Cryogenic Regenerative Heat Exchanger Performance Mapping using REGEN3.3

By

Renzhuo Wang

A thesis submitted in partial fulfilment of the requirements for the degree of:

Master of Science

Mechanical Engineering

at the

University of Wisconsin-Madison

2014

This thesis has been approved by

Professor John M. Pfotenhauer	**Data**
Professor Franklin Miller	**Data**
Professor Gregory F. Nellis	**Data**

Acknowledgements

I owed the deepest thanks to Yang, Shuonan and my parents. I can never do this without your love and support, it’s your love that help me through all the days and nights. And thank you for my advisors, Professor John Pfotenhauer and Professor Franklin Miller, for all the valuable time and advice you have given me, I learned from you every day, and those will be invaluable forever. Thanks Professor Sanford Klein for generously letting me use all the idle computers in SEL, I would still be waiting for the results otherwise.

I also need to thank all my friends and colleagues at SEL, Amir, Brad, Brian, Diego, Dongsheng, Fritz, Kyle, John, Russell, Wenjie and all the others I have met here in Madison. And my deepest thanks to my closet friends, hope the friendship never ends.

Contents

Acknowledgements v
Contents vi
List of Figures ix
Nomenclature ix
Abbreviations x
Variables xi
0. Introduction 1
1. Literature Review 4
1.1. History of Stirling-type Cryogenic Refrigerator 4
1.2. Applications 4
1.3. Regenerator 4
1.4. Advantages of Regenerators 5
1.5. Regenerator Simulation and Review 6
1.5.1. Sage 7
1.5.2. DELTA-EC 7
1.5.3. ARCOPTR 8
1.5.4. REGEN 8
1.6. Previous Work Review 9
1.7. Research Objectives 10
2. Regenerator Theory 12
2.1. Cryocooler Cyclic Analysis 12
2.2. Regenerator Theory and One Dimensional Model Analysis 13
2.3. Regen3.3 Variables 16
3. Calculation Procedure and Variable Setup 19
3.1. Optimization Procedure 19
3.2. Convergence Requirement 21
3.2.1. Final Cycles 21

3.2.2. Midpoint Temperature Ratio 24
3.2.3. Number of Mesh Points 25
3.2.4. Number of Time Steps per Cycle 27
3.2.5. Variables Choice Summary 28
3.3. Implementation of Calculation 28
3.3.1. The Innermost Loop 29
3.3.2. Mapping the Optimal Variables 32
4. Results and Discussion 34
4.1. Uncertainty Analysis 34
4.1.1. Sources of Uncertainty for CFD Modeling 34
4.1.2. Possible Sources of Uncertainties in REGEN3.3 35
4.1.3. Quantization of Uncertainty in REGEN3.3 36
4.1.4. Uncertainty Calculation 37
4.2. Optimal COP and Correspondent Variables versus Frequency 39
4.3. Optimal COP Shell 44
4.4. Convergence Issues for REGEN3.3 and Solutions 52
4.4.1. Description and Cause of Convergence Issues 52
4.4.2. Solutions to the Convergence Problem 53
4.4.3. Limits of REGEN3.3 56
4.5. Case Comparison of REGEN3.3 Modeling Results with Experimental Data 56
4.6. Scale Relationship Between Cross Sectional Area and Cooling Power 57
4.7. Conclusion 58
References 60
Appendix 62
A. Fitted Equations 62
B. Codes for Programms 64
1. C code for consecutively running Regen3.3 64
2. Matlab code for generating massive data files 65

3. Python code for reading massive output files 70
4. Matlab and python code for making the shell 77

List of Figures

Figure 1 Schematic of (a) wire mesh matrix and (b) randomly stacked sphere matrix [1] 1

Figure 2 Flow through an infinite randomly stacked woven-screen matrix, friction factor vs. Reynolds number[15] 10

Figure 3 Schematic of two Stirling-type cryocooler [16] 12

Figure 4 Optimization Procedure Loop 19

Figure 5 COP vs. Final Cycles 22

Figure 6 COP vs. Number of Final Cycles at 35K 23

Figure 7 COP vs. Number of Mesh Points for Three Cases 26

Figure 8 COP vs. Number of Time Steps 27

Figure 9 40mm, 40Hz, 80K, COP vs. Ag/mc 30

Figure 10 40mm, 40Hz, 80K, Optimal COP vs. Phase Angle 31

Figure 11 80K, 40Hz, 40mm, Optimal COP vs. Length 32

Figure 12 Optimal COP vs. Frequency at 80K, 60K and 35K predicting model 39

Figure 13 COP/Length vs. Frequency 40

Figure 14 Optimal Phase Angle vs. Frequency at 80K, 60K and 35K 41

Figure 15 Optimal Ag/mc vs. Frequency at 80K, 60K and 35K 42

Figure 16 Optimal Length vs. Frequency at 80K, 60K and 35K 43

Figure 17 cooling power per area at optimal performance vs. frequency at various temperatures 44

Figure 18 2% and 5% off optimal COP shell at 40Hz, 80K 45

Figure 19 COP shell 2% off optimum contour view, Length vs. Ag/mc at various phase angles 46

Figure 20 COP shell 2% off optimum contour view, Length vs. Phase angle for various Ag/mc 47

Figure 21 COP shell 2% off optimum contour view, Ag/mc vs phase angle at various lengths 48

Figure 22 98% and 95% of optimal COP, Phase angle vs. Length for optimal Ag/mc value 49

Figure 23 98% and 95% of optimal COP, Length vs. Ag/mc for optimal phase angle 50

Figure 24 98% and 95% of optimal COP, Ag/mc vs. Phase angle for optimal phase angle 51

Figure 25 Cooling Power vs. Cross Sectional Area at 60K, 30Hz, 0.065m 58

Nomenclature

Abbreviations

COOLING_MULT	correction factor for adjusting cooling power
EHTDIF	the difference between the maximum and minimum over the mesh of the integral average of the sum of the enthalpy flux and matrix heat flux
ENGBAL	energy inbalance at over per cycle
ENTFLX	the integral average of the enthalpy flux at the cold end of the regenerator over one cycle
GAS TEMP COLD	cold end gas temperature
GAS TEMP HOT	hot/warm gas end temperature
GTPNRM	normalized midpoint temperature output
HTFLUX	the heat flux due to thermal conduction through the cold side of the matrix
MID_TEMP_RATIO	midpoint temperature ratio
NTACOP	variable name for COP in REGENE3.3 output file
NTCADJ	net adjusted cooling power
NUM_STEPS_CYC	number of time steps per cycle
PRLOSS	estimate the effect of pressurization on the enthalpy flux of the gas
PVWK0T	warm end PV work done by gas
PVWKT1	cold end PV work done by gas
RGLOSS	heat loss due to regenerator ineffectiveness
TUBECD	thermal conduction through the tube containing the regenerator matrix

Variables

$\dot{W}_0$	warm end input power
$\dot{Q}_0$	transfer of heat between the flow out of the compressor and room temperature
T_0	room temperature
$\dot{Q}_c$	transfer of heat at cold end of regenerator
T_c	cold end temperature of the regenerator
$\dot{Q}_h$	heat rejection at between pulse tube and reservoir
T_h	heat rejection temperature
COP	coefficient of performance
ρ	density of fluid
v	gas flow velocity
t	time
x	longitudinal direction coordinate
p	pressure
e	internal energy per unit volume
f	friction factor
T	temperature
$\emptyset$	porosity of porous medium
A	regenerator cross sectional area
k_g	thermal conductivity of the gas
T_m	temperature of the matrix
D	thermal content of the matrix
k_m	thermal conductivity of the matrix

T_{min}	reference temperature for calculating thermal content of the matrix
H	heat transfer coefficient
D_h	hydraulic diameter
τ	period of a flow cycle
ρ_1	density of gas at the cold end of the regenerator
L	length of the regenerator
h_{tube}	thickness of the tube wall
Po	charging pressure
f	frequency
θ	phase different between pressure oscillation and cold end mass flow
d	inner diameter of the regenerator
Ag	gross cross sectional area, cross section area A multiplies porosity
U_y	uncertainty of the calculated variable/dependt variable
y	calculated variable/depend variable
U_x	uncertainty of the measured variable/independent variable
x	measured variable/independent variable

0. Introduction

Regenerative heat exchanger, or regenerator, is a type of heat exchanger that is commonly used in cryogenic refrigerators. A common cryogenic regenerator is a stainless steel tube filled with matrix. The matrix is a high heat capacity material stacked in the tube with void space as gas paths. Schematics of Two common matrix type, woven wire mesh matrix and randomly stacked spheres are shown in Figure 1.

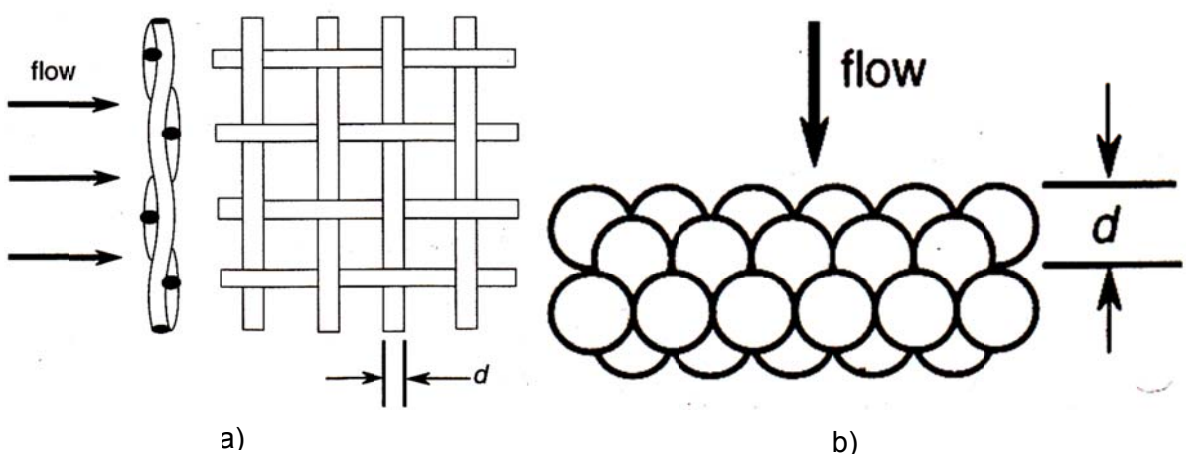

Figure 1 Schematic of (a) wire mesh matrix and (b) randomly stacked sphere matrix [1]

As a heat exchanger, regenerator is used for exchanging heat between two or more flow. Compared to a recuperative heat exchanger (counter or parallel flow heat exchanger) where two flow occupy separated space at all the time, the flow in the regenerator occupy the same space at separated period. During the cooling period, gas flows from the hot end flow through the regenerator, being gradually cooled by

the matrix and flow out at the cold end temperature; during the heating period, gas flows from the cold end through the regenerator, being gradually heated by the regenerator, and flow out at the hot end temperature. The heat exchange between the cold and hot fluid is done through a high heat capacity medium, the matrix. The purpose is for the gas to reach matrix temperature at two ends of the regenerator.

The regenerator conservation equations discussed in 2.2 does not have a close form solution. Ideal regenerator assumptions were proposed for solving the problem before computer age. The ideal regenerator assumptions include:

1. The thermal energy stored in the gas/fluid is much smaller compared to the thermal energy stored in the matrix material;
2. Both radial and circumferential flow is negligible compared to the longitudinal flow, thus the flow is one dimensional;
3. For the matrix, the thermal conductivity in the longitudinal direction is zero, and the thermal conductivity in the radial and circumferential direction is infinite;
4. The heat transfer coefficient between the fluid and the matrix is constant throughout the regenerator;
5. No mixing of fluid occurs during the reversal from hot to cold flow.

Ideal regenerator assumptions make the regenerator conservation equations solvable but are apparently not accurate in most applications. Therefore numerical models are often used.

