggg(float T,float uoH,float *gggprop);
float *vector(long nl, long nh);
void free_vector(float *v, long nl, long nh);
```

```
void spline(float x[],float y[],int n,float yp1,float ypn,float y2[]);
void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);
float TgsuoH(float sg, float uoH);

main()

{
        FILE *f_cyc;
        float  *gggprop,*dT,Tclow,Thhigh;
        float  cphe,Rhe,Thigh,Tlow,uoHhigh,uoHlow,count,ncount,mcount,Qref;
        float  *dsgduoHT,mrc;
        float  sg1,sg2,Prat,uoHexp,uoHcom,T1,T2,uoH,duoH,Thold;
        float  *Tc,*Th,dmr,*dsgdTuoH,dThe,*dTf,ff,Thlow,*Tci;
        int    i,j,n,m,g,choice,donecom2,pr;

        /*generate an array used only to store the results of the GGG property calculations*/
        gggprop=vector(1,3);
        /*perfect gas characteristics of helium*/
        cphe=5.192;
        Rhe=2.077;
        /*number of regenerator segments*/
        n=150;
        ncount=n+0.0;
        /*number of cycle steps-for magnetic cycle*/
        m=300;
        mcount=m+0.0;
        /*cycle parameters*/
        printf("Enter Thigh\n");
        scanf("%f",&Thigh);
        printf("Enter Tlow\n");
        scanf("%f",&Tlow);
        printf("Enter uoHhigh\n");
        scanf("%f",&uoHhigh);
        printf("Enter uoHlow\n");
        scanf("%f",&uoHlow);
        printf("Enter pressure ratio\n");
        scanf("%f",&Prat);
        f_cyc=fopen("COMCYC.dat","w");
        Th=vector(0,n);
        Tc=vector(0,n);
        Tci=vector(0,n);
        dT=vector(0,n);
        dTf=vector(0,n);
        dsgduoHT=vector(0,n);
        dsgdTuoH=vector(0,n);
        /*prompt for the fraction of the total applied field swing to be done during the*/
        /*flow portions of the cycle*/
        printf("Enter the flow field fraction\n");
        scanf("%f",&ff);
        /*calculate the applied field intermediate steps based on the flow fraction*/
        uoHcom=uoHlow+(uoHhigh-uoHlow)*ff;
        uoHexp=uoHhigh-(uoHhigh-uoHlow)*ff;
        ggg(Tlow,uoHhigh,gggprop);
```

```
sg1=gggprop[1];
Tc[0]=TgsuoH(sg1,uoHcom);
ggg(Thigh,uoHhigh,gggprop);
sg2=gggprop[1];
Tc[n]=TgsuoH(sg2,uoHcom);
/*record the initial temperature profile for future iterations*/
Tci[0]=Tc[0];
for(i=1;i<=n;i++)
        {
        count=i+0.0;
        Tc[i]=Tc[0]+(Tc[n]-Tc[0])*count/ncount;
        Tci[i]=Tc[i];
        }
        Thold=0.0;
donecom2=0;
do
        {
        /*either change the mass ratio and iterate again or allow the*/
        /*model to cycle through once more*/
        printf("change mass ratio (1) iterate cycle again (2)\n");
        scanf("%i",&g);
        if(g==1)
                {
                for(i=0;i<=n;i++) Tc[i]=Tci[i];
                printf("enter mass ratio\n");
                scanf("%f",&mrc);
                }
        else
                {
                for(i=0;i<=n;i++)
                        {
                        ggg(Th[i],uoHhigh,gggprop);
                        sg1=gggprop[1];
                        Tc[i]=TgsuoH(sg1,uoHcom);
                        }
                }
        printf("print cycles (1) yes (0) no\n");
        scanf("%i",&pr);
        dmr=mrc/mcount;
        uoH=uoHcom;
        duoH=(uoHlow-uoHcom)/mcount;
        Qref=0.0;
        for(j=1;j<=m;j++)
                {
                dT[n]=(Thigh-Tc[n]);
                ggg(Tc[n],uoH,gggprop);
                dsgdTuoH[n]=gggprop[2];
                dsgduoHT[n]=gggprop[3];
                /*determine the temperature difference and entropy derivatives for each node*/
                for(i=n-1;i>=0;i--)
                        {
                        ggg(Tc[i],uoH,gggprop);
                        dsgdTuoH[i]=gggprop[2];
                        dsgduoHT[i]=gggprop[3];
```

```
                dT[i]=(Tc[i+1]-Tc[i]);
                }
        /*use an energy balance to compute the temperature change as each*/
        /*differential mass ratio is sent down the axis*/
        for(i=0;i<=n;i++)
                {
                dTf[i]= dT[i]*cphe*n*dmr/(Tc[i]*dsgdTuoH[i]);
                dTf[i]=dTf[i]-(dsgduoHT[i]*duoH)/dsgdTuoH[i];
                }
        for(i=0;i<=n;i++)
                {
                Tc[i]=Tc[i]+dTf[i];
                if(Tc[i]>Thigh) Tc[i]=Thigh;
                }
        if(pr==1)
        {
        ggg(Tc[0],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Tc[0],gggprop[1]);
        ggg(Tc[50],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Tc[50],gggprop[1]);
        ggg(Tc[100],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Tc[100],gggprop[1]);
        ggg(Tc[150],uoH,gggprop);
        fprintf(f_cyc,"%f %f\n",Tc[150],gggprop[1]);
        }
        uoH=uoH-(uoHcom-uoHlow)/mcount;
        /*each subcooled differential helium mass adds to the*/
        /*magnetic portion of the refrigeration*/
        Qref=Qref+dmr*cphe*(Tlow-Tc[0]);
        printf("%i\n",j);
        }
/*adiabatic magnetization process*/
for(i=0;i<=n;i++)
        {
        ggg(Tc[i],uoHlow,gggprop);
        sg1=gggprop[1];
        Th[i]=TgsuoH(sg1,uoHexp);
        printf("Tc[%i]=%f Th[%i]=%f\n",i,Tc[i],i,Th[i]);
        }
uoH=uoHexp;
duoH=(uoHhigh-uoHexp)/mcount;
for(j=1;j<=m;j++)
        {
        if(pr==1)
        {
        ggg(Th[0],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Th[0],gggprop[1]);
        ggg(Th[50],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Th[50],gggprop[1]);
        ggg(Th[100],uoH,gggprop);
        fprintf(f_cyc,"%f %f ",Th[100],gggprop[1]);
        ggg(Th[150],uoH,gggprop);
        fprintf(f_cyc,"%f %f\n",Th[150],gggprop[1]);
        }
```

```
                        dT[0]=(Th[0]-Tlow);
                        ggg(Th[0],uoH,gggprop);
                        dsgdTuoH[0]=gggprop[2];
                        dsgduoHT[0]=gggprop[3];
                        /*determine the temperature difference and entropy derivatives for each node*/
                        for(i=1;i<=n;i++)
                                {
                                ggg(Th[i],uoH,gggprop);
                                dsgdTuoH[i]=gggprop[2];
                                dsgduoHT[i]=gggprop[3];
                                dT[i]=(Th[i]-Th[i-1]);
                                }
                        /*use an energy balance to compute the temperature change as each*/
                        /*differential mass ratio is sent down the axis*/
                        for(i=0;i<=n;i++)
                                {
                                dTf[i]= -dT[i]*cphe*n*dmr/(Th[i]*dsgdTuoH[i]);
                                dTf[i]=dTf[i]-(dsgduoHT[i]*duoH)/dsgdTuoH[i];
                                }
                        for(i=0;i<=n;i++)
                                {
                                Th[i]=Th[i]+dTf[i];
                                if(Th[i]<Tlow) Th[i]=Tlow;
                                }
                        uoH=uoH+(uoHhigh-uoHexp)/mcount;
                        printf("%i\n",j);
                        }
                Qref=Qref+Tlow*mrc*Rhe*log(Prat);
                /*determine whether the cycle has closed and iterate again if not*/
                printf("Qref=%f mr=%f Th[0]=%f Tlow=%f Th[n]=%f
                        Thigh=%f\n",Qref,mrc,Th[0],Tlow,Th[n],Thigh);
                printf("Th[50]=%f Th[50]old=%f\n",Th[50],Thold);
                Thold=Th[50];
                printf("iterate again (1)=yes (0)=no\n");
                scanf("%i",&g);
                if(g==0) donecom2=1;
                }while(donecom2==0);
        free_vector(Th,0,n);
        free_vector(Tc,0,n);
        free_vector(dT,0,n);
        free_vector(dTf,0,n);
        free_vector(dsgduoHT,0,n);
        free_vector(dsgdTuoH,0,n);
        free_vector(Tci,0,n);
        fclose(f_cyc);
}

float *vector(long nl, long nh)
/* This function allocates a float vector with subscript range v[nl..nh]
The code was adapted from Numerical Recipes*/

{
        float *v;
```

```
        v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
        return v-nl+NR_END;
}


void free_vector(float *v, long nl, long nh)
/*This function frees a float vector allocated with vector()
This code was adapted from Numerical Recipes*/

{
        free((FREE_ARG) (v+nl-NR_END));
}
```

A.6 Matrix Solution Subroutines

This section lists the code used to solve the linear system of equations generated by the first law equations discussed in Subsection 3.2.3. This section is divided into two subsections. In the first subsection, the subroutine used to decompose the equation matrix into its upper and lower triangular components is listed. In the second subsection, the subroutine used to solve the system of equations with an arbitrary vector of constants is listed.

A.6.1 Subroutine LUDCMP.c

This subsection contains the C subroutine LUDCMP.c. This subroutine is given the matrix corresponding to the set of linear equations (a) and the dimension of the matrix (n). The subroutine returns upper and lower triangular decomposition of the input matrix (these matrices are returned in the input array a) and the row permutation effected by the partial pivoting (indx). This subroutine was adapted from *Numerical Recipes in C, Second Edition.*

```
void ludcmp(float **a, int n, int *indx)

{
        void free_vector(float *v, long nl, long nh);
        float *vector(long nl, long nh);
        int   i,imax,j,k;
        float big,dum,sum,temp;
        float *vv;

        vv=vector(1,n);
        for (i=1;i<=n;i++)
                {
                big=0.0;
                for (j=1;j<=n;j++)
                if ((temp=fabs(a[i][j])) > big) big=temp;
                vv[i]=1.0/big;
                }
        for (j=1;j<=n;j++)
```

```
        {
        for (i=1;i<j;i++)
                {
                sum=a[i][j];
                for (k=1;k<i;k++) sum -= a[i][k]*a[k][j];
                a[i][j]=sum;
                }
        big=0.0;
        for (i=j;i<=n;i++)
                {
                sum=a[i][j];
                for (k=1;k<j;k++) sum -= a[i][k]*a[k][j];
                a[i][j]=sum;
                if ( (dum=vv[i]*fabs(sum)) >= big)
                        {
                        big=dum;
                        imax=i;
                        }
                }
        if (j != imax)
                {
                for (k=1;k<=n;k++)
                        {
                        dum=a[imax][k];
                        a[imax][k]=a[j][k];
                        a[j][k]=dum;
                        }
                vv[imax]=vv[j];
                }
        indx[j]=imax;
        if (a[j][j] == 0.0) a[j][j]=TINY;
        if (j != n)
                {
                dum=1.0/(a[j][j]);
                for (i=j+1;i<=n;i++) a[i][j] *= dum;
                }
        }
    free_vector(vv,1,n);
}
```

A.6.2 Subroutine LUBKSB.c

This subsection contains the C subroutine LUBKSB.c. This subroutine is given the lower and upper triangular decomposition of a matrix corresponding to a set of linear equations (both are contained in a), the dimension of the system (n), the row permutations used in the decomposition (indx), and the right hand side vector (b). The subroutine returns the solution in the input vector (b).

