THERMAL ANALYSIS OF STIRLING CYCLE REGENERATORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERCAN ÖZBAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2011

Approval of the thesis:

THERMAL ANALYSIS OF STIRLING CYCLE REGENERATORS

submitted by **SERCAN ÖZBAY** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

ABSTRACT

THERMAL ANALYSIS OF STIRLING CYCLE REGENERATORS

Özbay, Sercan
M.Sc. Department of Mechanical Engineering
Supervisor: Assoc. Prof. Dr. İlker Tarı

August 2011, 116 pages

Stirling cycle cryocoolers are used widely in military applications. The regenerator is the key element of Stirling cycle cryocoolers. It is known that performance of the regenerator directly affects the cryocooler performance. Therefore, any improvement on the regenerator will lead to a more efficient cryocooler. Thus, it is essential to have an idea about regenerator parameters and their effects on the system.

In this study Stirling engine regenerator, which is constructed by wire mesh screens, is accepted as a porous medium. Using energy balance and continuity equation, matrix and fluid thermal equations are derived. Simplified versions of these equations are obtained for not only the ideal case, but also two other cases which take into account the effects of longitudinal conduction and the effects of regenerator wall. A computer code is developed in Matlab to solve these equations using finite difference method. The developed code is validated by using Sage. Afterwards, effects of all regenerator parameters on regenerator performance are investigated in detail and results are presented. To make this investigation easier, a graphical user interface is also built (in Matlab) and used.

Keywords: Cryocooler, Stirling cycle, thermal analysis, finite difference.

ÖZ

STİRLİNG ÇEVRİMİ İLE ÇALIŞAN REJENATÖRLERİN TERMAL ANALİZİ

Özbay, Sercan
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. İlker Tarı

Ağustos 2011, 116 sayfa

Askeri uygulamalarda Stirling çevrimi ile çalışan krayojenik soğutucular sıklıkla kullanılmaktadır. Rejeneratör ise bu çevrim ile çalışan soğutucuların önemli bir elemanıdır. Rejeneratörün performansı soğutucunun perforamansını doğrudan etkilemektedir. Bu nedenle rejeneratöre yapılacak iyileştirmeler soğutucuyu daha verimli hale getirecektir. Bu nedenle rejeneratör parametrelerinin sistem üzerindeki etkilerine hakim olmak gerekmektedir.

Bu tez çalışmasında tel örgü disklerle oluşturulmuş, Stirling çevrimi ile çalışan bir rejeneratör, gözenekli bir yapı olarak kabul edilip enerjinin korunumu yasası ve süreklilik denklemi kullanılarak, matriks yapı ve akışkan için termal denklemler çıkartılmıştır. Bu denklemlerin basitleştirilmiş halleri, ideal durum, dikey ısı iletiminin olduğu durum ve rejeneratör duvarının etkilerinin göz önüne alındığı durum için elde edilmiştir. Elde edilen denklemleri sonlu farklar yöntemiyle çözmek için Matlab programında bir kod yazılmıştır.

Yazılan bu kod, Sage programı kullanılarak doğrulanmıştır. Daha sonra, rejeneratör parametrelerinin, rejenaratör performansı üzerindeki etkileri detaylı olarak incelenmiş ve sonuçlar sunulmuştur. Bu incelemeyi daha kolay hale getirmek için, Matlab programında kullanıcı arayüzü oluşturulmuş ve kullanılmıştır.

Anahtar Kelimeler: Krayojenik soğutucu, Stirling çevrimi, termal analiz, sonlu farklar yöntemi

To my family

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. İlker Tarı for his support, guidance and suggestions through out this research.

The examining committee members Assoc. Prof. Dr. Cemil Yamalı, Asst. Prof. Dr. Ahmet Yozgatlıgil, and Asst. Prof. Dr. Tuba Okutucu Özyurt greatly acknowledged for their participation, comments and suggestions.

I want to thank my company ASELSAN, my superiors Savaş Bektaş, Ekrem Nurdağ and our manager Yılmaz Oktay for allowing and supporting me to get this degree. Their incentive talks and understading made everything much easier.

I am deeply thankful to my colleague and friend Semih Çakıl for his encouragement and support. I gained a lot of confidence while challenging with his attention over minor details.

I want to thank my dear friends Sevil Demirci, Ceyda Elbaşıoğlu for keeping me company while preparing this thesis. Special thanks go to Ece Koçer, who is a sister rather than a friend, for being there whenever I needed. Emre Yılmaz, M.Haydar Şahin and Uğur Olgun should know I could not do this without you having in my life. Thank you guys.

Last but not least, I would like to express my endless gratitude to my parents for their love, endless support and trust throughout my life. Mom, words are not enough to express my appriciation to you for the sacrifices you made for me. I will always do my best and make you proud. I love you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

A_{ff}	Fluid axial free flow area (m^2)
A_r	Wall to matrix heat transfer area ratio (m^2)
A_s	Matrix total heat transfer area (m^2)
A_{wl}	Wall thermal conduction heat transfer area (m^2)
c_F	Compression factor
c_p	Specific heat at constant pressure (J/kg.K)
C_r	Capacity ratio
d_h	Hydraulic diameter (m)
f	Frequency (Hz)
h	Heat transfer coefficient ($W/m^2.K$)
K_m	Matrix thermal conductivity (W/m.K)
K_{wl}	Wall thermal conductivity (W/m.K)
L	Regenerator Length (m)
M	Mass (kg)
$\dot{m}$	Mass flow rate (kg/s)
M_m	Mass of matrix material (kg)
n	Screen mesh size
N_s	Number of screens
N_t	Number of time intervals
N_z	Number of longitudinal nodes
$\dot{Q}$	Heat flow rate (W)
Q_{fric}	Friction heat loss (J)
r	Radial coordinate
r_L	Regenerator length (m)
t	Time (s)
T_c	Regenerator cold end temperature (K)
t_F	Thickness inclination constant
T_f	Regenerator fluid temperature (K)

$(T_f)_c$	Regenerator fluid temperature during cooling period (K)
$(T_f)_h$	Regenerator fluid temperature during heating period (K)
T_{in}	Inlet temperature (K)
T_m	Regenerator matrix temperature (K)
$(T_m)_c$	Regenerator matrix temperature during cooling period (K)
$(T_m)_h$	Regenerator matrix temperature during heating period (K)
T_{out}	Outlet temperature (K)
t_{sc}	Screen thickness (m)
T_w	Regenerator warm end temperature (K)
U	Overall heat transfer coefficient (W/m^2.K)
V	Volume (m^3)
V_m	Matrix volume (m^3)
w	Velocity Vector
w_d	Wire diameter (m)
x_t	Transverse pitch (m)
z	Axial coordinate

Greek Letters

α	Matrix porosity
β	Area density (1/m)
ε	Efficiency
θ	Angular cylindrical coordinate
λ	Flow period (s)
ρ	Density (kg/m^3)

Subscripts

c	Cold
f	Fluid
h	Hot

i	Inlet
m	Matrix
o	Outlet

CHAPTER 1

INTRODUCTION AND BACKGROUND

1.1. Cryogenics

Cryogenics is the branch of physics that deals with refrigeration at temperatures lower than 120 K. Obtaining cryogenic temperatures is essential for preservation of biological material and food as well as densification of gases for production and transportation purposes. At cryogenic temperatures, due to quantum effects, it is also possible to have superconductive materials. Since thermal noise is low at cryogenic temperatures, it is widely used at infrared applications. These benefits became the driving force for scientists to build instruments which make refrigeration at cryogenic temperatures.

1.2. Cryocoolers

Cryocooler is a device which has an ability to produce refrigeration at cryogenic temperatures. This ability had found itself numerous application areas which are listed in Table 1.1: Crycooler applications To produce the refrigeration, the working gas goes through a specific thermodynamic cycle which includes compression and expansion in order to transfer energy from one state to the other. Removing the heat that is generated during compression, and using the temperature decrease that occurs during expansion is the basic working principle of cryocoolers (Williems, 2007).

Cryocoolers can be classified with respect to their cooling capacity and size (Walker, 1983) or with the thermodynamic cycle they are following but there is a general agreement that they are basically classified with the heat exchanger type; regenerative or recuperative (Walker, 1983) (Timmerhaus, 1996).

Recuperative heat exchangers have separate flow passages for hot and cold fluids. The heat is transferred by conduction across the solid wall between the two streams which may flow continuously or periodically (Walker, 1983).

In regenerative heat exchangers hot and cold fluids flow through a single set of flow passages alternately and periodically. Usually a porous media (spheres, metal wire screens, rolled foils etc.) forms the regenerative matrix. The matrix can be thought as a thermodynamic sponge which accepts and rejects heat in an alternating fashion while the hot or cold fluid flows through it (Walker, 1983).

Joule-Thompson, Brayton, Linde, Hampson, Claude and Collins are cryocoolers with recuperative heat exchangers. Stirling, Pulse Tube, Gifford-McMahon, Ericsson and Vuilleumier are cryocoolers with regenerative heat exchangers (Walker, 1983). Some of the cryocoolers mentioned here, are not very common today. In Figure 1.1 schematics of five common ones are shown.

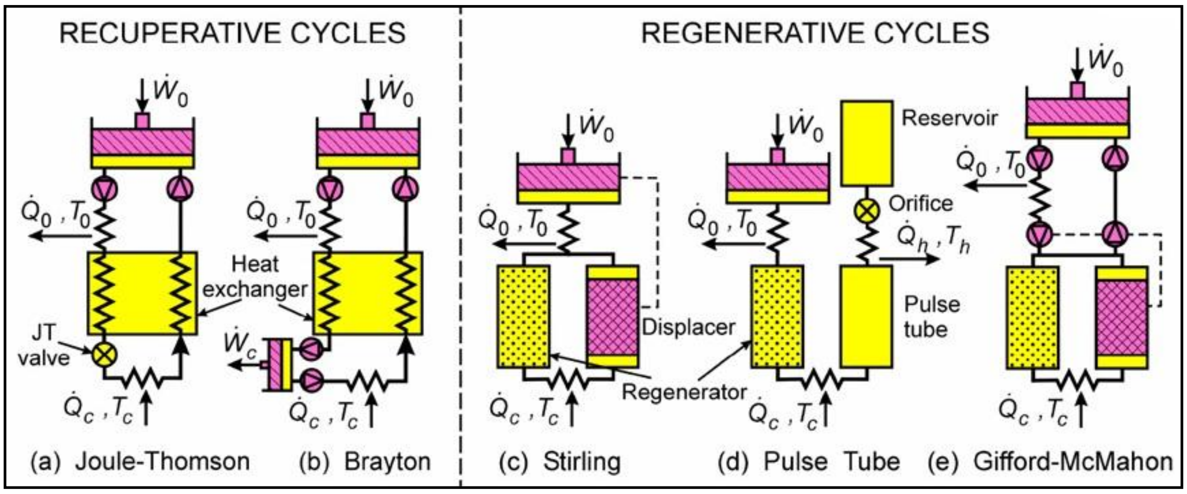


Figure 1.1: Schematics of five common types of cryocoolers (Radebaugh, 2009)

Table 1.1: Crycooler applications (Radebaugh, 1995)

Military	Infrared sensors for missile guidance Infrared sensors for surveillance (satellite based) Superconducting magnets for mine sweeping
Industrial and Commercial	Superconductors for high-speed communication Semiconductors for high-speed computers Cryopumps for semiconductor manufacture Low-level moisture sensors for ultrapure gases Infrared sensors for process monitoring
Medical	Superconducting magnets for MRI systems SQUID magnetometers for heart and brain studies Blood and semen storage Cryosurgery
Energy	LNG production at remote gas wells and for peak shaving Infrared sensors for thermal-loss measurements SMES for peak shaving and power conditioning
Environment	Infrared sensors for atmospheric studies of ozone hole and greenhouse effect Infrared sensors for pollution monitoring Cryotrapping air samples at remote locations
Transportation	Infrared sensors for aircraft night vision LNG for fleet vehicles
Agriculture and Biology	Biological specimens storage
Law Enforcement	Infrared sensors for night vision

1.3. Stirling Cycle Refrigerators

In Figure 1.2 main parts of a Stirling cycle refrigerator is shown. Working theory and detailed information about it can be found in the following sections.

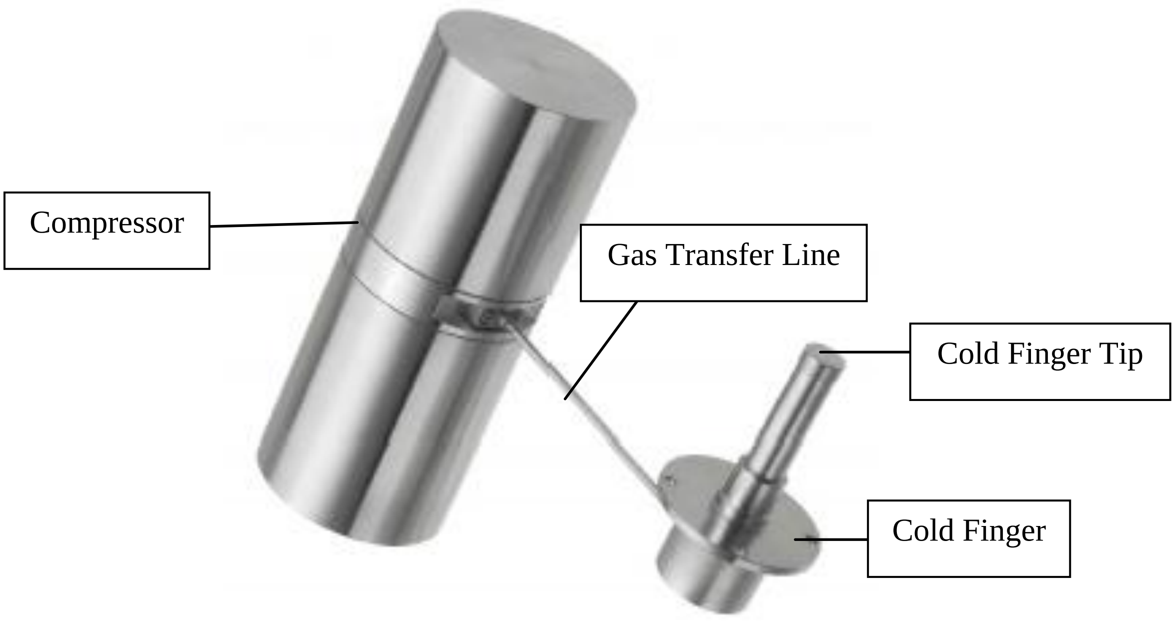


Figure 1.2: A modern Stirling cryocooler (Linear, split, pneumatically driven) (Radebaugh, 2010)

1.3.1. Theory

The ideal Stirling cycle is composed of four processes. Pressure-Volume and Temperature-Entropy diagrams for ideal Stirling refrigerator are shown in Figure 1.3. In ideal Stirling cycle diagrams, heat exchangers, regenerators, connecting elements are assumed to have zero volume (Heywood, 2004).

The four processes shown in Figure 1.3 and Figure 1.4 are explained in detail as follows;

- 1→2 Isothermal compression: Movement of compression piston to the right, while expansion piston stays still, compresses the working fluid. Due to isothermal compression, some amount of heat is removed from the system. Volume is reduced from V_1 to V_2. In an actual cryocooler, this is the heat that is removed (thrown out) from the bottom of cold finger (Figure 1.2).

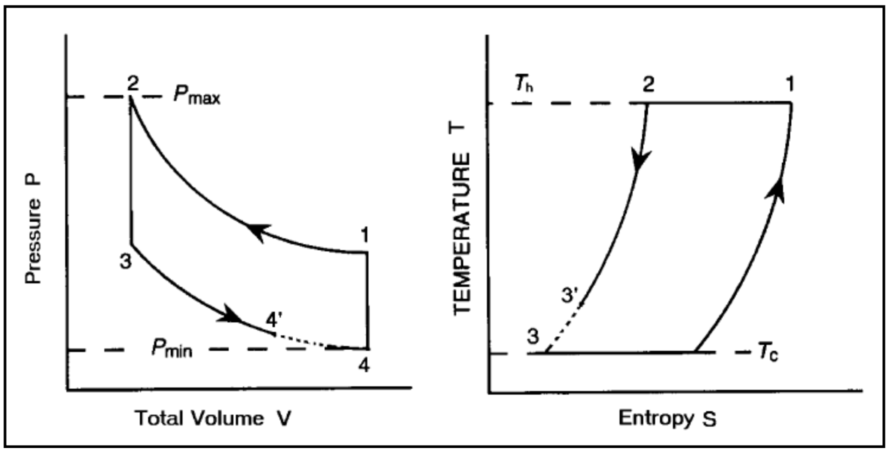


Figure 1.3: P-V and T-S diagrams for Stirling cycle (Refrigeration) (Yang & Chung, 2005)

- 2→3 Isochoric displacement: Keeping the volume constant, both pistons move to the right. The working gas is forced to pass through the regenerator. And while gas travels, heat is transferred to the regenerator causing the gas to cool down to T_c.

- 3→4 Isothermal expansion: This time expansion piston moves to the right while compression piston stays still. The volume has increased back to V_1 (V_4=V_1). Since expansion is isothermal, some amount of heat is transferred to the working fluid. In an actual cryocooler, this is the heat that is removed from the cold finger tip (Figure 1.2) i.e. the surface which is tried to be cooled.

- 4→1 Isochoric displacement: Again at constant volume, both pistons move to the left this time. This movement, forces the gas in the expansion area to flow back, increasing its temperature back to T_h and taking the system back to the original state.

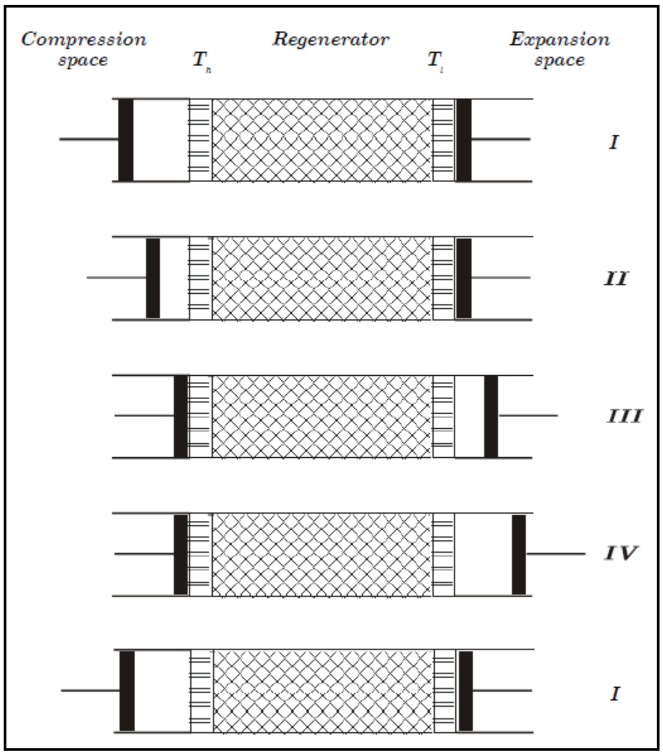


Figure 1.4: Steps in the Stirling refrigeration cycle (Williems, 2007)

1.3.2. Stirling Cycle Cryocoolers

As stated by Riabzev (2002) there are three common types of Stirling cycle cryocoolers. They are classified with respect to their compressor type and expander drive mechanism. Namely these Stirling cryocoolers are; rotary kinematic, rotary pneumatic, linear and linear pneumatic. In Figures 1.5 – 1.7, schematics for these coolers are shown respectively.

A rotary crankshaft drives the rotary kinematic Stirling cryocooler’s (Figure 1.5) piston and displacer by connecting rods. These rods are also used to adjust the phase angle difference between the compressor and displacer (Çakıl, 2010).

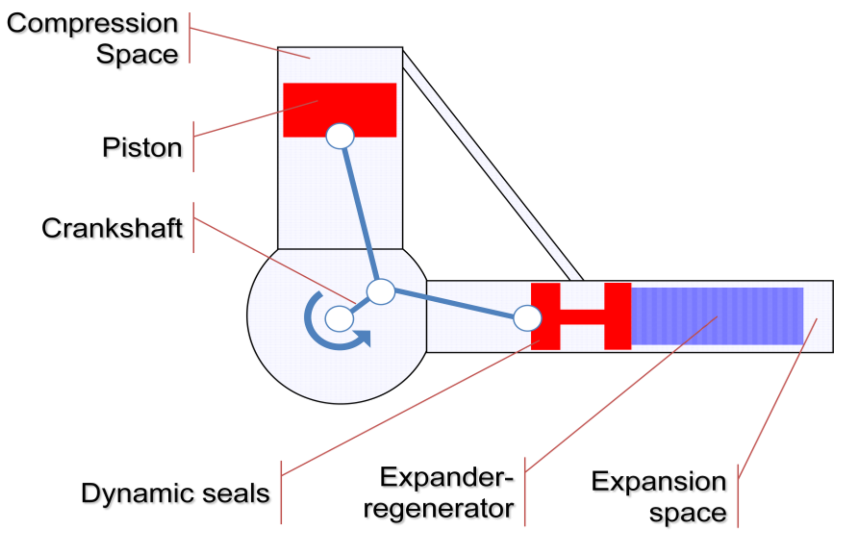


Figure 1.5: Rotary kinematic Stirling cryocooler (Riabzev, 2002)

In rotary pneumatic coolers (Figure 1.6), the piston is driven by a rotary crankshaft and the displacer is driven pneumatically. The spring at the bottom of the displacer is used to adjust the phase angle difference (Çakıl, 2010).

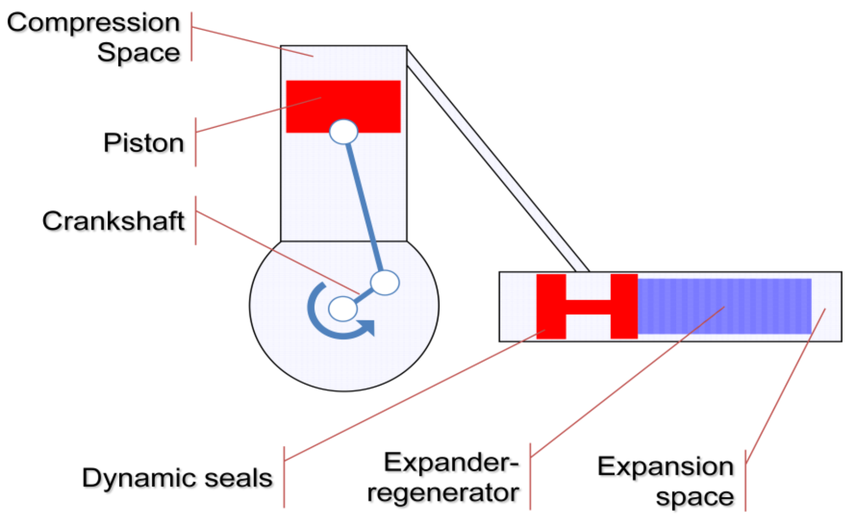


Figure 1.6: Rotary pneumatic Stirling cryocooler (Riabzev, 2002)

The most popular configuration, linear pneumatic Stirling cryocooler, uses an electric motor to drive the piston and the displacer is driven pneumatically (Figure 1.7). Like in rotary pneumatic configuration, phase angle is adjusted by the spring attached to the bottom of the displacer (Çakıl, 2010).

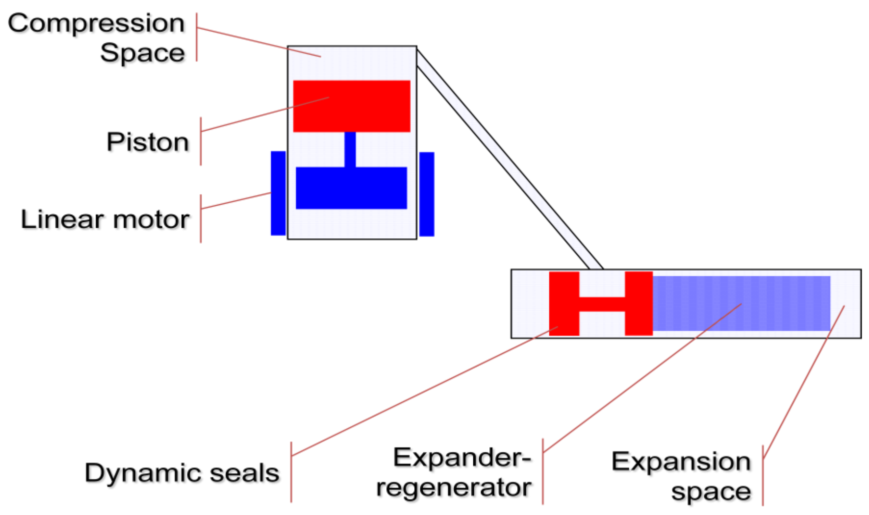


Figure 1.7: Linear pneumatic Stirling cryocooler (Riabzev, 2002)

1.3.3. Cold Finger

Cold finger is the name given to the structure that houses the displacer. Its main purpose is to absorb heat from the surface which needs to be at cryogenic temperature. The parts that form the cold finger are shown in Figure 1.8.

Displacer shown in Figure 1.8 is the part which carries the regenerator. The bearing rings are attached to displacer in order to prevent any mechanical interaction with the cold finger wall. These rings are made of a PTFE[1] based material which has low friction coefficient, and high wear resistance.

1.3.4. Regenerator

As stated earlier regenerator is the part which acts like a thermal sponge. It accepts and rejects heat periodically throughout the working cycle of the cryocooler.

[1] Polytetrafluoroethylene: A synthetic fluoropolymer of tetrafluoroethylene that finds numerous applications. PTFE is most well known by the DuPont brand name Teflon.

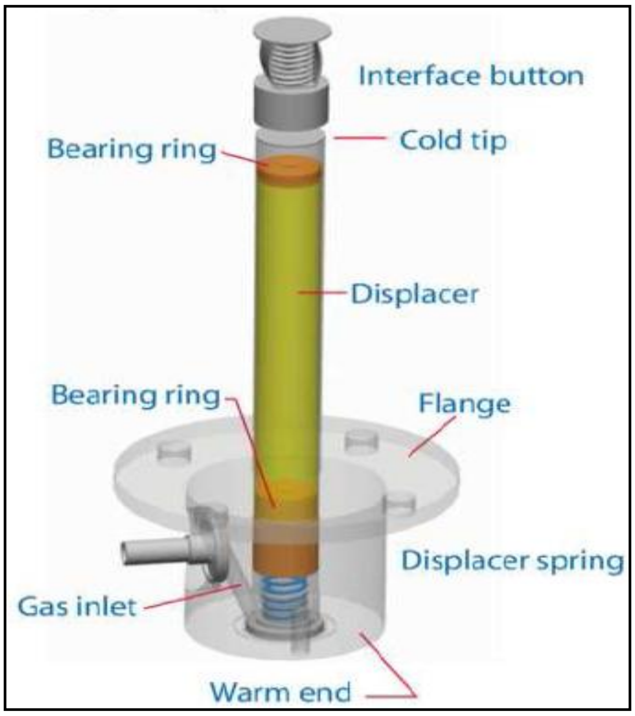


Figure 1.8: Cold finger (Thales, 2010)

As stated by many authors (Yang & Chung, 2005), (Shi J., 2007), (Pfotenhauer, Shi, & Nellis, 2004), (Cha, Ghiaasiaan, & C.S., 2008), regenerators are the crucial part of the regenerative cryocoolers. Minimizing the losses in the regenerators has always been an attractive subject for the related researchers. To do so, three main parameters can be adjusted; size, material and packing geometry. Among these three, size is usually constrained by surrounding components which belong to the satellite, the detector, etc. whatever the cooler is being used at. But other two parameters can be chosen as desired. More detailed information about these parameters is given in the upcoming section.