Chapter 1 is a review of regenerator history and common numerical models used for simulating cryogenic regenerators. Then there is a review of similar researches published in open literature. Finally, the purposes and motivation for this thesis work are presented. Chapter 2 is a review of regenerator theory and the numerical model of REGEN3.3. Chapter 3 is the calculation procedure and discussions about some the setup of some variables. Chapter 4 is results and discussion, including uncertainty analysis, correlations for optimal COP and corresponding variables and insights on an optimal COP shell that shows the broad range of variables with near optimum efficiency.

1. Literature Review

1.1. History of Stirling-type Cryogenic Refrigerator

Regenerative heat exchangers or regenerators were first introduced by Robert Stirling for use in his hot air engine, which is commonly known as a Stirling engine. John Herschel suggested that the same cycle could be used as a refrigerator for making ice [2]. W.E Gifford, H.O. McMahon and other researchers at the Philips Corp. developed small regenerative type cryogenic refrigerators to cool infrared detectors and maser amplifiers [3]. From then on, cryogenic refrigeration was transferred from the laboratory to industrial applications.

1.2. Applications

Stirling-type refrigerators are the most widely used cryogenic refrigerators. The applications include thermal shields for MRI superconducting magnets [4], superconducting power applications [5], cooling infrared sensors by the military to temperatures of about 80 K [6], x-ray and gamma ray detectors in space astronomical programs [7], and high temperature superconducting filters [8].

1.3. Regenerator

The regenerator is a crucial component for the Stirling-type refrigerator as it is the major heat exchanger within the refrigerator. For example, in a single stage

regenerative cryogenic refrigerator with 1 W of cooling power at 80 K, a well-designed regenerator with 99% efficiency is responsible for exchanging 99 W of heat between room temperature and 80 K. That also means that within the entire refrigerator, most of the entropy generation occurs in the regenerator. Therefore, optimizing the regenerator is a key issue for improving a refrigerators' performance.

1.4. Advantages of Regenerators

The advantages of regenerators when used in miniature cryocoolers are:

1. They are relatively cheap and easy to manufacture
2. They enable a very large surface area to be concentrated into a small volume. For example, a 50 mesh screen matrix has an area density on the order of 65 cm^2/cm^3, compared with a maximum of 13 to 20 cm^2/cm^3 for the most compact direct transfer type of parallel-, counter, or cross-flow extended surface heat exchangers. Many regenerators used in cryocoolers now have screens with 400 wires per inch, or 400 mesh screens. This makes regenerators even more compact compared to recuperators.
3. Properly designed regenerators have a low hydraulic resistance and thus allow the passage of large quantities of gas with a very small loss in pressure.
4. Periodic flow reversal eliminates permanent flow stagnation regions, and consequently the surface tends to be self-cleaning.

1.5. Regenerator Simulation and Review

Regenerators only work under an oscillating flow condition. Therefore, no closed form solution exists that couples mass, momentum and energy conservation equations at unsteady state, unless some ideal assumptions are made. These assumptions include infinite heat capacity of the matrix material, no pressure loss due to friction across the regenerator, and an infinite heat transfer coefficient between the fluid flow and matrix. After making these assumptions, a closed form solution is attainable. However, all real regenerators will certainly include pressure loss, which usually is a significant effect and must be considered in order to obtain an accurate solution. The heat capacity of the matrix material is usually several orders of magnitude larger than that of the fluid. However, especially at lower temperatures, the heat capacity of the matrix drops significantly, sometimes even to a value smaller than that of the fluid. These complications make numerical modeling a logical choice for solving the regenerator problem.

Through the years, many efforts at providing a reliable numerical model to simulate regenerator performance have been carried out. Several numerical code packages that offer reliable solutions are available nowadays. Among them, REGEN developed at NIST[9], the SAGE software package developed by David Gedeon [10], DELTA EC developed at Los Alamos National Laboratory [11] and ARCOPTR developed by NASA Ames Research Center [12] are the most frequently used codes. Also commercial CFD software [13], and general purpose programming languages

are frequently used to simulate regenerator performance. The following is a brief review of the various software packages.

1.5.1. Sage

Sage is a graphical interface that supports simulation and optimization of an underlying class of engineering models. The software package can be used to model and optimize Stirling cycle engines and coolers, pulse-tube cryocoolers, and other types of cryocoolers. Sage is based on a number of specialized components that encapsulate gas flow, heat transfer and other modeling details. Interconnected components form complete models that simulate complicated systems. Based on fully connected model, Sage can solve, map and optimize the model. [10]

1.5.2. DELTA-EC

DeltaEC (Design Environment for Low-Amplitude Thermo Acoustic Energy Conversion), formerly Delta-E, is a computer program that can calculate details of how thermoacoustic equipment performs and help the user to design equipment that achieves a desired performance. DeltaEC numerically integrates in a1-D spatial domain using a low-amplitude, “acoustic” approximation and sinusoidal time dependence. Regenerative cryocoolers are simulated by coupling the complex pressure amplitude equation, complex volumetric flow rate equation and energy conservation equations [11].

1.5.3. ARCOPTR

ARCOPTR is a program developed at NASA-Ames Research Center for modeling pulse tube coolers. ARCOPTR is a 1-D model that treats all the components of an orifice- or inertance- type pulse tube cooler. It analyzes the thermodynamic equations for the regenerator and assumes that all mass flows, pressure oscillations and temperature oscillations are small and sinusoidal. Mass flow and pressure are matched at the boundaries with the other components of the cooler. An optimizer is built into the ARCOPTR model that allows optimization of the cooler parameters to attain maximum efficiency [12].

1.5.4. REGEN

REGEN is a software package that models helium flow and heat transfer in a regenerator of the type used in cryocoolers. Only the flow through the regenerator is modeled (no other parts of a regenerative cooler). The model, like DELTA-EC and ARCOPTR, is also 1-D in spatial domain. In the latest version REGEN3.3, a fully implicit method is used. Certain combinations of boundary conditions need to be specified, in order for a solution to be calculated through a user-specified number of iterations. Among the various outputs from the code are the enthalpy flux at the cold end and an estimate of the cooling power, under the assumption of isothermal expansion space [9]. Because the software does not include a built-in optimization procedure, optimizing the performance of the regenerator based on some output

parameters such as the coefficient of performance (COP) as a function of the various input parameters requires an external iteration process running REGEN multiple times over the desired range of input values.

1.6. Previous Work Review

Very few reports exist in the open literature that attempt to map the performance of a cryogenic regenerator. A similar report by Pfotenhauer and Shi [14] exists using Regen3.2. The objective of their study was to optimize the regenerator COP calculated in Regen3.2 as a function of the geometry, mass flux, and phase angle at fixed cold end temperatures of 60K and 80K. They found that the optimal length is independent of the cooling power but inversely dependent on the cold end temperature and frequency. They also found that the optimized cross section area varies close to linearly with the desired cooling power. However due to limited computing time, their results were not accurate enough, and the scope of the study was also limited due to the limited computation power.

1.7. Research Objectives

In the book “Compact Heat Exchangers” by Kays and London[15], a series of graphs are used to show correlations between certain variables for certain types of heat exchangers. By collapsing large amounts of experimental or theoretical data into a few graphs, complicated correlations became intuitive for readers to understand. Designers can easily extrapolate correlations from the graphs without going into complicated calculations.

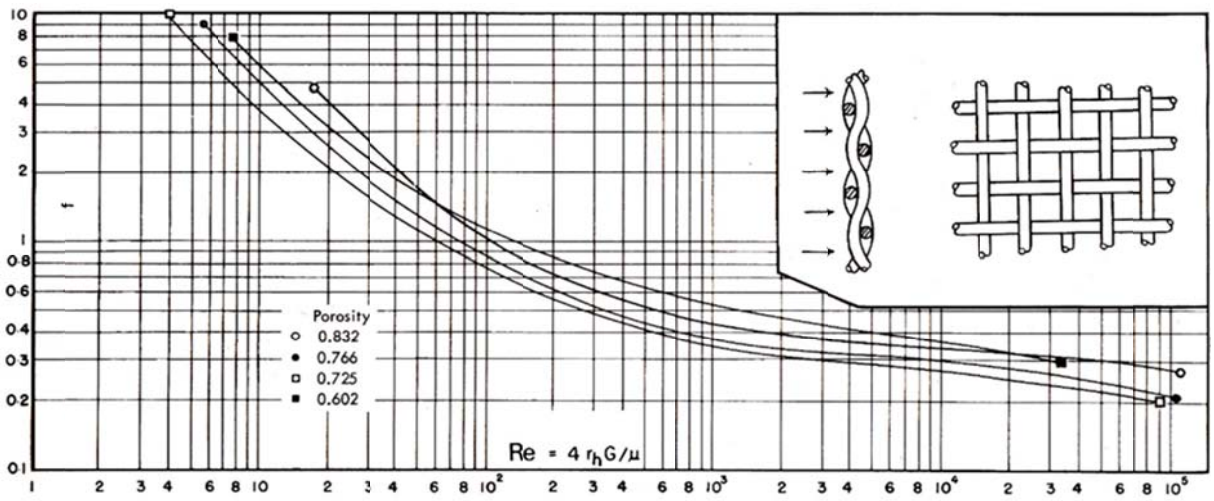

Figure 2 Flow through an infinite randomly stacked woven-screen matrix, friction factor vs. Reynolds number[15]

For example, Figure 2 shows the friction factor versus Reynolds number for flow through an infinite randomly stacked woven-screen matrix derived from experiment data with varying porosity [15]. As stated before, the graphs convey extensive qualitative and quantitative information with ease.

Offering graphs to describe the optimizations that result from a parametric study using REGEN is the motivation of this thesis’ work. The numerical models

mentioned in previous sections could be intimidating for new users. For instance, REGEN3.3 has over 60 input variables that can be freely defined by users, not to mention that it allows user definitions for heat transfer correlations and fluid properties. Even with the default variable settings, there are still at least 12 variables that must be defined from user input. The freedom in choice makes the numerical model more powerful and robust, as an almost infinite number of possibilities can be explored. However, for users without deep understanding of regenerators, or for those lacking the experience in numerical modeling, the freedom may become a serious challenge.

The objective of this thesis is to provide a mapping of cryocooler performance versus multiple regenerator design parameters at conditions of wide spread interest. By optimizing the COP over the set of variables including the cold end phase angle, average pressure, regenerator length, and mass flux for fixed values of cold end temperature and operating frequency, a set of optimized parameters are obtained at temperatures of 80 K, 60 K, and 35 K and with frequencies from 30 Hz to 300 Hz. The results allow approximate optimized designs for temperatures and frequencies intermediate to those studied here via interpolation. In addition, changes in design as a function of cooling power at any of the fixed temperature and frequency points are enabled by a simple scaling relationship. Results of the model are also compared with published cryocooler performance.

2. Regenerator Theory

The focus of this thesis is cryogenic regenerators, which is the major heat exchanger in a Stirling-type cryocooler. Understanding the working mechanism of a cryocooler is therefore important for understanding this thesis. The working mechanism of cryocoolers and the theory of regenerators is discussed in this section. The theory of regenerators will be combined with an explanation of Regen3.3's numerical model.