```
void lubksb(float **a, int n, int *indx, float b[])

{
    int   i,ii=0,ip,j;
```

```
        float sum;

        for (i=1;i<=n;i++)
                {
                ip=indx[i];
                sum=b[ip];
                b[ip]=b[i];
                if (ii) for (j=ii;j<=i-1;j++) sum -= a[i][j]*b[j];
                else if (sum) ii=i;
                b[i]=sum;
                }
        for (i=n;i>=1;i--)
                {
                sum=b[i];
                for (j=i+1;j<=n;j++) sum -= a[i][j]*b[j];
                b[i]=sum/a[i][i];
        }
}
```

A.7 Program REGEN.c

This section lists the program REGEN.c discussed in Chapter 3. This program is used to simulate a magnetically active regenerative refrigerator operating under nonideal conditions.

This program requires a large number of input parameters. These input parameters can be divided into three groups. The first group are parameters that characterize the refrigeration cycle. The second group are parameters which characterize the geometry and construction of the refrigeration device. The last group are computational parameters which specify how the device is simulated numerically. These parameters are listed in Tables A-1 through A-3.

The computational model runs for a number of cycles specified by the user. The model outputs four raw data files. These data files contain the helium temperature (th.dat), GGG temperature (tg.dat), helium mass flow (mf.dat), and helium pressure (p.dat) at each node and time step. The model also outputs ten interpreted data files. These files include the temperature and pressure gradient generated irreversibility (Sirrht.dat and Sirrp.dat), the entropy and enthalpy fluxes (hflx.dat and sflx.dat), the entropy change of the helium and GGG (dmsh.dat and dmsg.dat), the helium enthalpy (hh.dat), the helium entropy (sh.dat), and the GGG entropy (sg.dat) at every axial node and timestep. Finally, the model outputs a file containing the cycle characteristics calculated for each cycle (cycchar.dat) and a tag file containing the input parameters and the run name (input.txt).

Parameter	Variable	Typical Value	Units
compression process time	tcom	0.5	seconds
inlet process time	tin	0.5	seconds
expansion process time	texp	0.5	seconds
outlet process time	tout	0.5	seconds
high applied field value	uoHhigh	5	Tesla
applied field after compression process	uoHcom	2	Tesla
low applied field value	uoHlow	1	Tesla
applied field after expansion process	uoHexp	4	Tesla
high imposed pressure	Phigh	532.1	kPa
low imposed pressure	Plow	304	kPa
displacer stroke	stroke	0.03	m

Table A-1. Refrigeration Cycle Operating Parameters

Parameter	Variable	Typical Value	Units
regenerator length	length	0.121	meters
volume of cold end heat exchanger	Vexch	0.0	cubic meters
porosity of regenerator matrix	e	0.35	-
cross sectional area of regenerator	Areg	0.00096	meters squared
packed bed particle diameter	dp	0.00015	meters
area of displacer	Aexp	0.00096	meters squared
warm reservoir temperature	Thigh	12	K
cold reservoir temperature	Tcold	4.75	K

Table A-2. Refrigeration Device Design Parameters

Parameter	Variable	Typical Value	Units
number of axial nodes	nlstep	25	-
number of compression time nodes	ntstepcom	25	-
number of inlet time nodes	ntstepin	25	-
number of expansion time nodes	ntstepexp	25	-
number of outlet time nodes	ntstepout	25	-
helium nodal temperature weighting for heat transfer temperature characterization	wh	0.8	-
helium time history weighting for heat transfer temperature characterization	whts	0.1	-
helium axial history weighting for heat transfer temperature characterization	whas	0.1	-
GGG nodal temperature weighting for heat transfer temperature characterization	wg	0.9	-
GGG time history weighting for heat transfer temperature characterization	wgts	0.1	-
helium temperature weighting for axial step iteration - axial history	w1ha	0.9	-
helium temperature weighting for axial step iteration - second law results	w2h	0.1	-
helium temperature weighting for axial step iteration - time history	w1ht	0.9	-

cold end mass flow discrepancy correction weighting for downflow timestep iteration	wmf	0.5	-
warm end pressure discrepancy correction weighting for upflow timestep iteration	wPL	0.95	-
tolerance for convergence between first and second law results during axial step	dTas	0.05	K
tolerance for convergence to mass flow boundary condition during upflow timestep	dmf	0.02	g/s
tolerance for convergence to pressure boundary condition during downflow timestep	dP0	0.05	kPa

Table A-3. Computational Input Parameters

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
#define TINY 1.0e-20

void ggg(float T,float uoH,float *gggprop);
float *vector(long nl, long nh);
int *ivector(long nl, long nh);
float **matrix(long nrl, long nrh, long ncl, long nch);
void free_vector(float *v, long nl, long nh);
void free_ivector(int *v, long nl, long nh);
void free_matrix(float **m, long nrl, long nrh, long ncl, long nch);
void spline(float x[],float y[],int n,float yp1,float ypn,float y2[]);
void splie2(float x1a[], float x2a[], float **ya, int m, int n, float **y2a);
void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);
void splin2(float x1a[], float x2a[], float **ya, float **y2a, int m, int n, float x1, float x2, float *y);
float TgsuoH(float sg, float uoH);
float ThsP(float **entropy,float **entropy2,int m_prop,int n_prop,float *x1_prop,float *x2_prop,float sh,float P);
float shhP(float **entropy,float **entropy2,float **enthalpy,float **enthalpy2,int m_prop,int n_prop,float
*x1_prop,float *x2_prop,float hh,float P);
void ludcmp(float **a, int n, int *indx);
void lubksb(float **a, int n, int *indx, float b[]);

main()

{
        FILE *f_prop;
        FILE *f_iprof;
        FILE *f_th;
        FILE *f_tg;
        FILE *f_mf;
        FILE *f_p;
        FILE *f_sh;
        FILE *f_hh;
        FILE *f_sg;
        FILE *f_sflx;
        FILE *f_hflx;
```

```
FILE *f_dmsh;
FILE *f_Sirrht;
FILE *f_Sirrp;
FILE *f_dmsg;
FILE *f_char;
FILE *f_input;
float pr,mdismax,mr,hydar,hflxL,hflx0,sflxL,sflx0,hflxLiso,hflx0iso,sflxLiso,sflx0iso;
float ref,magref,mechref,irrchx,irrreg,irrwhx,PdV,mregmin,mregmax,magwork,PLt,wPL;
float *gggprop,hhp,rhohp;
float *x1_prop,*x2_prop;
float **density,**viscosity,**conductivity,**enthalpy,**entropy;
float **density2,**viscosity2,**conductivity2,**enthalpy2,**entropy2;
float gg,gg1,gg2,d0,s0,k0,h0,mu0;
int  g,m_prop,n_prop,itP0,itP0max,doneP0;
int nlstep,ntstepcom,ntstepin,ntstepexp,ntstepout,ncycle,tstep,lstep,cycle;
float tcom,tin,texp,tout;
float uoHhigh,uoHcom,uoHlow,uoHexp;
float Phigh,Plow,Thigh,Tlow;
float length,Vexch,stroke,e,Areg,dp,Aexp;
float cycletime,dx,dtcom,dtin,dtexp,dtout,dt,aa,rhoGGG,Mggg;
float *Tht,*Th,*Pt,*P,*Tgt,*Tg,*mf,shap;
float *Sirrht,*Sirrp,*sflx,*hflx,*sht,*sh,*sgt,*sg,*hh,*mht,*mh,*uh,*uht;
float **A,*B,count,ncount,phi,w1ha,w1ht,w2h,w1g,w2g;
int *indx;
int i,j,cond,done1,itmf,donemf,itas,doneas,show,itasmax,itmfmax;
float Thts,Thas,Th1,Th1a,Th1t,Th2,Tgts,Tg1,Tg2,Pts,Pas,P1,mfas,mf1,Thht,Tght;
float hh1,sh1,sg1,rhoh1,muh1,kh1,cph1,uh1,hh2,rhoh2,hh1a,hh1t,cph1a,cph1t;
float rhohas,hhas,uhts,shas;
float drhohdTP1,drhohdPT1,dhhdPT1,dvhdTP1,dvhdPT1,duhdPT1,duhdTP1;
float dsgdTuoH1,dsgduoHT1,drhohdPTL,rhohL;
float Redp,Nudp,htc,Vo,dPdx,Pr,dPdx1,dPdx2,m,n,dPdt,dPdtL;
float mfin,P0,uoH,Vexp,mfout;
float dPdt0,duoHdt,dVexpdt,hhLiso,hh0iso,shLiso,sh0iso;
float wh,whts,whas,wg,wgts,w1,w2,dP0con,dP0conold;
float Sirrht1,Sirrp1,sh2,sg2,q,Th2old,Tg2old,PL,PLold,PLtemp;
float mfinold,wmf,ci,pf,mfintemp,dmfconold,dTas,dT2,dmf,dmfcon,mreg,dP0;
char R[81],date[81];

/*generate an array used only to store the results of the GGG property calculations*/
gggprop=vector(1,3);
/*generate the temperature coordinate (x1) and pressure coordinate (x2) arrays for the helium
  property array interpolation scheme*/
x1_prop=vector(1,71);
x2_prop=vector(1,20);
m_prop=71;
n_prop=20;
for(i=1;i<=20;i++) x2_prop[i]=202.7+71*(i-1);
for(i=1;i<=71;i++) x1_prop[i]=3.0+0.2*(i-1);
/*create the arrays to hold the helium properties*/
density=matrix(1,71,1,20);
viscosity=matrix(1,71,1,20);
conductivity=matrix(1,71,1,20);
enthalpy=matrix(1,71,1,20);
entropy=matrix(1,71,1,20);
```

```
/*create the arrays to hold the helium property second derivatives*/
density2=matrix(1,71,1,20);
viscosity2=matrix(1,71,1,20);
conductivity2=matrix(1,71,1,20);
enthalpy2=matrix(1,71,1,20);
entropy2=matrix(1,71,1,20);
/*set up arrays and matrices to use in linear energy equation solution*/
A=matrix(1,13,1,13);
B=vector(1,13);
indx=ivector(1,13);
/*fill up the helium property tables from the data file*/
f_prop=fopen("heprop2","r");
rewind(f_prop);
for(j=1;j<=n_prop;j++)
        {
        for(i=1;i<=m_prop;i++)
                {
                fscanf(f_prop,"%f %f %f %f %f %f %f %f %f %f", &gg, &gg, &d0, &s0, &gg, &k0,
              &h0, &gg, &gg, &mu0);
                density[i][j]=d0*4.00192;
                entropy[i][j]=s0/4.00192;
                enthalpy[i][j]=h0/4.00192;
                conductivity[i][j]=k0/1000;
                viscosity[i][j]=mu0/1000;
                }
        }
fclose(f_prop);
/*calculate the second derivatives in the temperature coordinate for each property table*/
splie2(x1_prop,x2_prop,density,m_prop,n_prop,density2);
splie2(x1_prop,x2_prop,entropy,m_prop,n_prop,entropy2);
splie2(x1_prop,x2_prop,enthalpy,m_prop,n_prop,enthalpy2);
splie2(x1_prop,x2_prop,conductivity,m_prop,n_prop,conductivity2);
splie2(x1_prop,x2_prop,viscosity,m_prop,n_prop,viscosity2);
/*set up the grid size*/
nlstep=15;
ntstepcom=15;
ntstepin=15;
ntstepexp=15;
ntstepout=15;
ncycle=3;
/*set up the computational parameters*/
wh=0.8;
whas=0.1;
whts=0.1;
wg=0.9;
wgts=0.1;
w1ha=0.8;
w1ht=0.1;
w2h=0.1;
w1g=0.9;
w2g=0.1;
wmf=0.5;
dTas=0.05;
itasmax=500;
```

```
phi=0.55;
dT2=0.0001;
itmfmax=500;
dmf=0.02;
dP0=0.05;
itP0max=500;
wPL=0.95;
/*assign default parameters*/
tcom=0.5;
tin=0.5;
texp=0.5;
tout=0.5;
uoHhigh=5;
uoHcom=2;
uoHlow=1;
uoHexp=4;
Phigh=532.10;
Plow=304.050;
Thigh=12;
Tlow=4.75;
length=0.121;
Vexch=0.0;
stroke=0.03;
e=0.35;
Areg=0.00096;
dp=0.00015;
Aexp=0.00096;
/*Allow User to Alter Input Parameters*/
done1=0;
do
        {
        printf("        CONDITIONS\n");
        printf("Cycle time conditions per stroke:\n");
        printf(" compression: %f s  inlet: %f s\n",tcom,tin);
        printf(" expansion: %f s   outlet: %f s\n",texp,tout);
        printf("Applied field profile:\n");
        printf(" high applied field: %f T  ",uoHhigh);
        printf(" applied field after compression: %f T\n",uoHcom);
        printf(" low applied field: %f T  ",uoHlow);
        printf(" applied field after expansion: %f T\n",uoHexp);
        printf("Heat exchanger temperatures:\n");
        printf(" Thigh: %f K  ",Thigh);
        printf(" Tlow: %f K\n",Tlow);
        printf("Pressure profile:\n");
        printf(" high pressure: %f kPa  ",Phigh);
        printf(" low pressure: %f kPa\n",Plow);
        printf("Geometry:\n");
        printf(" length:  %f m  regenerator area:  %f  m^2\n",length,Areg);
        printf(" porosity: %f   displacer area:   %f m^2\n",e,Aexp);
        printf(" stroke:  %f m  hydraulic diameter: %f m\n",stroke,dp);
        printf(" volume of cold end heat exchanger: %f m^3\n",Vexch);
        printf("Mesh conditions\n");
        printf(" # axial steps=%i   # cycles=%i\n",nlstep,ncycle);
        printf(" # comp time steps=%i # inlet time steps=%i\n",ntstepcom,ntstepin);
```

```
printf("  # exp time steps=%i  # outlet time steps=%i\n",ntstepexp,ntstepout);
printf("(1) Use Conditions  ");
printf("(2) Change cycle time conditions\n");
printf("(3) Change applied field profile  ");
printf("(4) Change heat exchanger temperatures\n");
printf("(5) Change pressure profile  ");
printf("(6) Change geometry\n");
printf("(7) Change mesh conditions\n");
scanf("%i",&cond);
if(cond==1)
        {
        done1=1;
        }
if (cond==2)
        {
        printf("time of compression (s)\n");
        scanf("%f",&tcom);
        printf("time of inlet stroke (s)\n");
        scanf("%f",&tin);
        printf("time of expansion (s)\n");
        scanf("%f",&texp);
        printf("time of outlet stroke (s)\n");