1.3.5. Regenerator Packing Geometries

There are four commonly used regenerator packing geometries in regenerative cryocoolers; annular gap, wire mesh screen, packed sphere and etched foil regenerator. The first three are illustrated in Figure 1.9 and the fourth one in Figure 1.11.

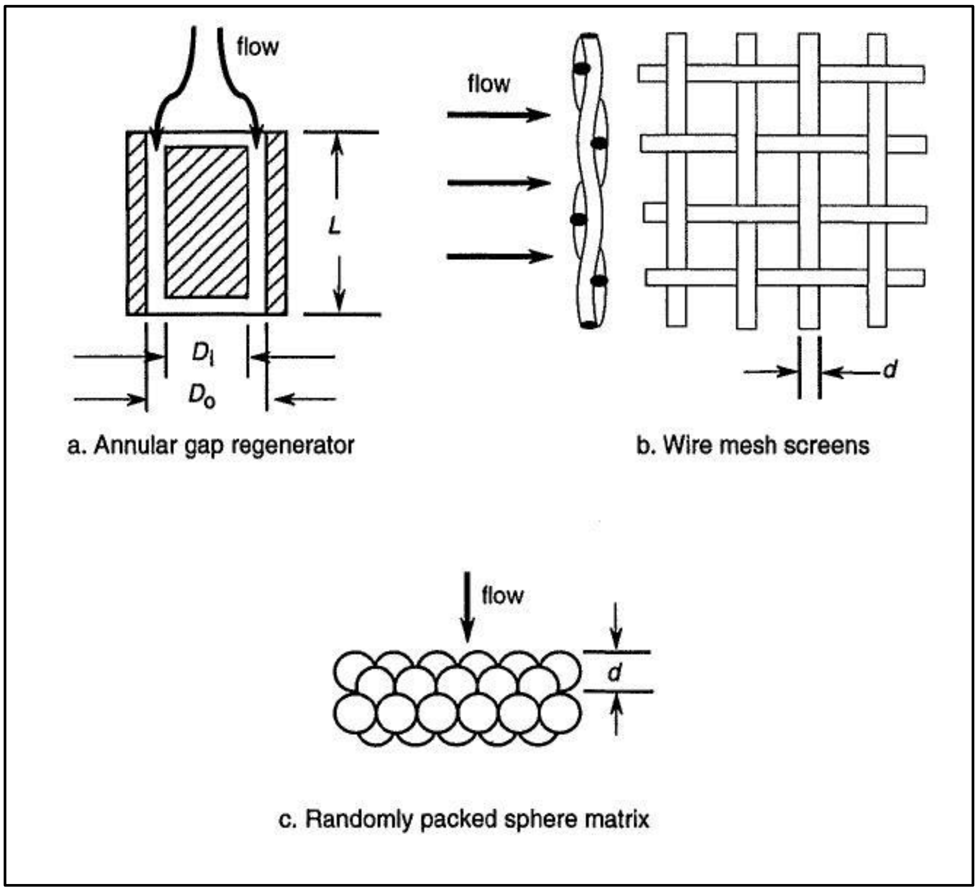


Figure 1.9: Common regenerator matrix geometries (Ackermann, 1997)

Regenerator matrix geometry should have the following characteristics to be efficient (Ackermann, 1997):

- Maximum heat transfer area
- Minimum axial conduction
- Minimum pressure drop
- Minimum dead volume

Among these, heat transfer area to fluid pressure drop ratio is the critical parameter in the definition of the effectiveness of a regenerator pack. Maximizing this ratio with minimum cost is a major design consideration (Ackermann, 1997).

1.3.5.1. Annular Gap Regenerator

Due to their simple configuration, annular gap regenerators are used by early designed Stirling cryocoolers. The space between two cylinders (Figure 1.9a) is used as heat transfer surface area. It has a simple design. Not only easy to construct, but also, the pressure drop values are significantly lower compared to wire mesh screens and packed spheres as it can be seen in Figure 1.10. The graph is obtained from the correlations given by Arp and Radebaugh (1987) which bases its results to the data obtained by Kays and London (1984). Values of the used parameters are given in Table 1.2.

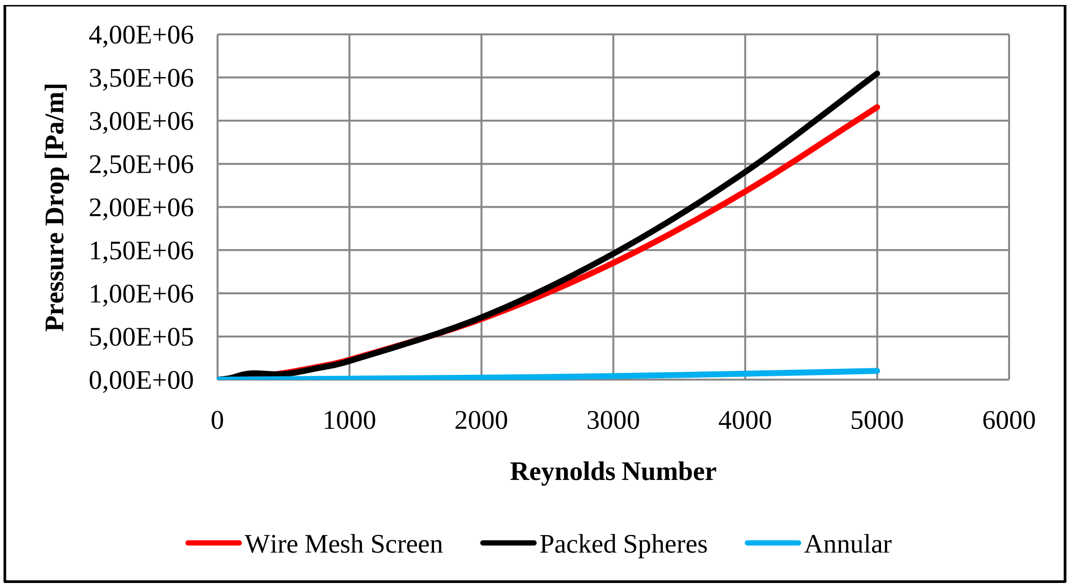


Figure 1.10: Pressure drop values for various regenerator geometries

Despite its simplicity and low pressure drop characteristics, annular regenerators suffer from limited heat transfer area.

Table 1.2: Parameters used to obtain Figure 1.10

Gas	Helium
Porosity	0.7
Viscosity	1.44E-5 Pa.s
Density	0.15 kg/m^3
Hydraulic Diameter	0.15 mm

1.3.5.2. Wire Mesh Screens

The most commonly used regenerator material is the woven wire mesh screen (Figure 1.9b). It provides high heat transfer area with lower entropy generation rate. A comparison about these characteristics was made by Barclay and Sarangi (1984). They define a figure of merit (FOM) for the refrigerator, which the tests have been carried on, as follows to make the comparison. Results are given in Table 1.3.

$$FOM \equiv \frac{Ideal\ (Carnot) Power\ Required}{Actual\ Power\ Required} = \frac{\dot{q}_c \left(\frac{T_h}{T_c} - 1\right)}{\dot{q}_c \left(\frac{T_h}{T_c} - 1\right) + T_h \dot{s}}$$

It is evident from the table that, wire mesh screens causes less entropy generation, thus they increase the efficiency.

Table 1.3: Results of Barclay and Sarangi (1984)

Geometry	Hydraulic Diameter (mm)	Frequency (Hz)	FOM
Wire Mesh	0.158	0.1	0.969
	0.079	1	0.936
	0.032	10	0.873
Packed Spheres	0.631	0.1	0.909
	0.316	1	0.823
	0.126	10	0.683
Annular Gap	0.251	0.1	0.948
	0.126	1	0.895
	0.063	10	0.797

Beside, constructing a regenerator with wire mesh discs is relatively inexpensive. Since same geometry has been used for certain types of filter production, it is easy to acquire. Moreover, it allows porosity to be adjusted with different mesh density (number of openings per inch square) configurations. This flexibility has been used by many for regenerator optimization (Harvey, 1999) (Imura, 2007) (Prajapati & Oza, 2007). It is also possible to use different material screens at different locations of the regenerator, which can be useful for low temperature (below 40K) applications where traditional materials (stainless steel etc.) losses their ability to store heat. By this method Qui, *et al*. were able to reach 11.1 K with a pulse tube cryocooler which used to operate around 20-40 K before (Qiu, 2007).

1.3.5.3. Packed Spheres

Packed spheres (Figure 1.9c) are generally used for very low temperature applications where specific heat of commercially available materials drops to values that are close to the working fluid specific heat (Ackermann, 1997). In order to eliminate this inefficiency factor, some special material compounds which have higher heat capacity at low temperatures, are used. These compounds are mostly available at spherical form. Using packed spheres decreases porosity and increases pressure drop therefore, it is efficient at temperatures below 25K where the viscosity of the working fluid is low.

1.3.5.4. Etched Foil Regenerator

The foil regenerator has a pattern with transverse slots and axial standoffs (Figure 1.11) which are produced by etching. The foil is rolled to obtain a cylindrical shape which can fit into cold finger. If the regenerator is viewed in flow direction, concentric annular flow channels will be observed. These flow channels are created by the standoffs. The transverse slots are used for smoothing the flow. They also reduce longitudinal conduction since the path of the conduction is disturbed.

This type of regenerators introduces less pressure drop compared to wire mesh screen regenerators (Mitchell & Fabris, 2003). However irregularities in the flow passages caused by etching process, uneven flow distribution due to assembly, low porosity of the foil, greatly reduces heat transfer as well as pressure drop.

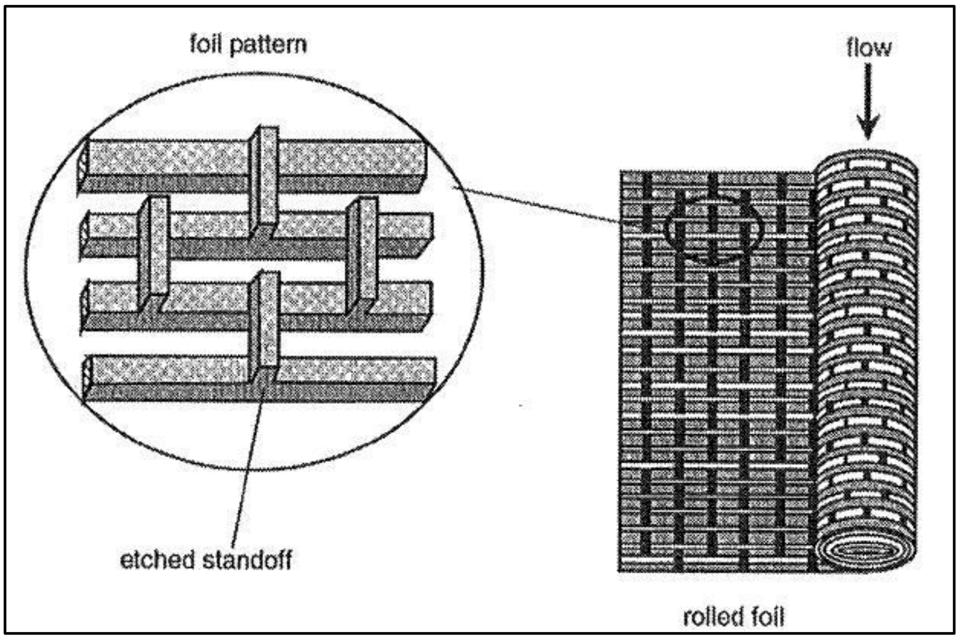


Figure 1.11: Etched foil regenerator (Ackermann, 1997)

1.3.6. Regenerator Materials

Objective of the regenerator is to store heat therefore the material used to build the regenerator should act accordingly. To reach this goal, the regenerator material must have high volumetric heat capacity. In Figure 1.12 volumetric heat capacity variation of commonly used regenerator materials are given. As it can be observed from Figure 1.12, as temperature decreases, volumetric heat capacity decreases sharply for most of the materials. But it should be kept in mind that the graph is for temperatures below 120K.

Stainless steel seems to be the best regenerator material if the working range is between 60 and 300K. Lead can be a good alternative between 20 and 60K since it is inexpensive compared to other complex compounds. But to go below 20K, usage of rare earth compounds is unavoidable.

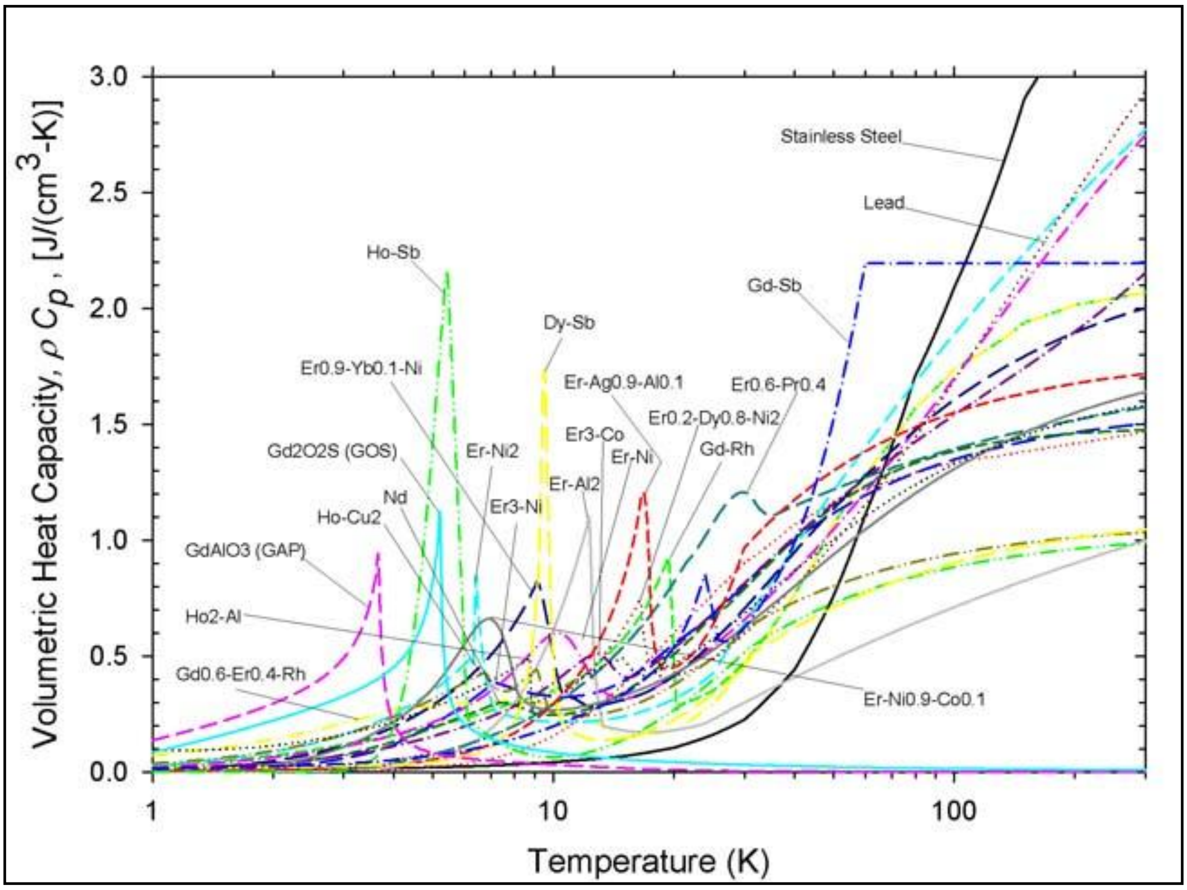


Figure 1.12: Volumetric heat capacity variation of various regenerator materials at cryogenic temperatures (NIST, 2005)

1.4. Motivation and Scope of the Thesis

All of the mentioned advantages and popularity of the wire mesh screen made it an attractive research subject. To predict the performance of different regenerator configurations by software saves time and decreases the cost of the research. Sage (Gedeon, 1995) and REGEN (O'Gallagher, Gary, Radebaugh, & Marquardt, 2001) are the two most commonly used programs to do this prediction.

Sage is a graphical interface that supports simulation and optimization of an underlying class of engineering models. The underlying model class represents something like a spring-mass-damper resonant system, a stirling-cycle machine, or anything else that has been properly coded to work with Sage (Gedeon, 2009).

The model classes of Sage are not just fixed-geometry models. Each may contain an unlimited number of variations or instances. A model instance, or just plain model for short, is a particular collection of component building blocks, connected and assembled in a particular way, with particular data values, forming a complete system representing whatever it is tried to be simulated. In other words, not just numerical data values are added within the confines of a presumed geometry. The geometry can be modified too. Each particular instance of a given model class resides in its own disk file with a unique name but common file extension (such as .stl for stirling models).

Despite its advantages, Sage is a commercial software, so it is not free of charge. Since it is originally written in Pascal and upgraded in Delphi (Gedeon, 2009), (which are rather old programming languages) it is not very user friendly. To build the model, making the connections, and getting the system running needs considerable effort. Even changing a parameter value is tricky.

REGEN is a free software developed by the cryogenic technologies group scientists (Cryogenics Technologies Group, 1995), whom led by Dr. Ray Radebaugh, in NIST. In this software physical model of a regenerator is a tube filled with a porous medium. An oscillatory flow of helium passes though the void space in the porous medium or matrix. The fluid is alternatively heated and cooled as the flow direction is reversed. The model is based on a numerical solution of the one dimensional equations for the flow of the helium gas through a porous matrix with an additional thermal conservation equation for the temperature of the matrix. The regenerator domain is subdivided into cells with the gas and matrix temperature computed for each cell along with the mass flux and pressure in the gas. A finite difference approximation is used to convert the system of differential equations into discrete equations that are marched forward in time until a nearly time-periodic solution is obtained. A correlation is used to obtain the pressure drop in the gas due to viscous flow through the porous matrix and another correlation is used for the heat transfer between the gas and matrix (O’Gallagher, Gary, Radebaugh, & and Marquardt, 2008).

REGEN can provide accurate results, but the program is not visual in terms of input/output. Similar to Sage, first version of REGEN was developed in 1980s (O’Gallagher, Gary, Radebaugh, & and Marquardt, 2008), probably that is the reason why both of these programs fails to meet today’s visualization demands.

In this thesis work, thermal partial differential equations are obtained for a regenerator which is formed by wire mesh screen discs, these equations are solved numerically by finite difference method. In order to evaluate it from every aspect, an easy to use Matlab graphical user interface (GUI) is also prepared.

CHAPTER 2

MODELLING AND SOLUTION

2.1. Introduction

While modeling a cryocooler regenerator in order to obtain a solution, basic concept that needs to be dealt with is the exchange of the thermal energy between the matrix and fluid. The necessary equations describing this exchange are obtained from Ackermann's book (Ackermann, 1997). They are the equation for the conservation of mass, and the equations of the motion for the fluid. These equations establish the energy balance for the fluid and the matrix material.

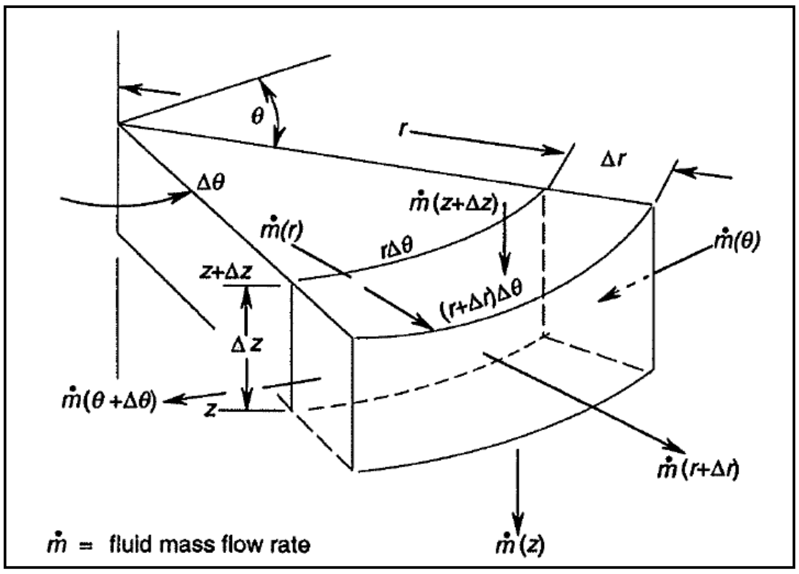


Figure 2.1: Regenerator control volume (Ackermann, 1997)

2.1.1. Continuity Equation

As a first step for development, the control volume in Figure 2.1, ΔV, is considered with a mass, ΔM, which includes fluid and matrix masses:

$$\Delta M = \Delta M_\mathrm{m} + \Delta M_\mathrm{f} = \rho_\mathrm{m} \delta V_\mathrm{m} + \rho_\mathrm{f} \delta V_\mathrm{f} \tag{2.1}$$

Where the control volume is given by

$$\Delta V = \Delta r (r\Delta \theta)(\Delta z) \tag{2.2}$$

Equation of continuity states that the net mass change in control volume is equal to the difference between the mass entering and leaving the control volume per unit time. This can be expressed as:

$$\frac{\partial(\rho \Delta V)}{\partial t} = \sum mass\ inflow - \sum mass\ outflow \tag{2.3}$$

From Figure 2.1, equations for the mass flows that crosses the control volume boundaries are

$$\begin{aligned} \sum mass\ inflow &= \dot{m}(r) + \dot{m}(\theta) + \dot{m}(z) \\ &= \rho(w_r)(r\Delta\theta)(\Delta z) + \rho(w_\theta)(\Delta r)(\Delta z) + \rho(w_z)(r\Delta\theta)(\Delta r) \end{aligned} \tag{2.4}$$

$$\sum mass\ outflow = \dot{m}(r+\Delta r) + \dot{m}(\theta + \Delta\theta) + \dot{m}(z + \Delta z) \tag{2.5}$$

To relate outflow to inflow, following Taylor expansion is used.

$$f(n+\Delta n) = f(n) + \frac{df(n)}{dn}(\Delta n) + \frac{d^2 f(n)}{dn^2}(\Delta n)^2 + \cdots \tag{2.6}$$

If we let control volume approach to zero, following equation is obtained for the mass outflow

$$\begin{aligned}\sum mass\ outflow &= \rho(w_r)(r\Delta\theta)(\Delta z) + \rho(w_\theta)(\Delta r)(\Delta z) + \rho(w_z)(r\Delta\theta)(\Delta r) \\ &+ \frac{\partial(\rho w_r)}{\partial r}(\Delta r)(r\Delta\theta)(\Delta z) + \frac{1}{r}\frac{\partial(\rho w_\theta)}{\partial\theta}(r\Delta\theta)(\Delta r)(\Delta z) \\ &+ \frac{\partial(\rho w_z)}{\partial z}(\Delta z)(r\Delta\theta)(\Delta r)\end{aligned} \tag{2.7}$$

Substituting (2.4) and (2.7) into (2.3) continuity for a compressible fluid can be obtained as

$$\frac{\partial(\rho\Delta V)}{\partial t} + \left[\frac{\partial(\rho w_r)}{\partial r} + \frac{1}{r}(\rho w_r) + \frac{1}{r}\frac{\partial(\rho w_\theta)}{\partial\theta} + \frac{\partial(\rho w_z)}{\partial z}\right]\Delta V = 0 \tag{2.8}$$

For the flow in a regenerator, where small pressure changes occur, no work is done by, or on, the fluid, as it flows throughout the regenerator, meaning ΔV is constant. The fluid velocities are well below the speed of sound and working pressures are generally at moderate level which points out that the flow is incompressible (Ackermann, 1997). The conditions above lead to the continuity equation for the flow defined as;

$$\frac{\partial(\rho w_r)}{\partial r} + \frac{1}{r}(\rho w_r) + \frac{1}{r}\frac{\partial(\rho w_\theta)}{\partial\theta} + \frac{\partial(\rho w_z)}{\partial z} = 0 \tag{2.9}$$

2.1.2. Energy Equation

Considering the control volume in Figure 2.1, heat inflow during *dt* and heat generated by internal sources during *dt* must be equal to heat outflow during *dt* and the change in internal energy during *dt*. Here inflow and outflow consists of two things. These are, the conducted heat through the matrix material, dQ_c, in and out of control volume and transported heat, dQ_f. Frictional heating dQ_η is the source of internal energy change and it is given by the mass, and the enthalpy of the material enclosed by the control volume (ΔM de), which can be expressed as

$$\begin{bmatrix} dQ_c + dQ_f \end{bmatrix}\begin{matrix} r \\ \theta \\ z \end{matrix} + \begin{bmatrix} dQ_\eta \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} = \begin{bmatrix} dQ_c + dQ_f \end{bmatrix}\begin{matrix} r+\Delta r \\ \theta+\Delta\theta \\ z+\Delta z \end{matrix} + \begin{bmatrix} M\,de \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} \tag{2.10}$$

For matrix material, energy balance in the control volume is

$$\begin{bmatrix} dQ_c \end{bmatrix}\begin{matrix} r \\ \theta \\ z \end{matrix} + \begin{bmatrix} dQ_h \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} = \begin{bmatrix} dQ_c \end{bmatrix}\begin{matrix} r+\Delta r \\ \theta+\Delta\theta \\ z+\Delta z \end{matrix} + \begin{bmatrix} (\rho c_p \delta V)_m dT_m \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} \tag{2.11}$$

And for the fluid

$$\begin{bmatrix} dQ_f \end{bmatrix}\begin{matrix} r \\ \theta \\ z \end{matrix} + \begin{bmatrix} dQ_\eta - dQ_h \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} = \begin{bmatrix} dQ_f \end{bmatrix}\begin{matrix} r+\Delta r \\ \theta+\Delta\theta \\ z+\Delta z \end{matrix} + \begin{bmatrix} (\rho c_p \delta V)_f dT_f \end{bmatrix}\begin{matrix} r+\Delta r/2 \\ \theta+\Delta\theta/2 \\ z+\Delta z/2 \end{matrix} \tag{2.12}$$

$$dQ_h = h\delta A_s\left(T_m - T_f\right)dt \tag{2.13}$$

Where dQ_h denotes the heat transfer between the matrix and fluid, h denotes the heat transfer coefficient between the fluid and matrix, δA_s represents the heat transfer area of the matrix contained within the element, and temperature difference between matrix and fluid is denoted by (T_m - T_f).

While developing these equations, it is assumed that no conduction occurs through fluid. Also the pressure change is small so, specific heat at constant pressure is valid.

2.1.2.1. The Matrix Energy Equation

In Equation (2.11) the heat added to the matrix volume δV_m consists of the convection across the fluid - matrix boundary and the conduction through matrix material. These two terms can be expressed as follows.

The convective term

$$\frac{dQ_h}{dt} = h\delta A_s\left(T_m - T_f\right) \tag{2.14}$$

Here T_f is the fluid temperature just outside of the matrix element surroundings, δA_s is the heat transfer surface area of the matrix in contact with the fluid element.