2.1. Cryocooler Cyclic Analysis

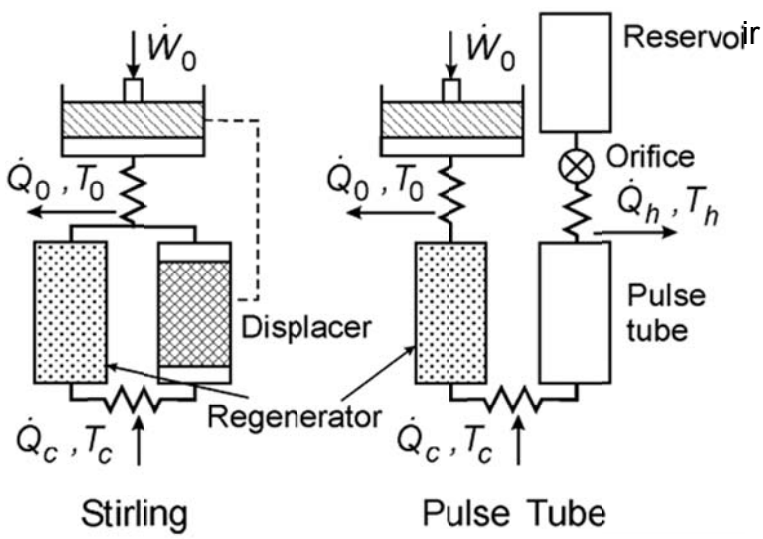

Figure 3 Schematic of two Stirling-type cryocooler [16]

There are two common variations of Stiriling-type cryocoolers, Stiriling and Pulse tube type. Schematics of the two are shown in Figure 3. Here we use the pulse tube cryocooler as an example and define the contained gas as the system. In the compression cycle, the input work $\dot{W}_0$ is done on the system through compression. The gas exchanges heat $\dot{Q}_0$ with the surroundings at room temperature T_0. After

being cooled by the regenerator matrix, the gas expands in the pulse tube, which is essentially an empty tube, while absorbing heat $\dot{Q}_c$ at cold end temperature T_c in the cold heat exchanger. The gas rejects heat $\dot{Q}_h$ in the hot heat exchanger at temperature T_h. When the compression space expands, the cold fluid in the pulse tube will flow back to the compression chamber, cooling the heated regenerator matrix in the process. The regenerator works as a thermal energy storage medium in the process, cooling the warm fluid when it flows from room temperature to the cold end, and being cooled by the cold fluid that flows back to the compression chamber from the pulse tube. The coefficient of performance for the cryocooler is defined as

$$COP = \frac{\dot{Q}_c}{\dot{W}_0} \tag{1}$$

Therefore, an optimal COP is associated with a maximum cooling power for a certain input power or a minimum input power for a certain cooling power.

2.2. Regenerator Theory and One Dimensional Model Analysis

The theory associated with the function of a regenerator should include the thermal energy exchange occurring between the matrix material and fluid. To develop the equations describing this exchange, a differential element of the regenerator is established for the matrix material and fluid. The conservation of mass, momentum and energy are employed. A one-dimensional spatial model will be discussed as an example, which is the same as that used in Regen3.3.

The fluid flow in a regenerator is similar to flow in an open tube, which can be described by the following three conservation equations:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0 \qquad (2)$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial(\rho v^2 + p)}{\partial x} = 0 \qquad (3)$$

$$\frac{\partial e}{\partial t} + \frac{\partial((e+p)v)}{\partial x} = 0 \qquad (4)$$

representing the conservation of mass, momentum, and energy respectively. Here □ is the density, v the velocity, p the pressure, and e the specific internal energy that has units of [J/m^3]. These equations are sufficient for flow in an open tube, but they are not adequate for flow in regenerators, which could be better described as flow in porous medium. The derivation of the flow equations used in Regen3.3 originates from those in Kays and London [15]. The flow equations assume that the effects of the porous media on the momentum equation can be modeled by the addition of a "friction" term in the momentum equation and a heat transfer term in the energy equation. The modified momentum equation becomes:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + \frac{1}{\rho}\left(\frac{\partial p}{\partial x} - f(p,T,v)\right) = 0 \qquad (5)$$

where $f(p,T,v)$ is the extra friction term, which is a function of pressure, temperature and flow velocity.

The energy conservation accounts for both the fluid and the matrix material. For the fluid, the energy conservation becomes:

$$\frac{\partial \emptyset A e}{\partial t} + \frac{\partial(\emptyset A(e+p)v)}{\partial x} - \frac{\partial\left(\emptyset A k_g \frac{\partial T}{\partial x}\right)}{\partial x} - \emptyset A q(p, T, T_m, v) = 0 \qquad (6)$$

For the regenerator matrix, the energy conservation becomes:

$$\frac{\partial D}{\partial t} - \frac{(1-\emptyset) A k_m \frac{\partial T_m}{\partial x}}{\partial x} + \emptyset A q(p, T, T_m, v) = 0 \qquad (7)$$

Where D is the matrix thermal content, defined by

$$D(x, T) = \int_{T_{min}}^{T} (1 - \emptyset) A\, c_m(x, T) dT \qquad (8)$$

in which c_m is the heat capacity of the matrix.

The heat transfer term is written as:

$$q(p, T, T_m, v) = \frac{4H(p,T,v)(T_m - T)}{D_h} \qquad (9)$$

For the four equations above, ∅ is porosity of the matrix, A is the total cross sectional area, E is the specific internal energy of the fluid, p is the pressure, v is the velocity of the fluid, T_m is the temperature of the matrix, T is the temperature of the fluid, $k_g$is the fluid thermal conductivity, k_m is the matrix thermal conductivity, H is the heat transfer coefficient between fluid and the matrix within the porous medium, and D_h is the hydraulic diameter for the matrix.

The friction term $f(p, T, v)$ and heat transfer coefficient $H(p, T, v)$ used for Regen3.3 are obtained by fitting data for steady flow correlations from Kays and London [15].

The one-dimensional numerical model of the regenerator is developed by writing the above equations ($\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0$ (2, $\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + \frac{1}{\rho}\left(\frac{\partial p}{\partial x} - f(p,T,v)\right) = 0$

(5 $\partial D \partial t - \frac{(1-\emptyset)Ak_m\frac{\partial T_m}{\partial x}}{\partial x} + \emptyset Aq(p,T,T_m,v) = 0$ (7) into a discrete form, which is the basis for the Regen3.3 model.

2.3. Regen3.3 Variables

The variable name for COP in Regen3.3 is NTACOP, which is written as

$$NTACOP = \frac{NTCADJ}{|PVWK0T|} \quad (10)$$

Here NTACOP is the coefficient of performance, NTCADJ is net adjusted cooling power and PVWK0T is the work done by the gas at the warm end of the regenerator, which is usually a negative number. PVWK0T is calculated assuming isothermal compression at warm end of the regenerator, and thus can be seen as the ideal compression work done to the gas, or the warm end PV work.

The net adjusted cooling power (NTCADJ) is written in the form of the variables used in Regen3.3 as following:

$$NTCADJ = (PVWK1 - PRLOSS) \cdot COOLING_\text{MULT} - RGLOSS - HTFLUX - TUBECD \quad (11)$$

Where PVWK1 is the cold end PV work, PRLOSS is the correction term used to estimate the effect of pressurization on the enthalpy flux of the gas,

COOLING_MULT is a factor specified by the user to reduce the gross refrigeration power (which is PVWK1-PRLOSS), RGLOSS is the loss due to regenerator ineffectiveness, HTFLUX is the heat flux due to thermal conduction through the cold side of the matrix and TUBECD is thermal conduction through the tube containing the regenerator matrix.

Further break down these variables:

1. PVWK1 is computed from the integral of $\int_{t-\tau}^{t} pdV$, where τ is the period of a cycle, p is the pressure and V is the volume of gas in the isothermal expansion chamber, both p and V are evaluated at the cold end. This is equivalent to the following integral:

$$\int_{t-\tau}^{t} \frac{\emptyset Av(L,t)p(t)\rho(L,t)}{\rho_1 \tau} dt \qquad (12)$$

where ρ_1 is the density at the cold end temperature T_1.

PVWK0T is also calculated with a similar integral, but using properties at the warm end instead of the cold end. Both PVWK1 and PVWK0T are in units of watts.

2. PRLOSS is a correction term used to estimate the effect of pressurization on the enthalpy flux of the gas, which is written as:

$$\max[\int_{t-\tau}^{t} \frac{\emptyset Av(0,t)h(\mathrm{p}(t),T_0)\rho(0,t)}{\tau} dt \ , \int_{t-\tau}^{t} \frac{\emptyset Av(L,t)h(\mathrm{p}(t),T_\mathrm{c})\rho(L,t)}{\tau} dt] \qquad (13)$$

where h is the enthalpy of the fluid. PRLOSS also has the unit of watts.

3. RGLOSS is the loss due to regenerator ineffectiveness, which is written as;

$$RGLOSS = ENTFLX - PRLOSS \tag{14}$$

where ENTFLX is the integral average of the enthalpy flux at the right side of the regenerator over one cycle. This excludes the loss due to pressurization and conduction. RGLOSS also has the unit of watts.

4. HTFLUX is the heat flux due to thermal conduction through the cold side of the matrix. HTFLUX is also in watts.
5. TUBECD is the thermal conduction through the tube containing the regenerator matrix. The tube is assumed to be stainless steel with a steady state temperature profile and the thermal exchange between the tube and the matrix is ignored. TUBECD is written as following:

$$\mathrm{TUBECD} = -\frac{\mathrm{h_{tube}}\sqrt{4\pi \mathrm{A}}}{\mathrm{L}} \int_{\mathrm{T_0}}^{\mathrm{T_c}} k_m(\mathrm{T})\mathrm{dT} \tag{15}$$

where A_r is the regenerator cross sectional area, h_{tube} is the thickness of the tube wall, L is the regenerator length, T_c and T_0 is the hot and cold end temperature, respectively and $k_m(\mathrm{T})$ is the conductivity of stainless steel. The above equation is assuming steady with no heat transfer from the tube to the matrix. The integral is computed using the temperature profile along the regenerator at the end of the cycle.

3. Calculation Procedure and Variable Setup

3.1. Optimization Procedure

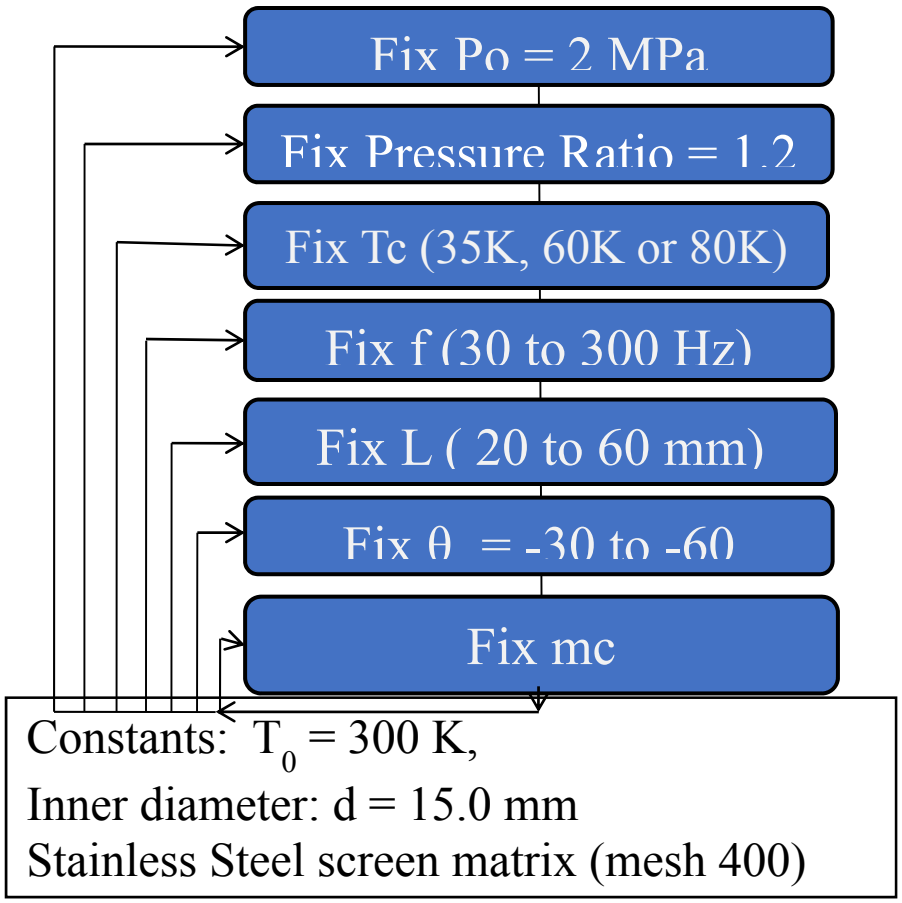

Figure 4 Optimization Procedure Loop

The optimizing goal, like stated in Chapter 2, is COP. The optimization procedure follows a nested loop as shown above in **Error! Reference source not found.**. First the charging pressure is fixed at 2MPa, and the pressure ratio is fixed at 1.2. Both are common operating conditions that can usually be found in many applications. As

the optimization focus on a single stage cryocooler, the warm end temperature would be 300K. Next a cold end temperature is chosen. Three different cold end temperatures have been investigated, 35K, 60K and 80K. After fixing the cold end temperature, the frequency is fixed next. Then the regenerator length, phase angle between pressure and mass flow oscillations, and finally the amplitude of the mass flow oscillation. The intervals of length, phase angle and mass flow rate were subject to change due to the shift of optimal intervals for them. For instance, the optimal interval of length might not be the same for 30Hz and 50Hz. In order to make the best use of limited computational power and focus on the optimal region, the upper and lower limits of length need to be adjusted, as do the limits for mass flow rate and phase angle.