        scanf("%f",&tout);
        }
if (cond==3)
        {
        printf("highest applied field (T)\n");
        scanf("%f",&uoHhigh);
        printf("applied field after compression (T)\n");
        scanf("%f",&uoHcom);
        printf("lowest applied field (T)\n");
        scanf("%f",uoHlow);
        printf("applied field after expansion (T)\n");
        scanf("%f",&uoHexp);
        }
if (cond==4)
        {
        printf("warm end heat exchanger temperature (K)\n");
        scanf("%f",&Thigh);
        printf("cold end heat exchanger temperature (K)\n");
        scanf("%f",&Tlow);
        }
if (cond==5)
        {
        printf("highest pressure (kPa)\n");
        scanf("%f",&Phigh);
        printf("lowest pressure (kPa)\n");
        scanf("%f",&Plow);
        }
if (cond==6)
        {
        printf("length of regenerator (m)\n");
        scanf("%f",&length);
        printf("volume of cold end heat exchanger (m^3)\n");
```

```
                scanf("%f",&Vexch);
                printf("stroke of displacer (m)\n");
                scanf("%f",&stroke);
                printf("porosity\n");
                scanf("%f",&e);
                printf("area of regenerator (m^2)\n");
                scanf("%f",&Areg);
                printf("hydraulic diameter (m)\n");
                scanf("%f",&dp);
                printf("area of displacer (m^2)\n");
                scanf("%f",&Aexp);
                }
        if (cond==7)
                {
                printf("Number of axial steps\n");
                scanf("%i",&nlstep);
                printf("Number of compression time steps\n");
                scanf("%i",&ntstepcom);
                printf("Number of inlet time steps\n");
                scanf("%i",&ntstepin);
                printf("Number of expansion time steps\n");
                scanf("%i",&ntstepexp);
                printf("Number of outlet time steps\n");
                scanf("%i",&ntstepout);
                printf("Number of cycles\n");
                scanf("%i",&ncycle);
                }
        }while(done1==0);
/*allow the user to select a diagnostic run in which all results are output to
  screen rather than to data file*/
/*assign secondary parameters*/
cycletime=tcom+tin+texp+tout;
dx=length/(nlstep-1);
dtcom=tcom/ntstepcom;
dtin=tin/ntstepin;
dtexp=texp/ntstepexp;
dtout=tout/ntstepout;
rhoGGG=7100;
Mggg=rhoGGG*Areg*(1-e)*1000*dx;
aa=6/dp;
/*create arrays to hold the cycle data*/
Th=vector(1,nlstep);
Tht=vector(1,nlstep);
Tg=vector(1,nlstep);
Tgt=vector(1,nlstep);
P=vector(1,nlstep);
Pt=vector(1,nlstep);
mf=vector(0,nlstep);
/*create arrays to hold the secondary cycle data*/
Sirrht=vector(1,nlstep);
Sirrp=vector(1,nlstep);
sflx=vector(0,nlstep);
hflx=vector(0,nlstep);
sht=vector(1,nlstep);
```

```
sh=vector(1,nlstep);
hh=vector(1,nlstep);
sgt=vector(1,nlstep);
sg=vector(1,nlstep);
mht=vector(1,nlstep);
mh=vector(1,nlstep);
uht=vector(1,nlstep);
uh=vector(1,nlstep);
/*use the contents of the data file iprof to generate the initial temperature and
   pressure profiles which are used as the temporal boundary condition*/
printf("Obtain initial profiles from data file? (1)=yes (0)=no\n");
scanf("%i",&g);
if (g==1)
        {
        f_iprof=fopen("b:iprof.dat","r");
        rewind(f_iprof);
        for(i=1;i<=nlstep;i++) fscanf(f_iprof,"%f ",&Tht[i]);
        for(i=1;i<=nlstep;i++) fscanf(f_iprof,"%f ",&Tgt[i]);
        for(i=1;i<=nlstep;i++) fscanf(f_iprof,"%f ",&Pt[i]);
        fclose(f_iprof);
        }
else
        {
        ncount=nlstep;
        for(i=1;i<=nlstep;i++)
                {
                count=i+0.0;
                Tht[i]=Thigh-(Thigh-Tlow)*(count-1)/(ncount);
                Tgt[i]=Thigh-(Thigh-Tlow)*(count-1)/(ncount);
                Pt[i]=Plow;
                }
        }
/*open output files*/
f_th=fopen("th.dat","w");
f_tg=fopen("tg.dat","w");
f_mf=fopen("mf.dat","w");
f_p=fopen("p.dat","w");
f_sh=fopen("sh.dat","w");
f_hh=fopen("hh.dat","w");
f_sg=fopen("sg.dat","w");
f_sflx=fopen("sflx.dat","w");
f_hflx=fopen("hflx.dat","w");
f_dmsh=fopen("dmsh.dat","w");
f_Sirrht=fopen("Sirrht.dat","w");
f_Sirrp=fopen("Sirrp.dat","w");
f_dmsg=fopen("dmsg.dat","w");
f_input=fopen("input.txt","w");
f_char=fopen("cycchar.dat","w");
/*set up the necessary secondary arrays */
for(i=1;i<=nlstep;i++)
        {
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,Tht[i],Pt[i],&hh1);
        ggg(Tgt[i],uoHhigh,gggprop);
        splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,Tht[i],Pt[i],&sh1);
```

```
                splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,Tht[i],Pt[i],&rhoh1);
                mht[i]=rhoh1*e*Areg*dx*1000;
                uht[i]=hh1-Pt[i]/rhoh1;
                sht[i]=sh1;
                sgt[i]=gggprop[1];
                }
        /*create cycle loop*/
for(cycle=1;cycle<=ncycle;cycle++)
        {
        /*initialize cycle characteristics*/
                hflx0=0.0;
                hflxL=0.0;
                hflx0iso=0.0;
                hflxLiso=0.0;
                sflx0=0.0;
                sflxL=0.0;
                sflx0iso=0.0;
                sflxLiso=0.0;
                PdV=0.0;
                magwork=0.0;
                mdismax=0.0;
                mregmax=0.0;
                mregmin=99999999;
                /*guess an inlet mass flow for the first timestep*/
                 mfin=1.5;
                P0=Plow;
                Vexp=Vexch;
                uoH=uoHhigh;
                /*create downflow process timestep loop*/
                for(tstep=1;tstep<=(ntstepcom+ntstepin);tstep++)
                        {
                        /*Compression Conditions*/
                        if(tstep<=ntstepcom)
                                {
                                dPdt0=(Phigh-Plow)/tcom;
                                duoHdt=(uoHcom-uoHhigh)/tcom;
                                dVexpdt=0.0;
                                dt=dtcom;
                                P0=P0+dPdt0*dtcom;
                                uoH=uoH+duoHdt*dtcom;
                                Vexp=Vexp+dVexpdt*dtcom;
                                }
                        /*Inlet Stroke Conditions*/
                        if((tstep>ntstepcom)&&(tstep<=(ntstepcom+ntstepin)))
                                {
                                dPdt0=0.0;
                                dVexpdt=(stroke*Aexp)/tin;
                                duoHdt=(uoHlow-uoHcom)/tin;
                                dt=dtin;
                                P0=P0+dPdt0*dtin;
                                uoH=uoH+duoHdt*dtin;
                                Vexp=Vexp+dVexpdt*dtin;
                                }
                        /*reset mass flow convergence indicators*/
```

```
dmfconold=0.0;
donemf=0;
itmf=0;
/*timestep iteration loop*/
do
        {
        /*calculate the fluxes entering the regenerator at the warm end*/
        splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,Thigh,P0,&sh1);
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,Thigh,P0,&hh1);
        mf[0]=mfin;
        sflx[0]=mf[0]*sh1*dt;
        hflx[0]=mf[0]*hh1*dt;
        /*using this set of axial boundary conditions, solve to the warm end*/
        for(lstep=1;lstep<=nlstep;lstep++)
                {
                itas=0;
                doneas=0;
                /*setup the nodal axial and temporal history*/
                if(lstep==1)
                        {
                        Thas=Thigh;
                        mfas=mfin;
                        Pas=P0;
                        }
                else
                        {
                        Thas=Th[lstep-1];
                        mfas=mf[lstep-1];
                        Pas=P[lstep-1];
                        }
                Thts=Tht[lstep];
                Tgts=Tgt[lstep];
                Pts=Pt[lstep];
                Th1a=Thas;
                Th1t=Thts;
                Tg1=Tgts;
                P1=Pas;
                /*heat and momentum transfer characteristics are not iterated*/
                splin2(x1_prop,x2_prop,viscosity,viscosity2,m_prop,n_prop,
                        Th1a,P1,&muh1);
                splin2(x1_prop,x2_prop,conductivity,conductivity2,m_prop,n_prop,
                        Th1a,P1,&kh1);
                splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                        Th1a,P1,&rhoh1);
                splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                        Th1a+0.05,P1,&rhohp);
                drhohdTP1=(rhohp-rhoh1)/0.05;
                splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                        Th1a,P1+0.5,&rhohp);
                drhohdPT1=(rhohp-rhoh1)/0.5;
                splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
                        Th1a,P1,&shas);
                rhohas=rhoh1;
                splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
```

```
        Th1a,P1,&hh1);
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a+0.05,P1,&hhp);
cph1=(hhp-hh1)/0.05;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a,P1+0.5,&hhp);
dhhdPT1=(hhp-hh1)/0.5;
hhas=hh1;
Vo=mfas/(Areg*rhoh1*1000);
if (Vo<0) Vo=-Vo;
Redp=(rhoh1*Vo*dp)*1000000/muh1;
Pr=(muh1*cph1)/(10*kh1);
m=pow(Pr,0.33333);
n=pow(Redp,0.5);
if(m*n>50)
        {
        Nudp=2.0+2.0*m*n;
        htc=(kh1*Nudp)/(1000*dp);
        }
else
        {
        Nudp=-4.2+1.5*m*n;
        if(Nudp<=.75) Nudp=0.75;
        htc=(kh1*Nudp)/(1000*dp);
        }
dPdx1=((1-e)*(1-e)*muh1*Vo)/(1000000*e*e*e*e*dp*dp*phi*phi);
dPdx2=(1.75*(1-e)*rhoh1*Vo*Vo)/(e*e*e*e*dp*phi);
if(mfas>0) dPdx= -(dPdx1+dPdx2)/1000;
else dPdx=(dPdx1+dPdx2)/1000;
dPdt=(Pas+dPdx*dx-Pts)/dt;
P1=Pas+dPdx*dx;
/*lengthstep iteration loop*/
do
        {
        /*Energy Considerations*/
        /*helium properties and derivatives*/
        splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                Th1t,P1,&rhoh1);
        splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                Th1t+0.05,P1,&rhohp);
        drhohdTP1=(rhohp-rhoh1)/0.05;
        splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                Th1t,P1+0.5,&rhohp);
        drhohdPT1=(rhohp-rhoh1)/0.5;
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
                Th1a,P1,&hh1a);
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
                Th1a+0.05,P1,&hhp);
        cph1a=(hhp-hh1a)/0.05;
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
                Th1a,P1+0.5,&hhp);
        dhhdPT1=(hhp-hh1a)/0.5;
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
                Th1t,P1,&hh1t);
```

```
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
      Th1t+0.05,P1,&hhp);
cph1t=(hhp-hh1t)/0.05;
uh1=hh1t-P1/rhoh1;
dvhdTP1= -drhohdTP1/(rhoh1*rhoh1);
dvhdPT1= -drhohdPT1/(rhoh1*rhoh1);
duhdTP1=cph1t-P1*dvhdTP1;
duhdPT1= -Th1t*dvhdTP1-P1*dvhdPT1;
/*GGG properties and derivatives*/
ggg(Tg1,uoH,gggprop);
sg1=gggprop[1];
dsgdTuoH1=gggprop[2];
dsgduoHT1=gggprop[3];
for(i=1;i<=13;i++)
        {
        for(j=1;j<=13;j++) A[i][j]=0.0;
        B[i]=0.0;
        }
/*setup the system of linear equations*/
A[1][1]=1.0;
A[1][2]=Areg*e*1000;
A[2][1]=hhas;
A[2][2]=Areg*e*uht[lstep]*1000;
A[2][3]=mfas;
A[2][4]=mht[lstep]/dx;
A[2][5]=1.0;
A[3][5]=1.0;
A[3][6]= -rhoGGG*(1-e)*Areg*Tgts*1000;
A[4][2]=1.0;
A[4][7]= -drhohdTP1;
B[4]=drhohdPT1*dPdt;
A[5][3]=1.0;
A[5][8]= -cph1a;
B[5]=dhhdPT1*dPdx;
A[6][4]=1.0;
A[6][7]= -duhdTP1;
B[6]=duhdPT1*dPdt;
A[7][6]=1.0;
A[7][9]= -dsgdTuoH1;
B[7]=dsgduoHT1*duoHdt;
A[8][5]=1.0;
A[8][10]= -aa*Areg*htc;
A[8][11]= aa*Areg*htc;
A[9][10]=1.0;
A[9][12]= -wh;
B[9]=whts*Thts+whas*Thas;
A[10][11]=1.0;
A[10][13]= -wg;
B[10]=wgts*Tgts;
A[11][12]=1.0;
A[11][8]= -dx;
B[11]=Thas;
A[12][12]=1.0;
A[12][7]= -dt;
```

```
B[12]=Thts;
A[13][13]=1.0;
A[13][9]= -dt;
B[13]=Tgts;
/*decompose A into its lower and upper triangular components*/
ludcmp(A,13,indx);
/*use the decomposition of A to solve the energy balance*/
lubksb(A,13,indx,B);
/*The linear system solution is stored in B*/
Th1=B[12];
Tg1=B[13];
Thht=B[10];
Tght=B[11];
q=B[5];
mf1=mfas+B[1]*dx;