The conductive term

By Fourier equation, dQ_c is defined as,

$$\frac{dQ_c}{dt} = \dot{Q}_c = -(K\delta A_\eta)_m \frac{\partial T_m}{\partial \eta} \tag{2.15}$$

Where thermal conductivity is denoted by K_m and the surface area of the element is denoted by $A_{\eta m}$. Summing $\dot{Q}_c$ across each of the faces, and employing the Taylor series to equate the heat leaving the elemental volume to the heat entering, the equation for the conduction of heat through the matrix is

$$\begin{aligned} q_c = \left[\frac{\partial}{\partial r}\left(K_m \frac{\partial T_m}{\partial r}\right) + \frac{K_m}{r}\frac{\partial T_m}{\partial r} + \frac{\partial}{\partial \theta}\left(\frac{K_m}{r^2}\frac{\partial T_m}{\partial \theta}\right) + \frac{\partial}{\partial z}\left(K_m \frac{\partial T_m}{\partial z}\right)\right] \\ \times (\delta r\, r\delta\theta\, \delta z) \end{aligned} \tag{2.16}$$

Substituting (2.15) and (2.16) into (2.11) matrix thermal equation can be obtained as below

$$\begin{aligned} \left[\frac{\partial}{\partial r}\left(K_m \frac{\partial T_m}{\partial r}\right) + \frac{K_m}{r}\frac{\partial T_m}{\partial r} + \frac{\partial}{\partial \theta}\left(\frac{K_m}{r^2}\frac{\partial T_m}{\partial \theta}\right) + \frac{\partial}{\partial z}\left(K_m \frac{\partial T_m}{\partial z}\right)\right] \times \delta V_m \\ + h\delta A_s\left(T_f - T_m\right) = (\rho c_p \delta V)_m \frac{\partial T_m}{\partial t} \end{aligned} \tag{2.17}$$

2.1.2.2. Fluid Energy Equation

Three terms are associated with the fluid heat, convection between the matrix and the fluid, frictional heating, and the heat crosses the boundary of the elemental volume. Related equations are as follows:

The convective term

As expressed before convective term is

$$\frac{dQ_h}{dt} = h\delta A_s\left(T_f - T_m\right) \tag{2.18}$$

Frictional heating

Frictional heating is derived as (Schlichting, 1955)

$$\left[\frac{dQ_h}{dt}\right]\begin{matrix} r + \Delta r/2 \\ \theta + \Delta\theta/2 \\ z + \Delta z/2 \end{matrix} = (\delta V_f)\mu\Phi \tag{2.19}$$

where μ denotes fluid viscosity and Φ denotes the dissipation function defined by the velocity gradients as

$$\begin{aligned}\Phi = 2\left[\left(\frac{\partial w_r}{\partial r}\right)^2 + \frac{1}{r}\left(\frac{\partial w_\theta}{\partial \theta}\right)^2 + \frac{w_r}{r^2}\left(\frac{\partial w_\theta}{\partial \theta}\right) + \left(\frac{\partial w_z}{\partial z}\right)^2\right] \\ + \left[\frac{\partial w_\theta}{\partial r} + \frac{1}{r}\left(\frac{\partial w_r}{\partial \theta}\right)\right]^2 - \frac{\partial w_\theta}{r}\left[\frac{\partial w_\theta}{\partial r} + \frac{1}{r}\left(\frac{\partial w_r}{\partial \theta}\right)\right] + \left[\frac{\partial w_r}{\partial z} + \frac{\partial w_z}{\partial r}\right]^2 \\ + \left[\frac{\partial w_\theta}{\partial z} + \frac{1}{r}\left(\frac{\partial w_z}{\partial \theta}\right)\right]^2\end{aligned} \tag{2.20}$$

Complex structure of frictional heating makes it very difficult to obtain a solution for thermal fluid equation. Ackermann (1997), observed in his experiments that, frictional heating is ten orders magnitude smaller than total heat transfer. So it is justifiable to neglect it.

Heat transported across the boundary

The rate at which heat, dQ_f / dt, is transported across the boundaries of the element is defined by

$$\frac{dQ_f}{dt} = (\rho w \delta A_n)_f c_{pf} T_f \tag{2.21}$$

where A_{nf} is the surface area of the element through which heat is transported by the fluid. Equating the inflow to the outflow through the Taylor series gives

$$\begin{aligned}&\left[\frac{dQ_f}{dt}\right]\begin{matrix}r\\ \theta\\ z\end{matrix} - \left[\frac{dQ_f}{dt}\right]\begin{matrix}r+\Delta r/2\\ \theta+\Delta\theta/2\\ z+\Delta z/2\end{matrix}\\ &= -(\delta V_f)\left[\frac{\partial}{\partial r}(\rho w_r c_p T)_f + \frac{(\rho w_r c_p T)_f}{r} + \frac{1}{r}\frac{\partial}{\partial \theta}(\rho w_r c_p T)_f + \frac{\partial}{\partial z}(\rho w_r c_p T)_f\right]\end{aligned} \tag{2.22}$$

Differentiating the term $(\rho w_r c_p T)_f$ and applying the continuity equation (2.9), general form of the heat transported into the control volume is obtained as below.

$$\begin{aligned}&\left[\frac{dQ_f}{dt}\right]\begin{matrix}r\\ \theta\\ z\end{matrix} - \left[\frac{dQ_f}{dt}\right]\begin{matrix}r+\Delta r/2\\ \theta+\Delta\theta/2\\ z+\Delta z/2\end{matrix}\\ &= -(\rho \delta V_f)\left[w_r\frac{\partial}{\partial r}(c_p T)_f + \frac{w_\theta}{r}\frac{\partial}{\partial \theta}(c_p T)_f + w_z\frac{\partial}{\partial z}(c_p T)_f\right]\end{aligned} \tag{2.23}$$

Substituting (2.18) and (2.23) into (2.12), fluid thermal equation is obtained as follows,

$$\begin{aligned}&h\delta A_s(T_f - T_m)\\ &+ (\rho\delta V)_f\left[w_r\frac{\partial}{\partial r}(c_p T)_f + \frac{w_\theta}{r}\frac{\partial}{\partial \theta}(c_p T)_f + w_z\frac{\partial}{\partial z}(c_p T)_f\right]\\ &= -(\rho c_p \delta V)_f\frac{\partial T_f}{\partial t}\end{aligned} \tag{2.24}$$

2.1.3. The Ideal Regenerator

The thermal equations derived for regenerator in previous section are too complex for closed form solutions that none exists. In order to obtain a manageable form, some common assumptions can be made. Regenerator will be considered as static type which is generally the case in reciprocating coolers. Packing wire mesh screens in a cylindrical housing (Figure 2.2) and closing both ends with a cap, completes the construction of the regenerator. The heating period, which is half of the cycle, starts as the hot fluid enters to the regenerator from the right side. The remaining half is named as cooling period where cold fluid enters from the other end and travels to the other end. This sequence is periodically reversed.

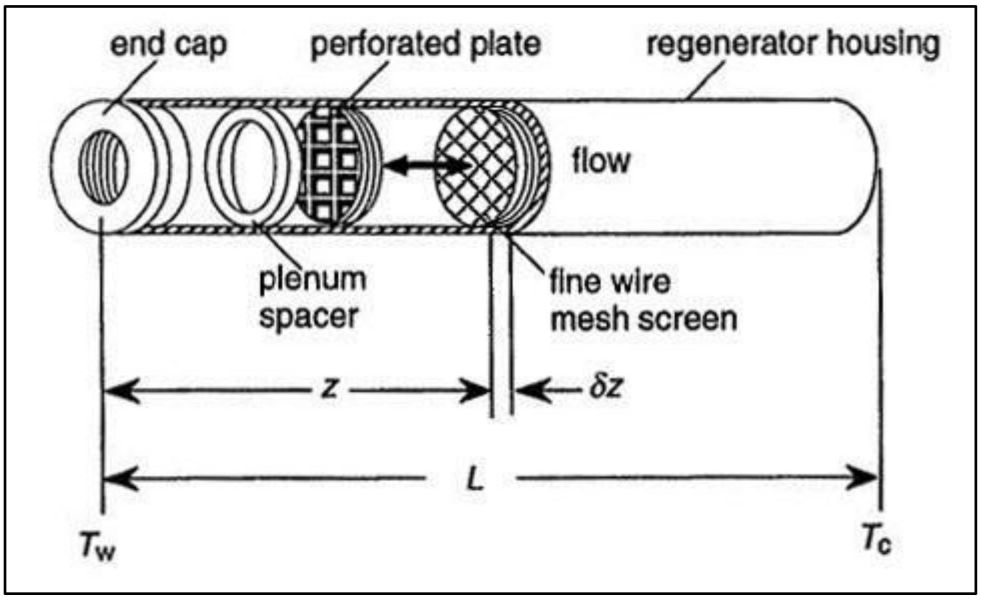


Figure 2.2: Static regenerator layout (Ackermann, 1997)

In an ideal regenerator the constant temperature hot fluid enters the regenerator, gives up its heat to the matrix and leaves with a lower temperature at the cold end. The flow of hot fluid is then cut off. After all of the hot fluid leaves the regenerator from the opposite end, the flow is reversed, with cold gas entering the cold end with constant temperature. The cold gas cools the matrix as it leaves the regenerator with a variable warmer temperature at the hot end. After several cycles, steady state condition is reached. In this condition at any location in the regenerator, matrix and gas temperatures will be the same in every cycle while temperature distribution will vary periodically with time.

Ideal regenerator assumptions are (Ackermann, 1997):

1. Heat stored in the matrix material is much greater compared to the heat stored in the fluid.
$$\frac{(\rho c_P \delta V)_f}{(\rho c_P \delta V)_m} \ll 1$$

   Heat capacity of the matrix is nearly three orders of magnitude greater than the heat capacity of the fluid around 80-300 K range. But as the temperature decreases this assumption loses its validity, as the ratio shown above is around 0.1 for 6 K where helium is the operating fluid.
2. The flow is one dimensional
3. Thermal conductivity of the matrix is zero in the longitudinal direction and infinite in the radial and circumferential directions.
4. The fluid and matrix properties are constant with temperature.
5. The heat transfer coefficient between the fluid and the matrix is constant throughout the regenerator.
6. The fluids pass in counterflow directions.
7. Entering fluid temperatures are uniform over the flow cross section and constant with time.
8. Regular periodic conditions are established for all matrix elements.
9. No mixing of the fluids occurs during the reversal from hot to cold flows.

With these assumptions continuity equation (2.9), becomes

$$\frac{\partial(\rho w_z)}{\partial z} = 0 \qquad \dot{m} = \rho A_{ff} w_z = constant \tag{2.25}$$

where A_{ff} is the fluid axial free flow area. The matrix thermal equation reduces (2.17), to

$$h\delta A_s (T_f - T_m) = (Mc_p)_m \frac{\partial T_m}{\partial t} \tag{2.26}$$

The fluid thermal equation (2.24), reduces to

$$hA_s(T_f - T_m) = (\rho c_p \delta V)_f w_z \frac{\partial T_f}{\partial z} \tag{2.27}$$

2.1.4. Boundary and Initial Conditions

The formulation derived above is completed with definition of boundary and initial conditions. Cooling and heating periods are designated with subscripts h and c, respectively. Position in the longitudinal direction z, is measured from the entrance point of the fluid to the matrix.

Boundary Conditions

- Over heating period the fluid enters the warm end of the regenerator with constant warm temp T_w, where λ_h donates heating flow period;

$$(T_f)_h(0,\lambda_h) = T_w$$

- Over cooling period the fluid enters the cold end of the regenerator with constant cold temp, T_c, where λ_c donates cooling flow period;

$$(T_f)_c(0,\lambda_c) = T_c$$

Initial Condition

- The temperature at any point in the matrix at the end of one period is equal to that at the same point at the beginning of the next period.

$$(T_m)_h(z,0) = (T_m)_c(L-z, \lambda_c)$$

$$(T_m)_c(z,0) = (T_m)_h(L-z, \lambda_h)$$

Together with above conditions there are two more conditions. First, for cooling and heating periods, the mass flow rate entering the regenerator is equal and constant. And second, thermal conduction at the boundaries of the regenerator is zero;

$$\left(\frac{\partial T_m}{\partial r}\right)_{r=R} = \left(\frac{\partial T_m}{\partial z}\right)_{z=0} = \left(\frac{\partial T_m}{\partial z}\right)_{z=L} = 0$$

2.2. Numerical Solution For the Ideal Regenerator Case

2.2.1. Finite Difference Equations

Based on the idealization in previous chapter, the open form solution is developed from the ideal regenerator equations:

Matrix thermal equation

$$h\delta A_s\left(T_f - T_m\right) = (\rho c_p \delta V)_m \frac{\partial T_m}{\partial t} \tag{2.28}$$

Fluid thermal equation

$$h\delta A_s\left(T_f - T_m\right) = -(\rho c_p \delta V)_f w_z \frac{\partial T_f}{\partial z} \tag{2.29}$$

As a starting point, regenerator and fluid are divided into nodes as shown in Figure 2.3:

$$i = 1,2,3,\ldots,N_z$$

Second the heating and cooling periods are subdivided into a series of small time intervals:

$$j = 1,2,3,\ldots, N_t$$

Third the first order derivatives are replaced by the finite differences

$$\lim_{\Delta z \to 0} \frac{f\left(z + \frac{1}{2}\Delta z\right) - f\left(z - \frac{1}{2}\Delta z\right)}{\Delta z} = f'(z)$$

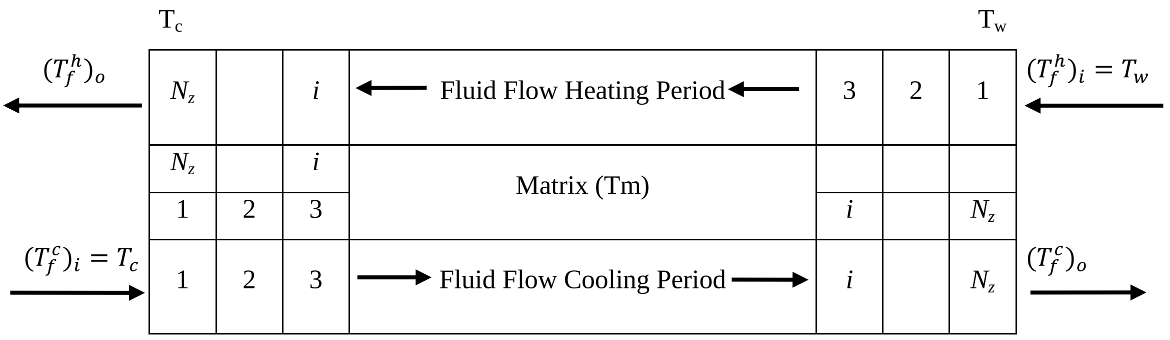


Figure 2.3: Regenerator finite element (Ackermann, 1997)

With these definitions, the ideal regenerator equations (2.28) and (2.29) can be expressed as the following finite difference equations:

$$\frac{T_m(t + \Delta t, z) - T_m(t, z)}{\Delta t} = \left[\frac{hA_s}{(Mc_p)_m}\right]_{\Delta z} \overline{\Delta T} \qquad (2.30)$$

$$\frac{T_f\left(t, z + \frac{1}{2}\Delta z\right) - T_f\left(t, z - \frac{1}{2}\Delta z\right)}{\Delta z} = -\left[\frac{hA_s}{(\dot{m}c_p)_f}\right]_{\Delta z} \overline{\Delta T} \qquad (2.31)$$

where $\overline{\Delta T}$ the average temperature difference between the fluid and the matrix over time interval is

$$\overline{\Delta T} = \frac{1}{2}\left[T_f\left(t, z + \frac{\Delta z}{2}\right) + T_f\left(t, z - \frac{\Delta z}{2}\right)\right] - \frac{1}{2}\left[T_m(t + \Delta t, z) + T_m(t, z)\right] \qquad (2.32)$$

Using an easier to follow notation temperature equations (2.30), (2.31) and (2.32) can be expressed respectively as

$$\Delta T_m = [(T_m)_o - (T_m)_i] = [T_m(t+\Delta t, z) - T_m(t, z)] \tag{2.33}$$

$$\Delta T_f = \left[(T_f)_o - (T_f)_i\right] = \left[T_f\left(t, z+\frac{\Delta z}{2}\right) - T_f\left(t, z-\frac{\Delta z}{2}\right)\right] \tag{2.34}$$

$$\overline{\Delta T} = \frac{1}{2}\left[(T_f)_o + (T_f)_i\right] - \frac{1}{2}[(T_m)_o + (T_m)_i] \tag{2.35}$$

where *i* and *o* designates the inlet and outlet temperatures across the spatial and time intervals.

2.2.2. Parallel Flow Model

Parallel flow model treats each node as a parallel-flow heat exchanger with the equivalent of a metal stream with the characteristics of the matrix material, flowing in parallel with the fluid stream. Considering the heating period the curves (Figure 2.4) depict the temperature difference

$$(\Delta T)_o = (T_f)_o - (T_m)_o \tag{2.36}$$

that exists between the fluid and matrix material as the fluid leaves the nodal element Δ*z* after warming the matrix material. Because the matrix temperature, $T_m(t)$ is changing uniformly over the length of the element, the fluid flowing through the element would respond accordingly with the temperature change, T_f (Figure 2.4). Thus, if the matrix material

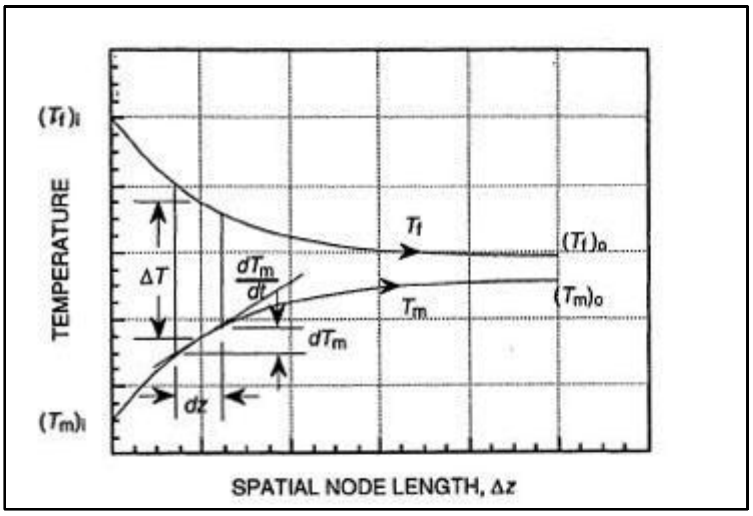


Figure 2.4: Parallel flow numerical model (Ackermann, 1997)

contained in Δz is considered to correspond to a second stream with a nodal thermal capacity $(C_r)_{\Delta z}$,

$$(C_r)_{\Delta z} = \left[\frac{(Mc_p)_m}{\Delta t}\right]_{\Delta z} \tag{2.37}$$

the pattern of the temperature changes occurring in the element is analogous to that observed in a parallel-flow heat exchanger. For a parallel-flow heat exchanger the temperature difference between the two streams across the element is given by

$$\Delta T = \left(T_f - T_m\right) = \Delta T_i \left\{ exp\left[-\frac{(hA_s)_{\Delta z}}{(\dot{m}c_p)_f}\right]\left[1 + \frac{(\dot{m}c_p)_f \Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\frac{z}{\Delta z}\right\} \tag{2.38}$$

where $(\Delta T)_i=(T_f)_i-(T_m)_i$.

It is also known from Figure 2.4 that

$$\frac{dT_m}{dt} = \frac{dT_m}{dz}\frac{\Delta z}{\Delta t}$$

By substituting these expressions into the ideal regenerator equations, (2.26) and (2.27), following is obtained.

$$-\Delta z\frac{dT_f}{dz} = \frac{(hA_s)_{\Delta z}}{(\dot{m}c_p)_f}\Delta T_i \left\{ exp\left[-\frac{hA_s}{(\dot{m}c_p)_f}\right]\left[1 + \frac{\left(\dot{m}c_p\right)\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\frac{z}{\Delta z}\right\} \tag{2.39}$$

$$\Delta z\frac{dT_m}{dz} = \frac{(hA_s)_{\Delta z}\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\Delta T_i \left\{ exp\left[-\frac{hA_s}{(\dot{m}c_p)_f}\right]\left[1 + \frac{\left(\dot{m}c_p\right)\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\frac{z}{\Delta z}\right\} \tag{2.40}$$

Integrating these equations across the nodal element, Δz, yields the two finite difference equations:

$$\left(T_f\right)_o = \left(T_f\right)_i - K_1\left[\left(T_f\right)_i - (T_m)_i\right] \tag{2.41}$$

$$(T_m)_o = (T_m)_i + K_2\left[\left(T_f\right)_i - (T_m)_i\right] \tag{2.42}$$

where the constants K_1 and K_2 are

$$K_1 = \frac{1}{\left\{1 + \frac{(\dot{m}c_p)_f \Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right\}}\left\{1 - exp\left[-\frac{(hA_s)_{\Delta z}}{(\dot{m}c_p)_f}\right]\left[1 + \frac{(\dot{m}c_p)_f \Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\right\} \tag{2.43}$$

$$K_2 = \frac{1}{\left\{1 + \frac{\left[(Mc_p)_m\right]_{\Delta z}}{(\dot{m}c_p)_f \Delta t}\right\}}\left\{1 - exp\left[-\frac{(hA_s)_{\Delta z}}{(\dot{m}c_p)_f}\right]\left[1 + \frac{(\dot{m}c_p)_f \Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\right\} \tag{2.44}$$

Temperature distribution in the regenerator is computed by the iterative solutions of equations (2.41) and (2.42) according to the schematic shown in Figure 2.3. The nodal geometry for the heating and cooling periods and the relationship between the fluid and matrix nodes are described in the figure. Equations (2.41) and (2.42) are solved for each of the spatial Nz and time nodes N_t for heating and cooling periods. The calculations proceeded by assuming an initial temperature distribution for the matrix material between T_w and T_c, where superscript h denotes the heating period and superscript c denotes the cooling period,

$$[T'_m(1,i)]_{initial} = T_w + \frac{(i-1)}{N_z}[T_w - T_c] \qquad i = 1,2,3 \ldots N_z$$

Setting the first node's inlet temperature equal to the heating period boundary condition:

$$T_f^h(j,1) = T_w \qquad j = 1,2,3, \ldots N_t$$

and calculating the outlet temperatures $[T_m^h(1,i)]_o$ and $[T_m^h(1,i)]_o$ using equations (2.41) and (2.42). Repeating the computational process for each node (1,2,3…N_z) by using the calculated outlet temperatures as the input temperatures for the next nodal calculation completes the calculations for the first time step (see Appendix A.1). To complete the solutions of the heating period, this stepwise procedure must be repeated for all spatial nodes and time steps, which means a matrix of N_z by N_t will be solved. After completing the calculations for the heating period, the final matrix temperatures are used as the initial conditions for the start of the cooling period, the reversal conditions:

$$T_m^c(1,i) = T_m^h(\lambda_h, \zeta)$$

where $\zeta = [N_z - (i\text{-}1)]$ and the outlet temperatures are computed by setting the first node's inlet temperatures equal to the cooling period boundary condition:

$$T_f^c(j,1) = T_c \qquad j = 1,2,3, \dots N_t$$

This procedure is repeated for each period until steady state behavior of the temperature distributions is achieved. Steady state behavior occurs when the matrix temperature distribution becomes cyclic and reversal condition, consisting of similar fluid and matrix temperature distributions, exists at the beginning of the heating and at the end of the cooling period.

2.3. Numerical Solution For the Nonideal Cases

2.3.1. Including Effects of Longitudinal Conduction

The contact between wire mesh screen layers causes a thermal longitudinal conduction. The nonideal effects of the longitudinal thermal conduction can be included by extending the open-form analysis. This can be achieved by also using difference equations to replace the higher-order differential equations in the conduction terms. With longitudinal conduction, the matrix thermal equation (2.28) becomes:

$$\frac{\partial}{\partial z}\left(K_m \frac{\partial T_m}{\partial z}\right)\delta V_m + h\delta A_s\left(T_f - T_m\right) = (\rho c_p \delta V)_m \frac{\partial T_m}{\partial t} \tag{2.45}$$

For a finite regenerator element of width Δ*z*, the matrix volume is defined by the matrix material area normal to the flow, A_m, and the differential volume with,

$$\delta V_m = A_m \Delta z$$

The matrix conductivity, K_m, is defined by interfacial conductance between matrix elements, which was experimentally determined (Ackermann, 1997) for a screen matrix as,

$$K_m = K_{m0}\left(\frac{T_m}{300\ \mathrm{K}}\right)^a \tag{2.46}$$

$$\frac{\partial K_m}{\partial T_m} = \frac{aK_m}{T_m} \tag{2.47}$$

The value of the constant *a* is changes with the regenerator material. It is taken as 0.8 for stainless steel screens, and 0.97 for phosphor bronze. Substitution of the volume and the conductance terms into equations (2.45) and (2.29) gives the finite element matrix and fluid thermal equations:

Matrix thermal equation

$$\begin{gathered}\left[\frac{aK_m}{T_m}\left(\frac{\partial T_m}{\partial z}\right)^2 + K_m \frac{\partial^2 T_m}{\partial z^2}\right] A_m \Delta z + (hA_s)_{\Delta z}\left(T_f - T_m\right) \\ = \left[(Mc_p)_m\right]_{\Delta z} \frac{\partial T_m}{\partial t}\end{gathered} \tag{2.48}$$

Fluid thermal equation

$$(hA_s)_{\Delta z}\left(T_f - T_m\right) = -(\dot{m}c_p)_f \frac{\partial T_f}{\partial z} \tag{2.49}$$

The open-form solution to equations (2.48) and (2.49) is found by replacing the derivatives with the finite central difference equations:

$$f'(z) = \frac{1}{2(\Delta z)}[f(z+\Delta z) - f(z-\Delta z)] \tag{2.50}$$

$$f''(z) = \frac{1}{(\Delta z)^2}[f(z+\Delta z) - 2f(z) + f(z-\Delta z)] \tag{2.51}$$

Note that central difference equations will not work for the first and last node. So for the first node, forward difference equations;

$$f'(z) = \frac{1}{(\Delta z)}[f(z+\Delta z) - f(z)] \tag{2.52}$$

$$f''(z) = \frac{1}{(\Delta z)^2}[f(z+2\Delta z) - 2f(z+\Delta z) + f(z)] \tag{2.53}$$

and for the last node backward difference equations;

$$f'(z) = \frac{1}{(\Delta z)}[f(z) - f(z-\Delta z)] \tag{2.54}$$

$$f''(z) = \frac{1}{(\Delta z)^2}[f(z-2\Delta z) - 2f(z-\Delta z) + f(z)] \tag{2.55}$$

are used.

Then, using the parallel-flow approach to express the temperature difference between the fluid and matrix (T_f - T_m), and using the finite difference equations to replace the conduction differentials, the nonlinear second-order matrix differential equation is reduced to an ordinary differential equation:

$$\begin{aligned}\Delta z\frac{dT_m}{dz} &= \frac{(hA_s)_{\Delta z}\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\Delta T_i\left\{exp\left[-\frac{hA_s}{(\dot{m}c_p)_f}\right]\left[1+\frac{(\dot{m}c_p)\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}}\right]\frac{z}{\Delta z}\right\} \\ &\quad + \left[\frac{A_{mn}\Delta t}{\left[(Mc_p)_m\right]_{\Delta z}\Delta z}\right]\left[\frac{aK_m}{4T_m}(T'_m)_i^2 + K_m(T''_m)_i\right]\end{aligned} \tag{2.56}$$

where the conduction finite temperature differences $(T'_m)_i$ and $(T''_m)_i$ are defined (using central difference except first and last node) by the temperature difference between the node that the calculation is being performed on and its surrounding nodes at the beginning of the time interval calculation:

$$\frac{dT_m}{dz} = \frac{1}{2(\Delta z)}\left[T_m(t,z+\Delta z) - T_m(t,z-\Delta z)\right] = \frac{1}{2(\Delta z)}(T'_m)_i \tag{2.57}$$

$$\begin{aligned}\frac{d^2T_m}{dz^2} &= \frac{1}{(\Delta z)^2}\left[T_m(t,z+\Delta z) - 2T_m(t,z) + T_m(t,z-\Delta z)\right] \\ &= \frac{1}{(\Delta z)^2}(T''_m)_i\end{aligned} \tag{2.58}$$

Integrating the above matrix equation produces the algebraic matrix temperature equation for the new matrix temperature at the end of the time interval Δt based on the initial matrix and fluid temperatures:

$$(T_m)_o = (T_m)_i + K_2\left[(T_f)_i - (T_m)_i\right] + K_3\left[\frac{a}{4(T_m)_i}(T'_m)_i^2 + (T''_m)_i\right] \tag{2.59}$$

The constant K_2 is calculated using equation (2.44) and K_3 with

$$K_3 = \frac{K_m A_m \Delta t}{\left[(Mc_p)_m\right]_{\Delta z} \Delta z} \tag{2.60}$$

The fluid equation (2.41) is the same as ideal case, where K_1 is calculated from equation (2.43).