Although the mass flow rate was being varied during the calculation, the actual variable of interest is the mass-specific cross sectional area (Ag/mc), where Ag is the cross sectional area times the porosity and mc is the mass flow rate at the cold end of the regenerator. As noted by Radebaugh [17], the COP varies with the ratio of Ag/mc, but does not depend separately on Ag or mc. That means when varying the ratio of Ag/mc, either Ag or mc can remain fixed while the other is changed. For the calculations presented below, the cross sectional area is fixed at a value of 1.767E-5 m^2. The thermal conduction through the tube containing the regenerator matrix (TUBECD) is set to be zero throughout this thesis' research. The thickness of the tube varies with individual design. Excluding TUBECD makes the scaling up the

cooling power more applicable to different designs, and TUBECD can also be easily added as an extra loss of cooling power.

3.2. Convergence Requirement

Numerical models require a certain number of iterations and mesh points to converge, as well as a set of boundary conditions and initial guess values for the key parameters of interest. Because a regenerator involves an oscillating flow, an oscillatory steady state condition is desired. That is, one for which all parameters are essentially the same at the beginning and end of a cycle. In this section, the number of time steps per cycle, the number spatial nodes, and the number of iterations (final cycles) necessary for REGEN3.3 to converge on a cyclic steady state are discussed, and the choices of these three variables are proposed and verified. The guess value for the initial temperature profile is based on the choice of a midpoint temperature ratio, which is also discussed in this section.

3.2.1. Final Cycles

The first thing is the number of iterations (cycles) for REGEN3.3 to converge. Final cycle is the corresponding variable name in REGEN3.3 that must be specified by the user. A simple study using COP as the convergence indicator, and carried out by Radebaugh [17], is shown below:

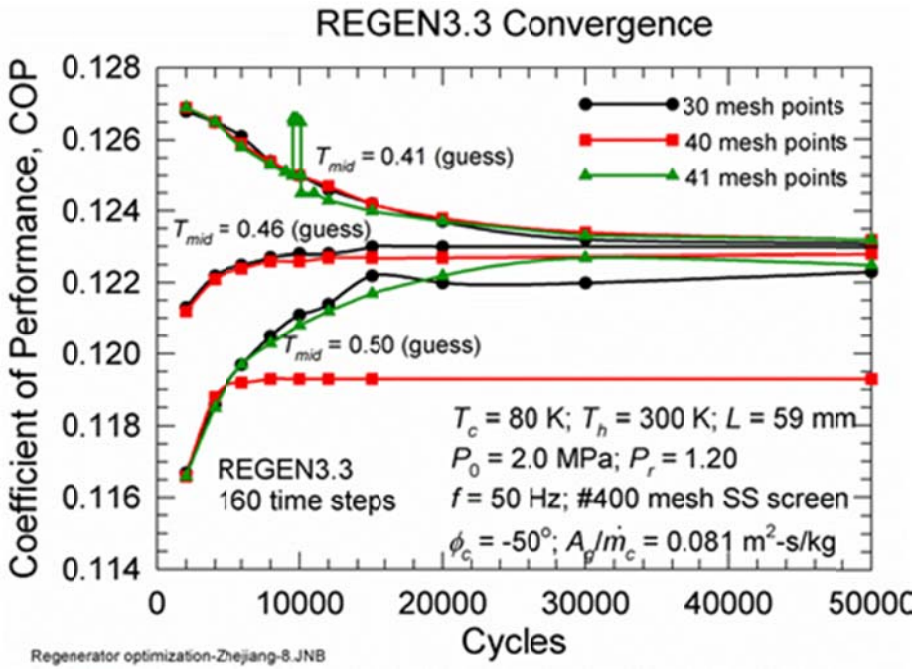

Figure 5 COP vs. Final Cycles

The figure shows how the coefficient of performance varies with the number of final cycles, and with the number of mesh points and mid-point temperature as variable parameters. For the fixed conditions of end temperatures, frequency, charge pressure, and pressure amplitude shown in **Error! Reference source not found.**, the number of cycles required for convergence, is between 20000 and 30000. To be conservative, 30000 final cycles was chosen to be the lower limit final cycles used at 80K, though. Two other variables of interest was also investigated and shown in the figure, the midpoint temperature ratio and the number of mesh points. The number of time steps per cycle is 160. The number of time steps per cycle is discussed in later section.

A similar study was done at 35K cold end temperature. The figure below shows the COP vs number of final cycles used at 35K, 50Hz frequency, 59mm length, -60 degree phase angle, 1.72E-3 kg/s mass flow rate, 0.5 midpoint temperature guess, 0.0001767 m^2 cross sectional area and 41 mesh points with 300 time steps. The COP is close to a stable value of 3.60E-03 with less than 1e-3 variance after 10000 final cycles. This validate the use of 30000 final cycles at the lowest temperature 35K.

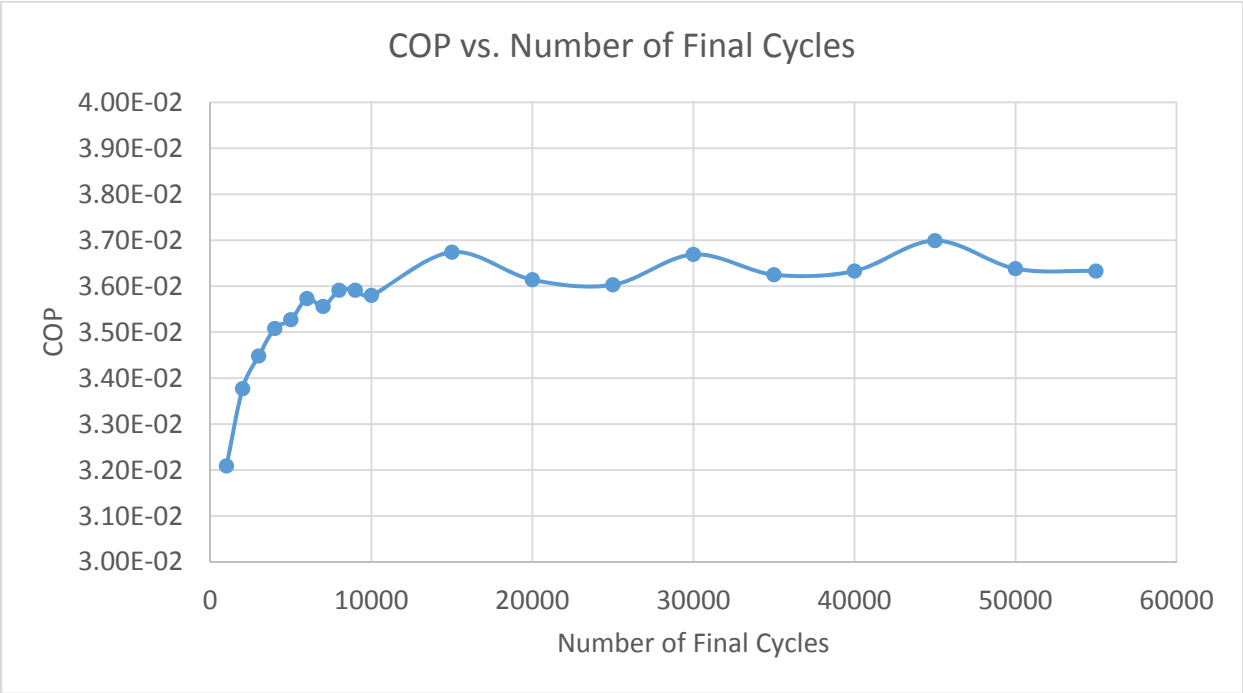

Figure 6 COP vs. Number of Final Cycles at 35K

At frequencies higher than 100Hz at 35K, 60K and 80K, due to convergence issues stated in 4.4.1 and the 'save and load' procedure stated in 4.4.2 in order to maintain 41 mesh points and 300 time steps per cycle resolution, 90000 or even more final cycles were used.

3.2.2. Midpoint Temperature Ratio

The initial guess value for the midpoint temperature ratio, identified in REGEN3.3 as MID_TEMP_RATIO, is the normalized initial temperature at the midpoint of the regenerator. This value is used to set the initial temperature profile for the matrix and gas. The value of the midpoint temperature is the difference between the initial temperature at the midpoint of the regenerator and the inflow temperature, GAS TEMP COLD, at the cold end divided by the difference between the two end point temperatures, GAS TEMP COLD and GAS TEMP HOT. If the value of MID TEMP RATIO is 0.5 then the initial temperature is a linear function of distance along the regenerator. Otherwise a quadratic polynomial is fit through the temperature at the ends and the midpoint to determine the initial temperature as a function of distance along the regenerator.

A good choice for midpoint temperature can speed up the convergence to a periodic state if the final temperature profile is not linear [9]. An output variable named GTPNRM represents the final value of the midpoint temperature after the final number of iterations. This variable can be used to update the midpoint temperature guess for subsequent calculations.

For the three cases with the same number of mesh points shown in Figure 5, the converged COP weakly depended on initial midpoint temperature guess. The midpoint temperature guess of 0.46 seemed to provide the best value for this case, as it converged with fewest cycles, and the final COP for the other two guesses

approached the same value of COP as with the mid-point guess value of 0.46. A careful choice of the initial midpoint temperature is necessary and can be accomplished by sweeping the interval of interest with fewer mesh points and time steps. The resulting value of GTPNRM can then be used for the input midpoint temperature, and more accurate solutions can be achieved.

3.2.3. Number of Mesh Points

As shown in Figure 5, for the same midpoint temperature guess, three different values for the number of mesh points were tested. The values represent the number of axial nodes from the cold to the warm end. There appears to be no significant trend for the three mesh points used. Radebaugh [17] recommends using 41 mesh points as the choice is time efficient, sufficiently accurate and stable. The effect of a higher number of mesh points on stability is discussed in a later section. A further investigation into the influence of the number of mesh points is shown in the figure below.