/*Entropy Considerations*/
/*recalculate the nodal temperatures using a pair of second law
  balances*/
shap=shhP(entropy,entropy2,enthalpy,enthalpy2,m_prop,n_prop,
          x1_prop,x2_prop,hhas,P1);
Sirrp1=fabs(mfas*dt*(shap-shas));
Sirrht1=fabs(q*dt*dx*(1/Tght-1/Thht));
sh2=(Sirrp1+sflx[lstep-1]+mht[lstep]*sht[lstep]-q*dt*dx/Thht)
    /(mfas*dt+mht[lstep]);
sg2=sgt[lstep]+(q*dt*dx/Thht+Sirrht1)/Mggg;
Tg2=TgsuoH(sg2,uoH);
Th2=ThsP(entropy,entropy2,m_prop,n_prop,
          x1_prop,x2_prop,sh2,P1);
if(itas>itasmax)
        {
        printf("CONVERGENCE OVER A LENGTH STEP
              HAS FAILED\n");
        break;
        }
/*determine if first and second law convergence has been
  obtained*/
if((((fabs(Th1-Th2)<dTas)&&(fabs(Tg1-Tg2)<dTas))||((fabs(Th2-
  Th2old)<dT2)&&(fabs(Tg2-Tg2old)<dT2)))
        {
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,
              m_prop,n_prop,Th2,P1,&hh2);
        splin2(x1_prop,x2_prop,density,density2,
              m_prop,n_prop,Th2,P1,&rhoh2);
        /*update the parameter matrices*/
        mh[lstep]=rhoh2*e*Areg*dx*1000;
        mf1=mfas+(mht[lstep]-mh[lstep])/dt;
        sflx[lstep]=mf1*sh2*dt;
        hflx[lstep]=mf1*hh2*dt;
        uh[lstep]=(hh2-P1/rhoh2);
        sh[lstep]=sh2;
        hh[lstep]=hh2;
        Sirrp[lstep]=Sirrp1;
        Sirrht[lstep]=Sirrht1;
        Th[lstep]=Th2;
```

```
                    Tg[lstep]=Tg2;
                    P[lstep]=P1;
                    mf[lstep]=mf1;
                    sg[lstep]=sg2;
                    printf("cycle=%i comp or inlet;tstep=%i \n",cycle,tstep);
                    printf("Th[%i]=%f Tg[%i]=%f mf[%i]=%f \n", lstep,
                          Th[lstep], lstep, Tg[lstep], lstep, mf[lstep]);
                    doneas=1;
                    }
               else
                    {
                    /*if convergence has not been obtained then re-iterate
                      using a weighted average of the first and second law
                      temperatures to obtain the properties*/
                    itas=itas+1;
                    Th1a=w1ha*Thas+w2h*Th2;
                    Th1t=w1ht*Thts+w2h*Th2;
                    Tg1=w1g*Tgts+w2g*Tg2;
                    Tg2old=Tg2;
                    Th2old=Th2;
                    }
               }while(doneas==0);
          }
/*set boundary conditions at cold end based on the results of our axial solution*/
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,Tlow,P[nlstep],&rhohL);
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
     Tlow,P[nlstep]+0.5,&rhohp);
drhohdPTL=(rhohp-rhohL)/0.5;
dPdtL=(P[nlstep]-Pt[nlstep])/dt;
/*mass into displacer by continuity*/
mfout=1000*(Vexp*drhohdPTL*dPdtL+rhohL*dVexpdt);
/*calculate the lack of convergence*/
dmfcon=(mf[nlstep]-mfout);
if(itmf>itmfmax)
          {
          printf("MODEL FAILED TO CONVERGE TO COLD END MASS
                FLOW B.C.\n");
          break;
          }
/*determine if convergence has been obtained*/
if(fabs(dmfcon)<dmf)
          {
          printf("Convergence Obtained\n");
          donemf=1;
          mreg=0.0;
          magwork=0.0;
          for(i=1;i<=nlstep;i++)
                    {
                    mreg=mreg+mh[i];
                    magwork=magwork+Mggg*Tg[i]*(sgt[i]-sg[i]);
                    fprintf(f_th,"%f ",Th[i]);
                    fprintf(f_tg,"%f ",Tg[i]);
                    fprintf(f_p,"%f ",P[i]);
                    fprintf(f_hh,"%f ",hh[i]);
```

```
            fprintf(f_sh,"%f ",sh[i]);
            fprintf(f_Sirrp,"%f ",Sirrp[i]);
            fprintf(f_Sirrht,"%f ",Sirrht[i]);
            fprintf(f_sg,"%f ",sg[i]);
            fprintf(f_dmsh,"%f ",sh[i]*mh[i]-sht[i]*mht[i]);
            fprintf(f_dmsg,"%f ",Mggg*(sg[i]-sgt[i]));
            }
    for(i=0;i<=nlstep;i++)
            {
            fprintf(f_sflx,"%f ",sflx[i]);
            fprintf(f_hflx,"%f ",hflx[i]);
            fprintf(f_mf,"%f ",mf[i]);
            }
    fprintf(f_th,"\n");
    fprintf(f_tg,"\n");
    fprintf(f_p,"\n");
    fprintf(f_hh,"\n");
    fprintf(f_sh,"\n");
    fprintf(f_Sirrp,"\n");
    fprintf(f_Sirrht,"\n");
    fprintf(f_sg,"\n");
    fprintf(f_dmsh,"\n");
    fprintf(f_dmsg,"\n");
    fprintf(f_sflx,"\n");
    fprintf(f_hflx,"\n");
    fprintf(f_mf,"\n");
    /*update cycle characteristics*/
    hflx0=hflx0+hflx[0];
    sflx0=sflx0+sflx[0];
    hflxL=hflxL+hflx[nlstep];
    sflxL=sflxL+sflx[nlstep];
    if(mdismax<Vexp*1000*rhohL) mdismax=Vexp*1000*rhohL;
    if(mregmax<mreg) mregmax=mreg;
    if(mregmin>mreg) mregmin=mreg;
    splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
           Tlow,P[nlstep],&hhLiso);
    splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
           Thigh,P0,&hh0iso);
    splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
           Tlow,P[nlstep],&shLiso);
    splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
           Thigh,P0,&sh0iso);
    hflx0iso=hflx0iso+mf[0]*dt*hh0iso;
    hflxLiso=hflxLiso+mf[nlstep]*dt*hhLiso;
    sflx0iso=sflx0iso+mf[0]*dt*sh0iso;
    sflxLiso=sflxLiso+mf[nlstep]*dt*shLiso;
    PdV=PdV+P[nlstep]*dVexpdt*dt;
    for(i=1;i<=nlstep;i++)
            {
            uht[i]=uh[i];
            sht[i]=sh[i];
            sgt[i]=sg[i];
            mht[i]=mh[i];
            Tht[i]=Th[i];
```

```
                                Pt[i]=P[i];
                                Tgt[i]=Tg[i];
                                }
                        }
                else
                        {
                        if((dmfconold/dmfcon)>=0.0)
                                {
                                printf("dmfconold=%f dmfcon=%f no
                                        bracketing\n",dmfconold,dmfcon);
                                itmf=itmf+1;
                                mfinold=mfin;
                                dmfconold=dmfcon;
                                mfin=mfin-wmf*dmfcon;
                                printf("changed: mfin=%f mfinold=%f mf[nlstep]=%f\n",
                                        mfin,mfinold,mf[nlstep]);
                                }
                        else
                                {
                                printf("dmfconold=%f dmfcon=%f bracketing\n",
                                        dmfconold,dmfcon);
                                itmf=itmf+1;
                                ci=fabs(dmfcon)+fabs(dmfconold);
                                pf=fabs(dmfcon)/ci;
                                mfintemp=mfin;
                                mfin=mfin+(mfinold-mfin)*pf;
                                mfinold=mfintemp;
                                dmfconold=dmfcon;
                                printf("changed: ci=%f pf=%f mfin=%f mfinold=%f
                                        mf[nlstep]=%f\n",ci,pf,mfin,mfinold,mf[nlstep]);
                                }
                        }
                }while(donemf==0);
        }
/*create initial guess for cold end pressure*/
PL=P[nlstep]+10;
PLt=P[nlstep];
for(tstep=1;tstep<=(ntstepexp+ntstepout);tstep++)
        {
        /*Expansion Conditions*/
        if(tstep<=ntstepexp)
                {
                dPdt0=(Plow-Phigh)/texp;
                duoHdt=(uoHexp-uoHlow)/texp;
                dVexpdt=0.0;
                dt=dtexp;
                P0=P0+dPdt0*dtexp;
                uoH=uoH+duoHdt*dtexp;
                Vexp=Vexp+dVexpdt*dtexp;
                }
        /*Outlet Stroke Conditions*/
        if((tstep>ntstepexp)&&(tstep<=(ntstepexp+ntstepout)))
                {
                dPdt0=0.0;
```

```
            dVexpdt=-(stroke*Aexp)/texp;
            duoHdt=(uoHhigh-uoHexp)/texp;
            dt=dtexp;
            P0=P0+dPdt0*dtexp;
            uoH=uoH+duoHdt*dtexp;
            Vexp=Vexp+dVexpdt*dtexp;
            }
/*reset pressure convergence indicators*/
dP0conold=0.0;
doneP0=0;
itP0=0;
/*timestep iteration loop*/
do
            {
            /*set boundary conditions at cold end based on the guessed cold end pressure*/
            splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,Tlow,PL,&rhohL);
            splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,Tlow,PL+0.5,&rhohp);
            drhohdPTL=(rhohp-rhohL)/0.5;
            dPdtL=(PL-PLt)/dt;
            mfout= -1000*(Vexp*drhohdPTL*dPdtL+rhohL*dVexpdt);
            /*calculate the fluxes entering the regenerator at the cold end*/
            splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,Tlow,PL,&sh1);
            splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,Tlow,PL,&hh1);
            mf[nlstep]= mfout;
            sflx[nlstep]=mf[nlstep]*sh1*dt;
            hflx[nlstep]=mf[nlstep]*hh1*dt;
            /*using this set of axial boundary conditions, solve to the warm end*/
            for(lstep=nlstep;lstep>=1;lstep--)
                        {
                        /*reset lengthstep convergence indicators*/
                        itas=0;
                        doneas=0;
                        /*setup the nodal axial and temporal history*/
                        if(lstep==nlstep)
                                    {
                                    Thas=Tlow;
                                    mfas=mfout;
                                    Pas=PL;
                                    }
                        else
                                    {
                                    Thas=Th[lstep+1];
                                    mfas=mf[lstep];
                                    Pas=P[lstep+1];
                                    }
                        Thts=Tht[lstep];
                        Tgts=Tgt[lstep];
                        Pts=Pt[lstep];
                        Th1a=Thas;
                        Th1t=Thts;
                        Tg1=Tgts;
                        P1=Pas;
                        /*heat and momentum transfer characteristics are not iterated*/
                        splin2(x1_prop,x2_prop,viscosity,viscosity2,m_prop,n_prop,
```

```
        Th1a,P1,&muh1);
splin2(x1_prop,x2_prop,conductivity,conductivity2,m_prop,n_prop,
        Th1a,P1,&kh1);
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
        Th1a,P1,&rhoh1);
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
        Th1a+0.05,P1,&rhohp);
drhohdTP1=(rhohp-rhoh1)/0.05;
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
        Th1a,P1+0.5,&rhohp);
drhohdPT1=(rhohp-rhoh1)/0.5;
splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
        Th1a,P1,&shas);
rhohas=rhoh1;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a,P1,&hh1);
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a+0.05,P1,&hhp);
cph1=(hhp-hh1)/0.05;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a,P1+0.5,&hhp);
dhhdPT1=(hhp-hh1)/0.5;
hhas=hh1;
Vo=mfas/(Areg*rhoh1*1000);
if (Vo<0) Vo=-Vo;
Redp=(rhoh1*Vo*dp)*1000000/muh1;
Pr=(muh1*cph1)/(10*kh1);
m=pow(Pr,0.33333);
n=pow(Redp,0.5);
if(m*n>50)
        {
        Nudp=2.0+2.0*m*n;
        htc=(kh1*Nudp)/(1000*dp);
        }
else
        {
        Nudp=-4.2+1.5*m*n;
        if(Nudp<=.75) Nudp=0.75;
        htc=(kh1*Nudp)/(1000*dp);
        }
dPdx1=((1-e)*(1-e)*muh1*Vo)/(1000000*e*e*e*dp*dp*phi*phi);
dPdx2=(1.75*(1-e)*rhoh1*Vo*Vo)/(e*e*e*dp*phi);
if(mfas>0) dPdx= -(dPdx1+dPdx2)/1000;
else dPdx=(dPdx1+dPdx2)/1000;
dPdt=(Pas+dPdx*dx-Pts)/dt;
P1=Pas+dPdx*dx;
/*lengthstep iteration loop*/
do
        {
        /*Energy Considerations*/
        /*helium properties and derivatives*/
        splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
                Th1t,P1,&rhoh1);
        splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
```

```
        Th1t+0.05,P1,&rhohp);
drhohdTP1=(rhohp-rhoh1)/0.05;
splin2(x1_prop,x2_prop,density,density2,m_prop,n_prop,
        Th1t,P1+0.5,&rhohp);
drhohdPT1=(rhohp-rhoh1)/0.5;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a,P1,&hh1a);
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a+0.05,P1,&hhp);
cph1a=(hhp-hh1a)/0.05;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1a,P1+0.5,&hhp);
dhhdPT1=(hhp-hh1a)/0.5;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1t,P1,&hh1t);