2.3.2. Including Wall Effects

More significant effect is the increase in inefficiency caused by heat transfer with the regenerator wall. The wall effect occurs when a metallic housing with large heat capacity is used to contain the matrix material. In this case, the heat transfer between the fluid and wall, and the longitudinal conduction along the wall, introduces an irreversible heat transfer from the matrix to the wall that reduces the effectiveness of the matrix material. The addition of the wall introduces a third differential equation into the analysis that describes both the heat transfer between the fluid and the wall, and the longitudinal conduction along the wall, where the wall thermal conductivity for stainless steel can be expressed as (Ackermann, 1997)

$$K_{wl} = K_{wl0}\left(\frac{T_{wl}}{300\ \mathrm{K}}\right)^{b} \qquad\qquad b = 0.38$$

Considering the wall effect, the three differential equations describing the heat transfer in a small element of the regenerator are:

Matrix thermal equation

$$\begin{gathered}\left[\frac{aK_m}{T_m}\left(\frac{\partial T_m}{\partial z}\right)^2 + K_m \frac{\partial^2 T_m}{\partial z^2}\right] A_m \Delta z + (hA_s)_{\Delta z}\left(T_f - T_m\right) \\ = \left[(Mc_p)_m\right]_{\Delta z} \frac{\partial T_m}{\partial t}\end{gathered} \tag{2.61}$$

Wall thermal equation

$$\left[\frac{bK_{wl}}{T_{wl}}\left(\frac{\partial T_{wl}}{\partial z}\right)^2 + K_{wl}\frac{\partial^2 T_{wl}}{\partial z^2}\right]A_{wl}\Delta z + (hA_{wl})_{\Delta z}\left(T_f - T_m\right) = \left[(Mc_p)_{wl}\right]_{\Delta z}\frac{\partial T_{wl}}{\partial t} \tag{2.62}$$

Fluid thermal equation

$$(hA_s)_{\Delta z}\left(T_f - T_m\right) + (hA_{wl})_{\Delta z}\left(T_f - T_{wl}\right) = (\dot{m}c_p)_f\frac{\partial T_f}{\partial z} \tag{2.63}$$

The solution for equations (2.61), (2.62), and (2.63) follows the same procedures described above with the matrix initial and boundary conditions also used for the wall.

Regarding equations are;

$$(T_m)_o = (T_m)_i + K_2\left[\left(T_f\right)_i - (T_m)_i\right] + K_3\left[\frac{a}{4(T_m)_i}(T_m')_i^2 + (T_m'')_{\boldsymbol{i}}\right] \tag{2.64}$$

$$(T_{wl})_o = (T_{wl})_i + K_5\left[\left(T_f\right)_i - (T_{wl})_i\right] + K_6\left[\frac{a}{4(T_{wl})_i}(T_{wl}')_i^2 + (T_{wl}'')_{\boldsymbol{i}}\right] \tag{2.65}$$

$$\left(T_f\right)_o = \left(T_f\right)_i - K_1\left[\left(T_f\right)_i - (T_m)_i\right] - K_4\left[\left(T_f\right)_i - (T_{wl})_i\right] \tag{2.66}$$

where,

$$K_4 = \frac{1}{\left\{1 + \frac{\left(\dot{m}c_p\right)_f\Delta t}{\left[\left(Mc_p\right)_{wl}\right]_{\Delta z}}\right\}}\left\{1 - exp\left[-\frac{(hA_{wl})_{\Delta z}}{\left(\dot{m}c_p\right)_f}\right]\left[1 + \frac{\left(\dot{m}c_p\right)_f\Delta t}{[(Mc_{wl})_m]_{\Delta z}}\right]\right\} \tag{2.67}$$

$$K_5 = \frac{1}{\left\{1 + \frac{\left[(Mc_p)_{wl}\right]_{\Delta z}}{(\dot{m}c_p)_f \Delta t}\right\}} \left\{1 - exp\left[-\frac{(hA_{wl})_{\Delta z}}{(\dot{m}c_p)_f}\right]\left[1 + \frac{(\dot{m}c_p)_f \Delta t}{\left[(Mc_p)_{wl}\right]_{\Delta z}}\right]\right\} \tag{2.68}$$

$$K_6 = \frac{K_{wl} A_{wl} \Delta t}{\left[(Mc_p)_{wl}\right]_{\Delta z} \Delta z} \tag{2.69}$$

and (again using central difference except first and last node).

$$\frac{dT_{wl}}{dz} = \frac{1}{2(\Delta z)}\left[T_{wl}(t, z + \Delta z) - T_{wl}(t, z - \Delta z)\right] = \frac{1}{2(\Delta z)}\left(T'_{wl}\right)_{\boldsymbol{i}} \tag{2.70}$$

$$\begin{aligned}\frac{d^2T_{wl}}{dz^2} &= \frac{1}{(\Delta z)^2}\left[T_{wl}(t, z + \Delta z) - 2T_{wl}(t, z) + T_{wl}(t, z - \Delta z)\right] \\ &= \frac{1}{(\Delta z)^2}\left(T''_{wl}\right)_{\boldsymbol{i}}\end{aligned} \tag{2.71}$$

Equations for three cases (ideal, with longitudinal conduction, with longitudinal conduction and wall effect) are solved with the described method using a computer code developed in Matlab in this thesis work.

CHAPTER 3

1D REGENERATOR SIMULATOR

3.1. Introduction

Up to now, matrix and fluid thermal equations are derived using energy balance and continuity equation. Simplified versions of these equations are obtained for three different cases. Details of finite difference method, which will be used for solution of the obtained, PDEs are also presented. In this chapter, details of the regenerator parameters are given within the context of the graphical user interface that has been prepared. Also the equations and the approach that has been followed while obtaining the results are explained.

3.2. GUI

Graphical user interface (GUI) allows user to change regenerator parameters without making any change within the code. In this section, information about the parameters that can be adjusted by user with using the GUI (Figure 3.1) is presented.

3.2.1. Matrix Properties

3.2.1.1. Matrix Material

From matrix material menu, the material of the matrix screens can be chosen. It can be either phosphor-bronze or stainless steel. It is possible to increase the number of the options by just adding the material properties of the new materials to the code. By choosing the matrix material, density, thermal conduction constants, specific heat capacity values of the mesh is set.

3.2.1.2. Screen Diameter

It is possible to set the diameter of the discs from this menu. Together with the wall thickness this parameter determines the diameter of the displacer. It has also effects on the mass and the total thermal capacity of the regenerator.

Minimum value for the screen diameter is 3mm, where as the maximum is 28mm.

3.2.1.3. Wire Diameter

Diameter of the wires that are used to form the discs can be set from this menu. It is possible to go down to 0.01 mm. Since values above 0.11 mm are not practical, it is taken as the upper limit.

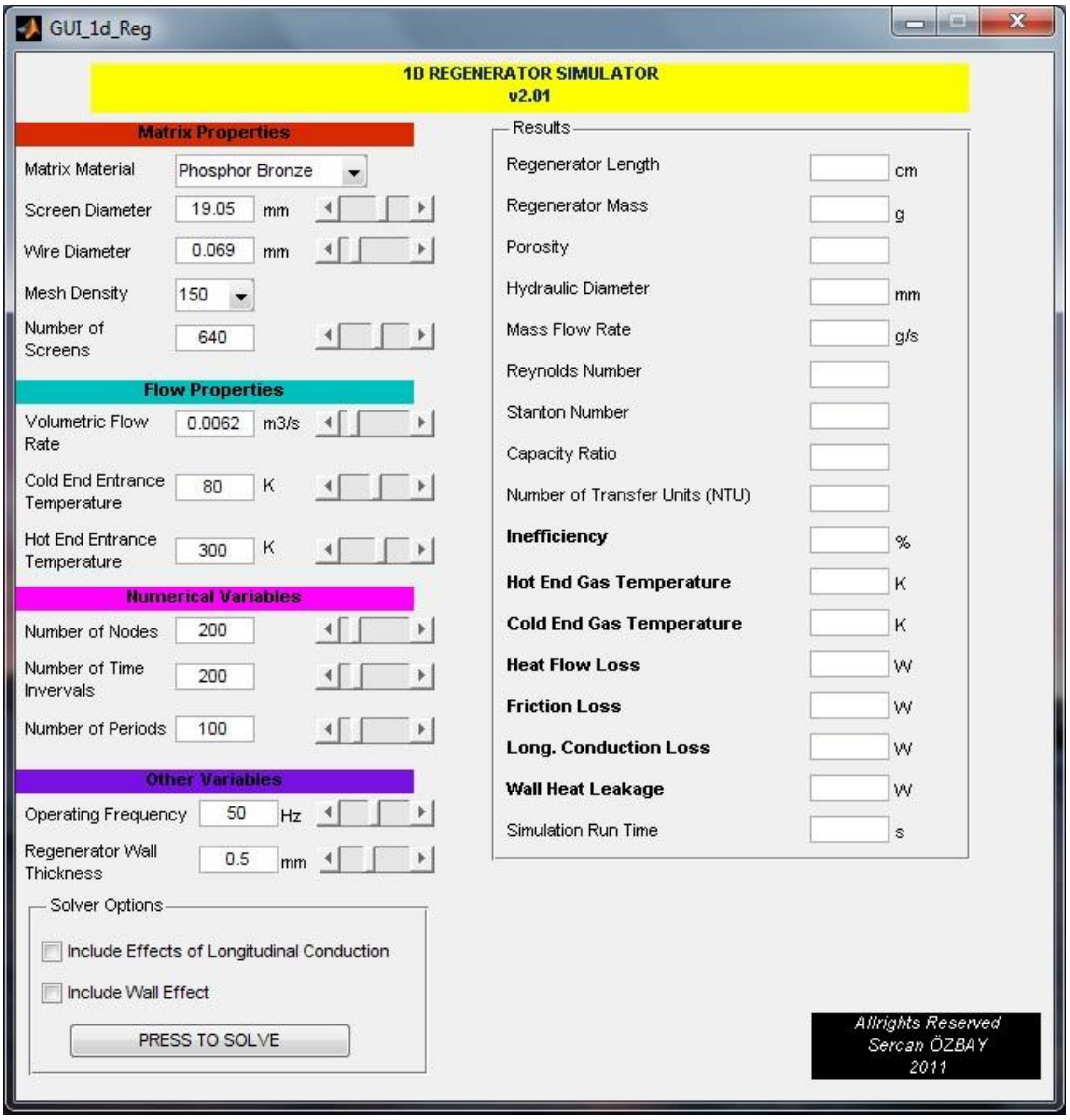
Diameter of the screen wires affects porosity, hydraulic radius, screen thickness, and regenerator mass directly.

3.2.1.4. Mesh Density

Mesh density defines the number of openings in a screen disc. As an industry standard, it is defined as number of openings per inch square. It has a strong effect on porosity.

3.2.1.5. Number of Screens

This term sets the number of screens that will be stacked on top of each other to form the regenerator. Since it is practically impossible to manipulate the alignment of the discs, they are stacked randomly. Thus, regenerator obtains a porous medium form rather than micro heat exchanger. Together with the disc thickness the number of screens determines the length of the regenerator. It has also an obvious effect on the regenerator mass.



GUI_1d_Reg

1D REGENERATOR SIMULATOR
v2.01

Matrix Properties

Matrix Material: Phosphor Bronze
Screen Diameter: 19.05 mm
Wire Diameter: 0.069 mm
Mesh Density: 150
Number of Screens: 640

Flow Properties

Volumetric Flow Rate: 0.0062 m3/s
Cold End Entrance Temperature: 80 K
Hot End Entrance Temperature: 300 K

Numerical Variables

Number of Nodes: 200
Number of Time Inverval s: 200
Number of Periods: 100

Other Variables

Operating Frequency: 50 Hz
Regenerator Wall Thickness: 0.5 mm

Solver Options
- [ ] Include Effects of Longitudinal Conduction
- [ ] Include Wall Effect

PRESS TO SOLVE

Results

Regenerator Length: cm
Regenerator Mass: g
Porosity:
Hydraulic Diameter: mm
Mass Flow Rate: g/s
Reynolds Number:
Stanton Number:
Capacity Ratio:
Number of Transfer Units (NTU):
Inefficiency: %
Hot End Gas Temperature: K
Cold End Gas Temperature: K
Heat Flow Loss: W
Friction Loss: W
Long. Conduction Loss: W
Wall Heat Leakage: W
Simulation Run Time: s

Allrights Reserved
Sercan ÖZBAY
2011

Figure 3.1: The graphical user interface of the 1-D regenerator simulator

3.2.2. Flow Properties

3.2.2.1. Flow Rate

This box is used to set the volumetric fluid flow rate. The fluid is helium by default. It is quite rare for any other fluid to be used in state of art cryocoolers so fluid type is not optional.

3.2.2.2. Cold End Entrance Temperature

As mentioned in section 2.1.3, the code works on a constant end temperature assumption. That constant value is set from here. The lowest value in this box is 40 K, because below 40K many parameters that have been assumed to be constant will vary significantly with decreasing temperature. And since cryogenic coolers are defined for temperatures below 120K upper limit is set as such.

3.2.2.3. Hot End Entrance Temperature

The hot end temperature is usually taken as the environment temperature of the cooler. It is set as 300K by default (standard room temperature). Upper limit is 350K and lower limit is 250K.

3.2.3. Numerical Variables

3.2.3.1. Number of Nodes

As it is stated earlier, the simplified versions of the continuity, matrix and fluid thermal equations are solved by finite difference method. The regenerator is divided into nodes and the equations are solved for each node. To obtain an accurate solution minimum 50 of these nodes should be used. Increasing nodes, leads to more accurate results but increases simulation run time also. The numerical variables are bounded by the physical memory of the computer. After several trials, it is observed that increasing number of nodes more than 550 do not improve the result. So it is set as the upper limit.

3.2.3.2. Number of Time Intervals

Change in the temperature of a node in the matrix not only will affect the neighboring nodes in the matrix but also the fluid node at that point. The implicit approach first takes care of the change in the matrix nodes (excluding the change in fluid) and then the fluid nodes. The small amount of time passes between these two steps is named as time interval. Minimum time interval is set to 20 to obtain a meaningful solution again by experimenting. Maximum value is 520.

3.2.3.3. Number of Periods

The flow in the regenerator is assumed to be periodically steady. The heating period is followed by the cooling period. Combination of heating and cooling periods completes one cycle. Number of periods indicates the number of these cycles. Generally minimum of 20 cycles need to be completed to observe a steady state behavior. Some cases may need more, but increasing this value beyond 220 does not help much.

3.2.4. Other Properties

3.2.4.1. Operating Frequency

Operating frequency indicates the number of cycles completed per second. In a real cryocooler this parameter is usually between 20 and 60 Hz. Covering and extending this range, values between 1-100 Hz is accepted in this work.

3.2.4.2. Regenerator Wall Thickness

Regenerator disc screens require a structure to hold them together in line. Such a structure is usually made of stainless steel (wall material selection is not optional). Wall thickness must be as small as possible. But the housing also needs to be strong and perfectly flat. That is why, smaller values than 0.1mm is not preferred. Increasing wall thickness beyond 1mm increases inefficiency too much so that is not preferred as well.

3.2.5. Solver Options

3.2.5.1. Include Effects of Longitudinal Conduction

Wire mesh screens are in contact with each other at some ambiguous points. This causes a thermal longitudinal conduction along the regenerator matrix which introduces a loss in the mechanism. To obtain a more realistic solution, this loss can be taken into account by checking the associated box. To include effects of the longitudinal thermal conduction, it is required to solve more complex differential equation set. As a result the computation time and required computer memory space increases.

3.2.5.2. Include Wall Effect

Another source of inefficiency is the heat transferred to and along the regenerator wall. This wall effect occurs when a metallic housing with large heat capacity is used to contain the matrix material. Similar to effects of longitudinal thermal conduction case, by including wall effect more realistic solutions are obtained in the expense of increased computational time and required computational memory space.

3.2.6. Results

3.2.6.1. Regenerator Length

Ideally regenerator length (r_L) is equal to the screen thickness (t_{sc}) multiplied by number of regenerator discs (N_s). And also for a woven mesh screen (Figure 3.2), screen thickness is equal to the length of two the wire diameter (w_d).

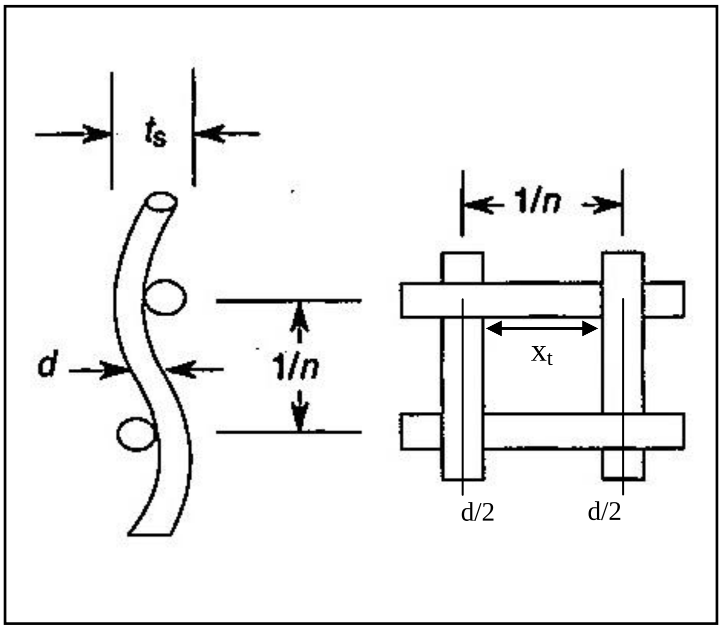


Figure 3.2: Geometry of woven screen (Ackermann, 1997)

But in reality, due to inclination caused by weaving, screens are 5-15% thicker than the ideal case. This value (t_F) is determined experimentally (Harvey, 1999) and to calculate the screen thickness following equation is used. In this code, t_F is taken as 1.1.

$$t_{sc} = 2w_d\, t_F \tag{3.1}$$

Mesh stacking density is another parameter that affects the regenerator length. Since it is not possible to stack the screens perfectly, small gaps will appear between them. This effect is taken care of by a constant (c_F) which can vary between 1.03 and 1.08 (Harvey, 1999). 1.044 is chosen as an average value in this work.

$$r_L = N_s t_{sc} c_F \tag{3.2}$$

3.2.6.2. Regenerator Mass

Mass of the regenerator (M) is calculated by the following equation:

$$M = A_r r_L d_m (1 - \alpha) \tag{3.3}$$

Here A_r donates regenerator frontal area, d_m donates density of mesh material and α donates porosity.

3.2.6.3. Porosity

Porosity (α) is defined as;

$$\alpha \equiv \frac{\text{total volume of connected void spaces}}{\text{total volume of matrix}}$$

Analytically the porosity is calculated by considering a small segment of screen with a transverse pitch x_t (Figure 3.2), designating the transverse spacing between wires and a lateral pitch, x_l, designating the longitudinal spacing between wires. Referring to Figure 3.2 the pitches are related to mesh density (n) and screen thickness by

$$\frac{1}{n} = x_t + d \tag{3.4}$$

If a perfect stacking of square mesh screens in which weaving causes no inclination of the wires and the screen layers are not separated, these idealizations lead to a matrix packing where screen thickness is equal to $2w_d$ and the ideal porosity is given by

$$\alpha = 1 - \frac{2\left[\left(\frac{\pi}{4}\right) w_d^2\right](x_t + d)}{(x_t + d)^2 (2d)} = 1 - \frac{\pi w_d n}{4} \tag{3.5}$$

As mentioned above, the result obtained from above equation is for ideal case. This value is multiplied by t_F and used as such.

3.2.6.4. Hydraulic Diameter

Hydraulic diameter (d_h) is defined as follows (Ackermann, 1997);

$$\mathrm{d_h} = \frac{2\alpha}{\beta} \tag{3.6}$$

where, β, area density is

$$\beta \equiv \frac{\textit{total surface area of the connected voids}}{\textit{total volume of the matrix}}$$

3.2.6.5. Total Wetted Area

Total wetted area (A_s) is the total area that heat transfer between the fluid and the matrix occurs.

$$A_s = \beta A_r r_L \tag{3.7}$$

Here A_r donates frontal area of a screen disc.

3.2.6.6. Matrix Heat Capacity

Regenerator matrix heat capacity (C_m) is calculated simply by multiplying specific heat capacity (C_{p_m}) of the used material with the mass of regenerator.

$$C_m = Mc_{p\_m} \tag{3.8}$$

3.2.6.7. Capacity Ratio

The fluid heat capacity rate ratio measures the thermal imbalance of the flow streams (Ackermann, 1997):

$$C \equiv Fluid\ heat\ capacity\ rate\ ratio \equiv \frac{C_{min}}{C_{max}} \tag{3.9}$$

where C_{min} and C_{max} are the smaller and the larger of the two magnitudes C_h and C_c

$$C_h = (\dot{m}c_p)_h = warm\ fluid\ heat\ capacity\ rate$$

$$C_c = (\dot{m}c_p)_c = cold\ fluid\ heat\ capacity\ rate$$

The matrix capacity ratio (C_r, capacity ratio in short) measures the thermal capacity of the matrix relative to the minimum flow stream capacity:

$$C_r = \frac{C_m}{C_{min}} = \frac{Mc_{p\_m}}{\lambda(\dot{m}c_p)_{min}} \tag{3.10}$$

Here, λ donates flow period i.e. time required for one complete cycle.

The larger the matrix capacity ratio, the smaller is the matrix temperature swing and, in general, the more efficient the regenerator.

3.2.6.8. Number of Transfer Units

The number of heat transfer units (NTU) is a nondimensional expression that is related to a heat exchanger's "heat transfer size". When NTU is small the exchanger effectiveness is low and when the NTU is large the effectiveness approaches a limit physically imposed by flow and thermodynamic considerations. It is defined as (Ackermann, 1997)

$$NTU = \frac{A_s\bar{U}}{C_{min}} \tag{3.11}$$

where

$$\frac{1}{A_s\bar{U}} = \left[\left(\frac{1}{\bar{h}A_s}\right)_c + \left(\frac{1}{\bar{h}A_s}\right)_h\right]$$

A_s stands for matrix total heat transfer area, h is convective heat transfer coefficient and U is overall heat transfer conductance between the warm and cold fluids.

3.2.6.9. Inefficiency

The effectiveness defines how well a real heat exchanger is performing relative to an ideal exchanger operating across the same temperature differences. "Effectiveness" and "efficiency" are used interchangeably, and are defined as (Ackermann, 1997)

$$\varepsilon = \frac{Q}{Q_{ideal}} \tag{3.12}$$

where Q is the actual heat exchange between fluids, and Q_{ideal} is an ideal amount that could be exchanged if no temperature difference existed between the inlet and outlet streams. Thus, referring to Figure 3.3, the two energy terms can be written as

$$Q = \left(\dot{m}c_p\right)_h \lambda_h (T_{in} - \bar{T}_{out})_w = \left(\dot{m}c_p\right)_c \lambda_c (\bar{T}_{out} - T_{in})_c \tag{3.13}$$

$$Q_{ideal} = \left(\dot{m}c_p\right)_{min} \lambda (T_{w,in} - T_{c,in}) \tag{3.14}$$

Substituting the above terms into the effectiveness equation provides the general expression for a heat exchanger's thermal performance:

$$\varepsilon = \frac{C_h(T_{in} - \bar{T}_{out})_w}{C_{min}(T_{w,in} - T_{c,in})} = \frac{C_c(\bar{T}_{out} - T_{in})_c}{C_{min}(T_{w,in} - T_{c,in})} \tag{3.15}$$

For the particular case where the flows are balanced $C_h/C_c=1$, the effectiveness reduces to

$$\varepsilon = \frac{T_{w,in} - \bar{T}_{c,out}}{T_{w,in} - T_{c,in}} \tag{3.16}$$

In cryogenics, very often the regenerator performance is given in terms of an inefficiency, Ie, where

$$\varepsilon = 1 - Ie$$

$$Ie = 1 - \frac{C_h(T_{in} - \bar{T}_{out})_w}{C_{min}(T_{w,in} - T_{c,in})} \tag{3.17}$$

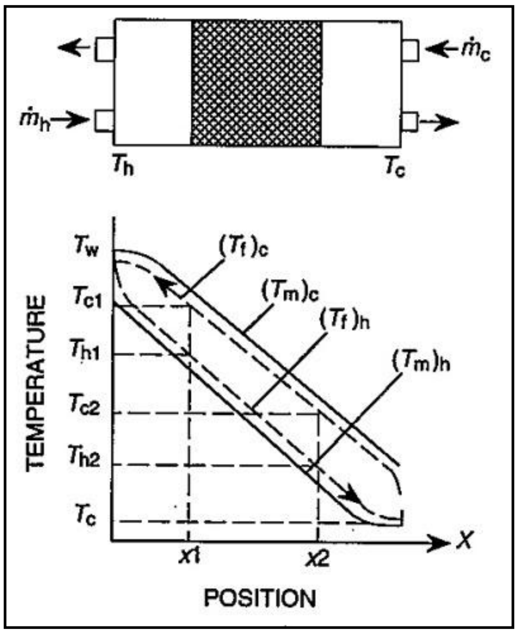


Figure 3.3: Fluid temperature distribution (Ackermann, 1997)

3.2.6.10. Hot End Gas Temperature

After the simulation completed, value of the fluid node at the hot end is presented.

3.2.6.11. Cold End Gas Temperature

After the simulation completed, value of the fluid node at the cold end is presented.

3.2.6.12. Regenerator Thermal Loss Mechanisms

There are three main thermal loss mechanisms that exists in regenerator; heat flow loss, loss due to conduction in matrix material and wall heat leakage. The sum of these three is named as regenerator thermal loss.

Heat flow loss is the loss that is caused by the net enthalpy flow due to mass flux. In an ideal regenerator, it will be the only thermal loss. So for ideal case, it can be calculated with the regenerator thermal loss expression (Ackermann, 1997)

$$\dot{Q}_{reg} = \dot{m}c_p Ie(T_w - T_c) \tag{3.18}$$

But for the case where longitudinal conduction in the matrix is taken into account, regenerator thermal loss will include both flow loss and conduction loss. To distinguish one from another, the code first solves the problem as if it was the ideal case, and then moves on to longitudinal conduction case. The difference in the regenerator thermal loss between the two cases comes from the loss mechanism that is due longitudinal conduction.

As mentioned in section 2.3.2, wall heat leakage consists of the loss due to fluid-wall heat transfer and the thermal conduction along the wall. To calculate it, similar procedure is followed. The code first runs for the ideal case, then for the longitudinal conduction case. After these two, longitudinal conduction loss and heat flow loss is known. As a last step, code runs the case with wall effect and calculates the regenerator thermal loss. Using the results obtained from previous two runs, wall heat leakage value is calculated.