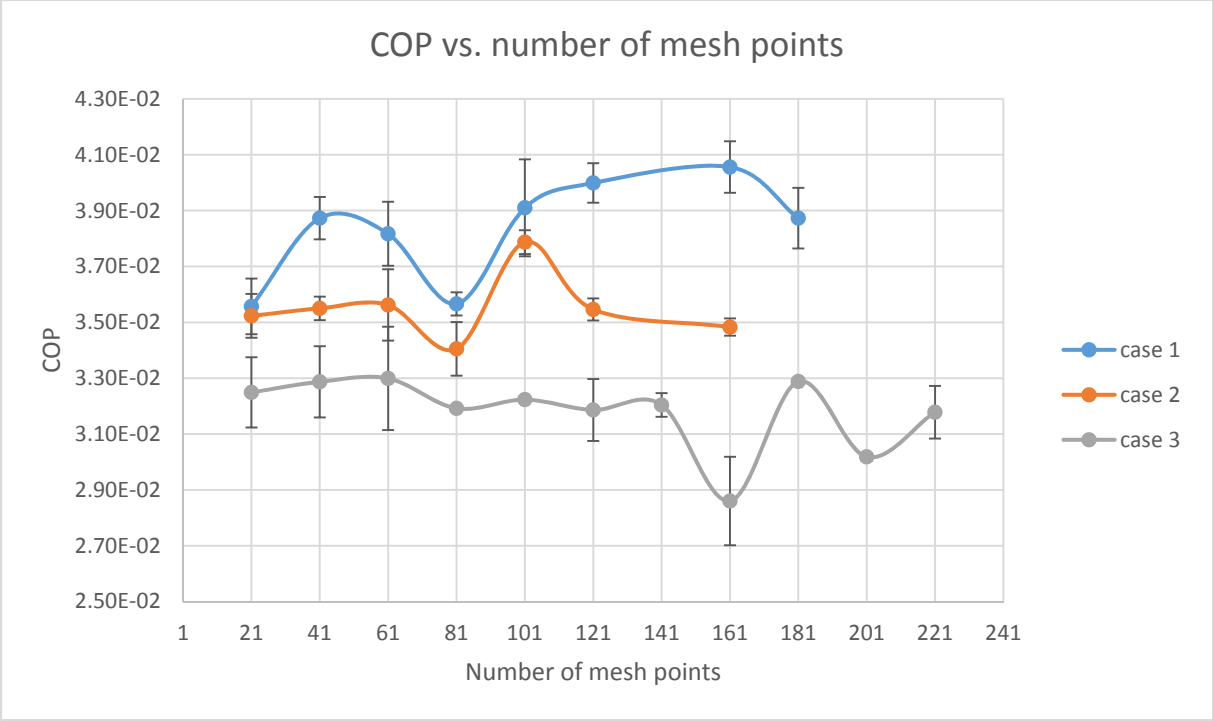

Figure 7 COP vs. Number of Mesh Points for Three Cases

As shown in Figure 7, the number of mesh points was varied between 21 and 221. Using a higher number of mesh points such as 161 or even 221 doesn't offer an obvious advantage for convergence. However, using a higher number of mesh points introduces a possible instability in the final COP for all three cases. A mesh size of 41 points was enough for most calculations because it is both stable and sufficiently accurate.

The number of mesh points was varied for each case. The fixed variables for case 1 are: the number of final cycles is 30000, the cold end temperature is 35K, the frequency is 300Hz, the mass flow rate at cold end is 1.38E-3 kg/s, the length is

0.02m, the phase angle is -20 degree, the number of time steps per cycle is 300, and the midpoint temperature ratio is 0.5. Compared to case 1, the only different variable for case 2 is the mass flow rate, which is 1.48E-3 kg/s. Compared to case 1, the only different variable for case 3 is the frequency, which is 30Hz.

3.2.4. Number of Time Steps per Cycle

Another discretization variable that influences the numerical results is the number of time steps per cycle, identified by REGEN3.3 as num_steps_cyc. The period of a cycle divided by the number of time steps gives the actual time step. Since the choice of the number of time steps is important for convergence, a case study was carried out. The result is shown in the figure below:

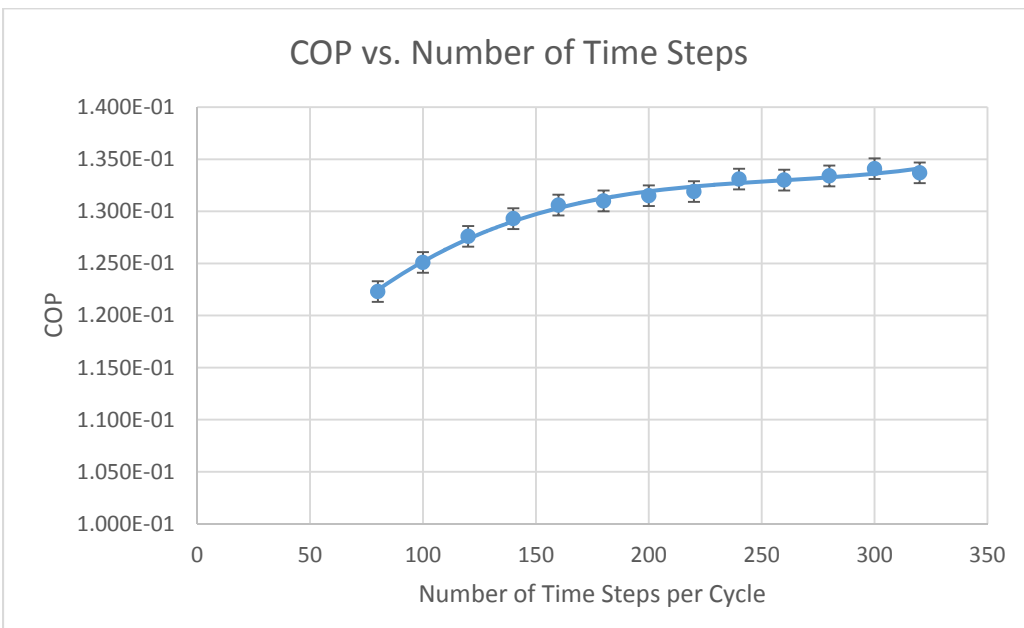

Figure 8 COP vs. Number of Time Steps

As the other variables were all fixed, the number_steps_cyc was varied between 80 and 320 for the three cases. As shown in figure 6 the COP appears to approach a limiting value by 250 time steps. Although no significant difference occurs between 240 steps and 320 steps, the relative difference between using 300 steps and 140 steps is about 4%, and the relative difference between using 300 steps and 80 steps is almost 10%. Therefore 300 time steps is chosen for better convergence and accuracy.

3.2.5. Variables Choice Summary

To summarize, the final choice of midpoint temperature ratio was determined on a case to case basis; 41 mesh points were used for most cases, while 81 mesh points were used at 80K; 300 time steps were used for most cases; at least 30000 final cycles were used for every case.

3.3. Implementation of Calculation

As stated in chapter 1 above, REGEN3.3 does not include a mapping or optimization procedure. Therefore a stand-alone shell-script was created to run REGEN3.3 iteratively. The software package REGEN3.3 provides a graphic user interface that calls rg33.exe, which in turn obtains input data from a file named data.dat, writes output data files named rgpr plus a file number, and writes user specified information to a graphic output file.

A shell program is written in C to repeatedly run this procedure in order to carry out the parametric studies. The code can be found in Appendix B. Additionally a large amount of input data files were generated by a separate sub-program written in Matlab, the code of which is also included in Appendix B. Finally, a third small program, written in Python, is used to extrapolate the desired variables from the massive output data files.

The general procedure for the parametric studies is as follows: First, input the desired variable range of interest into Matlab, and generate a set of data files. Second, run the shell program, defining the first and final data file numbers, and allow the program to run REGEN3.3 continuously until it finishes calculating all the cases specified.

3.3.1. The Innermost Loop

The following paragraphs provide an example of the optimization procedure that follows the steps given above, and illustrates how the set of optimal variables are found for the conditions of a cold end temperature of 80K, and operating frequency of 40Hz.

As shown in **Error! Reference source not found.**, the outer four loops are fixed and the optimization procedure scans over all the possible combinations identified in the three innermost loops. The regenerator length is initially set at a value of 40mm, and REGEN3.3 is run for phase angles from -20 degree to -55 degree, and for mass

flow rates from 3.8E-4 kg/s to 1.36E-4 kg/s. The Matlab code is used to generate the associated data files with these various sets of input values and the shell program is used to run REGEN3.3 with the defined inputs. The output files are then read and graphed as shown in Figure 9.

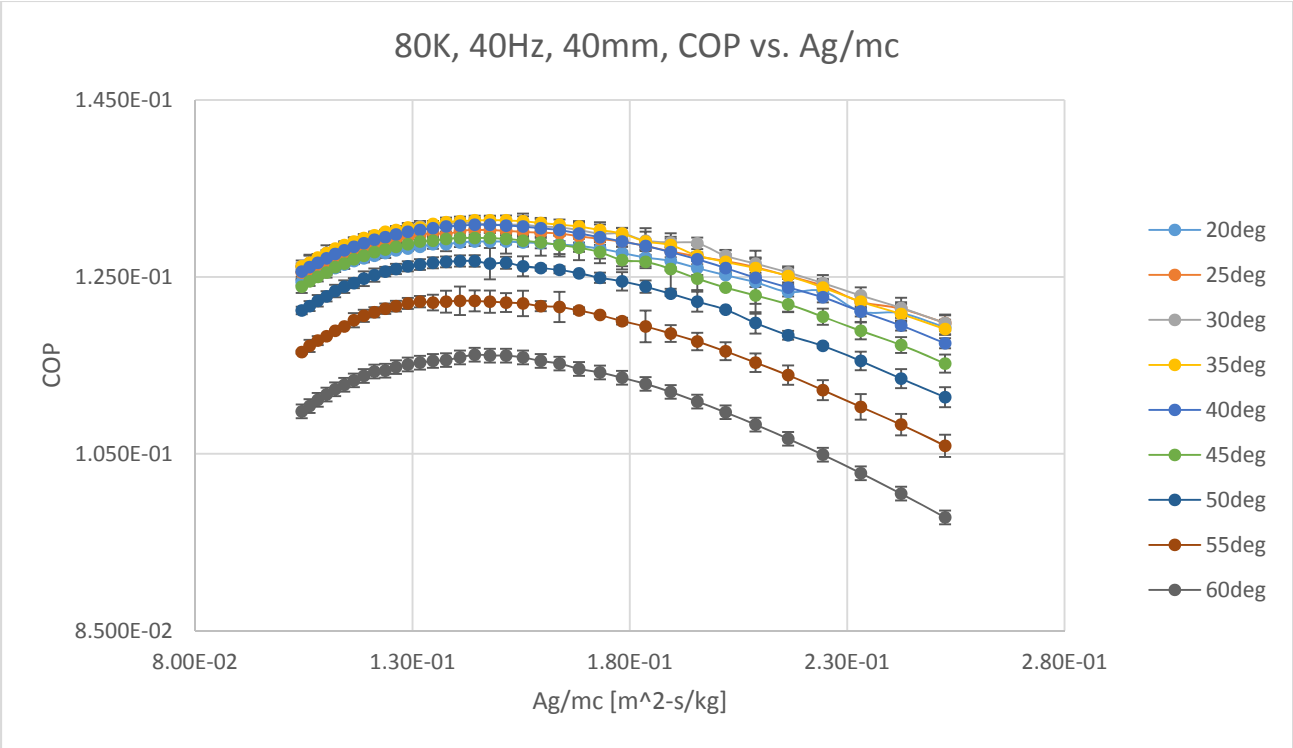

Figure 9 40mm, 40Hz, 80K, COP vs. Ag/mc

The above figure is a summary of calculations performed at 40mm, 40Hz and with a cold end temperature equal to 80K. The COP at each specific cross sectional area (Ag/mc) is plotted, and different curves represent different phase angles. The optimal COP and corresponding set of variables for these length can be identified by reading the plot. The optimal COP is 1.332E-1, the optimal phase angle is -35 degree, and the optimal value of Ag/mc is 1.64E-1 m^2-s/kg.