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
        Th1t+0.05,P1,&hhp);
cph1t=(hhp-hh1t)/0.05;
uh1=hh1t-P1/rhoh1;
dvhdTP1= -drhohdTP1/(rhoh1*rhoh1);
dvhdPT1= -drhohdPT1/(rhoh1*rhoh1);
duhdTP1=cph1t-P1*dvhdTP1;
duhdPT1= -Th1t*dvhdTP1-P1*dvhdPT1;
/*GGG properties and derivatives*/
ggg(Tg1,uoH,gggprop);
sg1=gggprop[1];
dsgdTuoH1=gggprop[2];
dsgduoHT1=gggprop[3];
for(i=1;i<=13;i++)
          {
          for(j=1;j<=13;j++) A[i][j]=0.0;
          B[i]=0.0;
          }
/*setup the system of linear equations*/
A[1][1]=1.0;
A[1][2]=Areg*e*1000;
A[2][1]=hhas;
A[2][2]=Areg*e*uht[lstep]*1000;
A[2][3]=mfas;
A[2][4]=mht[lstep]/dx;
A[2][5]=1.0;
A[3][5]=1.0;
A[3][6]= -rhoGGG*(1-e)*Areg*Tgts*1000;
A[4][2]=1.0;
A[4][7]= -drhohdTP1;
B[4]=drhohdPT1*dPdt;
A[5][3]=1.0;
A[5][8]= -cph1a;
B[5]=dhhdPT1*dPdx;
A[6][4]=1.0;
A[6][7]= -duhdTP1;
B[6]=duhdPT1*dPdt;
A[7][6]=1.0;
A[7][9]= -dsgdTuoH1;
```

```
B[7]=dsgduoHT1*duoHdt;
A[8][5]=1.0;
A[8][10]= -aa*Areg*htc;
A[8][11]= aa*Areg*htc;
A[9][10]=1.0;
A[9][12]= -wh;
B[9]=whts*Thts+whas*Thas;
A[10][11]=1.0;
A[10][13]= -wg;
B[10]=wgts*Tgts;
A[11][12]=1.0;
A[11][8]= -dx;
B[11]=Thas;
A[12][12]=1.0;
A[12][7]= -dt;
B[12]=Thts;
A[13][13]=1.0;
A[13][9]= -dt;
B[13]=Tgts;
/*decompose A into its lower and upper triangular components*/
ludcmp(A,13,indx);
/*use the decomposition of A to solve the energy balance*/
lubksb(A,13,indx,B);
Th1=B[12];
Tg1=B[13];
Thht=B[10];
Tght=B[11];
q=B[5];
mf1=mfas+B[1]*dx;
/*Entropy Considerations*/
shap=shhP(entropy,entropy2,enthalpy,enthalpy2,m_prop,n_prop,
          x1_prop,x2_prop,hhas,P1);
Sirrp1=fabs(mfas*dt*(shap-shas));
Sirrht1=fabs(q*dt*dx*(1/Tght-1/Thht));
sh2=(Sirrp1+sflx[lstep]+mht[lstep]*sht[lstep]-q*dt*dx/Thht)
    /(mfas*dt+mht[lstep]);
sg2=sgt[lstep]+(q*dt*dx/Thht+Sirrht1)/Mggg;
Tg2=TgsuoH(sg2,uoH);
Th2=ThsP(entropy,entropy2,m_prop,n_prop,
         x1_prop,x2_prop,sh2,P1);
/*determine if first and second law convergence has been
  obtained*/
if(itas>itasmax)
        {
        printf("CONVERGENCE HAS FAILED OVER A
                LENGTH STEP\n");
        break;
        }
if(((fabs(Th1-Th2)<dTas)&&(fabs(Tg1-Tg2)<dTas))||((fabs(Th2-
  Th2old)<dT2)&&(fabs(Tg2-Tg2old)<dT2)))
        {
        splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,
               n_prop,Th2,P1,&hh2);
        splin2(x1_prop,x2_prop,density,density2,m_prop,
```

```
                        n_prop,Th2,P1,&rhoh2);
                    /*update the parameter matrices*/
                    mh[lstep]=rhoh2*e*Areg*dx*1000;
                    mf1=mfas+(mht[lstep]-mh[lstep])/dt;
                    sflx[lstep-1]= mf1*sh2*dt;
                    hflx[lstep-1]= mf1*hh2*dt;
                    uh[lstep]=(hh2-P1/rhoh2);
                    sh[lstep]=sh2;
                    hh[lstep]=hh2;
                    Sirrp[lstep]=Sirrp1;
                    Sirrht[lstep]=Sirrht1;
                    Th[lstep]=Th2;
                    Tg[lstep]=Tg2;
                    P[lstep]=P1;
                    mf[lstep-1]=mf1;
                    sg[lstep]=sg2;
                    printf("cycle=%i exp or outlet;tstep=%i \n",cycle,tstep);
                    printf("Th[%i]=%f Tg[%i]=%f mf[%i]=%f\n",
                        lstep,Th[lstep],lstep,Tg[lstep],lstep,mf[lstep]);
                    doneas=1;
                    }
                else
                    {
                    itas=itas+1;
                    Th1a=w1ha*Thas+w2h*Th2;
                    Th1t=w1ht*Thts+w2h*Th2;
                    Tg1=w1g*Tgts+w2g*Tg2;
                    Tg2old=Tg2;
                    Th2old=Th2;
                    }
                }while(doneas==0);
        }
dP0con=(P[1]-P0);
if(itP0>itP0max)
        {
        printf("MODEL FAILED TO CONVERGE TO WARM END
                PRESSURE B.C.\n");
        break;
        }
/*determine if convergence has been obtained*/
if(fabs(dP0con)<dP0)
        {
        printf("Convergence Obtained\n");
        doneP0=1;
        mreg=0.0;
        magwork=0.0;
        for(i=1;i<=nlstep;i++)
                {
                mreg=mreg+mh[i];
                magwork=magwork+Mggg*Tg[i]*(sgt[i]-sg[i]);
                fprintf(f_th,"%f ",Th[i]);
                fprintf(f_tg,"%f ",Tg[i]);
                fprintf(f_p,"%f ",P[i]);
                fprintf(f_hh,"%f ",hh[i]);
```

```
            fprintf(f_sh,"%f ",sh[i]);
            fprintf(f_Sirrp,"%f ",Sirrp[i]);
            fprintf(f_Sirrht,"%f ",Sirrht[i]);
            fprintf(f_sg,"%f ",sg[i]);
            fprintf(f_dmsh,"%f ",sh[i]*mh[i]-sht[i]*mht[i]);
            fprintf(f_dmsg,"%f ",Mggg*(sg[i]-sgt[i]));
            }
for(i=0;i<=nlstep;i++)
            {
            fprintf(f_sflx,"%f ",-sflx[i]);
            fprintf(f_hflx,"%f ",-hflx[i]);
            fprintf(f_mf,"%f ",-mf[i]);
            }
fprintf(f_th,"\n");
fprintf(f_tg,"\n");
fprintf(f_p,"\n");
fprintf(f_hh,"\n");
fprintf(f_sh,"\n");
fprintf(f_Sirrp,"\n");
fprintf(f_Sirrht,"\n");
fprintf(f_sg,"\n");
fprintf(f_dmsh,"\n");
fprintf(f_dmsg,"\n");
fprintf(f_sflx,"\n");
fprintf(f_hflx,"\n");
fprintf(f_mf,"\n");
/*update cycle characteristics*/
hflx0=hflx0-hflx[0];
sflx0=sflx0-sflx[0];
hflxL=hflxL-hflx[nlstep];
sflxL=sflxL-sflx[nlstep];
if(mdismax<Vexp*1000*rhohL) mdismax=Vexp*1000*rhohL;
if(mregmax<mreg) mregmax=mreg;
if(mregmin>mreg) mregmin=mreg;
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
      Tlow,P[nlstep],&hhLiso);
splin2(x1_prop,x2_prop,enthalpy,enthalpy2,m_prop,n_prop,
      Thigh,P0,&hh0iso);
splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
      Tlow,P[nlstep],&shLiso);
splin2(x1_prop,x2_prop,entropy,entropy2,m_prop,n_prop,
      Thigh,P0,&sh0iso);
hflx0iso=hflx0iso-mf[0]*dt*hh0iso;
hflxLiso=hflxLiso-mf[nlstep]*dt*hhLiso;
sflx0iso=sflx0iso-mf[0]*dt*sh0iso;
sflxLiso=sflxLiso-mf[nlstep]*dt*shLiso;
PdV=PdV+P[nlstep]*dVexpdt*dt;
for(i=1;i<=nlstep;i++)
            {
            uht[i]=uh[i];
            sht[i]=sh[i];
            sgt[i]=sg[i];
            mht[i]=mh[i];
            Tht[i]=Th[i];
```

```
                                        Pt[i]=P[i];
                                        Tgt[i]=Tg[i];
                                        }
                                PLt=PL;
                                }
                        else
                                {
                                if((dP0conold/dP0con)>=0.0)
                                        {
                                        printf("dP0conold=%f dP0con=%f no
                                                bracketing\n",dP0conold,dP0con);
                                        itP0=itP0+1;
                                        PLold=PL;
                                        dP0conold=dP0con;
                                        PL=PL-wPL*dP0con;
                                        printf("changed: P0=%f PL=%f PLold=%f
                                                P[1]=%f\n",P0,PL,PLold,P[1]);
                                        }
                                else
                                        {
                                        printf("dP0conold=%f dP0con=%f bracketing\n",
                                                dP0conold,dP0con);
                                        itP0=itP0+1;
                                        ci=fabs(dP0con)+fabs(dP0conold);
                                        pf=fabs(dP0con)/ci;
                                        PLtemp=PL;
                                        PL=PL+(PLold-PL)*pf;
                                        PLold=PLtemp;
                                        dP0conold=dP0con;
                                        printf("changed: P0=%f ci=%f pf=%f PL=%f PLold=%f
                                                P[1]=%f\n",P0,ci,pf,PL,PLold,P[1]);
                                        }
                                }
                        }while(doneP0==0);
                }
        /*print the cycle characteristics*/
        fprintf(f_char,"%f %f %f %f %f %f %f %f %f %f %f %f %f\n",
                hflx0iso,hflx0,hflxL,hflxLiso,sflx0iso,sflx0,sflxL,sflxLiso,PdV,
                magwork,mregmin,mregmax,mdismax);
        }
/*create a file which stores the input parameters - computational, operational, and design -
  which were used to generate this run*/
printf("Model has finished %i cycles of simulation\n",ncycle);
printf("Enter data to generate tag file\n");
printf("Enter Simulation Label\n");
scanf("%s",&R);
fprintf(f_input,"\n\n             Run %s\n\n",R);
printf("Enter Date of Simulation\n");
scanf("%s",&R);
fprintf(f_input,"      date of completion: %s\n\n",R);
fprintf(f_input,"        CONDITIONS\n");
fprintf(f_input,"Cycle time conditions per stroke:\n");
fprintf(f_input,"  compression: %f s  inlet: %f s\n",tcom,tin);
fprintf(f_input,"  expansion: %f s    outlet: %f s\n",texp,tout);
```

```
    fprintf(f_input,"Applied field profile:\n");
    fprintf(f_input,"  high applied field: %f T  ",uoHhigh);
    fprintf(f_input,"  applied field after compression: %f T\n",uoHcom);
    fprintf(f_input,"  low applied field: %f T  ",uoHlow);
    fprintf(f_input,"  applied field after expansion: %f T\n",uoHexp);
    fprintf(f_input,"Heat exchanger temperatures:\n");
    fprintf(f_input,"  Thigh: %f K  ",Thigh);
    fprintf(f_input,"  Tlow: %f K\n",Tlow);
    fprintf(f_input,"Pressure profile:\n");
    fprintf(f_input,"  high pressure: %f kPa  ",Phigh);
    fprintf(f_input,"  low pressure: %f kPa\n",Plow);
fprintf(f_input,"Geometry:\n");
fprintf(f_input,"  length:  %f m  regenerator area:  %f  m^2\n",length,Areg);
fprintf(f_input,"  porosity: %f    displacer area:    %f m^2\n",e,Aexp);
fprintf(f_input,"  stroke:  %f m  hydraulic diameter: %f m\n",stroke,dp);
fprintf(f_input,"  volume of cold end heat exchanger: %f m^3\n",Vexch);
fprintf(f_input,"Mesh conditions\n");
fprintf(f_input,"  # axial steps=%i    # cycles=%i\n",nlstep,ncycle);
fprintf(f_input,"  # comp time steps=%i # inlet time steps=%i\n",ntstepcom,ntstepin);
fprintf(f_input,"  # exp time steps=%i  # outlet time steps=%i\n",ntstepexp,ntstepout);
fprintf(f_input,"Numerical Parameters:\n");
fprintf(f_input,"  mass flow convergence tolerance:%f g/s\n",dmf);
fprintf(f_input,"  pressure convergence tolerance:%f kPa\n",dP0);
fprintf(f_input,"  heat transfer temperature:\n");
fprintf(f_input,"    helium axial step: %f\n",whas);
fprintf(f_input,"    helium time step:  %f\n",whts);
fprintf(f_input,"    helium node:       %f\n",wh);
fprintf(f_input,"    GGG time step:     %f\n",wgts);
fprintf(f_input,"    GGG node:          %f\n",wg);
fprintf(f_input,"  required length step temperature convergence: %f K\n",dTas);
/*close output files*/
fclose(f_th);
fclose(f_tg);
fclose(f_mf);
fclose(f_p);
fclose(f_sh);
fclose(f_hh);
fclose(f_sg);
fclose(f_sflx);
fclose(f_hflx);
fclose(f_dmsh);
fclose(f_Sirrht);
fclose(f_Sirrp);
fclose(f_dmsg);
fclose(f_char);
fclose(f_input);
/*free all arrays and matrices used within the code*/
free_ivector(indx,1,13);
free_vector(B,1,13);
free_vector(x1_prop,1,71);
free_vector(x2_prop,1,20);
free_vector(Th,1,nlstep);
free_vector(Tht,1,nlstep);
free_vector(Tg,1,nlstep);
```

```
        free_vector(Tgt,1,nlstep);
        free_vector(P,1,nlstep);
        free_vector(Pt,1,nlstep);
        free_vector(mf,0,nlstep);
        free_vector(sflx,0,nlstep);
        free_vector(hflx,0,nlstep);
        free_vector(Sirrht,1,nlstep);
        free_vector(Sirrp,1,nlstep);
        free_vector(sht,1,nlstep);
        free_vector(sh,1,nlstep);
        free_vector(sgt,1,nlstep);
        free_vector(sg,1,nlstep);
        free_vector(hh,1,nlstep);
        free_vector(mht,1,nlstep);
        free_vector(mh,1,nlstep);
        free_vector(uht,1,nlstep);
        free_vector(uh,1,nlstep);
        free_matrix(A,1,13,1,13);
        free_matrix(density,1,71,1,20);
        free_matrix(viscosity,1,71,1,20);
        free_matrix(conductivity,1,71,1,20);
        free_matrix(enthalpy,1,71,1,20);
        free_matrix(entropy,1,71,1,20);
        free_matrix(density2,1,71,1,20);
        free_matrix(viscosity2,1,71,1,20);
        free_matrix(conductivity2,1,71,1,20);
        free_matrix(enthalpy2,1,71,1,20);
        free_matrix(entropy2,1,71,1,20);
}
```