3.2.6.13. Friction Loss

Pressure gradient is estimated using the correlation given in the report prepared by NIST (O'Gallagher, Gary, Radebaugh, & and Marquardt, 2008). This is an empirical correlation for the data that Kays and London obtained. In the reference friction factor is defined in terms of two functions, *A(R)* and *Y(R)* where R is the Reynolds number. The pressure gradient is given by

$$\frac{\partial p}{\partial x} = \frac{2Y(R)R^2\mu^2}{A(R)D_h^3\rho} \tag{3.19}$$

And the constants *A(R)* and *Y(R)* are expressed as below, where α is porosity.

$$A(R)=0.0074R \qquad for \quad R\leq 10$$

$$A(R)=0.129-0.0058(\log(R/200))^2 \qquad for \quad 10<R\leq 3000$$

$$A(R)=0.149-0.0239\log(R/200) \qquad for \quad 10<R\leq 3000$$

$$Y(R)=aR^{-0.43}$$

$$a=0.715(5.6+\alpha(-16.363+\alpha 13.928))$$

Pressure drop along the regenerator is calculated by multiplying the gradient with the length of the regenerator. To obtain the energy loss in terms of watts, pressure drop is multiplied with the average mass flow rate of the fluid and divided by the density.

$$\dot{Q}_{fric}=r_L\frac{\partial p}{\partial x}\frac{\dot{m}}{\rho} \tag{3.20}$$

3.2.6.14. Simulation Run Time

Simulation run time box shows the simulation time i.e. measured time between the "Press to Solve" is clicked and, "DONE" is revealed.

CHAPTER 4

RESULTS AND DISCUSSION

4.1. Model Verification

4.1.1. Sage modeling

The model will be verified by using the Sage built in Stirling cycle, split type cryocooler model. An overview of the model is shown in Figure 4.1.

This model does not have an element that sets the flow rate in regenerator. So by playing around the compressor parameters and system pressure, the system is adjusted such that both regenerators (the one in this thesis work and the one in Sage model) operate under the same average mass flow rate.

The tests have been performed for three different regenerators. Properties for these are given in Table 4.1.

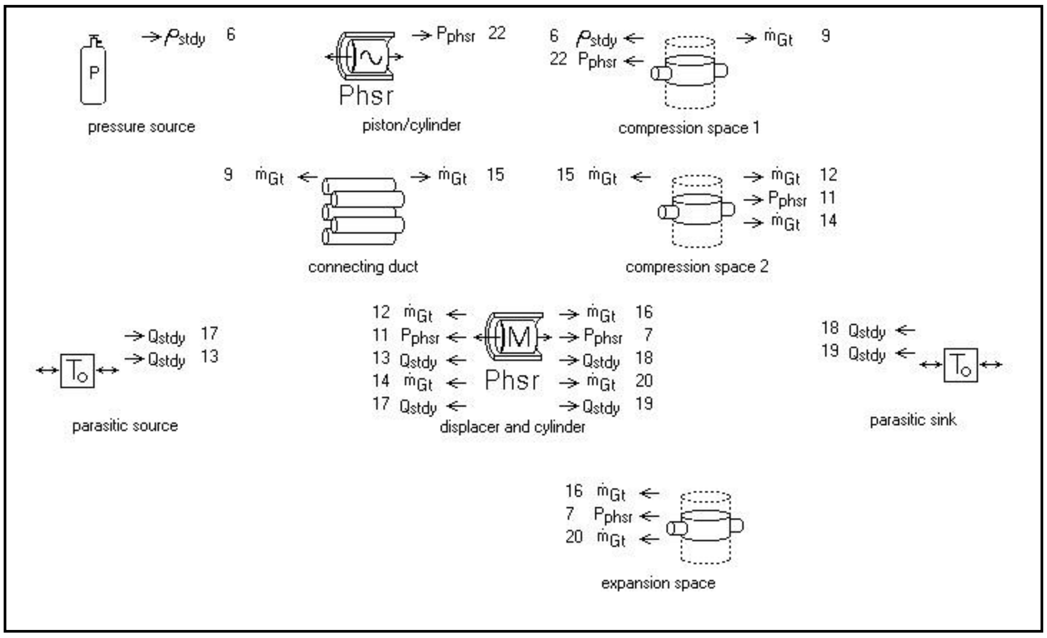


Figure 4.1: Sage model

Table 4.1: Properties of the regenerators used for validation

Regenerator Properties	**Reg-1**	**Reg-2**	**Reg-3**
Material	SS	SS	Ph-Br
Screen Diameter (mm)	10	12	19
Wire Diameter (mm)	0.035	0.045	0.07
Mesh Density	350	200	150
Porosity	0.62	0.72	0.68
Number of Screens	995	580	630
Regenerator Length (cm)	8	6	10.1
Volumetric Flow Rate (m^3/s)	0.0062	0.01	0.015
Mass Flow Rate (g/s)	0.93	1.5	2.25
Cold End Temp. (K)	80	80	80
Hot End Temp. (K)	300	300	300
Operating Frequency (Hz)	60	60	60
Wall Thickness (mm)	0.5	0.5	0.5

One thing to mention here is that length of the regenerator and porosity are given under results section of the GUI built but they are supplied as input in Sage. The methodology used in this work is closer to the industrial type perspective because it is not possible to order disc screens by porosity. And what determines the regenerator length is the number of disc stacks, not the other way around.

4.1.2. Comparison

Results of two programs for the regenerator configurations listed in Table 4.1 are presented in Table 4.2.

Table 4.2: Result comparison with Sage

Results	1D Regenerator Simulator			Sage		
	Reg-1	Reg-2	Reg-3	Reg-1	Reg-2	Reg-3
Regenerator Mass (g)	18.57	14.72	59.98	18.62	14.82	60.13
Hydraulic Diameter (mm)	0.063	0.128	0.16	-	-	-
Reynolds Number	84	164	131	84	162	130
Stanton Number	0.14	0.11	0.12	-	-	-
Capacity Ratio	138	68	64	-	-	-
NTU	356	100	149	-	-	-
Hot End Gas Temp (K)	298.2	297.1	297.7	331	329.2	330
Cold end Gas Temp (K)	82.2	83.3	82.7	93.7	103.7	98.6
Heat Flow Loss (W)	11.09	27.17	32.98	10.7	26.8	31.87
Friction Loss (W)	2.64	1.36	1.08	2.58	1.32	1.07
Long. Cond. Loss (W)	0.05	0.11	0.04	0.18	0.21	0.11
Wall Heat Leakage (W)	0.35	0.21	0.96	-	-	-
Inefficiency (%)	1.08	1.61	1.32	-	-	-

Regenerator mass is calculated by both programs with a maximum difference of 0.25%. Likewise Reynolds number predictions of the two have 1.3% difference ranges which is acceptable. Sage does not produce outputs for hydraulic diameter, Stanton number, capacity ratio or NTU so these parameters are left as blank.

The obvious deviation of two programs is about the end temperatures. Sage predicts the hot end temperature about 30K higher than the 1D Regenerator Simulator does. For the cold end this gap is between 10 and 20 K. Since in Sage whole system is modeled, heat generations at the bottom and top of the cold finger are also taken into account.

Loss mechanism outputs of the two programs are again close. Maximum difference of the heat flow loss results is 3.5% where maximum difference in friction loss is 3%. Longitudinal conduction is a bit underestimated by 1D Regenerator Simulator. Results differ from each other by more than %50 but, the trend is the same and the maximum difference is around 0.1W level. The reason may be the material property values i.e. thermal conduction coefficients that are being used. Since Sage is a commercial software, it does not give any information about what is happening in the background.

Wall heat leakage is not given in Sage. Instead another term, discrepancy loss is used but this term covers not only wall effect but also other losses. Therefore it is not possible to make any comparison.

Regenerator inefficiency is a term where, like in this thesis work, the regenerator itself is analyzed only (Ackermann, 1997). Authors prefer to give results such as cold end temperature values at certain heat loads or overall system efficiency. Sage is no different at this point. However to obtain these results, system level modeling must be done which is beyond the scope of this thesis.

Sage is a widely used program whose results show good match with experimental works that has been carried (Harvey, 2003), (Kashani, Helvensteijn, Kittel, Gschneidner, Pecharsky, & Pecharsky, 2001), (Wilson & Gedeon, 2003). It has a system modeling capacity. 1D Regenerator Simulator produces similar results, which are mostly within 5% range.

4.2. Effects of Numerical Parameters

To obtain results with minimum error, it is required to work at a point where the system is independent of the number of nodes, time intervals and period. As it can be seen from Figure 4.2, increasing the number of time intervals or the number of periods above 100 will do nothing other than increasing the computational time. Results in the upcoming sections are obtained under these conditions. However the case is not the same for the number of nodes. Until 200 nodes, inefficiency decreases sharply, and it seems to be settled around 450. In this work, 500 nodes are used to obtain the results in the following sections.

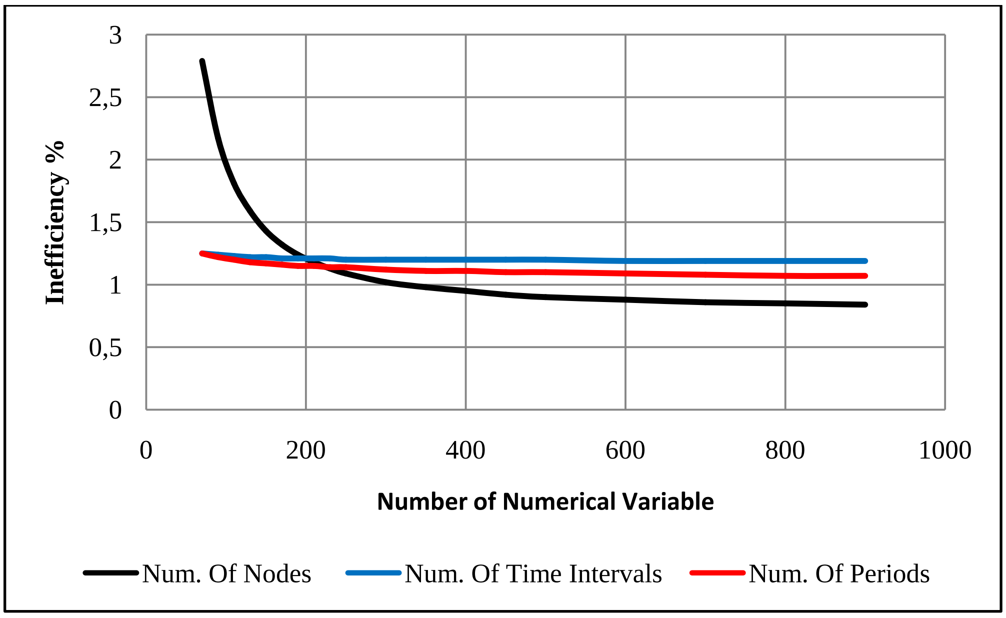


Figure 4.2: Effect of numerical variables on inefficiency

4.3. Effect of Matrix Properties

In this section effects of different matrix properties on inefficiency and loss mechanisms are investigated. All regenerator configurations are the same (except the varying parameter) and all of them are operated under the same conditions, unless otherwise stated. Default configuration and operation parameter values can be found in Table 4.3.

4.3.1. Matrix Material

Matrix materials are defined by three parameters; density, longitudinal conduction coefficient, and specific heat constant. Density and specific heat constant affect heat capacity therefore heat capacity rate ratio. As mentioned earlier, heat capacity rate ratio measures the thermal capacity of the matrix relative to the minimum flow stream capacity and it has a direct effect on inefficiency.

In Figure 4.3, heat capacity rate ratio of two regenerators that are formed from different materials for various flow rate values can be found.

Table 4.3: Default configuration and operating parameters

Material	SS
Screen Diameter (mm)	10
Wire Diameter (mm)	0.035
Mesh Density	300
Number of Screens	746
Volumetric Flow Rate (m^3/s)	0.007
Cold End Temp. (K)	80
Hot End Temp. (K)	300
Operating Frequency (Hz)	28
Wall Thickness (mm)	0.5
Number of Nodes	500
Number of Time Intervals	100
Number of Periods	100

Due to density difference of two metals, a big difference in heat capacity rate ratio values is observed at low flow rates. Increase in flow rate, increases heat capacity of the fluid, which lowers the heat capacity rate ratio values. The increase in flow rate also closes the gap between two materials' heat capacity rate ratio values but, even for 0.01 (m^3/s) case, stainless steel regenerator's heat capacity rate ratio is three times greater than the phosphor-bronze one.

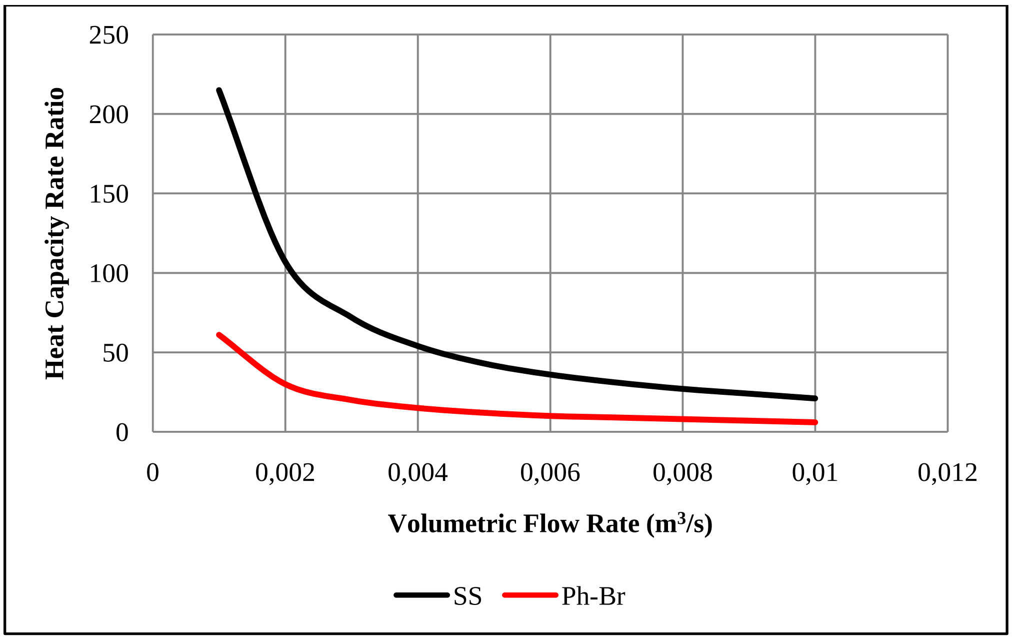


Figure 4.3: Heat capacity rate ratio comparison of stainless steel and phosphor-bronze screens

The difference in the heat capacity rate ratio of two materials affects inefficiency considerably (Figure 4.4). Stainless steel screens can resist more to the increase in fluid heat capacity, which results a less increase in inefficiency compared to the phosphor-bronze screens.

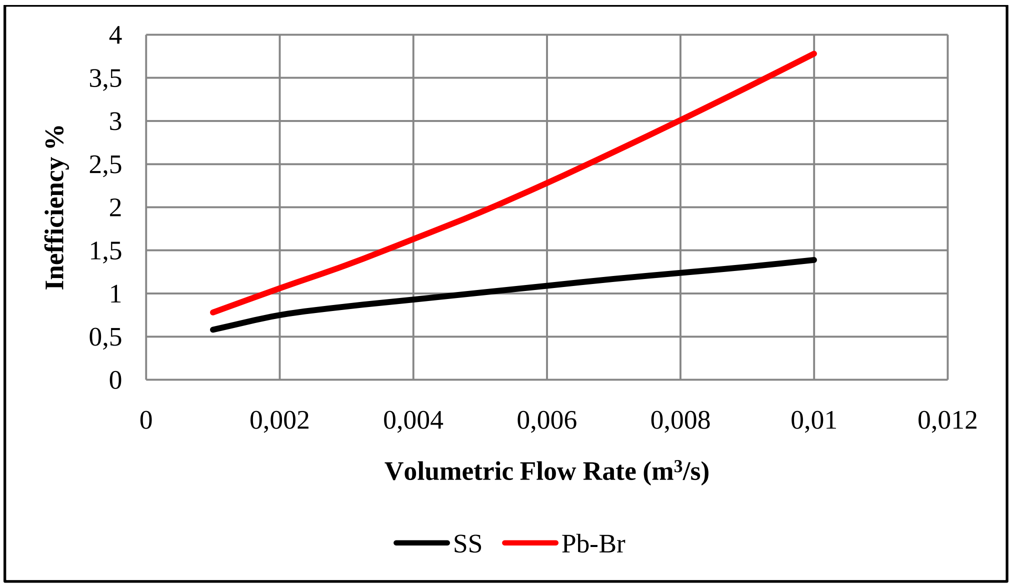


Figure 4.4: Inefficiency results for stainless steel and phosphor-bronze screens

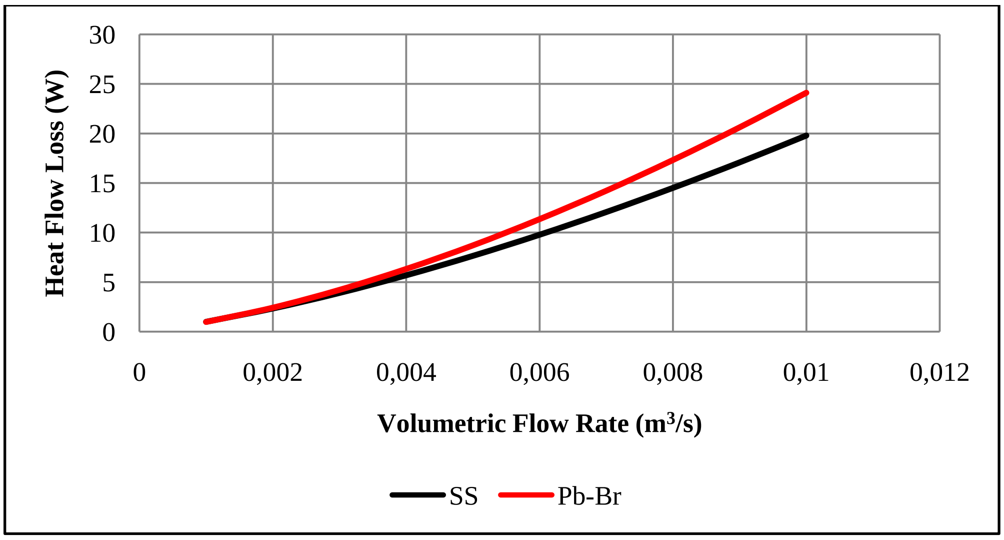


Figure 4.5: Heat flow loss for stainless steel and phosphorous bronze screens

However, not much of a difference exists from the heat loss perspective (Figure 4.5). This points out that, another loss mechanism must be dominant. Due to its lower mass, phosphor bronze screens suffer from the wall leakage (Figure 4.6). It is evident from the graph that as the flow rate increases the fluid exchanges more heat with the wall than it does with the regenerator.

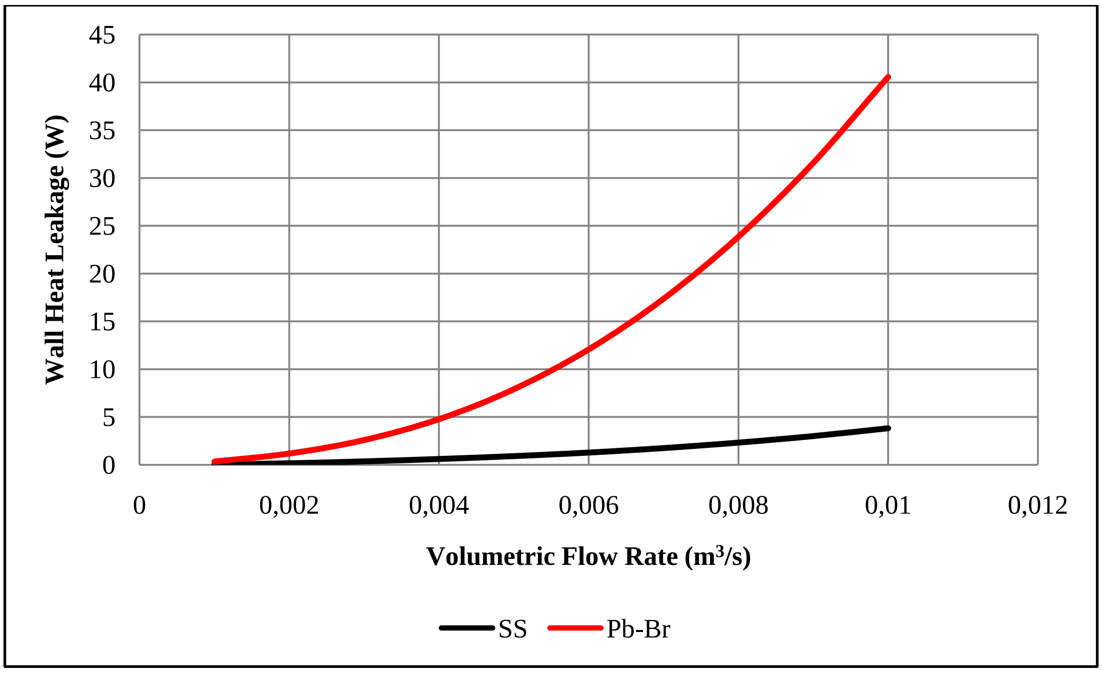


Figure 4.6: Wall heat leakage values for stainless steel and phosphor-bronze screens

4.3.2. Screen Diameter

In regenerator designs, screen diameter is generally determined by the cold tip area. The cold tip, which lifts heat from the required surface, has its area set according to requirements; therefore together with the wall thickness, the screen diameter will be set. In Figure 4.7, Reynolds number and number of transfer units (NTU) for different screen diameters can be seen. The screens are formed form 0.04 mm wires and have a mesh density of 250.

As the screen diameter increases, the flow velocity decreases since the flow rate is constant. Therefore Reynolds number decreases as the screen diameter increases. However, total heat transfer area will increase with the increasing screen diameter, which causes NTU to increase also. These affects can be observed in Figure 4.8.

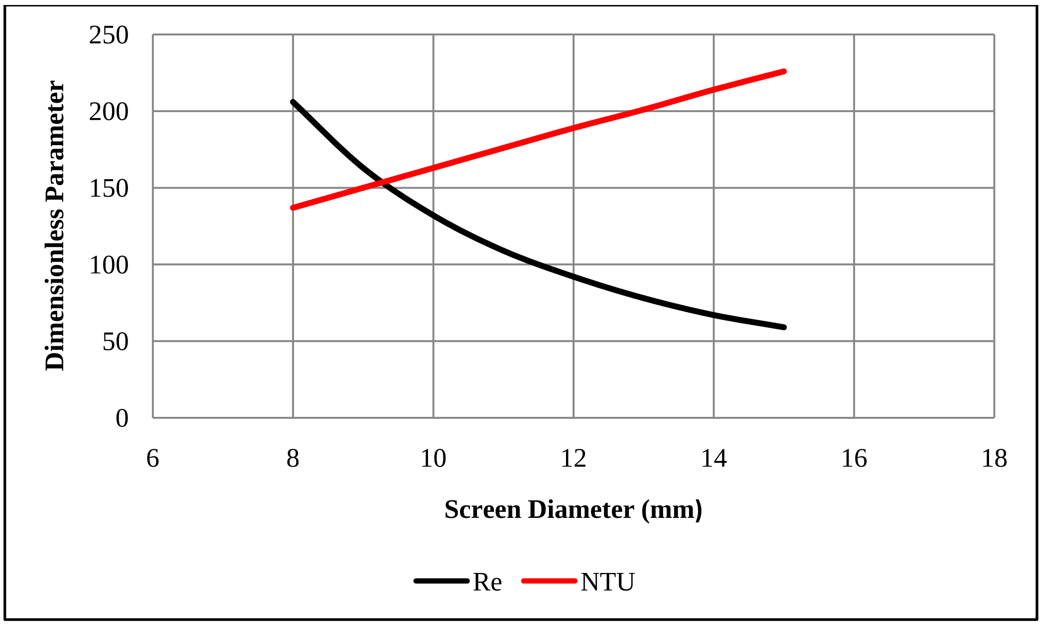


Figure 4.7: Variation of Reynolds number and number of transfer units with screen diameters

All of the loss mechanisms tend to decrease with increasing screen size except the longitudinal conduction loss (Figure 4.8 Figure 4.9). Heat flow loss decreases due to increasing heat capacity of the regenerator, likewise the wall heat leakage. Decrease in the flow velocity so as Reynolds number, decreases the friction loss.

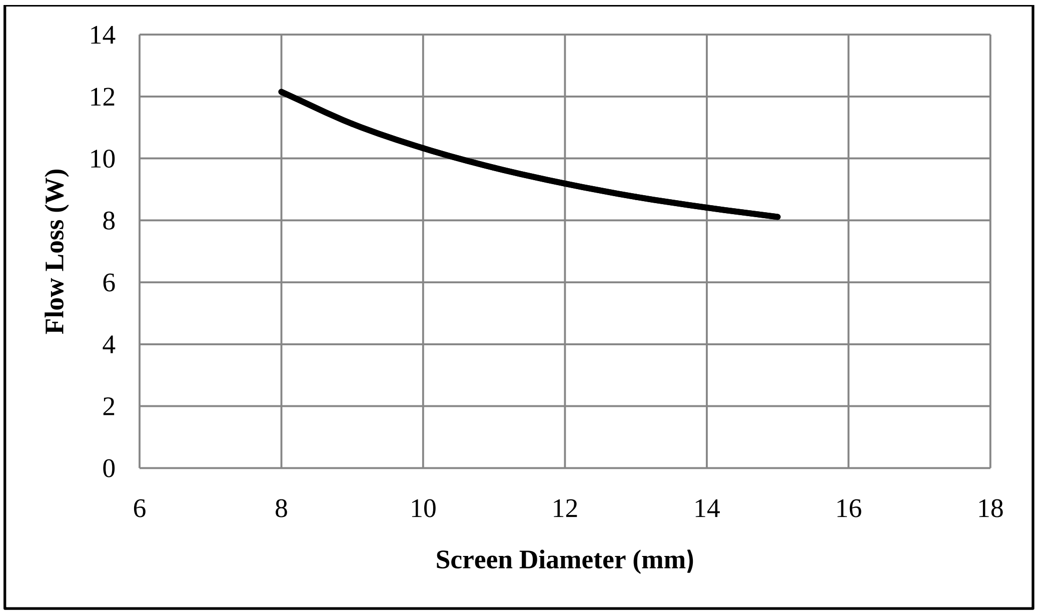


Figure 4.8: Heat flow loss vs. screen diameter

Increase in the longitudinal conduction is not very significant compared to the changes in other mechanisms. This increase in the surface area of the discs generates more interaction points for longitudinal conduction to occur.