The following graph shows another way to represent the results from this loop of calculation,

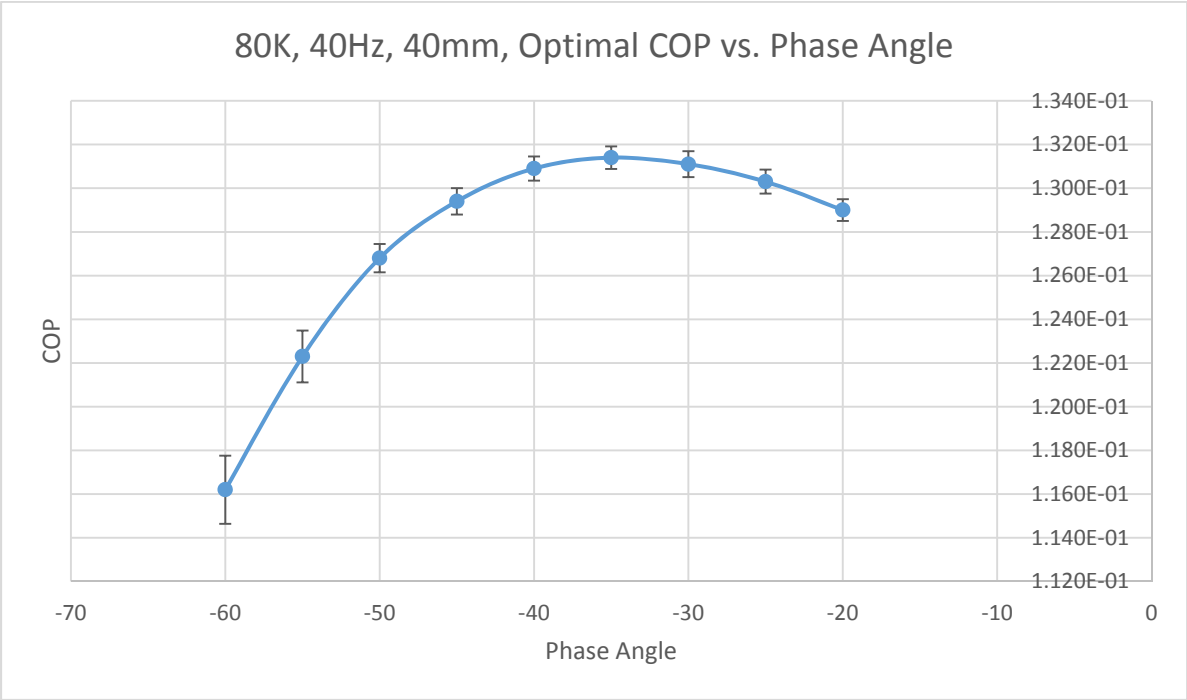

Figure 10 40mm, 40Hz, 80K, Optimal COP vs. Phase Angle

The figure shows the optimal COP at each phase angle varies with phase angle. The optimal phase angle is -35 degree and the optimal COP is 0.1318 for this length. Also tracing back to the previous figure, the optimal value of Ag/mc can be identified as well.

3.3.2. Mapping the Optimal Variables

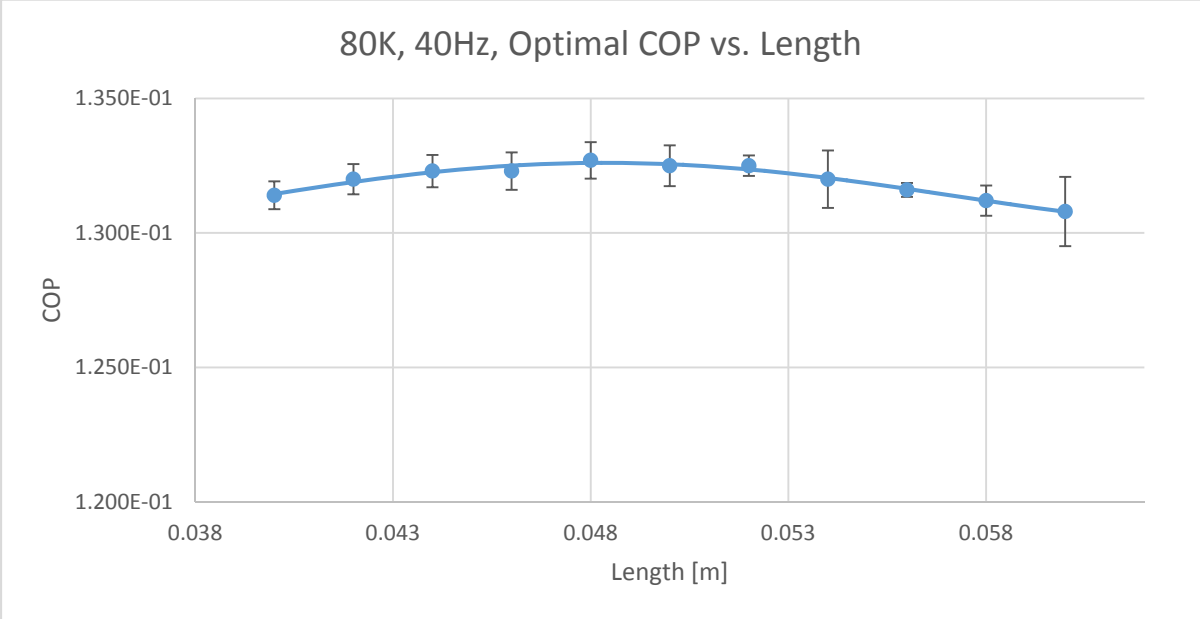

Figure 11 80K, 40Hz, 40mm, Optimal COP vs. Length

The next step involves changing the regenerator length (RG_LENTH) and repeating the above calculations, graphing the results so that the optimal COP and corresponding set of variables can be identified at the new length. As shown in Figure 11, a graph of optimal COP values at various lengths at 40Hz is plotted. The optimal length can be easily identified from the graph.

Often it takes at least ten or more different mass flux values to find the optimal COP at a phase angle, and three to five phase angles in order to locate the optimal variables at a given length. A consistent step size of 5 degrees has been used in exploring the phase angle dependence. At least three different lengths are used to

find the optimal length at a given frequency. Repeating this procedure at several different frequencies, enables one to determine the optimal variables for a regenerator operating at a fixed cold end temperature. This overall procedure has been carried out for three different cold end temperatures, 35 K, 60 K and 80 K

4. Results and Discussion

4.1. Uncertainty Analysis

4.1.1. Sources of Uncertainty for CFD Modeling

In physical science, whenever a physical process is quantified, uncertainties associated with modeling and computer simulation can arise. Types of numerical error that are of concern in the numerical solution of partial differential equations are: spatial discretization error in finite element and finite difference methods, temporal discretization error in time dependent simulations, and error due to discrete representation of strongly nonlinear features. There are, in general, six phases of modeling and simulation. The phases are 1) conceptual modeling of the physical system, 2) mathematical modeling of the conceptual model, 3) discretization and algorithm selection for the mathematical model, 4) computer programming of the discrete model, 5) numerical solution of the computer program model, and 6) representation of the numerical solution [18]. Each of these phases can introduce numerical uncertainties.

Uncertainty with conceptual modeling comes from the uncertainty of physical boundary conditions, ideal assumptions made that might not be exactly the same in physical world, or that neglect some physical process that might affect the solution. These uncertainties are not errors but rather often compromises that have to be made when describing a physical process. For example, when modeling flow

through an open pipe, the inlet and outlet temperature are often assumed to be uniformly distributed over the cross sectional area. These compromises are often insignificant compared to other features of interest.

There are many ways to mathematically model a physical system, for instance, the choice of using a one-dimensional spatial model instead of a three-dimensional one for REGEN3.3. Again, these are often necessary due to the finite computational power and coding time. But the choice requires careful consideration. For most regenerators, a three-dimensional or even two-dimensional model is not necessary because the effects of flow in one direction are dominant.

Discretization and algorithm selection involves choosing an order of accuracy and solution algorithm, as well as the discretization of spatial meshes. These, along with the computer programming of discretization, affect the error introduced by discretization and the numerical solving algorithm. Also the round-off error and the finite number of digits used contribute to the uncertainty of numerical models.

4.1.2. Possible Sources of Uncertainties in REGEN3.3

The conceptual modeling of REGEN3.3, as discussed in former sections, is based on adding extra frictional and heat transfer terms to the conservation equations for one-dimensional open pipe flow. The correlations for the friction factor and heat transfer coefficients are based on curve fitting of data for steady state flow through porous medium. Therefore, the discrepancy from actual physical process could

result from two sources: first is the possible insufficiency due to the simplification of a three dimensional system using one dimensional model; second is the uncertain accuracy of steady state correlations when applied to oscillating conditions and also the uncertainty in the correlations themselves.

Most regenerators operate at conditions where the influence of the oscillatory nature of the flow on pressure drop and heat transfer is not likely to be important even at very low operating temperatures [19], therefore the REGEN3.3 model is reasonable in regards to friction factor and heat transfer coefficients.

The mathematical modeling of the conceptual model is based on established equations describing energy, momentum and mass conservations, and a Newtonian iteration method is used to reach the final solution. The iteration termination criteria is specified by user. The variable name is EPS_NEWTON, which represents the error tolerance for each time step. For the discretization of equations, first order accuracy is used within the scope of this thesis. Higher order accuracy is optional in REGEN3.3, but it takes much more computational time. Using a lower order accuracy could introduce some round-off uncertainties. Also, the round-off error due to a finite number of digits used might contribute to the uncertainty as well.

4.1.3. Quantization of Uncertainty in REGEN3.3

There are two variables in the output from REGEN3.3 that can be used as indicators of uncertainty or error of the solution, EHTDIF and ENGBAL. EHTDIF is the

difference between the maximum and minimum over the mesh of the integral average of the sum of the enthalpy flux and matrix heat flux. When a periodic flow is achieved this value should be small in comparison to the value of EHTFLX. ENGBAL is an estimate of the accuracy of energy conservation in the gas. Given a perfectly accurate method the value of ENGBAL would vanish. Comparing its value to the enthalpy flux, ENTFLX, gives some indication of the discretization error in the computation. Both terms indicate whether the solution is a good one or not. If EHTDIF is 5% to 10% of EHTFLX and ENGBAL is 5% to 10% of ENTFLX, the solution is considered to have converged and valid. On the other hand, if EHTDIF is 20% to 30% of EHTFLX or ENGBAL is 20% to 30% of ENTFLX, the solution is not a good one.

Referring to Equation 11 and 14, ENTFLX is part of the RGLOSS term, and the combination of ENTFLX and HTFLUX is the same as EHTFLX. Both error terms therefore provide equivalent information. In this thesis, ENGBAL is used to characterize the uncertainty of NTCADJ (net adjusted cooling power).

4.1.4. Uncertainty Calculation

To propagate the uncertainties in each of the measured variables into the value of the calculated uncertainty, the method in use is described in NIST Technical Note 1297 [20]. Assuming the individual measurements are uncorrelated and random, the uncertainty in the calculated quantity can be determined as:

$$U_y = \sqrt{\sum_i (\frac{\partial y}{\partial x_i})^2 U_{x_i}^2} \quad (16)$$

where U represents the uncertainty of the variable. U_x is the measured variables uncertainty and U_y is the propagated calculated variable uncertainty.

ENGBAL is considered as the uncertainty of NTCADJ, therefore by using the uncertainty propagation function in Engineering Equation Solver, the uncertainty of COP is solved. There is an obvious limit to this method of representing the uncertainty. Rather than an accurate uncertainty value, ENGBAL was coded only as an indicator of uncertainty.