Appendix B. Cubic Spline Interpolation

This appendix explains the implementation of the cubic spline interpolation technique. The appendix is divided into five sections. In the first section, the mathematical principles behind cubic spline interpolation are described. The next two sections contain listings of SPLINE.c and SPLINT.c, the two functions required to perform cubic spline interpolation in one dimension. The final two sections contain listings of SPLIE2.c and SPLIN2.c, the two additional functions required to perform cubic spline interpolation in two dimensions (i.e. bicubic spline interpolation).

B.1 Theory of Cubic Spline Interpolation

A cubic spline interpolation in one dimension is given a series of abscissa values (x_i) and the corresponding functional values (y_i). The number of points (x_i, y_i) will be referred to as m. In order for the second derivative to be continuous, the interpolation scheme must be third order or higher. In effect, a third order polynomial is constructed within each interval as shown in Figure B-1.

We begin by assuming that in addition to having a vector of functional values (y_i) and abscissas (x_i), we also have a vector of tabulated second derivatives at each abscissa (y''_i). This gives the four boundary conditions necessary to construct the third order polynomial within each interval. For example, in the interval between x_6 and x_7 in Figure B-1, the third order interpolating polynomial can be written as:

$$y = F_0 + F_1(x - x_6) + F_2(x - x_6)^2 + F_3(x - x_6)^3 \qquad \text{(B.1)}$$

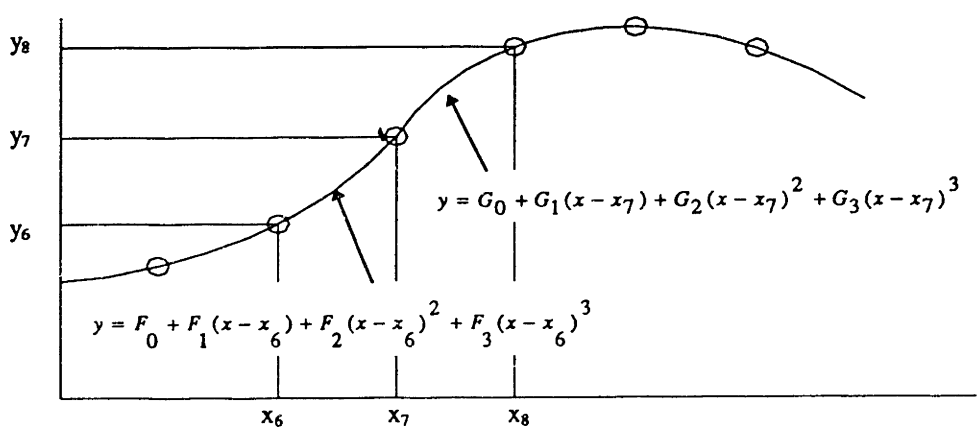

Figure B-1. Cubic Spline Interpolation Development

Four boundary conditions are required in order to calculate the four unknown coefficients F_0, F_1, F_2, and F_3. These boundary conditions are obtained by requiring that the value of y at each abscissa (i.e. x_6 and x_7) match the tabulated value of y (i.e. y_6 and y_7) and that the second derivative of Equation B.1 at the abscissas match the tabulated second derivatives (i.e. y''_6 and y''_7). These boundary conditions are:

$$y_6 = F_0 \tag{B.2}$$

$$y_7 = F_0 + F_1(x_7 - x_6) + F_2(x_7 - x_6)^2 + F_3(x_7 - x_6)^3 \tag{B.3}$$

$$y''_6 = 2 \cdot F_2 \tag{B.4}$$

$$y''_7 = 2 \cdot F_2 + 6 \cdot F_3(x_7 - x_6) \tag{B.5}$$

These four equations can be solved simultaneosly for the equation coefficients in terms of the known values of the abscissas at each side of the interval, the tabulated functional values at each side of the interval, and the tabulated second derivative on each side of the interval.

$$F_0 = y_6 \tag{B.6}$$

$$F_1 = \frac{y_7 - y_6}{x_7 - x_6} - \frac{y''_6(x_7 - x_6)^2}{2} - \frac{(y''_7 - y''_6) \cdot (x_7 - x_6)^2}{6} \tag{B.7}$$

$$F_2 = \frac{y''_6}{2} \tag{B.8}$$

$$F_3 = \frac{(y''_7 - y''_6)}{6(x_7 - x_6)} \tag{B.9}$$

After some algebra, the final interpolating equation can be expressed as[57]:

$$y = \left[\frac{x_7 - x}{x_7 - x_6}\right] y_6 + \left[\frac{x - x_6}{x_7 - x_6}\right] y_7 + \frac{1}{6}\left[\left(\frac{x_7 - x}{x_7 - x_6}\right)^3 - \left(\frac{x_7 - x}{x_7 - x_6}\right)\right](x_7 - x_6)^2 y''_6$$
$$+ \frac{1}{6}\left[\left(\frac{x - x_6}{x_7 - x_6}\right)^3 - \left(\frac{x - x_6}{x_7 - x_6}\right)\right](x_7 - x_6)^2 y''_7 \tag{B.10}$$

Similar interpolating relations exist between each node.

The second derivatives are obtained by using the requirement that the first derivative of the interpolating functions also be continuous. Referring back to Figure B-1, this implies that the first derivative of the equation which applies between x_6 and x_7 and of the equation which applies between x_7 and x_8 must be equal at x_7.

$$F_1 + 2 \cdot F_2(x_7 - x_6) + 3 \cdot F_3(x_7 - x_6)^2 = G_1 \tag{B.11}$$

However, we already have equations for F_1, F_2, and F_3 in terms of abscissas, functional values, and second derivatives at nodes 6 and 7. The same form of interpolating function applies between x_7 and x_8, so G_1 is also a known function. Using these definitions for the coefficients of the interpolating equations leads to a linear equation for the second derivative.

$$\left[\frac{x_7 - x_6}{6}\right] y''_6 + \left[\frac{x_8 - x_6}{3}\right] y''_7 + \left[\frac{x_8 - x_7}{6}\right] y''_8 = \left[\frac{y_8 - y_7}{x_8 - x_7} - \frac{y_7 - y_6}{x_7 - x_6}\right] \qquad \textbf{(B.12)}$$

An equation similar to Equation B.12 can be written for every interior node.

Unfortunately, there is no known interpolating function below node 1 and above node m. Therefore, there are only m-2 equations (for nodes 2 through m-1) for the m second derivatives (y''_1 through y''_m). This problem is typically remedied by specifying the second derivative at the extreme abscissas to be zero (i.e. the 'natural cubic spline'). In general, the first or second derivative at any two additional points may be used to supply the necessary additional boundary conditions.

The set of equations for the second derivatives (e.g. Equation B.12) is a tridiagonal system of linear equations. This system can be solved much more easily than a general system of linear equations.

The subroutine SPLINE.c is used to pre-calculate the tabulated second derivatives of a set of data for later use within the interpolating equations. SPLINE.c is listed in Appendix B.2. The subroutine SPLINT.c performs the actual interpolation. SPLINT.c is listed in Appendix B.3.

The method of cubic spline interpolation can easily be extended to two dimensions. The functional values are contained in a matrix y with two abscissas, x1 and x2 as illustrated in Figure B-2. No new theory needs to be developed in order to obtain an interpolated functional value at any values of x1a and x2a based on the functional values stored in y.

In order to save computational time when many interpolations will be needed, the second derivative vectors corresponding to each row should first be computed and stored in a separate matrix (y2). Then, a one dimensional cubic spline interpolation can be performed on each row of the table of functional values. The end result will be a vector of functional values which correspond to the correct value of x2 (i.e. x2a) over the range of x1. This vector can then be interpolated in one dimension to yield an approximation for the functional value at x1a and x2a.

The subroutine SPLIE2.c calculates the initial matrix of second derivatives for later use in bicubic spline interpolations. SPLIE2.c is listed in Appendix B.4. The subroutine SPLIN2.c performs the two dimensional interpolation. SPLIN2 is listed in Appendix B.5.