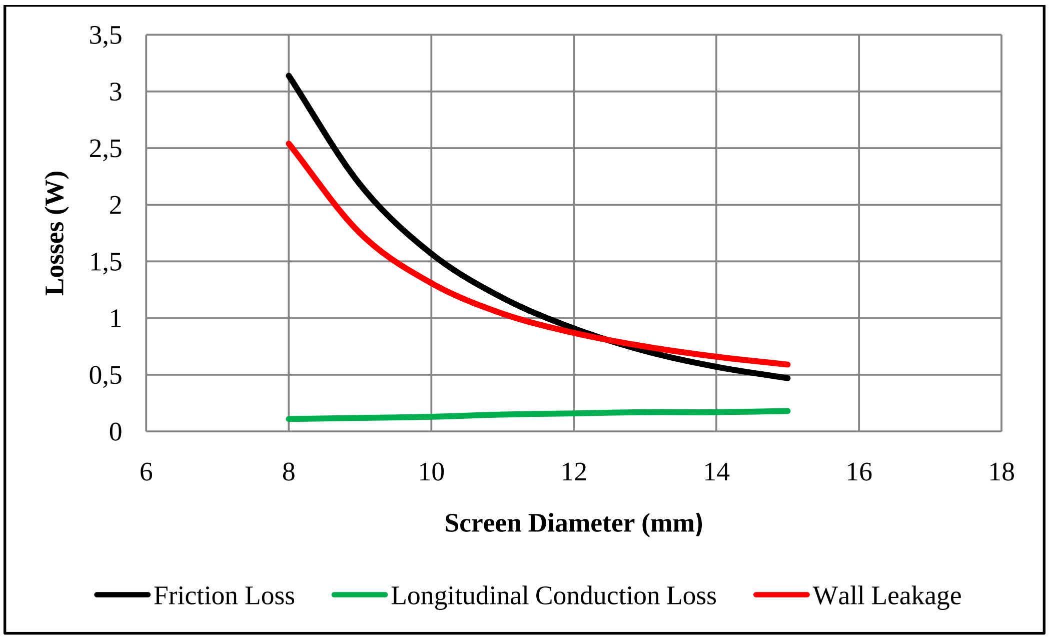


Figure 4.9: Losses vs. screen diameter

4.3.3. Wire Diameter

Wire diameter is a critical parameter especially for small cryocoolers. As the cold finger diameter reduces, to increase heat transfer area, it is a general trend to use higher mesh density screens which require smaller wire diameters. But due to manufacturing issues, values below 0.025 mm are very unlikely.

Regenerator length will increase as wire diameter increases for constant number of screens. It is important to eliminate the effects of having a longer regenerator, while investigating wire diameter effects. To do so, the number of screens is decreased while increasing the wire diameter and the regenerator length is kept constant at 4.6 cm. Screen diameter is chosen as 12 mm so obtaining a 300 x 300 mesh density will not be problem with a 0.05 mm wire diameter.

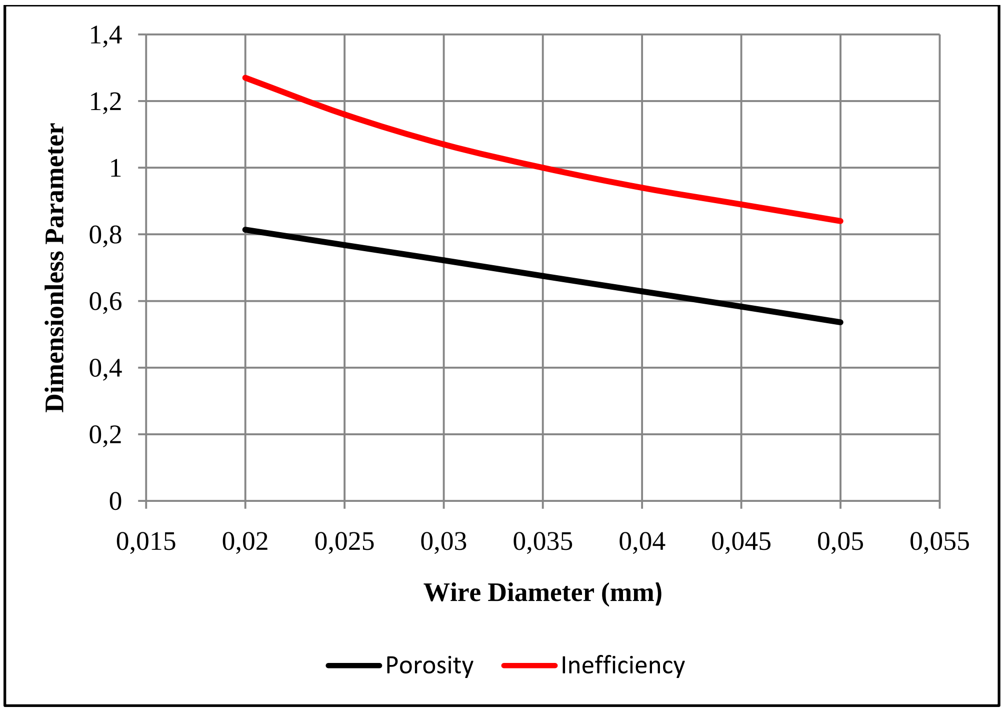


Figure 4.10: Inefficiency and porosity values for different wire diameters

As it can been seen from Figure 4.10, while the wire diameter increases, the porosity and inefficiency decrease linearly. This trend well matches with the literature (Walker, 1983). Decrease in inefficiency comes from increasing heat transfer area and heat capacity. Decrease in porosity is an expected result, since volume occupied by the material will increase with the increasing wire diameter.

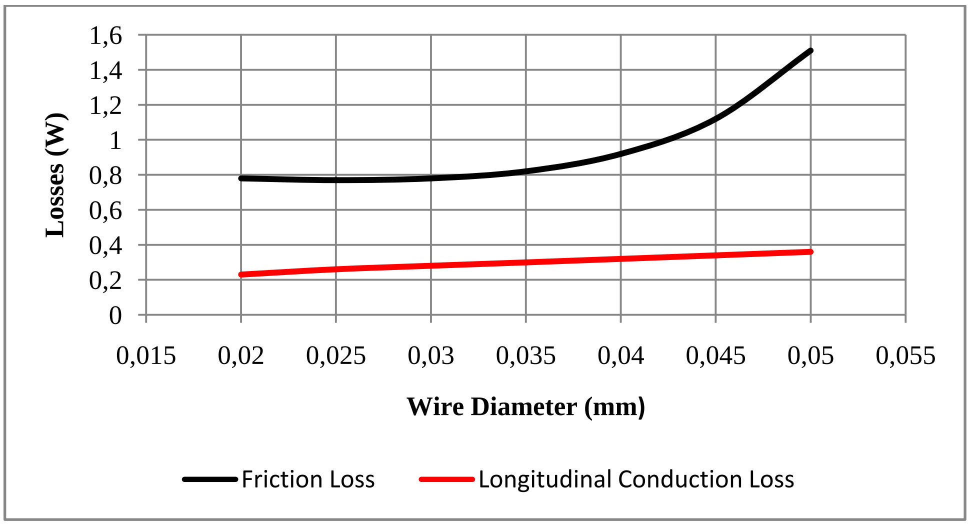


Figure 4.11: Variation of friction loss and longitudinal conduction loss with wire diameter

Increasing wire diameter also has negative effects. From Figure 4.11 an exponential increase in friction loss can be observed. This is due to decrease in the fluid flow area. A linear increase in longitudinal conduction loss is another outcome of increasing wire diameter. Like in screen diameter case, increase in the number of contact points is the reason behind it.

Wire diameter increase - as expected - increases regenerator mass, so as the heat capacity. One of the outcomes of this change is decrease in the heat flow loss (Figure 4.12).

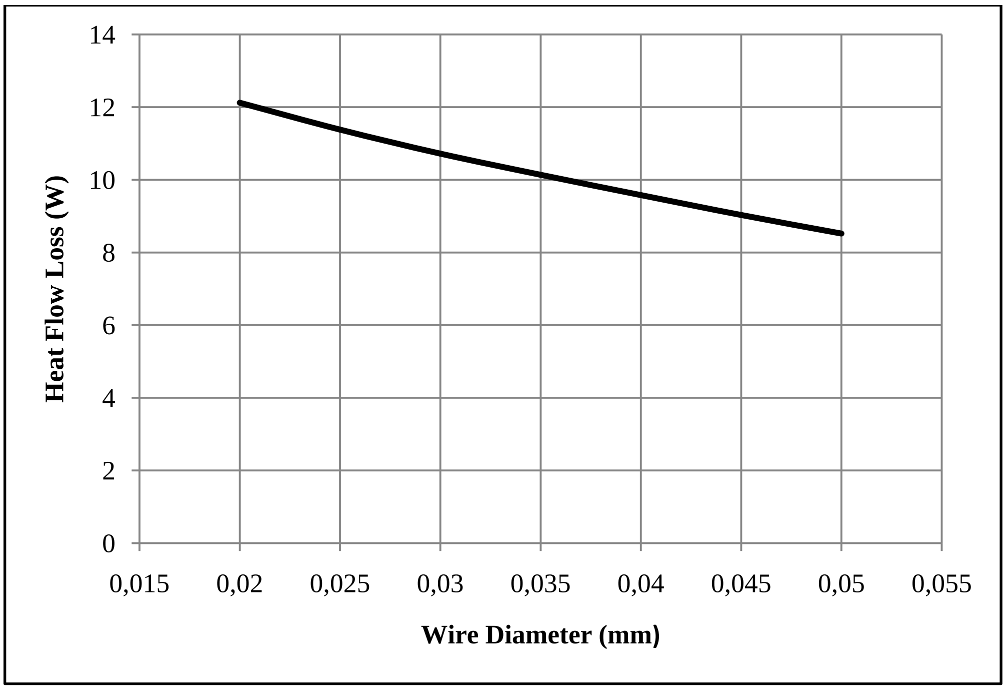


Figure 4.12: Heat flow loss vs wire diameter

4.3.4. Mesh Density

As mentioned in section 3.2.1.4, mesh density is defined as the number of openings in an inch square. Since the openings are square, number of openings in left-right and up-down directions is same. So it is very common to drop one of them i.e. using 300 instead of 300 x 300. This abbreviation is also used throughout this thesis.

Mesh density is directly related to porosity. As expected, increasing the mesh density decreases porosity. Porosity decrease increases the regenerator mass, so as the heat capacity of matrix. Increase in number of openings, increases the heat transfer surface area. These two effects, decreases inefficiency as it can be seen from Figure 4.13.

Inefficiency seems to be decreasing as porosity decreases but, it increases the friction exponentially (Figure 4.14). Therefore having a regenerator with 0.1 porosity will not work, since there will not be any flow area literally. Otherwise, a solid conductor bar would work fine as a regenerator. *Choi S. et al*. obtained similar results in their work (Choi, Nam, & Jeyong, 2004).

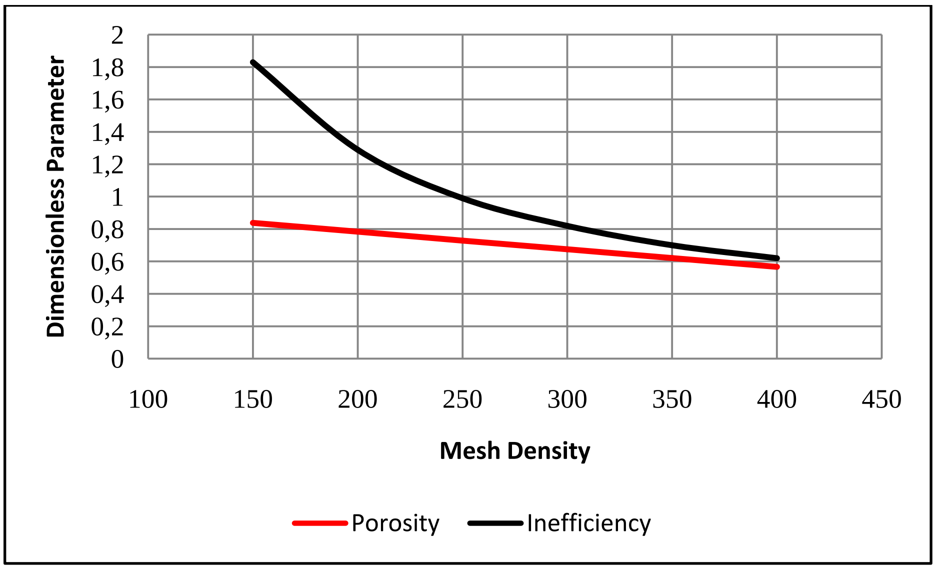


Figure 4.13: Effect of mesh density on porosity and inefficiency

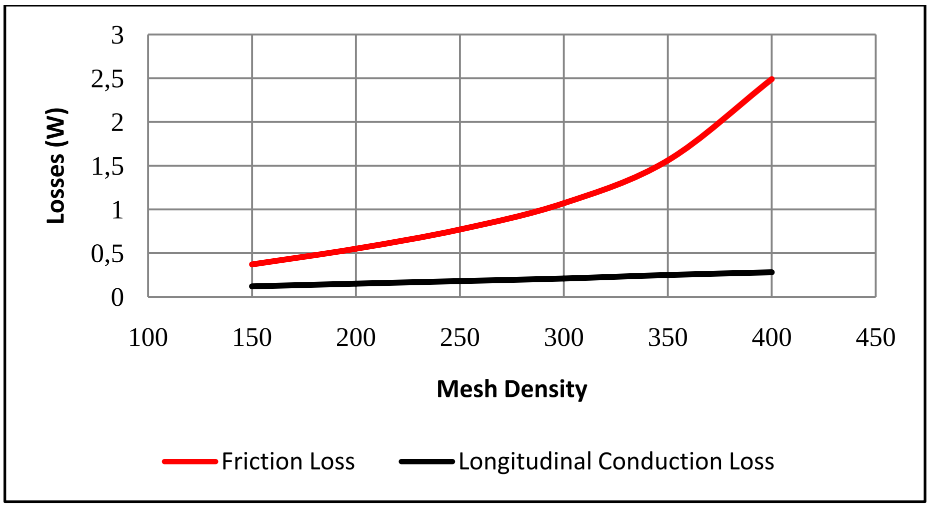


Figure 4.14: Effect of mesh density on friction and conduction loss

4.3.5. Volumetric Flow Rate

Effect of the volumetric flow rate is investigated under default values of regenerator parameters except mesh density value, which is set to 250.

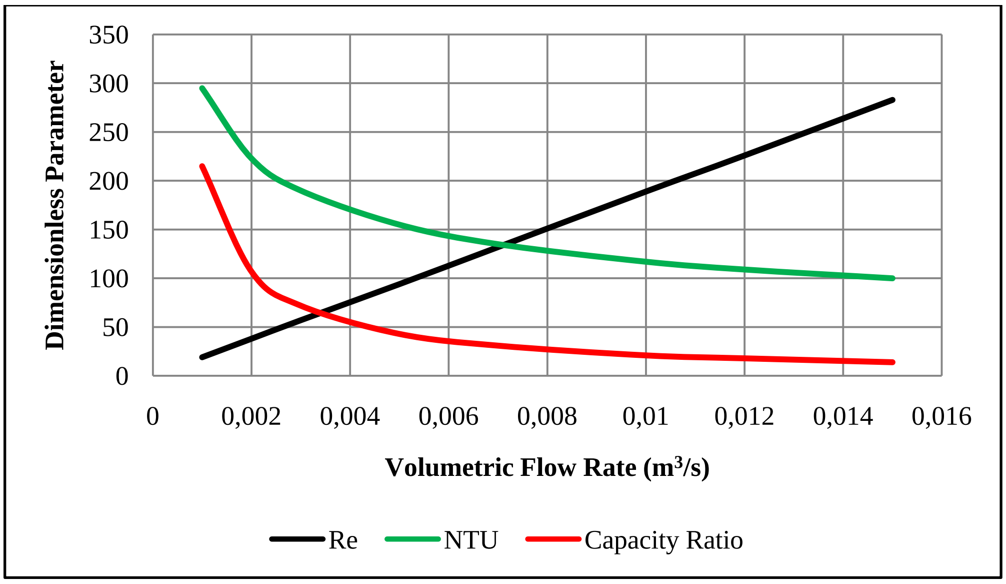


Figure 4.15: Reynolds number, NTU and capacity ratio variation with volumetric flow rate

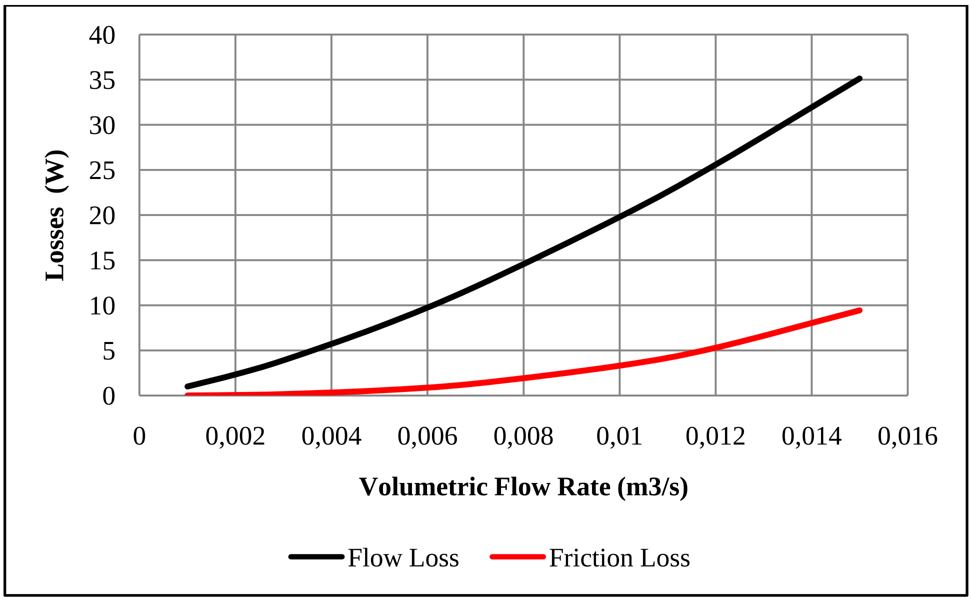


Figure 4.16: Heat flow and friction losses vs. volumetric flow rate

An increase in the volumetric flow rate increases both Reynolds and Prandtl numbers. This causes NTU (Figure 4.15) to drop since it is inversely proportional with these two dimensionless parameters. Increase in the volumetric flow rate, works in favor of the fluid heat capacity, therefore decreases the heat capacity rate ratio (Figure 4.15) as expected. Both the heat flow loss and the friction loss increase with increasing volumetric flow rate (Figure 4.16). They are triggered by the decrease in capacity ratio and increase in Reynolds number respectively. These effects cause the inefficiency to increase as shown in Figure 4.17.

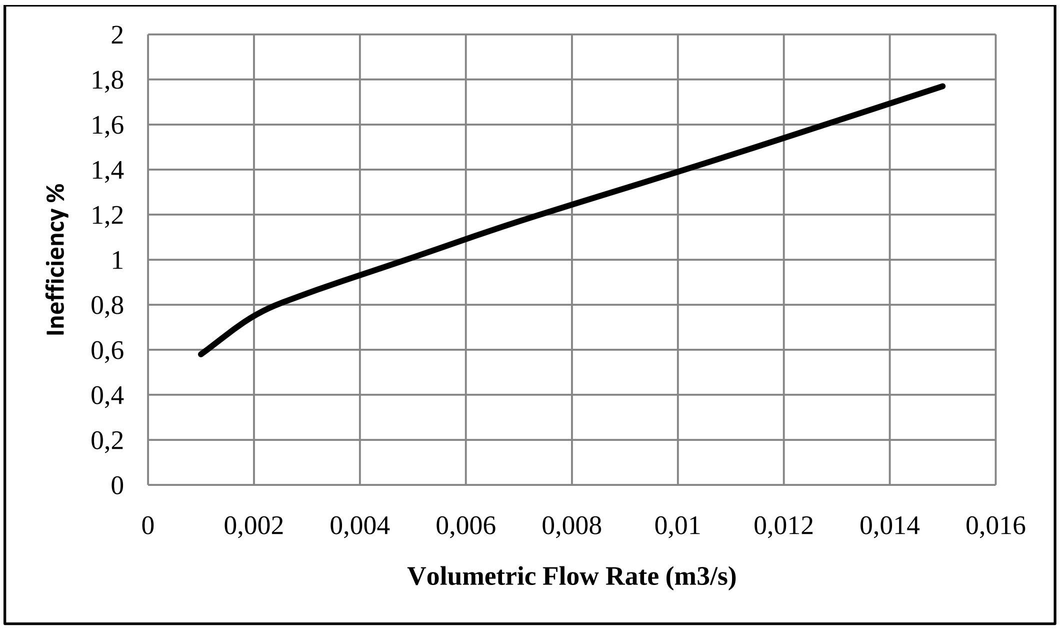


Figure 4.17: Volumetric flow rate vs inefficiency

4.3.6. Operating Frequency

Operating frequency is a critical parameter not only for the regenerator but also for the cryocooler itself (Koh, Hong, Park, Kim, & Lee, 2002). It is a known fact that increasing the operating frequency shortens cryocooler operating life. The wear in the moving parts with heavy masses (heavy compared to other parts of the cryocooler) is one of the main reasons behind it (Nachman, Veprik, & Pundak, 2007).

On the other hand, it is also stated in (Çakıl, 2010) that, increasing operating frequency, increases cryocooler efficiency. To obtain a compromise – from the regenerator point of view – inefficiency of regenerator is plotted for frequencies 10 to 100 Hz. in Figure 4.18.

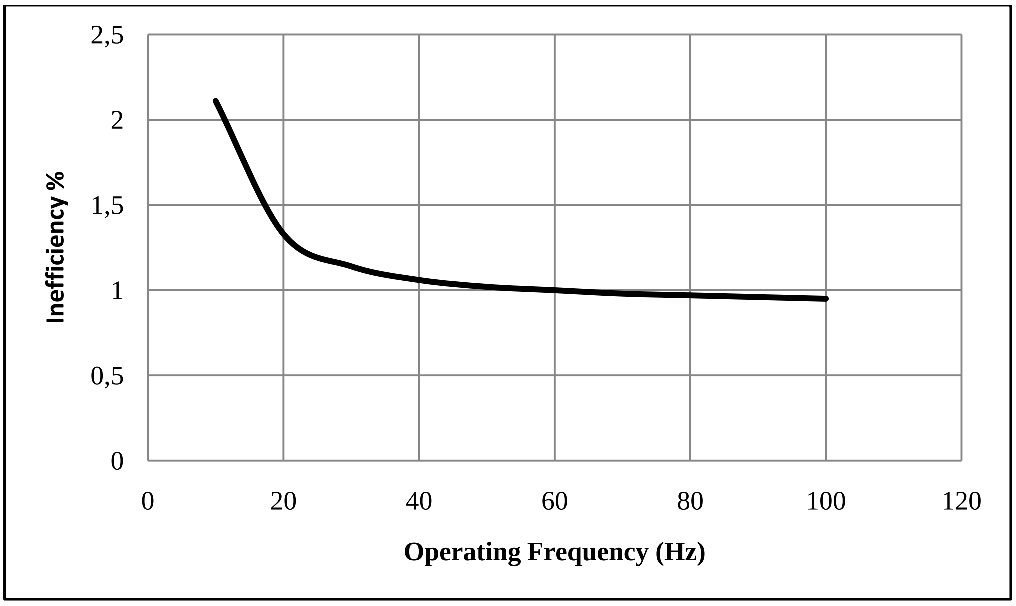


Figure 4.18: Operating frequency vs. inefficiency

A sharp decrease in the inefficiency is observed from 10 to 40 Hz. From 40 to 100 Hz, inefficiency continues to decrease but very slowly. Today's cryocoolers work mostly around 40-60 Hz (Koh, Hong, Park, Kim, & Lee, 2002) (Jensen S.M., 2007) and Figure 4.18 reveals one of the reasons behind this choice clearly.

4.3.7. Wall Thickness

As stated in section 2.3.2, regenerator wall is one of the main mechanisms that introduce loss to the system. The wall heat leakage becomes dominant especially for regenerator with low mass values. As the wall - regenerator heat capacity ratio shifts in favor of the wall, the fluid exchanges more heat with the wall which increases inefficiency. These affects can also be seen from Figure 4.19 and Figure 4.20 for a 10 gram regenerator.

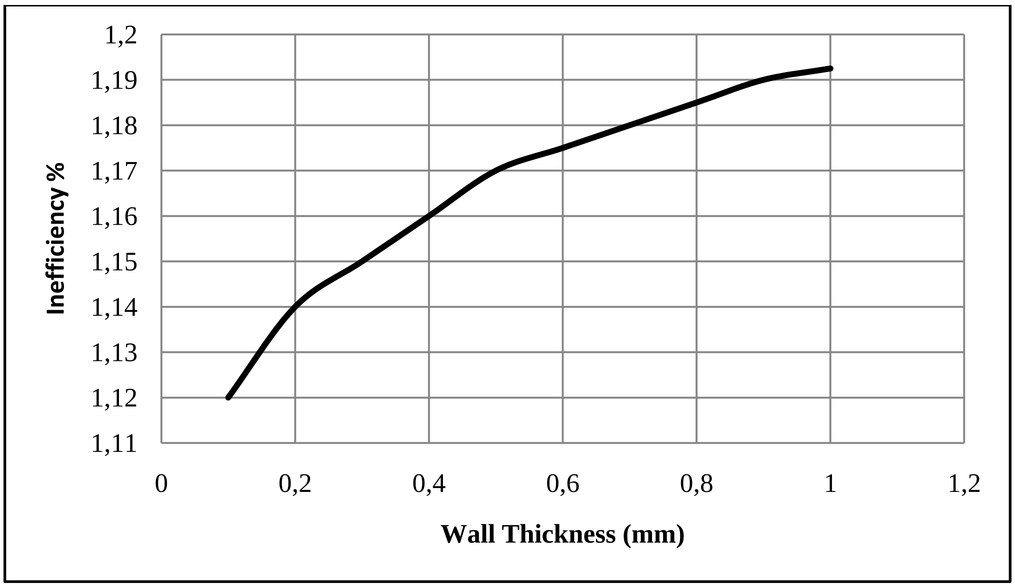


Figure 4.19: Wall thickness vs inefficiency

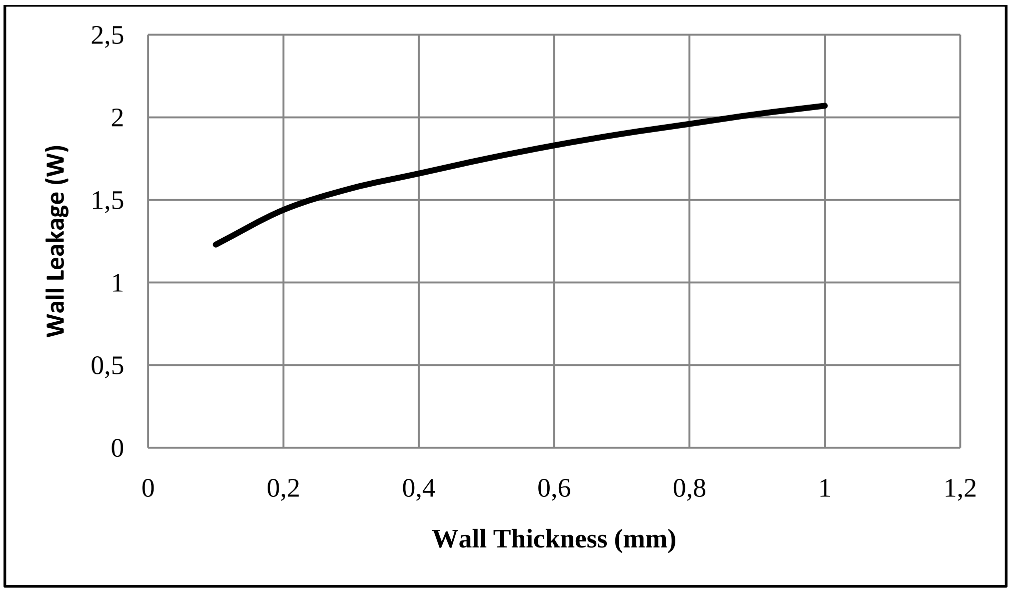


Figure 4.20: Wall thickness vs. wall leakage

4.4. Effect of Regenerator Efficiency on the System

Up to now, effects of the regenerator parameters on the regenerator inefficiency is presented. To conclude the results section it will be a good idea to relate regenerator efficiency with the overall system efficiency.