4.2. Optimal COP and Correspondent Variables versus Frequency

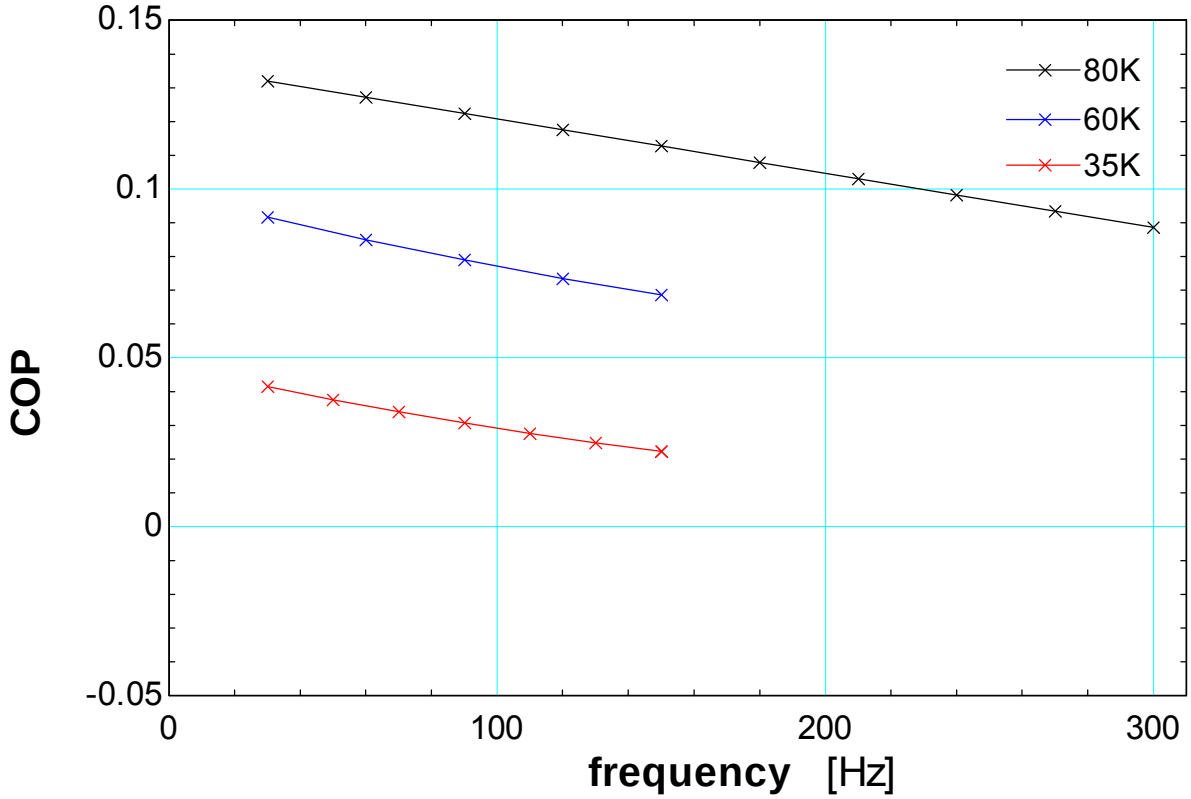

Figure 12 Optimal COP vs. Frequency at 80K, 60K and 35K predicting model

The figure above shows the correlation between optimal COP and the corresponding frequency at 80K, 60K and 35K based on curve fitting of actual data. This summarizes the optimization mapping. At a certain temperature, the optimal COP is almost a linear function of frequency, and decreases as the frequency increases. Operating a cryocooler at higher frequency could make the system more compact since the size of either regenerators or compressors is inversely proportional to frequency [21].

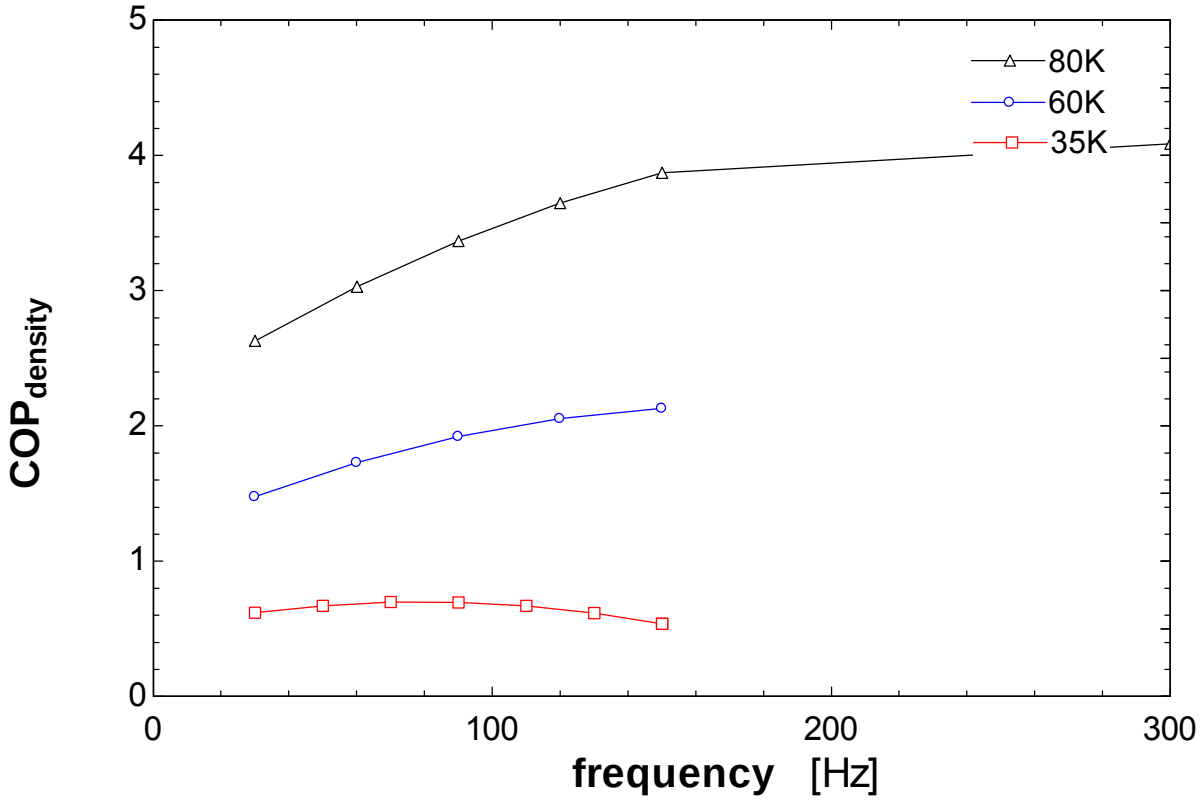

Figure 13 COP/Length vs. Frequency

The figure above shows the COP per unit Length versus the frequency. The figure could also be interpreted as COP per unit volume versus the frequency since the cross sectional area of the regenerator is fixed. Therefore if we borrow the concept of density, COP per unit Length can be named COP density. For a single stage regenerative cryocooler, when the cold end temperature is at 80K and 60 K, the COP density increases as frequency increases. That could be one more incentive to design cryocoolers operating at higher frequency.

However, for a single stage cryocooler with a cold end temperature of 35 K, there seems to be an optimal operating frequency from the view of COP density. The

probable optimal frequency is between 60Hz and 120Hz, though the COP density is not significantly different from 30Hz to 180Hz. One could also argue that for a single stage cryocooler with cold end temperature 35K, the COP density decreases slightly as frequency increases. Therefore it might not be ideal to operate a 35K single stage cryocooler at higher frequency.

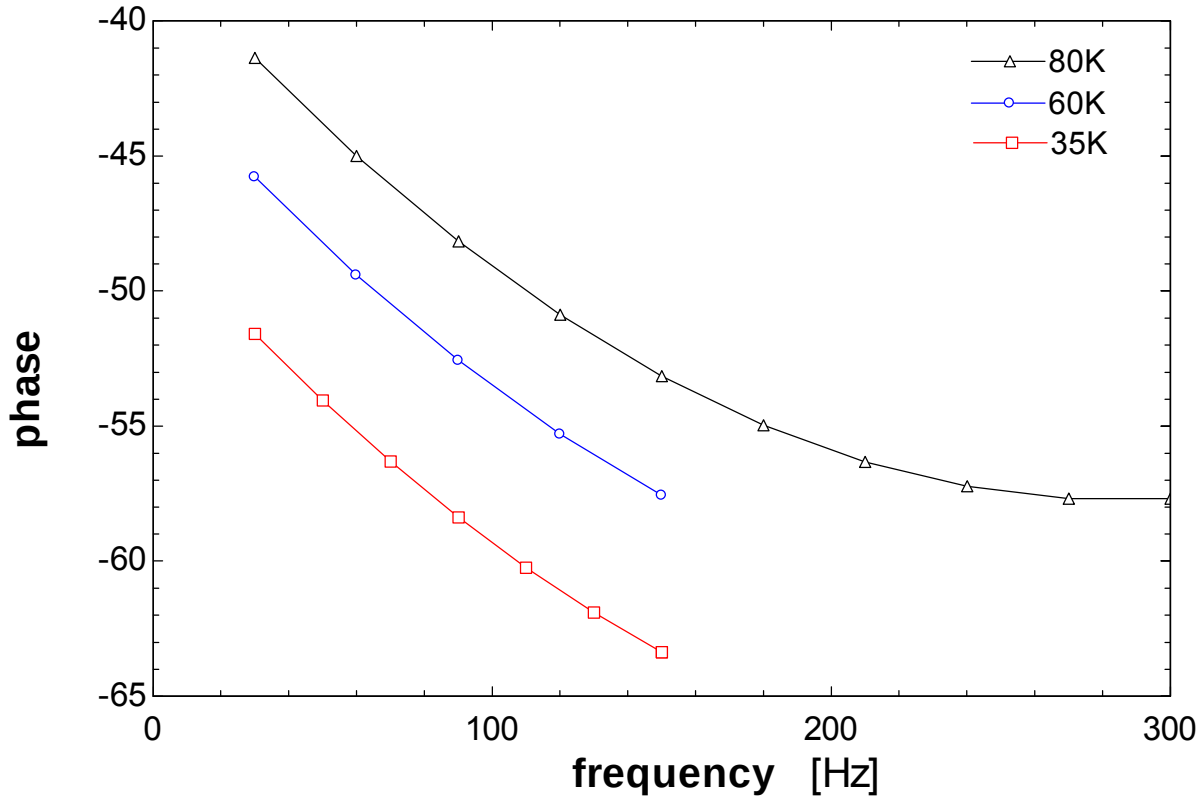

Figure 14 Optimal Phase Angle vs. Frequency at 80K, 60K and 35K

Figure 14 shows the phase angle that corresponds to the optimal COP at various frequencies. At higher frequencies, a greater phase shift is necessary for the optimal COP.

The discrepancy between the curve fitting and the actual calculated data is due to the step size used when changing phase angles. As explained before, the step size used when changing phase angle is 5 degree. That might not be small enough to accurately identify the optimal phase angle. But since the phase shift is not often controlled at a high precision, it is considered to be a minor issue that optimal phase angles are not precisely determined.

The following two figures show the correlations between the optimal specific cross sectional area, and the optimal length vs frequency at each temperature. They are both strong functions of frequency, and decrease as frequency increases.