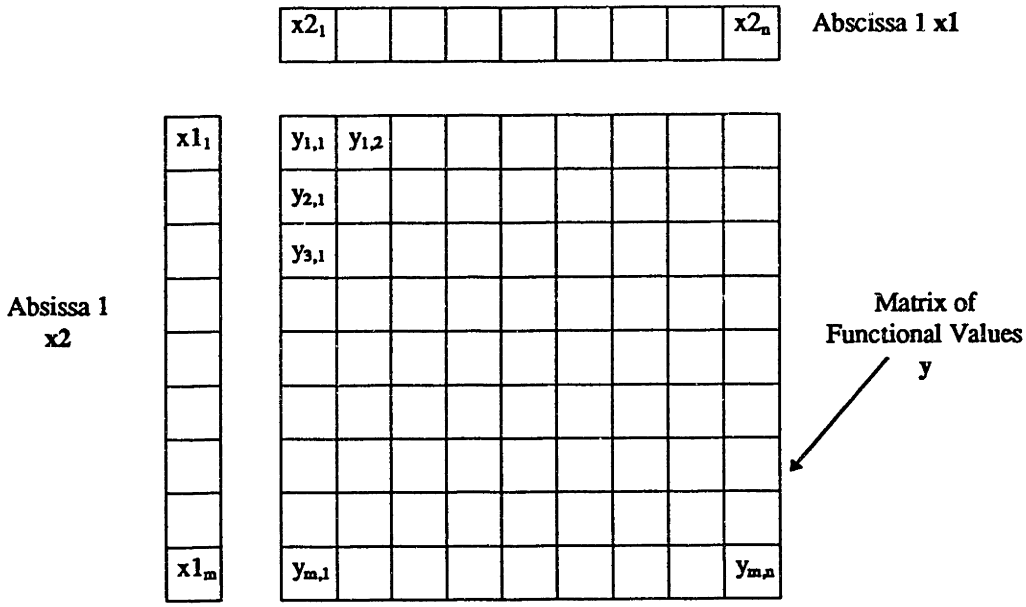

Figure B-2. Data Organization for Bicubic Spline Interpolation

B.2 Subroutine SPLINE.c

This is the C subroutine SPLINE.c discussed in Appendix B.1. This subroutine calculates the tabulated second derivatives required within the equations of a cubic spline interpolation. This subroutine requires a set of abscissas (x), the corresponding functional values (y), and the number of nodes (n). It also requires two additional boundary conditions in the form of the first derivative at the extreme nodes (yp1 and ypn). If these values are greater than 1e30, the routine implements a 'natural cubic spline'. The subroutine SPLINE.c returns the tabulated second derivatives in the vector y2. This subroutine is adapted from *Numerical Recipes in C, Second Edition.*

```
void spline(float x[], float y[], int n, float yp1, float ypn, float y2[])

{
        float *vector(long nl, long nh);
        void free_vector(float *v, long nl, long nh);


        int   i,k;
        float p,qn,sig,un,*u;

        u=vector(1,n-1);
        if (yp1 > 0.99e30)
                y2[1]=u[1]=0.0;
        else
                {
                y2[1] = -0.5;
```

```
                u[1]=(3.0/(x[2]-x[1]))*((y[2]-y[1])/(x[2]-x[1])-yp1);
                }
        for (i=2;i<=n-1;i++)
                {
                sig=(x[i]-x[i-1])/(x[i+1]-x[i-1]);
                p=sig*y2[i-1]+2.0;
                y2[i]=(sig-1.0)/p;
                u[i]=(y[i+1]-y[i])/(x[i+1]-x[i]) - (y[i]-y[i-1])/(x[i]-x[i-1]);
                u[i]=(6.0*u[i]/(x[i+1]-x[i-1])-sig*u[i-1])/p;
                }
        if (ypn > 0.99e30)
                qn=un=0.0;
        else
                {
                qn=0.5;
                un=(3.0/(x[n]-x[n-1]))*(ypn-(y[n]-y[n-1])/(x[n]-x[n-1]));
                }
        y2[n]=(un-qn*u[n-1])/(qn*y2[n-1]+1.0);
        for (k=n-1;k>=1;k--)
                y2[k]=y2[k]*y2[k+1]+u[k];
        free_vector(u,1,n-1);
}
```

B.3 Subroutine SPLINT.c

This is the C subroutine SPLINT.c discussed in Appendix B.1. This subroutine performs a cubic spline interpolation on a set of data. This subroutine requires a vector of abscissas (xa), functional values (ya), and tabulated second derivatives (y2a). The subroutine also requires the number of nodes (n) and the interpolating abscissa (x). SPLINT.c searches the abscissa vector to determine the correct interval of interpolation. It then appies an interpolating equation (e.g. Equation B.10) and returns the result in y. This subroutine is adapted from *Numerical Recipes in C, Second Edition.*

```
void splint(float xa[], float ya[], float y2a[], int n, float x, float *y)

{
        int   klo,khi,k;
        float h,b,a;

        klo=1;
        khi=n;
        while (khi-klo > 1)
                {
                k=(khi+klo) >> 1;
                if (xa[k] > x) khi=k;
                else klo=k;
                }
        h=xa[khi]-xa[klo];
        a=(xa[khi]-x)/h;
        b=(x-xa[klo])/h;
        *y=a*ya[klo]+b*ya[khi]+((a*a*a-a)*y2a[klo]+(b*b*b-b)*y2a[khi])*(h*h)/6.0;
```

```
}
```

B.4 Subroutine SPLIE2.c

This is the C subroutine SPLIE2.c discussed in Appendix B.1. This subroutine calculates the matrix of second derivatives for use in later bicubic spline operations. The subroutine requires the two abscissa vectors (x1a and x2a), the matrix of function values (y), and the dimensions of the matrix (m and n). The subroutine returns the calculated second derivatives in the matrix y2a. This subroutine is adapted from *Numerical Recipes in C, Second Edition.*

```
void splie2(float x1a[], float x2a[], float **ya, int m, int n, float **y2a)

{
        void spline(float x[], float y[], int n, float yp1, float ypn, float y2[]);
        int j;

        for (j=1;j<=m;j++)
                spline(x2a,ya[j],n,1.0e30,1.0e30,y2a[j]);
}
```

B.5 Subroutine SPLIN2.c

This is the C subroutine SPLIN2.c discussed in Appendix B.1. This subroutine performs a bicubic spline interpolation at a given interpolating point (x1, x2). The subroutine requires the two abscissa vectors (x1a and x2a), the matrix of functional values (ya), the matrix of precalculated second derivatives (y2a), and the dimensions of the matrix (m and n). The subroutine returns the interpolated vallue in y. This subroutine is adapted from *Numerical Recipes in C, Second Edition.*

```
void splin2(float x1a[], float x2a[], float **ya, float **y2a, int m, int n, float x1, float x2, float *y)

{
        float *vector(long nl, long nh);
        void spline(float x[], float y[], int n, float yp1, float ypn, float y2[]);
        void splint(float xa[], float ya[], float y2a[], int n, float x, float *y);
        void free_vector(float *v, long nl, long nh);

        int   j;
        float *ytmp,*yytmp;

        ytmp=vector(1,m);
        yytmp=vector(1,m);
        for (j=1;j<=m;j++)
                splint(x2a,ya[j],y2a[j],n,x2,&yytmp[j]);
        spline(x1a,yytmp,m,1.0e30,1.0e30,ytmp);
        splint(x1a,yytmp,ytmp,m,x1,y);
        free_vector(yytmp,1,m);
        free_vector(ytmp,1,m);
}
```

Appendix C. Ideal 'Mechanical' Cycle with Nonideal Helium Properties

This appendix carries out the analysis described in Subsection 2.6.4. The temperature profiles are obtained for a magnetically active regenerative refrigerator undergoing a 'mechanical' cycle with nonideal helium. The differential equation which governs the temperature profile was derived in Subsection 2.6.4 and is restated here:

$$\frac{c_p(P_{high}, T_2(x))}{T_2(x)} \cdot \frac{dT_2}{dx} = \frac{c_p(P_{low}, T_4(x))}{T_4(x)} \cdot \frac{dT_4}{dx} \tag{2.13}$$

This equation can be written in terms of nondimensional temperature (Θ) and nondimensional axial position ($\bar{x}$):

$$\frac{c_p(P_{high}, \Theta_2(\bar{x}))}{\Theta_2(\bar{x})} \cdot \frac{d\Theta_2(\bar{x})}{d\bar{x}} = \frac{c_p(P_{low}, \Theta_4(\bar{x}))}{\Theta_4(\bar{x})} \cdot \frac{d\Theta_4(\bar{x})}{d\bar{x}} \tag{C.1}$$

These nondimensional variables are defined in Equations C.2 and C.3.

$$\Theta = \frac{(T - T_{cold})}{(T_{warm} - T_{cold})} \tag{C.2}$$

$$\bar{x} = \frac{x}{L} \tag{C.3}$$

The nonideal specific heat of helium is simulated with two seventh order polynomial curve fits. The curve fit is accomplished using the software package SigmaPlot®. This packages uses the Marquardt-Levenberg algorithm to minimize residual errors. The curve fits are illustrated in Figure C-1. The resulting coefficients are listed in Table C-1.

$$c_p(P_{high}, \Theta) = \sum_{i=0}^{7} a_i \cdot \Theta^i \tag{C.4}$$

$$c_p(P_{low}, \Theta) = \sum_{i=0}^{7} b_i \cdot \Theta^i \tag{C.5}$$

The non-dimensional temperature profiles are assumed to be third order polynomials.

$$\Theta_2(\bar{x}) = \sum_{i=0}^{3} c_i \cdot \bar{x}^i \tag{C.6}$$

$$\Theta_4(\bar{x}) = \sum_{i=0}^{3} d_i \cdot \bar{x}^i \tag{C.7}$$

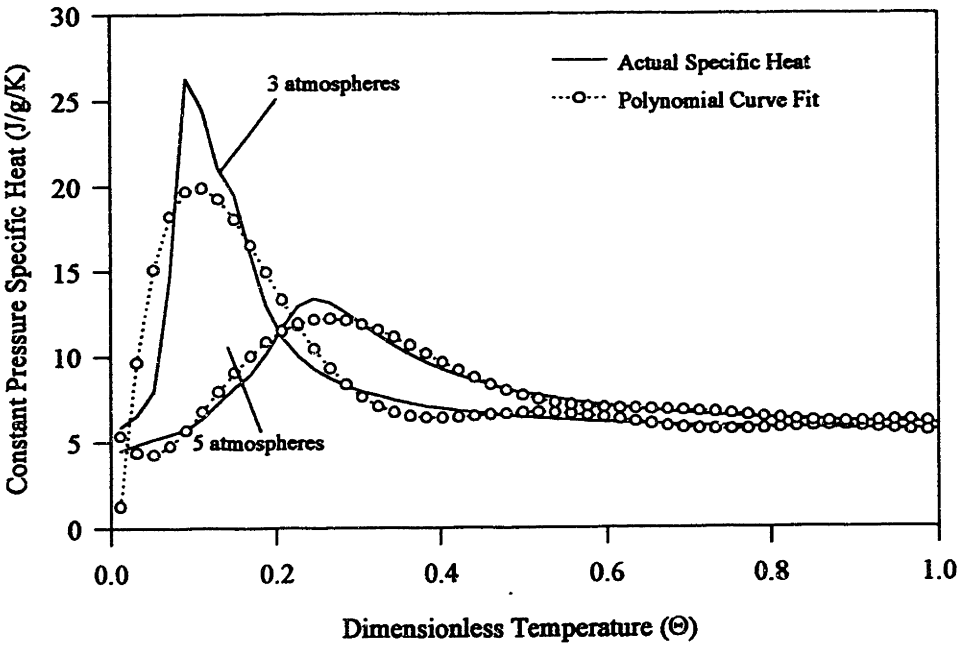

Figure C-1. Polynomial Curve Fit of Nonideal Helium Constant Pressure Specific Heat

Equation C-1 can now be rewritten in terms of the known specific heat coefficients.

$$\frac{\sum_{i=0}^{7} a_i \cdot \left[\sum_{j=0}^{3} c_j \cdot \bar{x}^{j}\right]}{\sum_{j=0}^{3} c_j \cdot \bar{x}^{j}} \cdot \sum_{j=1}^{3} j \cdot c_j \cdot \bar{x}^{j-1} = \frac{\sum_{k=0}^{7} b_k \cdot \left[\sum_{l=0}^{3} d_l \cdot \bar{x}^{l}\right]}{\sum_{l=0}^{3} d_l \cdot \bar{x}^{l}} \cdot \sum_{l=1}^{3} l \cdot d_l \cdot \bar{x}^{l-1} \qquad \textbf{(C.8)}$$

The dimensionless temperature profile is required to go through zero and unity at the two endpoints. Equation C.8 is written for several non-dimensional axial positions. The coefficients of the temperature profiles are chosen in order to minimize the error function for this nonlinear system of equations. The minimization was performed using the software package Mathcad®, which also uses a modified version of the Marquardt-Levenberg method.

The resulting non-dimensional temperature profile is illustrated in Figure C-2. The dimensionless temperatures are converted into absolute temperatures and shown as Figure 2-13. The coefficients which minimize the residual error in the dimensionless temperature profile are listed in Table C-1.