In the literature, any direct relation between efficiencies of the regenerator and the complete system is not available except for the ideal adiabatic model that is derived by Urieli (Urieli & Walker, 1990).

Using the proposed approach by Urieli (Urieli & Berchowitz, 1984), together with the Matlab code given (Urieli, 1980), Figure 4.21 is obtained by using the parameter values listed in Table 4.4.

Table 4.4: Cooler configuration

Cooler Type	Rotary Kinematic
Mean Pressure	200 kPa
Operating Frequency (Hz)	55
Cold Sink Temp. (K)	80
Hot Sink Temp. (K)	300
Regenerator Type	Wire Mesh Screen
Screen Diameter (mm)	10
Wire Diameter (mm)	0.035
Porosity	0.62
Regenerator Length (cm)	8

It is evident from Figure 4.21 that regenerator performance is crucial for cryocoolers as it is mentioned in the Chapter 1. Even 10% decrease (from 1 to 0.9) in the regenerator efficiency, decreases the cooler's efficiency 64% (from 0.7 to 0.25). It must be kept in mind that the cooler is modeled as ideal, thus in actual systems, performance degradation may even be more severe.

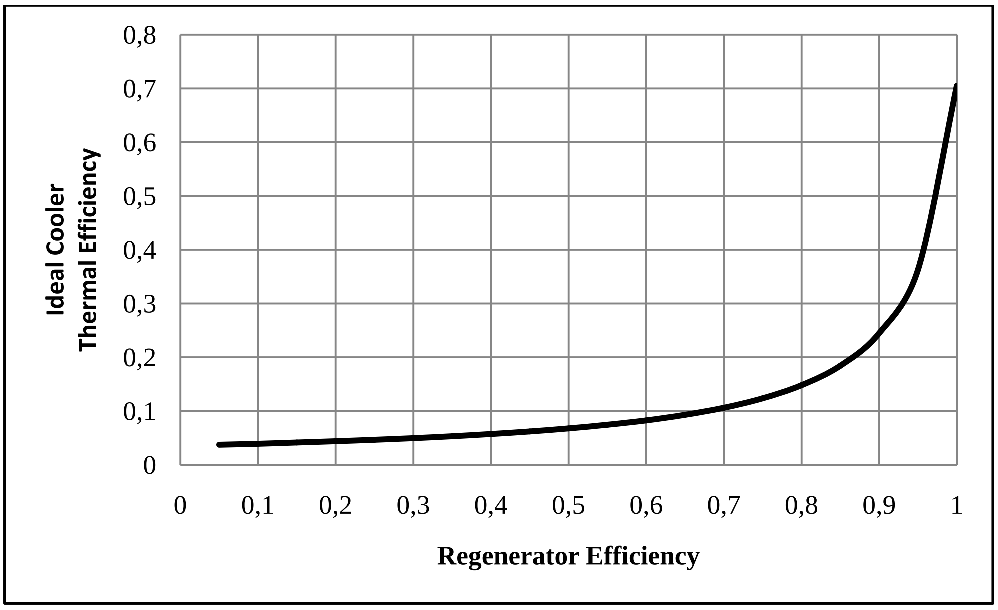


Figure 4.21: Effect of regenerator efficiency on ideal cooler

CHAPTER 5

CONCLUSIONS

In this thesis work, a Matlab code is developed together with a graphical user interface, for thermal analysis of regenerators which work in a Stirling cycle cryocooler. The code uses finite difference method for the solution of the thermal equations derived. The results obtained for three different regenerator configurations are compared with the results obtained by the commercially available Stirling cryocooler analysis software Sage. After the in house code -1d Regenerator Simulator- is verified, effects of regenerator parameters on inefficiency and on thermal losses are investigated in detail. From this investigation, following conclusions can be drawn:

- Number of nodes, time intervals or periods can affect results considerably. It is important to obtain results while the system is independent of these variables.
- Regenerator construction material must have high volumetric heat capacity. This is crucial especially for the flows with high mass flow rate.
- Woven screen wire diameter should be chosen as thick as possible while keeping an eye on limiting parameters; friction loss, longitudinal conduction loss, porosity.
- Even it increases friction loss, finer meshes work better.
- Regenerator best working frequency range matches well with today's cryocooler working frequency range.
- Regenerator wall introduces an unavoidable loss to the system. It must be as thin as possible. The limiting parameters (straightness, strength) may be improved with trying different construction materials.

- Regenerator efficiency has a strong influence on system efficiency. It must be above 98% for an efficient system.

5.1. Suggestions for Future Work

In this research the regenerator is assumed to be under a steady periodic flow. Even this assumption is good enough to evaluate its performance trends, more realistic results can be obtained with a sinusoidal varying mass flow rate which is the actual case for Stirling cryocoolers.

Assumption of one directional flow gives satisfactory results. But it does not take into account some details like local accelerations. To create a two dimensional or three dimensional model will not work since, the porous media structure is random. To obtain a better result, volume averaging technique (Whitaker, 1999) can be used.

The developed code handles the wire mesh discs only. It can be extended to handle other kinds of regenerator configurations like etched foil, packed spheres etc. Other commonly used regenerator materials' properties can also be added to the database to cover more configurations.

REFERENCES

Ackermann, R. A. (1997). *Cryogenic Regenerative Heat Exchangers.* New York: Plenum Press.

Arp, V., & Radebaugh, R. (1987). *Interactive program for microcomputers to calculate the optimum regenerator geometry for cryocoolers.*

Barclay, J., & Sarangi, S. (1984). Selection of regenerator geometry for magnetic regenerator applications. *5th ASME/AIChE/IIR Intersociety Cryogenic Symposium.* New Orleans.

Cha, J., Ghiaasiaan, S., & C.S., K. (2008). Oscillatory flow in microporous media applied in pulse – tube and Stirling – cycle cryocooler regenerators. *Experimental Thermal and Fluid Science 32 , 32*, p. 1264-1278.

Choi, S., Nam, K., & Jeyong, S. (2004). Investigation on the pressure drop characteristics of cryocooler regenerators under oscillating flow and pulsating pressure conditions. *Cryogenics , 44*, p. 203-210.

Cryogenics Technologies Group. (1995). (NIST) Last access date: July 28, 2011 http://cryogenics.nist.gov/Group%20Information/group.htm

Çakıl, S. (2010). *Computational Analysis for Peformance Prediction of Stirling Cryocoolers [MSc Thesis]*. METU.

Gedeon, D. (1995). Last access date: July 28, 2011 http://www.sageofathens.com/index.php

Gedeon, D. (2009). *Sage User's Guide V6.* Athens-OH.

Harvey, P. (2003). *Oscillatory compressible flow and heat transfer in porous media - Application to cryocooler regenerators [PhD Thesis]*. Georgia Institute of Technology.

Harvey, P. (1999). *Parametric Study of Cryocooler Regenerator Performance [MSc Thesis]*. Georgia Institute of Technology.

Heywood, D. (2004). Last access date: July 27, 2011, http://www.stevenpopkes.com/Stirling--SterlingCycleIntro.pdf

Imura, J. (2007). Development of high capacity Stirling type pulse tube cryocooler. *Physica C* , p. 1369-1371.

Jensen S.M., H. G. (2007). Thermal/Mechanical System Level Test Results of the GIFTS 2-Stage Pulse Tube Cryocooler. International Cryocooler Conference Vol 14.

Kashani, A., Helvensteijn, B., Kittel, P., Gschneidner, K., Pecharsky, V., & Pecharsky, A. (2001). New Regenerator Materials for Use in Pulse Tube Coolers. *11th International Cryocooler Conference*, (p. 433-441).

Kays, A., & London, W. (1984). *Compact Heat Exchangers*. McGraw Hill.

Koh, D., Hong, Y., Park, S., Kim, H., & Lee, K. (2002). A study on the linear compressor characteristics of the Stirling cryocooler. *Cryogenics 42* , p. 427-432.

Mitchell, M., & Fabris, D. (2003). Improved Flow Patterns in Etched Foil Regenerator. *Cryocoolers 12*, (p. 499-505).

Nachman, I., Veprik, A., & Pundak, N. (2007). Life test result of Ricor K529N 1 Watt linear Cryocooler. *Infrared Technology and Applications XXXIII.*

NIST. (2005). (NIST) Last access date: July 28, 2011 http://cryogenics.nist.gov/MPropsMAY/RegeneratorMaterials/RegenPlot.html

O’Gallagher, A., Gary, J., Radebaugh, R., & and Marquardt, E. (2008). *REGEN 3.3: User Manual.*

O’Gallagher, A., Gary, J., Radebaugh, R., & Marquardt, E. (2001). (NIST) Last access date: July 28, 2011 http://cryogenics.nist.gov/Software/software.htm

Pfotenhauer, J., Shi, J., & Nellis, G. (2004). A Parametric Optimization of a Single Stage Regenerator Using REGEN 3.2. *13th International Cryocooler Conference*, *13*, p. 463-470.

Prajapati, A., & Oza, T. (2007). Study Analysis Of Effect Of Different Parameters On Design Aspects Of Cryogenic Wire Mesh Heat Exchanger. *ICME.* Dhaka.

Qiu, L. (2007). Regenerator performance improvement of a single-stage pulse tube cooler reached 11.1 K. *Cryogenics 47*, (p. 49-55).

Radebaugh, R. (2010, May 17). Cryocooler Short Course Presentation. Atlanta, Georgia, USA.

Radebaugh, R. (2009). Cryocoolers: the state of the art and recent developments. *Journal of Physics* , *21*, p1-8.

Radebaugh, R. (1995). Recent Developments in Cryocoolers. *Proc. Int. Cong. Refigeration*, *3b*, p. 973. Paris.

Riabzev, S. (2002). *Ricor Cryogenics*. Last access date: July 27, 2011 www.ricor.com/_Uploads/58Stir-pt.pps

Schlichting, H. (1955). *Boundary Layer Theory.* New York: McGraw-Hill.

Shi J., P. J. (2007). Dimensionless Analysis for Regenerator Design. International Crycooler Conference 14.

Thales. (2010). Company presentation. Ankara.

Timmerhaus, K. (1996). Cryocooler Development. *AIChE Journal* , p. 3202-3211.

Urieli, I. (1980). A general purpose program for Stirling engine simulation. *Proe. 15th IECEC* , p. 1701-1705.

Urieli, I., & Berchowitz, D. (1984). *Stirling cycle engine analysis.* Adam Hilger Ltd.

Urieli, I., & Walker, G. (1990). An ideal adiabatic analysis of a stirling cryocoole rwith multiple expansion stages. *Low Temperature Engineering and Crvogenics Conference.*

Walker, G. (1983). *Cryocoolers.* Plenum Press.

Whitaker, S. (1999). *The method of volume averaging.* Kluwer Academic Publishers.

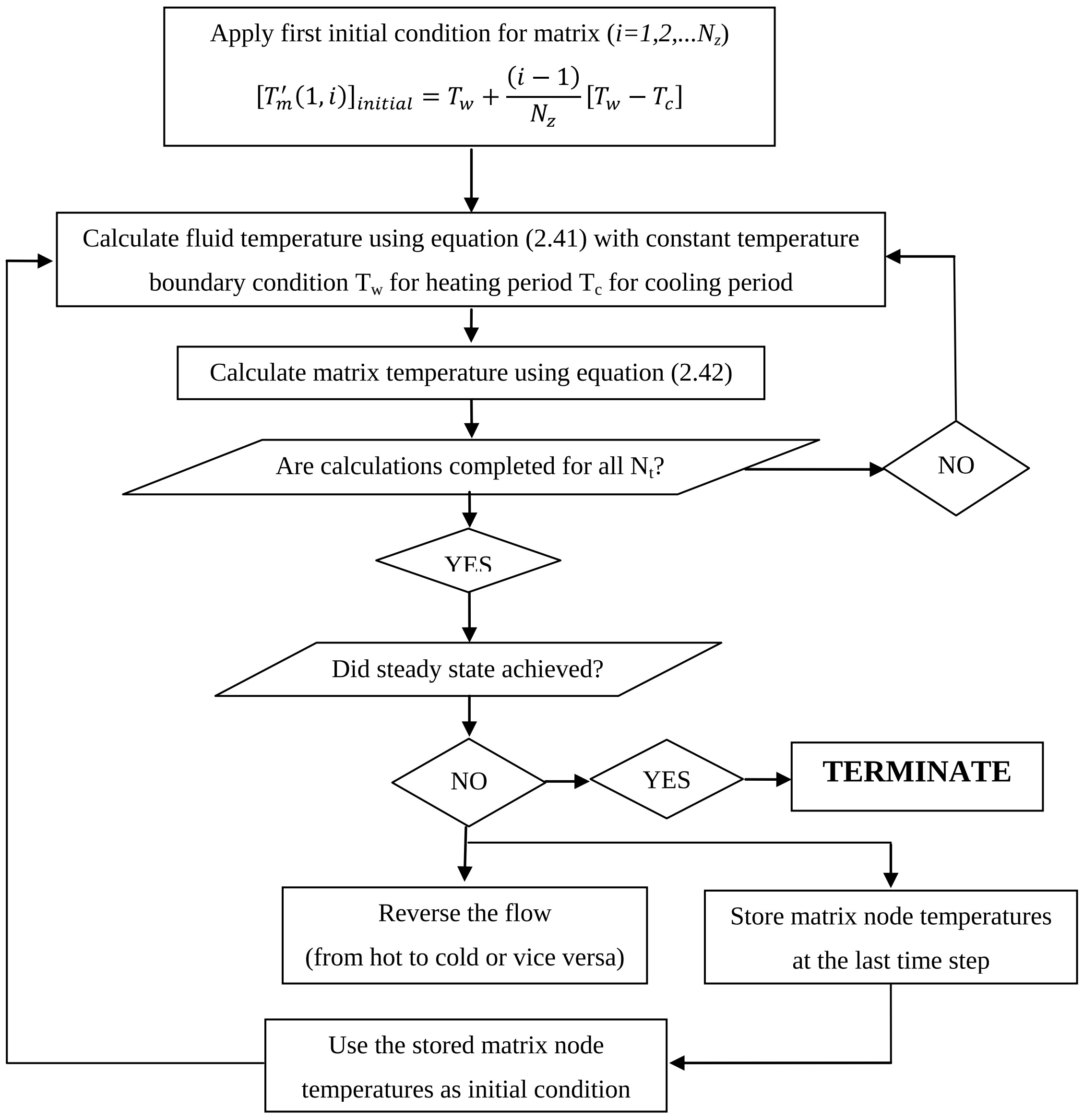
Williems, D. (2007). *High-Power Crycooling [PhD Thesis].* Technische Universiteit Eindhoven.

Wilson, K., & Gedeon, D. (2003). Development of Single and Two-Stage Pulse Tube Cryocoolers with Commercial Linear Compressors. *12th International Cryocooler Conference,* (p. 139-147).

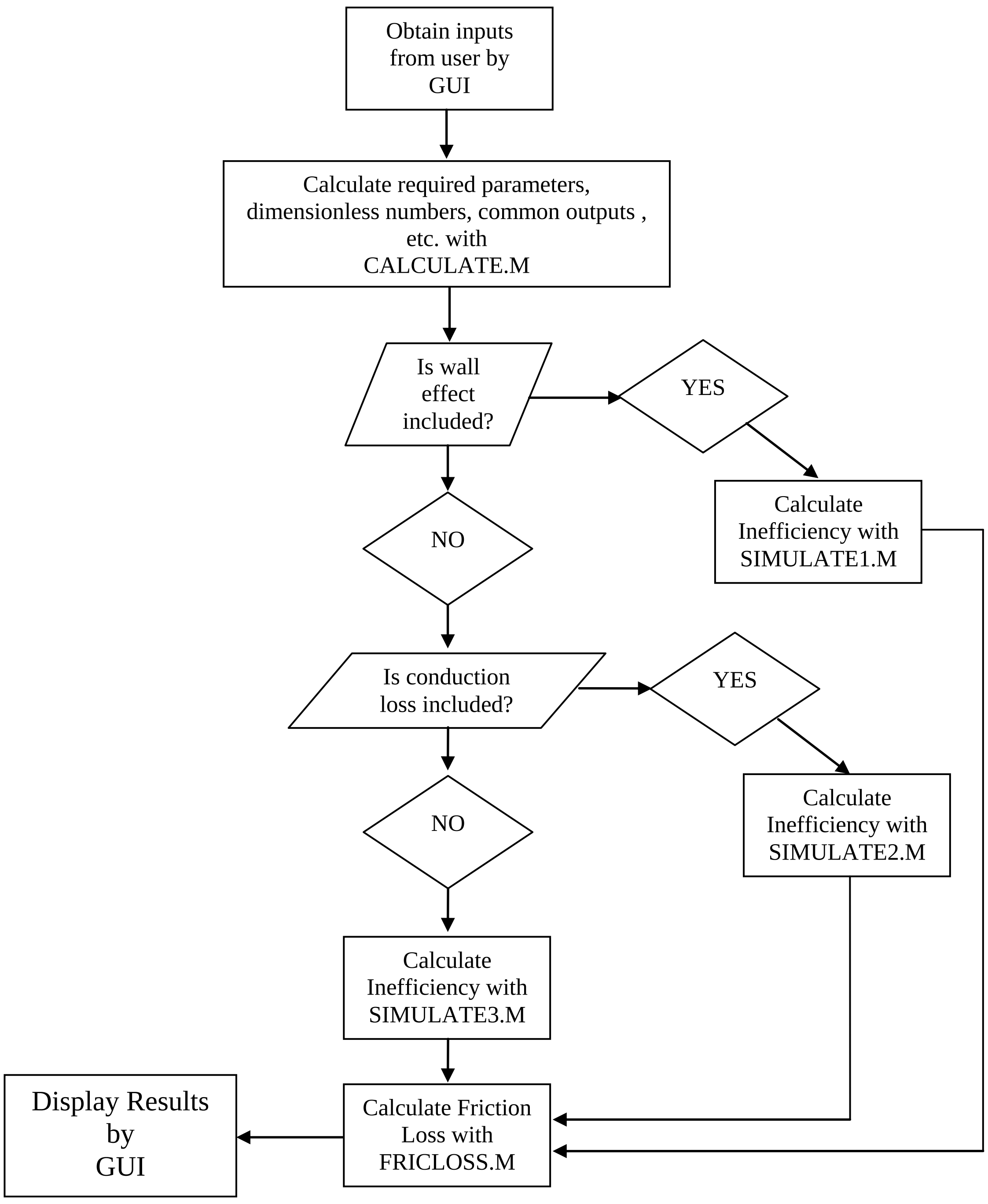
Yang, X., & Chung, J. (2005). Size effects on miniature Stirling cycle cryocoolers. *Cyogenics 45* , p. 537-545.

APPENDIX A

Temperature Solution Algorithm


Apply first initial condition for matrix ($i=1,2,\ldots N_z$)
$[T_m'(1,i)]_{initial} = T_w + \frac{(i-1)}{N_z}[T_w - T_c]$
Calculate fluid temperature using equation (2.41) with constant temperature boundary condition T_w for heating period T_c for cooling period
Calculate matrix temperature using equation (2.42)
Are calculations completed for all N_t?
NO
YES
Did steady state achieved?
NO
YES
TERMINATE
Reverse the flow (from hot to cold or vice versa)
Store matrix node temperatures at the last time step
Use the stored matrix node temperatures as initial condition

1D Simulator Flow Chart


Obtain inputs from user by GUI
Calculate required parameters, dimensionless numbers, common outputs , etc. with CALCULATE.M
Is wall effect included?
YES
Calculate Inefficiency with SIMULATE1.M
NO
Is conduction loss included?
YES
Calculate Inefficiency with SIMULATE2.M
NO
Calculate Inefficiency with SIMULATE3.M
Calculate Friction Loss with FRICLOSS.M
Display Results by GUI

APPENDIX B

MATLAB CODING

GUI_1d_Reg.m

```
function varargout = GUI_1d_Reg(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_1d_Reg_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_1d_Reg_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI_1d_Reg is made visible.
function GUI_1d_Reg_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_1d_Reg (see VARARGIN)

% Choose default command line output for GUI_1d_Reg
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes GUI_1d_Reg wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_1d_Reg_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in SOLVE.
function SOLVE_Callback(hObject, eventdata, handles)
% hObject    handle to SOLVE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

clc

timeStart = tic;
k=findobj('Tag','done');
set(k,'Visible','off');

pause(0.01);

k=findobj('Tag','busy');
set(k,'Visible','on');

pause(0.01);

set(handles.regeneratorlength,'String','');
set(handles.regeneratormass,'String','');
set(handles.porosity,'String','');
set(handles.hd,'String','');
set(handles.massflowrate,'String','');
set(handles.Re,'String','');
set(handles.St,'String','');
set(handles.capacityratio,'String','');
set(handles.NTU,'String','');
set(handles.inefficiency,'String','');
set(handles.Tcf,'String','');
set(handles.Thf,'String','');
set(handles.Qz,'String','');
set(handles.Qfric,'String','');
set(handles.simQx,'String','');
set(handles.simQw,'String','');
set(handles.runtime,'String','');

longcond = get(handles.longcond,'Value');
walleffect = get(handles.walleffect,'Value');

rD = str2num(get(handles.screendiameter,'String'));
wD = str2num(get(handles.wirediameter,'String'));
```

```
NOS = str2num(get(handles.numberofscreens,'String'));
flow_rate = str2num(get(handles.flowrate,'String'));
Tc = str2num(get(handles.Tc,'String'));
Tw = str2num(get(handles.Tw,'String'));
Nz = str2num(get(handles.Nz,'String'));
Nt = str2num(get(handles.Nt,'String'));
max_period = str2num(get(handles.maxperiod,'String'));
freq = str2num(get(handles.freq,'String'));
t_wl = str2num(get(handles.wallthickness,'String'));

matrixmaterial= get(handles.matrixmaterial,'Value');
meshdensity= get(handles.meshdensity,'Value');

[rL M porosity hd m_dot Re St Cr NTU ineff Th_out_ave Tc_out_ave Qz...
    Qfric simQx simQw] = calculate(matrixmaterial,meshdensity,rD,wD,...
    NOS,flow_rate,Tc,Tw,Nz,Nt,max_period,freq,t_wl,longcond,walleffect);

set(handles.regeneratorlength,'String',round(rL*10000)/100);
set(handles.regeneratormass,'String',(round(M*100000))/100);
set(handles.porosity,'String',(round(porosity*1000))/1000);
set(handles.hd,'String',(round(hd*1000000))/1000);
set(handles.massflowrate,'String',(round(m_dot*100000))/100);
set(handles.Re,'String',round(Re));
set(handles.St,'String',(round(St*100))/100);
set(handles.capacityratio,'String',round(Cr));
set(handles.NTU,'String',round(NTU));
set(handles.inefficiency,'String',(round(ineff*100))/100);
set(handles.Tcf,'String',(round(Th_out_ave*10))/10);
set(handles.Thf,'String',(round(Tc_out_ave*10))/10);
set(handles.Qz,'String',(round(Qz*100))/100);
set(handles.Qfric,'String',(round(Qfric*100))/100);

if simQx==0;
    set(handles.simQx,'String','N/A');
else
    set(handles.simQx,'String',(round(simQx*100))/100);
end

if simQw==0;
    set(handles.simQw,'String','N/A');
else
    set(handles.simQw,'String',(round(simQw*100))/100);
end

timeTotal = toc(timeStart);
set(handles.runtime,'String',(round(timeTotal*10))/10);

pause(0.01);

k=findobj('Tag','busy');
set(k,'Visible','off');

pause(0.01);

k=findobj('Tag','done');
set(k,'Visible','on');
```

```
% --- Executes on selection change in matrixmaterial.
function matrixmaterial_Callback(hObject, eventdata, handles)
% hObject    handle to matrixmaterial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function matrixmaterial_CreateFcn(hObject, eventdata, handles)
% hObject    handle to matrixmaterial (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function screendiameter_Callback(hObject, eventdata, handles)
% hObject    handle to screendiameter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','screendiameterslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function screendiameter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to screendiameter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function screendiameterslider_Callback(hObject, eventdata, handles)
% hObject    handle to screendiameterslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','screendiameter');
set(H,'string',num2str(x));
```

```
% --- Executes during object creation, after setting all properties.
function screendiameterslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to screendiameterslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function wirediameter_Callback(hObject, eventdata, handles)
% hObject    handle to text55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','wirediameterslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function wirediameter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text55 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function wirediameterslider_Callback(hObject, eventdata, handles)
% hObject    handle to wirediameterslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','wirediameter');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function wirediameterslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wirediameterslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on selection change in meshdensity.
function meshdensity_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to meshdensity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function meshdensity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to meshdensity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function numberofscreens_Callback(hObject, eventdata, handles)
% hObject    handle to numberofscreens (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','numberofscreensslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function numberofscreens_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numberofscreens (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function numberofscreensslider_Callback(hObject, eventdata, handles)
% hObject    handle to numberofscreensslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','numberofscreens');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function numberofscreensslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numberofscreensslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function flowrate_Callback(hObject, eventdata, handles)
% hObject    handle to flowrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','flowrateslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function flowrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to flowrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function flowrateslider_Callback(hObject, eventdata, handles)
% hObject    handle to flowrateslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','flowrate');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function flowrateslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to flowrateslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Tc_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Tc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','Tcslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function Tc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function Tcslider_Callback(hObject, eventdata, handles)
% hObject    handle to Tcslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','Tc');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function Tcslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tcslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Tw_Callback(hObject, eventdata, handles)
% hObject    handle to Tw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','Twslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
```

```
function Tw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function Twslider_Callback(hObject, eventdata, handles)
% hObject    handle to Twslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','Tw');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function Twslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Twslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Nz_Callback(hObject, eventdata, handles)
% hObject    handle to Nz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','Nzslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function Nz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```matlab
% --- Executes on slider movement.
function Nzslider_Callback(hObject, eventdata, handles)
% hObject    handle to Nzslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','Nz');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function Nzslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nzslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function Nt_Callback(hObject, eventdata, handles)
% hObject    handle to Nt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','Ntslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function Nt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function Ntslider_Callback(hObject, eventdata, handles)
% hObject    handle to Ntslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','Nt');
set(H,'string',num2str(x));
```

```
% --- Executes during object creation, after setting all properties.
function Ntslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ntslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function maxperiod_Callback(hObject, eventdata, handles)
% hObject    handle to maxperiod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','maxperiodslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function maxperiod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxperiod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function maxperiodslider_Callback(hObject, eventdata, handles)
% hObject    handle to maxperiodslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','maxperiod');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function maxperiodslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxperiodslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function freq_Callback(hObject, eventdata, handles)
% hObject    handle to freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
H=findobj('Tag','freqslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function freq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function freqslider_Callback(hObject, eventdata, handles)
% hObject    handle to freqslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','freq');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function freqslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freqslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function wallthickness_Callback(hObject, eventdata, handles)
% hObject    handle to wallthickness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'string');
```

```
H=findobj('Tag','wallthicknessslider');
set(H,'value',str2num(x));


% --- Executes during object creation, after setting all properties.
function wallthickness_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wallthickness (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function wallthicknessslider_Callback(hObject, eventdata, handles)
% hObject    handle to wallthicknessslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

x=get(gco,'value');
H=findobj('Tag','wallthickness');
set(H,'string',num2str(x));


% --- Executes during object creation, after setting all properties.
function wallthicknessslider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wallthicknessslider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in longcond.
function longcond_Callback(hObject, eventdata, handles)
% hObject    handle to longcond (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of longcond


% --- Executes on button press in walleffect.
```

```
function walleffect_Callback(hObject, eventdata, handles)
% hObject    handle to walleffect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of walleffect


function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function regeneratorlength_Callback(hObject, eventdata, handles)
% hObject    handle to regeneratorlength (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function regeneratorlength_CreateFcn(hObject, eventdata, handles)
% hObject    handle to regeneratorlength (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function regeneratormass_Callback(hObject, eventdata, handles)
% hObject    handle to regeneratormass (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function regeneratormass_CreateFcn(hObject, eventdata, handles)
% hObject    handle to regeneratormass (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function St_Callback(hObject, eventdata, handles)
% hObject    handle to St (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function St_CreateFcn(hObject, eventdata, handles)
% hObject    handle to St (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function fluidheatcapacity_Callback(hObject, eventdata, handles)
% hObject    handle to fluidheatcapacity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function fluidheatcapacity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fluidheatcapacity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function capacityratio_Callback(hObject, eventdata, handles)
% hObject    handle to capacityratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function capacityratio_CreateFcn(hObject, eventdata, handles)
% hObject    handle to capacityratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function NTU_Callback(hObject, eventdata, handles)
% hObject    handle to NTU (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function NTU_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NTU (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function inefficiency_Callback(hObject, eventdata, handles)
% hObject    handle to inefficiency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function inefficiency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inefficiency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function porosity_Callback(hObject, eventdata, handles)
% hObject    handle to porosity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function porosity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to porosity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function runtime_Callback(hObject, eventdata, handles)
% hObject    handle to runtime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB


% --- Executes during object creation, after setting all properties.
function runtime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to runtime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function hd_Callback(hObject, eventdata, handles)
% hObject    handle to hd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function hd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Re_Callback(hObject, eventdata, handles)
% hObject    handle to Re (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Re_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Re (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function massflowrate_Callback(hObject, eventdata, handles)
% hObject    handle to massflowrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function massflowrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to massflowrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Tcf_Callback(hObject, eventdata, handles)
% hObject    handle to Tcf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Tcf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tcf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Thf_Callback(hObject, eventdata, handles)
% hObject    handle to Thf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Thf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Thf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Qz_Callback(hObject, eventdata, handles)
% hObject    handle to Qz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Qz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Qz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Qfric_Callback(hObject, eventdata, handles)
% hObject    handle to Qfric (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function Qfric_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Qfric (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function simQx_Callback(hObject, eventdata, handles)
% hObject    handle to simQx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function simQx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simQx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function simQw_Callback(hObject, eventdata, handles)
% hObject    handle to simQw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function simQw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simQw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Calculate.m