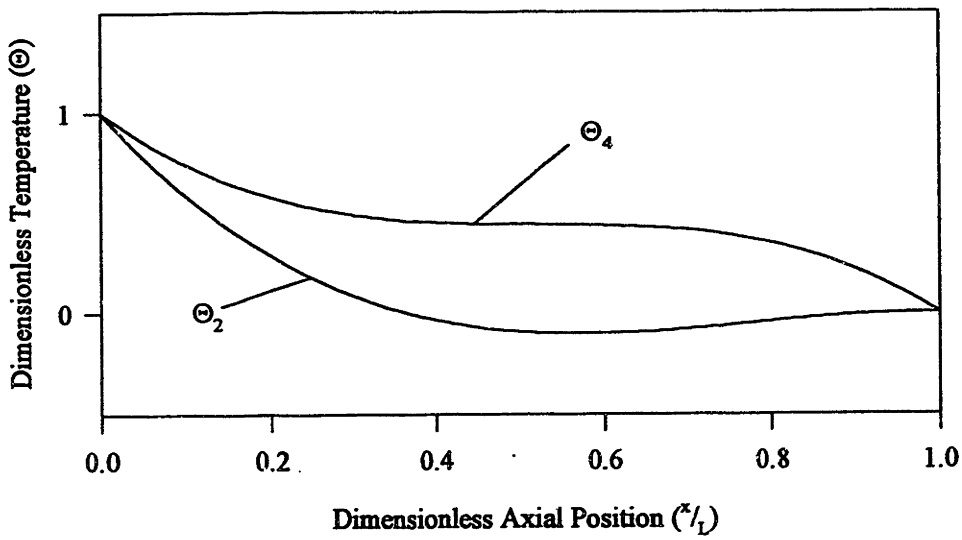

Figure C-1. Non-dimensional Temperature Profiles for 'Mechanical' Cycle with Nonideal Helium Properties

Coefficient Subscript	$c_p(P_{high},\Theta)$ a_i	$c_p(P_{low},\Theta)$ b_i	Inlet Profile Θ_2 c_i	Outlet Profile Θ_4 d_i
0	6.6158	-5.8768	1	1
1	-124.87	650.67	-4.772	-3.161
2	1994.6	-5719.2	6.585	6.012
3	-9897.8	2.1904e4	-2.813	-3.851
4	2.3148e4	-4.4269e4	-	-
5	-2.8326e4	4.9263e4	-	-
6	1.7582e4	-2.8570e4	-	-
7	-4377.0	6752.3	-	-

Table C-2. Curve Fit Coefficients for 'Mechanical' Cycle with Nonideal Helium Properties

Appendix D. Precoolant Regenerator Model

This appendix simulates the precoolant regenerator based on the equations developed in Subsection 5.4.4. The relevant equations are restated below.

$$T_{pc} = \frac{(\overline{mf} \cdot T_{warm} + mf_c \cdot T_{boil})}{\overline{mf} + mf_c} \tag{5.12}$$

$$\varepsilon = \frac{(T_{room} - \overline{T}_{superheat})}{(T_{room} - T_{pc}(mf_c))} \tag{5.13}$$

$$NTU = \frac{\ln\left[\dfrac{\varepsilon - 1}{C_r(mf_c)\cdot\varepsilon - 1}\right]}{C_r(mf_c) - 1} \qquad C_r(mf_c) = \frac{\overline{mf}}{\overline{mf} + mf_c} \tag{5.14}$$

$$NTU = \frac{N \cdot \pi \cdot Nu_D}{2 \cdot c_p \cdot \overline{mf}} \int_0^L k_{he}(T(x)) \cdot dx \tag{5.18}$$

Equations 5.12 through 5.14 have been combined to yield the number of transfer units required as a function of precoolant mass flow rate. This relation is illustrated in Figure 5-12. Examination of the heat transfer regime within the tubes leads to Equation 5.18.

The thermal conductivity of helium is a strong function of temperature as shown in Figure D-1. A second order polynomial in temperature is curve fit to the thermal conductivity in order to analytically evaluate the integral in 5.18. This curve fit is also illustrated in Figure D-1.

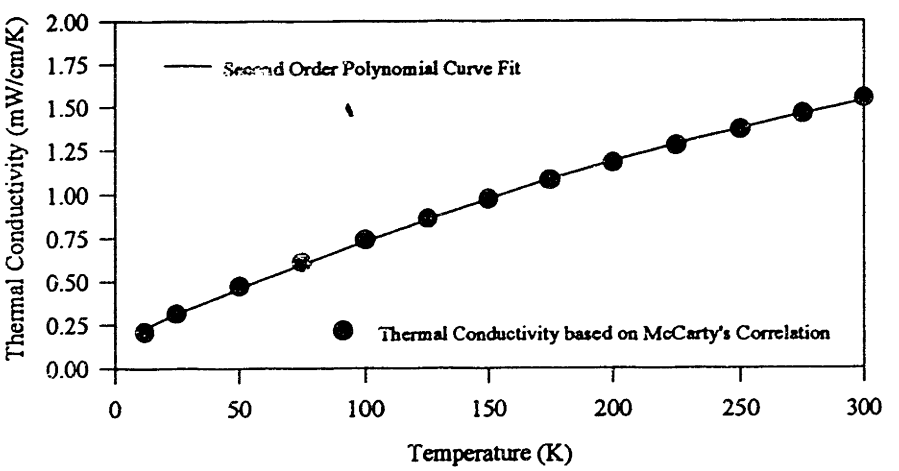

Figure D-1. Polynomial Curve Fit for Helium Thermal Conductivity

The polynomial curve fit is in terms of a dimensionless temperature (Θ). The dimensionless temperature is based on the total temperature spanned by the precoolant regenerator.

$$\Theta = \frac{(T - \overline{T}_{superheat})}{(T_{room} - \overline{T}_{superheat})} \tag{D.1}$$

$$k_{he}(T) = a_0 + a_1 \cdot \Theta + a_2 \cdot \Theta^2 \tag{D.2}$$

Initially, the temperature profile within the regenerator is assumed to be linear.

$$\Theta(x) = \frac{x}{L} \tag{D.3}$$

With this simplification, Equation 5.18 can be integrated directly.

$$NTU = \frac{N \cdot \pi \cdot Nu_D}{2 \cdot c_p \cdot \overline{mf}} \cdot (a_0 + \tfrac{1}{2} \cdot a_1 + \tfrac{1}{3} \cdot a_2) \cdot L \tag{D.4}$$

In order to validate the assumptions made in Section 5.4.4, the temperature profiles within the regenerator are examined. These profiles are governed by three equations.

$$(\overline{mf} + mf_c) \cdot c_p \cdot \frac{dT_c}{dx} = htc_{tot}(x) \cdot N \cdot \pi \cdot D \cdot (T_h(x) - T_c(x)) \tag{D.5}$$

$$\overline{mf} \cdot c_p \frac{dT_h}{dx} = htc_{tot}(x) \cdot N \cdot \pi \cdot D \cdot (T_h(x) - T_c(x)) \tag{D.6}$$

$$\frac{1}{htc_{tot}(x)} = \frac{D}{Nu_D \cdot k_{he}(T_h(x))} + \frac{D}{Nu_D \cdot k_{he}(T_c(x))} \tag{D.7}$$

Equation D.5 through D.7 can be numerically integrated from the cold end to the warm end using a simple finite difference technique. The results of this calculation with the design parameters specified in Table 5-6 are illustrated in Figure 5-16.

Appendix E. Displacer Flow Calculations

This appendix presents the numerical analysis which was used to specify the displacer geometry. In order to realistically estimate the pressure difference which will be seen by the displacer during a flow process, the pressure drop over the precoolant regenerator must be modelled. The viscosity of helium changes by an order of magnitude between room temperature and 12° K. This must be taken into account in any realistic model.

The viscosity of helium is relatively insensitive to pressure changes over this temperature range. Therefore, an approximate analytical expression for viscosity as a function of temperature at an average pressure may be applied regardless of pressure. A second order polynomial curve fit for helium viscosity as a function of temperature is obtained. This curve is illustrated in Figure E-1.

$$\mu_{he}(T) = b_0 + b_1 \cdot T + b_2 \cdot T^2 \tag{E.1}$$

The Reynolds Number within the precoolant regenerator, neglecting the entrained helium, will only change due to the viscosity variation.

$$\mathrm{Re}_D(T) = \frac{4 \cdot \overline{mf}}{N \cdot \pi \cdot D \cdot \mu(T)} \tag{E.2}$$

Since the viscosity of helium increases with temperature, the Reynolds Number will decrease at higher temperatures. Neglecting any transition effects, the friction factor should tend to increase with temperature. Furthermore, the pressure gradient may be determined from the friction factor.

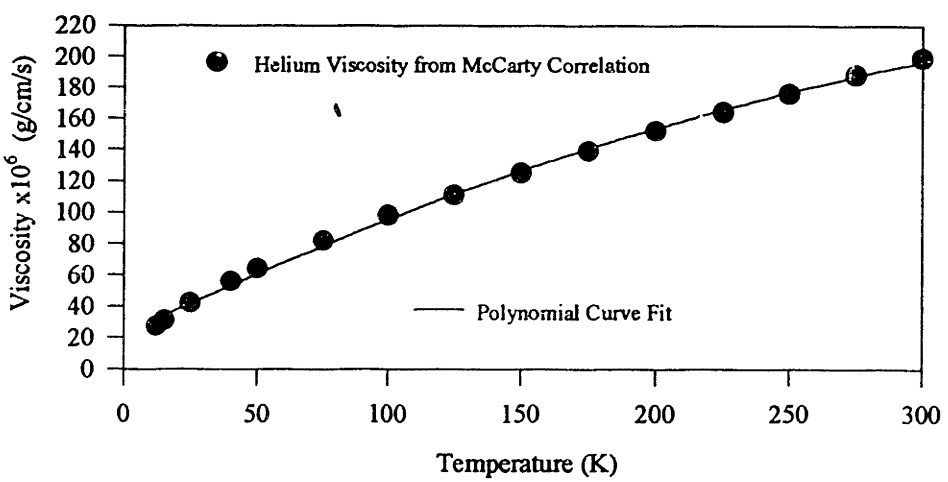

Figure E-1. Polynomial Curve Fit for Helium Viscosity as a Function of Temperature

$$\frac{dP}{dx} = \frac{f(\mathrm{Re}_D(T)) \cdot \rho \cdot v^2}{2 \cdot D} \tag{E.2}$$

Equation E.2 may be rewritten as a function of only temperature and pressure by assuming that helium behaves as a perfect gas and that the mass flow rate remains constant.

$$\frac{dP}{dx}(T,P) = \frac{f(\mathrm{Re}_D(T)) \cdot \overline{mf}^2 \cdot 8 \cdot R_{he} \cdot T}{D^5 \cdot N^2 \cdot \pi^2 \cdot P} \tag{E.3}$$

Examination of Equation E.3 reveals that the pressure drop will increase with temperature due to the viscosity increase and the density decrease. The approximate temperature profiles within the precoolant regenerator were developed within Appendix D. These profiles are illustrated in Figure 5-16. It was determined in Appendix D that the thermal conductivity effect will tend to pull the temperature profiles down. It is therefore a conservative approximation to use a linear temperature profiles in order to estimate the pressure drop.

If Equation E.3 were combined with a suitable analytical expression for friction factor as a function of Reynolds Number, the combination could be numerically integrated in order to approximate the pressure drop through the precoolant regenerator. Unfortunately, the flow within the precoolant regenerator is laminar in the warmer regions due to the increased viscosity. As the temperature drops, the flow becomes turbulent. There is no suitable analytic expression for friction factor in the transition region. Therefore, the Moody Diagram must be used to graphically determine the appropriate friction factor after each differential integration.

The results of the numerical integration are illustrated in Figure E-2. As expected, most of the pressure drop occurs within the warmer region of the regenerator. The total pressure drop across the regenerator is expected to be approximately 37 kPa.

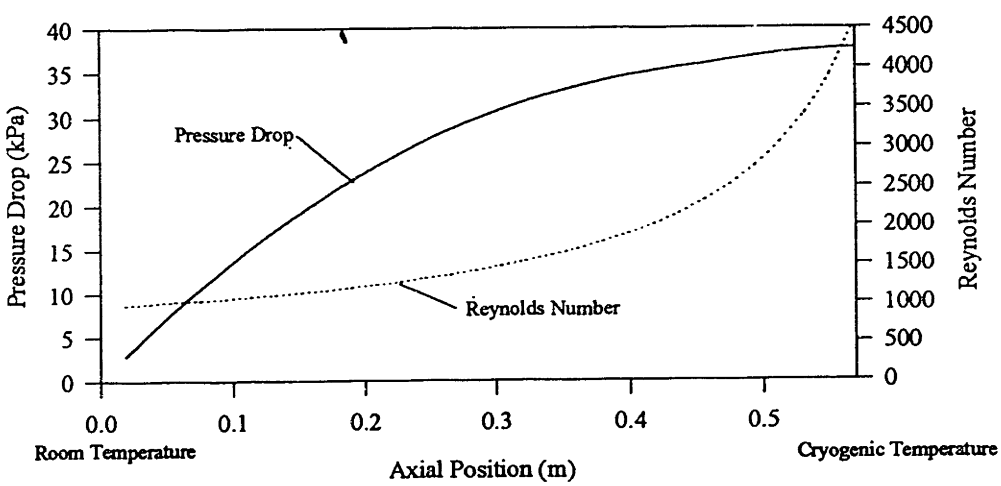

Figure E-2. Pressure Drop and Reynolds Number within the Precoolant Regenerator

THESIS PROCESSING SLIP

FIXED FIELD ill _____ name _____

index _____ biblio _____

▶ COPIES (Archives) Aero Dewey (Eng) Hum

Lindgren Music Rotch Science

TITLE VARIES ▶☐ _____

NAME VARIES: ▶☒ Francis

IMPRINT (COPYRIGHT) _____

▶ COLLATION: 173 leaves

▶ ADD. DEGREE: _____ ▶ DEPT.: _____

SUPERVISORS: _____

NOTES:

cat'r: date:

page: 598

▶ DEPT: ~~Mech. Eng.~~ M.E.

▶ YEAR: 1995 ▶ DEGREE: M.S.

▶ NAME: WELLIS, Gregory F.