```
function [rL M porosity hd m_dot Re St Cr NTU ineff Th_out_ave...
    Tc_out_ave Qz Qfric simQx simQw]= calculate(matrixmaterial,...
    meshdensity,rD,wD,NOS,flow_rate,Tc,Tw,Nz,Nt,max_period,freq,...
    t_wl,longcond,walleffect)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert Parameters to suitable units
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rD=rD*1E-3;                         % mm to m
wD=wD*1E-3;                         % mm to m
meshdensity=100+meshdensity*50;     % dropdown menu number to
                                    % number of mesh openings per inch


lambda=1/(2*freq);                  % -s- flow period
t_wl=t_wl*1E-3;                     % mm to m


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matrix Material Selection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if matrixmaterial==1
    density_m=  6430;          % [kg/m3]   Regenerator Density
    a=          0.97;          % [-]       Thermal Conductivity Constant
    Cp_m=       103.35;        % [J/kg-K]  Matrix Specific Heat
    Km0=        0.7;           % [-]       Thermal Conductivity Constant
elseif matrixmaterial==2
    density_m=  7800;
    a=          0.8;
    Cp_m=       300;
    Km0=        3.5;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Regenerator Property Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t_sc=wD*2.2;                            % [m]   Screen thickness
rL=NOS*t_sc*1.0444;                     % [m]   Regenerator Length
meshsize=meshdensity*100/2.54;          % [1/m] Size of a square mesh
beta=2*pi*wD*meshsize/t_sc;             % [1/m] Area Density
idealporosity=1-(pi*meshsize*wD/4);     % [-]   Ideal Porosity
porosity=idealporosity;
rh=porosity/beta;                       % [m]   Hydraulic Radius
hd=4*rh;                                % [m]   By definition Dh=4*rh
Ar=(rD^2)*pi/4;                         % [m2]  Frontal Area
As=beta*Ar*wD*2*NOS*1.0444;             % [m2]  Heat Transfer Area
Am=Ar*(1-idealporosity);                % [m2]  Matrix Area Normal
                                        %       to the flow

M=Ar*rL*(1-idealporosity)*density_m;    % [kg]  Total Regenerator Mass
HC_m=M*Cp_m;                            % [J/K] Matrix Heat Capacity
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wall Property Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Cp_wl=      500;            % [J/kg.K]  Wall Material Specific Heat
density_wl= 7800;           % [kg/m3]   Wall Material Density
b=          0.38;           % [-]       Wall Thermal Conductivity Constant
Awl_s=      pi*t_wl*(2*rD+t_wl);     % [m2]      Wall Section Area
Awl_h=      2*pi*rD*rL;              % [m2]      Wall Heat Transfer Area
Mwl=        Awl_s*rL*density_wl;     % [kg]      Wall Mass
HC_wl=      Mwl*Cp_wl;               % [J/K]     Wall Heat Capacity
Kwl0=       14.5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Fluid Property Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mu_f=       1.44e-5;            % [Pa.s]       Dynamic Fluid Viscosity
K_f=        0.1;                % [W/m-K]      Fluid Thermal Conductivity
den_f=      0.15;               % [kg/m3]      Fluid Density
Cp_f=       5.19E3;             % [J/kg-K]     Fluid Specific Heat

m_dot=flow_rate*den_f;        % [kg/s]       Mass Flow Rate
HC_f=m_dot*Cp_f;              % [W/K]        Fluid Heat Capacity
Pr=Cp_f*mu_f/K_f;             % [-]          Prandtl Number
G=m_dot/(Ar*porosity);        % [kg/m2.s]    Mass Flow Rate Per Unit Area
Re=G*4*rh/mu_f;                  % [-]          Reynolds Number
St=0.68*(Re^-0.4)*(Pr^-0.667);   % [-]          Stanton Number

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Regenerator Property Calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NTU=St*rL/(2*rh);       % [-]          Number of Transfer Units
h=NTU*2*HC_f/As;        % [W/m2-K]     Heat transfer Coefficent
Cr=HC_m/(HC_f*lambda);  % [-]          Capacity Ratio

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Execute the simulation & Calculate inefficiecy,
% Outlet Temperatures & Losses
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if walleffect==1;
    [ineff Th_out_ave Tc_out_ave Qz simQx simQw] = simulate1(Nz,Nt,...
        max_period,Tw,Tc,a,b,Km0,Kwl0,h,As,Am,Awl_h,HC_m,HC_f,HC_wl,...
        lambda,rL,Awl_s,m_dot,Cp_f);
elseif longcond==1;
    [ineff Th_out_ave Tc_out_ave Qz simQx] = simulate2(Nz,Nt,...
        max_period,Tw,Tc,a,Km0,h,As,Am,HC_m,HC_f,lambda,rL,m_dot,Cp_f);
    simQw=0;
else
    [ineff Th_out_ave Tc_out_ave Qz] = simulate3(Nz,Nt,max_period,...
        Tw,Tc,h,As,HC_m,HC_f,lambda,m_dot,Cp_f);
    simQx=0;
    simQw=0;
end
[Qfric] = fricloss(G,rL,den_f,rh,m_dot,Re,porosity,mu_f,hd);
```

Simulate1.m

```
function [ineff Th_out_ave Tc_out_ave Qz simQx simQw]= simulate1(Nz,...
    Nt,max_period,Tw,Tc,a,b,Km0,Kwl0,h,As,Am,Awl_h,HC_m,HC_f,HC_wl,...
    lambda,rL,Awl_s,m_dot,Cp_f)

    [ineff_w Th_out_ave_w Tc_out_ave_w]=...
        Ideal_Reg_Model_Long_Cond_and_Wall_Effect(Nz,Nt,max_period,...
    Tw,Tc,a,b,Km0,Kwl0,h,As,Am,Awl_h,HC_m,HC_f,HC_wl,lambda,rL,Awl_s);

    [ineff_c Th_out_ave_c Tc_out_ave_c]=Ideal_Reg_Model_Long_Cond...
        (Nz,Nt,max_period,Tw,Tc,a,Km0,h,As,Am,HC_m,HC_f,lambda,rL);

    [ineff_i Th_out_ave_i Tc_out_ave_i]=Ideal_Reg_Model(Nz,Nt,...
        max_period,Tw,Tc,h,As,HC_m,HC_f,lambda);

Th_out_ave=Th_out_ave_w;
Tc_out_ave=Tc_out_ave_w;
ineff=0.0352+ineff_w;

Qz=m_dot*Cp_f*ineff_i*0.01*(Tw-Tc);
simQx=m_dot*Cp_f*(ineff_c-ineff_i)*0.01*(Tw-Tc);
simQw=m_dot*Cp_f*(ineff-ineff_c)*0.01*(Tw-Tc);
```

Simulate2.m

```
function [ineff Th_out_ave Tc_out_ave Qz simQx]= simulate2(Nz,Nt,...
    max_period,Tw,Tc,a,Km0,h,As,Am,HC_m,HC_f,lambda,rL,m_dot,Cp_f)

    [ineff_c Th_out_ave_c Tc_out_ave_c]=Ideal_Reg_Model_Long_Cond(Nz,Nt,...
    max_period,Tw,Tc,a,Km0,h,As,Am,HC_m,HC_f,lambda,rL);

    [ineff_i Th_out_ave_i
Tc_out_ave_i]=Ideal_Reg_Model(Nz,Nt,max_period,...
    Tw,Tc,h,As,HC_m,HC_f,lambda);

Th_out_ave=Th_out_ave_c;
Tc_out_ave=Tc_out_ave_c;
ineff=ineff_c;

Qz=m_dot*Cp_f*ineff_i*0.01*(Tw-Tc);
simQx=m_dot*Cp_f*(ineff_c-ineff_i)*0.01*(Tw-Tc);
```

Simulate3.m

```
function [ineff Th_out_ave Tc_out_ave Qz]=
simulate3(Nz,Nt,max_period,Tw,Tc,...
        h,As,HC_m,HC_f,lambda,m_dot,Cp_f)

    [ineff_i Th_out_ave_i
Tc_out_ave_i]=Ideal_Reg_Model(Nz,Nt,max_period,...
    Tw,Tc,h,As,HC_m,HC_f,lambda);

Th_out_ave=Th_out_ave_i;
Tc_out_ave=Tc_out_ave_i;
ineff=ineff_i;

Qz=m_dot*Cp_f*ineff_i*0.01*(Tw-Tc);
```

Fricloss.m

```
function [Qfric] = fricloss(G,rL,den_f,rh,m_dot,Re,porosity,mu_f,hd)

a=0.715*(5.6+porosity*(-16.363+porosity*13.928));
Y=a*Re^-0.43;

if Re<=10;
    A=0.0074*Re;
elseif Re>10 && Re<=3000;
    A=0.129-0.0058*((log(Re/200))^2);
else
    A=0.149-0.0239*(log(Re/200));
end


dP=rL*Y*(Re^2)*(mu_f^2)/(A*(hd^3)*den_f);      % Pressure Drop [Pa]
Qfric=m_dot*dP/den_f;
```

Ideal_Reg_Model.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ineff_i Th_out_ave_i Tc_out_ave_i] = Ideal_Reg_Model(Nz,Nt,...
    max_period,Tw,Tc,h,As,HC_m,HC_f,lambda)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hot=1;
cold=2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NumericalVariables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_t=lambda/Nt;           % [s]           Time increment

Tm=zeros(2,Nt,Nz);
Tf=zeros(2,Nt,Nz);

Th_out_sum=0;
Tc_out_sum=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalcRegProp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_NTU=-(h*As/Nz)/HC_f;
delta_Cr=(HC_m/Nz)/(HC_f*delta_t);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 K1=(1/(1+1/delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));
 K2=(1/(1+delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));

for p=1:1:max_period;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % HeatingPeriod
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%   Initial Condition   %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if p==1;                                % At the beginning a linear
        for i = 1:Nz                        % distrubution is assumed.
        Tm(hot,1,i)=Tw-(i-1)*(Tw-Tc)/Nz;
        end
    else
        for i=1:Nz
        Tm(hot,1,i)=Tm(cold,Nt,Nz-(i-1));   % In later stages matrix
        end                                    % temperature distrubition
    end                                        % is taken from previous
                                            %  period (cooling period).
    %%%  Boundry Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    for j=1:Nt                               % Fluid hot end entrance
    Tf(hot,j,1)=Tw;                          % temperature is constant
    end                                      % and equal to Tw at all times

    Th_out_sum_local=0;

    %%% Heating Period Calculations %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:(Nt)

        for i=1:(Nz)

            Tf(hot,j,i+1)=Tf(hot,j,i)-K1*(Tf(hot,j,i)-Tm(hot,j,i));
            Tm(hot,j+1,i)=Tm(hot,j,i)+K2*(Tf(hot,j,i)-Tm(hot,j,i));

        end
        Th_out_sum_local=Tf(hot,j,Nz)+Th_out_sum_local;
    end

    Th_out_ave_local=(Th_out_sum_local/Nt);
    Th_out_sum=Th_out_ave_local+Th_out_sum;


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CoolingPeriod
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%  Initial Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%

    for i=1:Nz                                % Initial matrix temperature
    Tm(cold,1,i)=Tm(hot,Nt,Nz-(i-1));         % distrubition is equal to
    end                                       % the one at the end of
                                           % heating period
    %%%  Boundry Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:Nt                         % Fluid cold end entrance
    Tf(cold,j,1)=Tc;                   % temperature is constant and
    end                                % equal to Tc at all times

    Tc_out_sum_local=0;

    %%% Cooling Period Calculations %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:(Nt)

        for i=1:(Nz)
```

```
            Tf(cold,j,i+1)=Tf(cold,j,i)-K1*(Tf(cold,j,i)-Tm(cold,j,i));
            Tm(cold,j+1,i)=Tm(cold,j,i)+K2*(Tf(cold,j,i)-Tm(cold,j,i));

        end
        Tc_out_sum_local=Tf(cold,j,Nz)+Tc_out_sum_local;
    end

    Tc_out_ave_local=(Tc_out_sum_local/Nt);
    Tc_out_sum=Tc_out_ave_local+Tc_out_sum;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateIneff
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Th_out_ave_i=Th_out_sum/(max_period);
Tc_out_ave_i=Tc_out_sum/(max_period);
eff=(Tw-Th_out_ave_i)/(Tw-Tc);

ineff_i=(1-eff)*100;
```

Ideal_Reg_Model_Long_Cond.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ineff_c Th_out_ave_c Tc_out_ave_c] =...
    Ideal_Reg_Model_Long_Cond(Nz,Nt,max_period,Tw,Tc,a,Km0,h,As,Am,...
    HC_m,HC_f,lambda,rL)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hot=1;
cold=2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NumericalVariables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_t=lambda/Nt;           % [s]           Time increment

Tm=zeros(2,Nt,Nz);
Tf=zeros(2,Nt,Nz);

Th_out_sum=0;
Tc_out_sum=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalcRegProp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_z=rL/Nz;
delta_NTU=-(h*As/Nz)/HC_f;
delta_Cr=(HC_m/Nz)/(HC_f*delta_t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


 K1=(1/(1+1/delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));
 K2=(1/(1+delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));
 K3i=Am*delta_t/((HC_m/Nz)*delta_z);         %%% Without Km


 for p=1:1:max_period;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % HeatingPeriod
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%   Initial Condition   %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if p==1;
        for i = 1:Nz
        Tm(hot,1,i)=Tw-(i-1)*(Tw-Tc)/Nz;
        end
```

```
else
    for i=1:Nz
    Tm(hot,1,i)=Tm(cold,Nt,Nz-(i-1));
    end
end

%%%  Boundry Condition  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:Nt
Tf(hot,j,1)=Tw;
end

Th_out_sum_local=0;

%%% Heating Period Calculations %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j=1:(Nt)

    for i=1:(Nz)

        if i==1;
        Tm_p=2*(Tm(hot,j,i+1)-Tm(hot,j,i));
        Tm_dp=(Tm(hot,j,i+2)-2*Tm(hot,j,i+1)+Tm(hot,j,i));

        elseif i==Nz;
        Tm_p=2*(Tm(hot,j,i)-Tm(hot,j,i-1));
        Tm_dp=(Tm(hot,j,i-2)-2*Tm(hot,j,i-1)+Tm(hot,j,i));

        else
        Tm_p=(Tm(hot,j,i-1)-Tm(hot,j,i+1));
        Tm_dp=(Tm(hot,j,i-1)-2*Tm(hot,j,i)+Tm(hot,j,i+1));

        end

        Km=Km0*((Tm(hot,j,i)/300)^a);
        K3=K3i*Km;

        Tf(hot,j,i+1)=Tf(hot,j,i)-K1*(Tf(hot,j,i)-Tm(hot,j,i));
        Tm(hot,j+1,i)=Tm(hot,j,i)+K2*(Tf(hot,j,i)-Tm(hot,j,i))+...
            K3*(a*(Tm_p^2)/(4*Tm(hot,j,i))+Tm_dp);

    end
    Th_out_sum_local=Tf(hot,j,Nz)+Th_out_sum_local;
end

Th_out_ave_local=(Th_out_sum_local/Nt);
Th_out_sum=Th_out_ave_local+Th_out_sum;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CoolingPeriod
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%  Initial Condition  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    for i=1:Nz
    Tm(cold,1,i)=Tm(hot,Nt,Nz-(i-1));
    end

    %%%  Boundry Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    for j=1:Nt
    Tf(cold,j,1)=Tc;
    end

    Tc_out_sum_local=0;

    %%% Cooling Period Calculations %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:(Nt)

        for i=1:(Nz)

            if i==1;
            Tm_p=2*(Tm(cold,j,i+1)-Tm(cold,j,i));
            Tm_dp=(Tm(cold,j,i+2)-2*Tm(cold,j,i+1)+Tm(cold,j,i));

            elseif i==Nz;
            Tm_p=2*(Tm(cold,j,i)-Tm(cold,j,i-1));
            Tm_dp=(Tm(cold,j,i-2)-2*Tm(cold,j,i-1)+Tm(cold,j,i));

            else
            Tm_p=(Tm(cold,j,i-1)-Tm(cold,j,i+1));
            Tm_dp=(Tm(cold,j,i-1)-2*Tm(cold,j,i)+Tm(cold,j,i+1));

            end

            Km=Km0*((Tm(cold,j,i)/300)^a);
            K3=K3i*Km;

            Tf(cold,j,i+1)=Tf(cold,j,i)-K1*(Tf(cold,j,i)-Tm(cold,j,i));
            Tm(cold,j+1,i)=Tm(cold,j,i)+K2*(Tf(cold,j,i)-...
                Tm(cold,j,i))+K3*(a*(Tm_p^2)/(4*Tm(cold,j,i))+Tm_dp);

        end
        Tc_out_sum_local=Tf(cold,j,Nz)+Tc_out_sum_local;
    end

    Tc_out_ave_local=(Tc_out_sum_local/Nt);
    Tc_out_sum=Tc_out_ave_local+Tc_out_sum;

 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateIneff
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Th_out_ave_c=Th_out_sum/(max_period);
Tc_out_ave_c=Tc_out_sum/(max_period);
eff=(Tw-Th_out_ave_c)/(Tw-Tc);
ineff_c=(1-eff)*100;
```

Ideal_Reg_Model_Long_Cond_and_Wall_Effect.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ineff_w Th_out_ave_w Tc_out_ave_w] =...
    Ideal_Reg_Model_Long_Cond_and_Wall_Effect(Nz,Nt,max_period,...
    Tw,Tc,a,b,Km0,Kwl0,h,As,Am,Awl_h,HC_m,HC_f,HC_wl,lambda,rL,Awl_s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hot=1;
cold=2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NumericalVariables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_t=lambda/Nt;            % [s]           Time increment
delta_z=rL/Nz;                % [m]           Distance increment

Tm=zeros(2,Nt,Nz);
Tf=zeros(2,Nt,Nz);
Twl=zeros(2,Nt,Nz);

Th_out_sum=0;
Tc_out_sum=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalcRegProp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

delta_NTU=-(h*As/Nz)/HC_f;
delta_NTU_wl=-(h*Awl_h/Nz)/HC_f;

delta_Cr=(HC_m/Nz)/(HC_f*delta_t);
delta_Cr_wl=(HC_wl/Nz)/(HC_f*delta_t);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateK
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 K1=(1/(1+1/delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));
 K2=(1/(1+delta_Cr))*(1-exp((delta_NTU)*(1+1/delta_Cr)));
 K3i=Am*delta_t/((HC_m/Nz)*delta_z);         %%% Without Km

 K4=(1/(1+1/delta_Cr_wl))*(1-exp((delta_NTU_wl)*(1+1/delta_Cr_wl)));
 K5=(1/(1+delta_Cr_wl))*(1-exp((delta_NTU_wl)*(1+1/delta_Cr_wl)));
 K6i=Awl_s*delta_t/((HC_wl/Nz)*delta_z);          %%% Without Kwl
```

```
for p=1:1:max_period;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % HeatingPeriod
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%   Initial Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if p==1;
        for i = 1:Nz
        Tm(hot,1,i)=Tw-(i-1)*(Tw-Tc)/Nz;
        Twl(hot,1,i)=Tw-(i-1)*(Tw-Tc)/Nz;
        end
    else
        for i=1:Nz
        Tm(hot,1,i)=Tm(cold,Nt,Nz-(i-1));
        Twl(hot,1,i)=Twl(cold,Nt,Nz-(i-1));
        end
    end

    %%%  Boundry Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    for j=1:Nt
    Tf(hot,j,1)=Tw;
    end

    Th_out_sum_local=0;

    %%% Heating Period Calculations %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:(Nt)

        for i=1:(Nz)

            if i==1;
            Tm_p=2*(Tm(hot,j,i+1)-Tm(hot,j,i));
            Tm_dp=(Tm(hot,j,i+2)-2*Tm(hot,j,i+1)+Tm(hot,j,i));
            Twl_p=2*(Twl(hot,j,i+1)-Twl(hot,j,i));
            Twl_dp=(Twl(hot,j,i+2)-2*Twl(hot,j,i+1)+Twl(hot,j,i));

            elseif i==Nz;
            Tm_p=2*(Tm(hot,j,i)-Tm(hot,j,i-1));
            Tm_dp=(Tm(hot,j,i-2)-2*Tm(hot,j,i-1)+Tm(hot,j,i));
            Twl_p=2*(Twl(hot,j,i)-Twl(hot,j,i-1));
            Twl_dp=(Twl(hot,j,i-2)-2*Twl(hot,j,i-1)+Twl(hot,j,i));

            else
            Tm_p=(Tm(hot,j,i-1)-Tm(hot,j,i+1));
            Tm_dp=(Tm(hot,j,i-1)-2*Tm(hot,j,i)+Tm(hot,j,i+1));
            Twl_p=(Twl(hot,j,i-1)-Twl(hot,j,i+1));
            Twl_dp=(Twl(hot,j,i-1)-2*Twl(hot,j,i)+Twl(hot,j,i+1));

            end

            Km=Km0*((Tm(hot,j,i)/300)^a);
            K3=K3i*Km;
            Kwl=Kwl0*((Twl(hot,j,i)/300)^b);
            K6=K6i*Kwl;

            Tf(hot,j,i+1)=Tf(hot,j,i)-K1*(Tf(hot,j,i)-Tm(hot,j,i))-...
                K4*(Tf(hot,j,i)-Twl(hot,j,i));
            Tm(hot,j+1,i)=Tm(hot,j,i)+K2*(Tf(hot,j,i)-Tm(hot,j,i))+...
                K3*(a*(Tm_p^2)/(4*Tm(hot,j,i))+Tm_dp);
            Twl(hot,j+1,i)=Twl(hot,j,i)+K5*(Tf(hot,j,i)-Twl(hot,j,i))+...
                K6*(b*(Twl_p^2)/(4*Twl(hot,j,i))+Twl_dp);
        end
        Th_out_sum_local=Tf(hot,j,Nz)+Th_out_sum_local;
    end

    Th_out_ave_local=(Th_out_sum_local/Nt);
    Th_out_sum=Th_out_ave_local+Th_out_sum;


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % CoolingPeriod
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%  Initial Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:Nz
    Tm(cold,1,i)=Tm(hot,Nt,Nz-(i-1));
    Twl(cold,1,i)=Twl(hot,Nt,Nz-(i-1));
    end

    %%%  Boundry Condition  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for j=1:Nt
    Tf(cold,j,1)=Tc;
    end

    Tc_out_sum_local=0;

    %%% Cooling Period Calculations %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for j=1:(Nt)

        for i=1:(Nz)

            if i==1;
            Tm_p=2*(Tm(cold,j,i+1)-Tm(cold,j,i));
            Tm_dp=(Tm(cold,j,i+2)-2*Tm(cold,j,i+1)+Tm(cold,j,i));
            Twl_p=2*(Twl(cold,j,i+1)-Twl(cold,j,i));
            Twl_dp=(Twl(cold,j,i+2)-2*Twl(cold,j,i+1)+Twl(cold,j,i));

            elseif i==Nz;
            Tm_p=2*(Tm(cold,j,i)-Tm(cold,j,i-1));
            Tm_dp=(Tm(cold,j,i-2)-2*Tm(cold,j,i-1)+Tm(cold,j,i));
            Twl_p=2*(Twl(cold,j,i)-Twl(cold,j,i-1));
            Twl_dp=(Twl(cold,j,i-2)-2*Twl(cold,j,i-1)+Twl(cold,j,i));

            else
            Tm_p=(Tm(cold,j,i-1)-Tm(cold,j,i+1));
            Tm_dp=(Tm(cold,j,i-1)-2*Tm(cold,j,i)+Tm(cold,j,i+1));
            Twl_p=(Twl(cold,j,i-1)-Twl(cold,j,i+1));
            Twl_dp=(Twl(cold,j,i-1)-2*Twl(cold,j,i)+Twl(cold,j,i+1));

            end

            Km=Km0*((Tm(cold,j,i)/300)^a);
            K3=K3i*Km;
            Kwl=Kwl0*((Twl(cold,j,i)/300)^b);
            K6=K6i*Kwl;

            Tf(cold,j,i+1)=Tf(cold,j,i)-K1*(Tf(cold,j,i)-Tm(cold,j,i))...
                -K4*(Tf(cold,j,i)-Twl(cold,j,i));
            Tm(cold,j+1,i)=Tm(cold,j,i)+K2*(Tf(cold,j,i)-Tm(cold,j,i))...
                +K3*(a*(Tm_p^2)/(4*Tm(cold,j,i))+Tm_dp);
            Twl(cold,j+1,i)=Twl(cold,j,i)+K5*(Tf(cold,j,i)-...
                Twl(cold,j,i))+K6*(b*(Twl_p^2)/(4*Twl(cold,j,i))+Twl_dp);
        end
        Tc_out_sum_local=Tf(cold,j,Nz)+Tc_out_sum_local;
    end

    Tc_out_ave_local=(Tc_out_sum_local/Nt);
    Tc_out_sum=Tc_out_ave_local+Tc_out_sum;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CalculateIneff
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Th_out_ave_w=Th_out_sum/(max_period);
Tc_out_ave_w=Tc_out_sum/(max_period);
eff=(Tw-Th_out_ave_w)/(Tw-Tc);
ineff_w=(1-eff)*100;
```