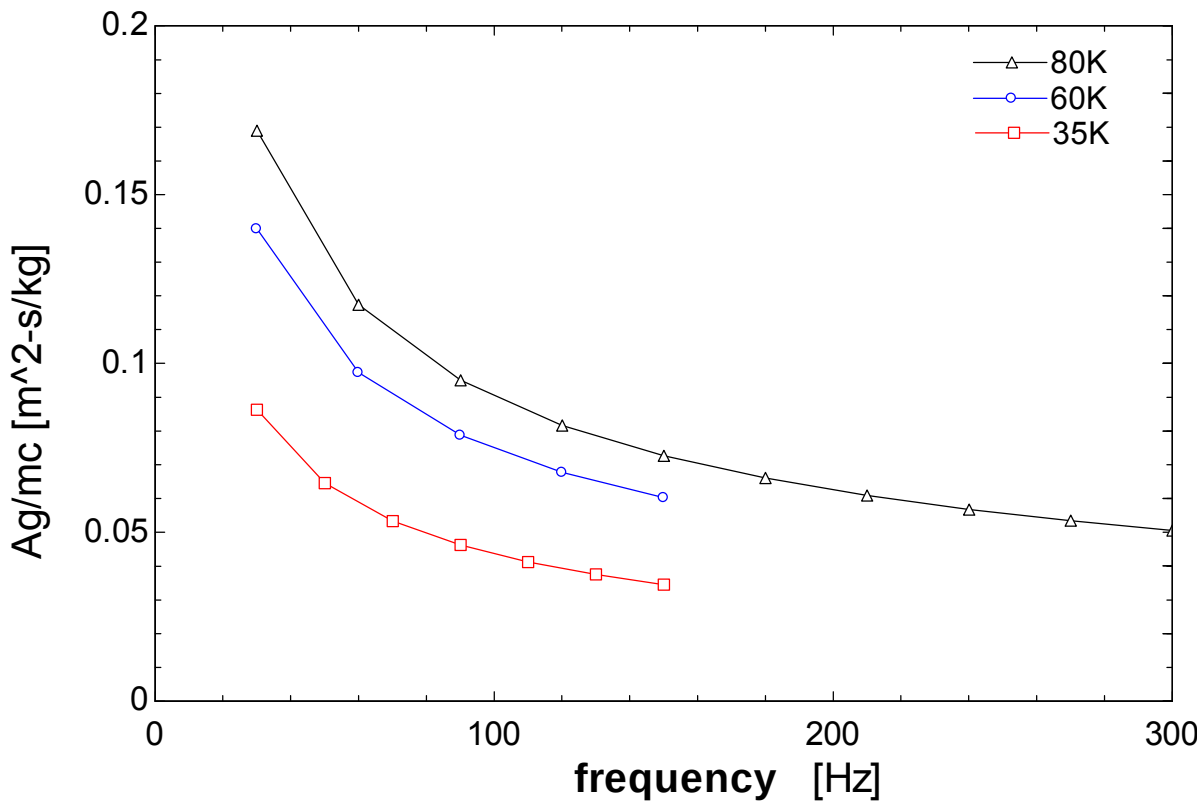

Figure 15 Optimal Ag/mc vs. Frequency at 80K, 60K and 35K

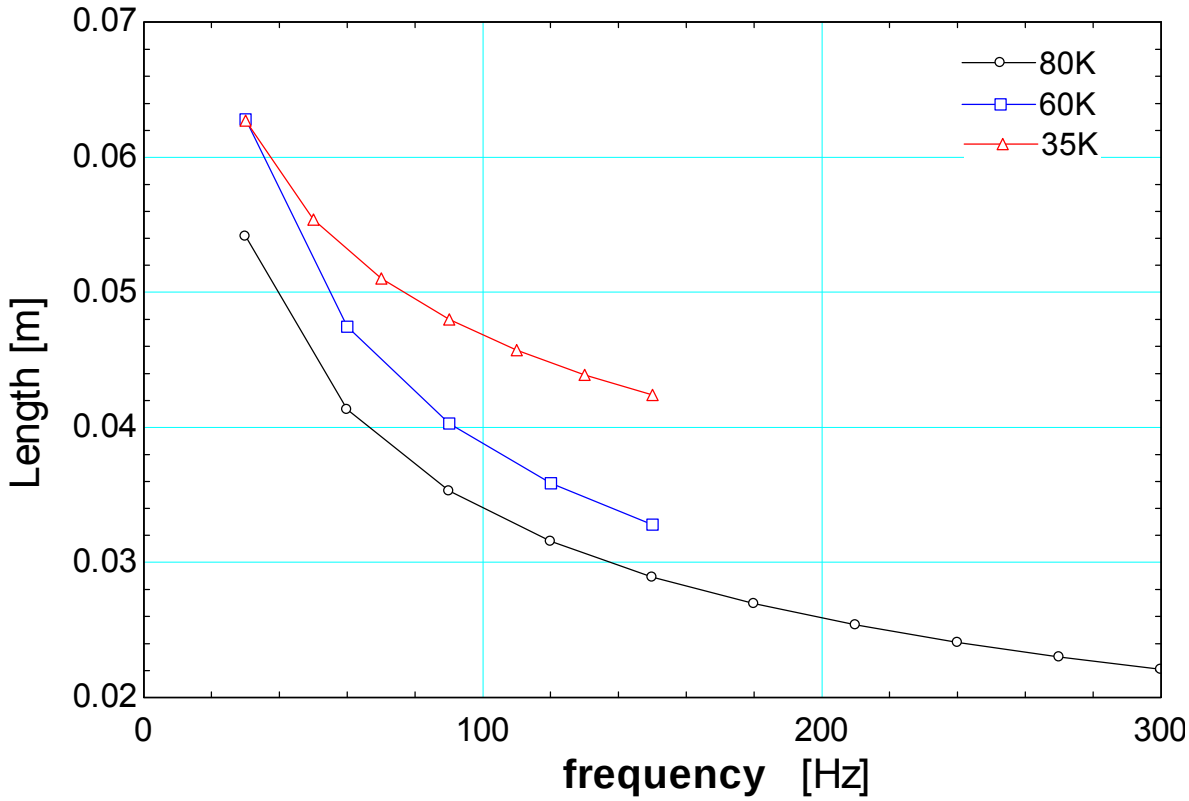

Figure 16 Optimal Length vs. Frequency at 80K, 60K and 35K

The fitted function and related EES code can be found in Appendix A and B respectively.

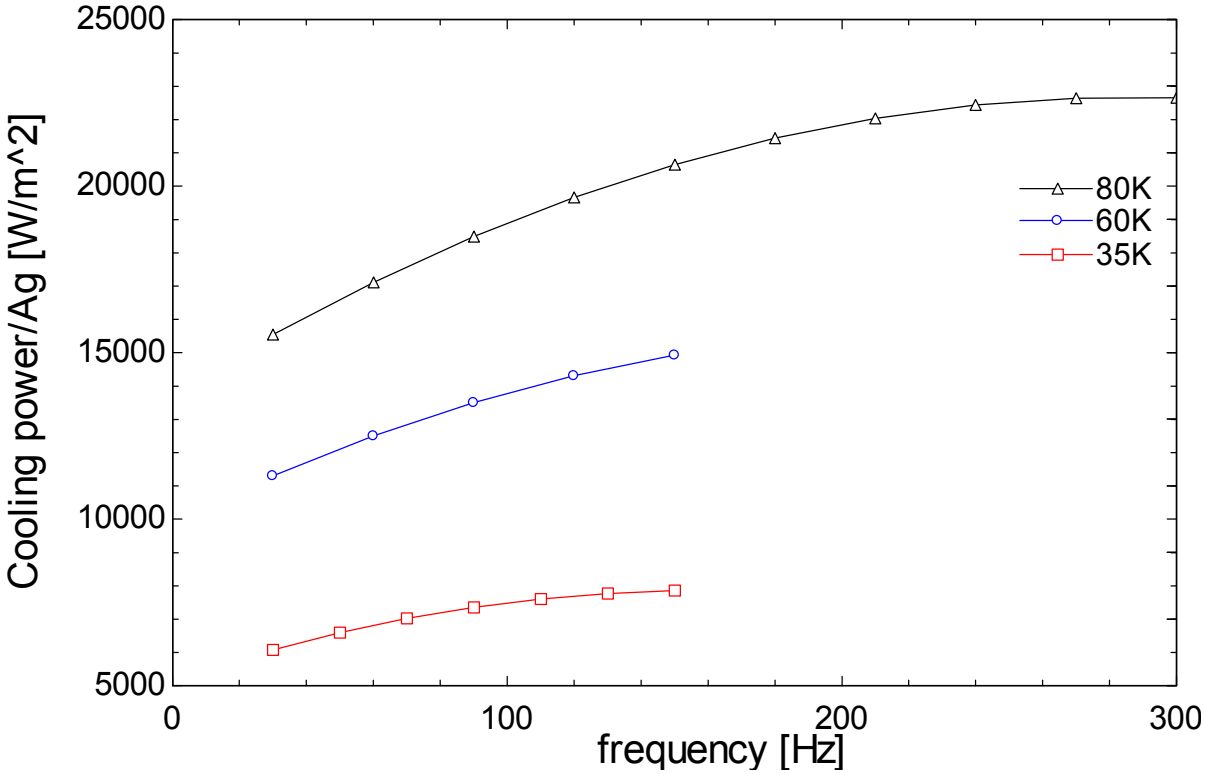

Figure 17 cooling power per area at optimal performance vs. frequency at various temperatures

The figure above shows the correlation between cooling power at optimal performance divided by the cross sectional area versus frequency at 80K, 60K and 35K. Cooling power per unit area as a function of Tc and f can be found in Appendix A.

4.3. Optimal COP Shell

A detailed look at Figure 9, Figure 10, and Figure 11 revealed that there is a broad region of Ag/mc, phase angle and lengths where COP downgrading is relatively insignificant. The following figures shows range of variables with2% and 5% off the

optimal COP at 80K, 40Hz.

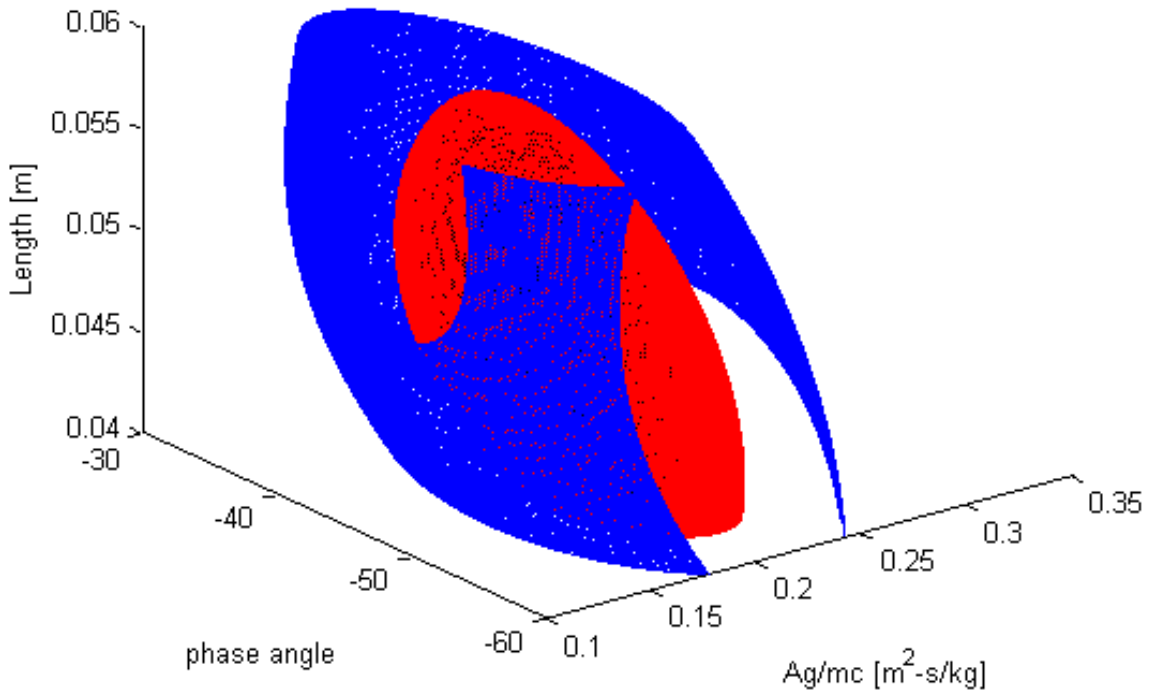

Figure 18 2% and 5% off optimal COP shell at 40Hz, 80K

A 3-D plot was made to present this broad region of variable set with small performance downgrading. The blue shell represents variable sets 5% off the optimal COP, the red shell represents variable sets 2% off the optimal COP. The optimal COP is evaluated at 80K and 40Hz. The figure shows that region for conditions of 40Hz, and 80K cold end temperature. A point in the 3-D plot represents a specific combination of length, Ag/mc and phase angle that has a certain value of COP. The red shell shows the combinations where the values of the COP are only 2%

off from the optimum value at this frequency. In other words, if the three variables are constrained to be located on the shell or within it, the COP corresponding to that combination would be at least 98% of the optimal COP at 40Hz, 80K. The same conclusion can be stated for the blue shell, except that the COP is 5% or less off the optimum when the three variables fall on or within the blue shell. Essentially Figure 18 is a 3-D contour plot, with the contour represented by a colored shell instead of connected lines as in a 2-D contour.

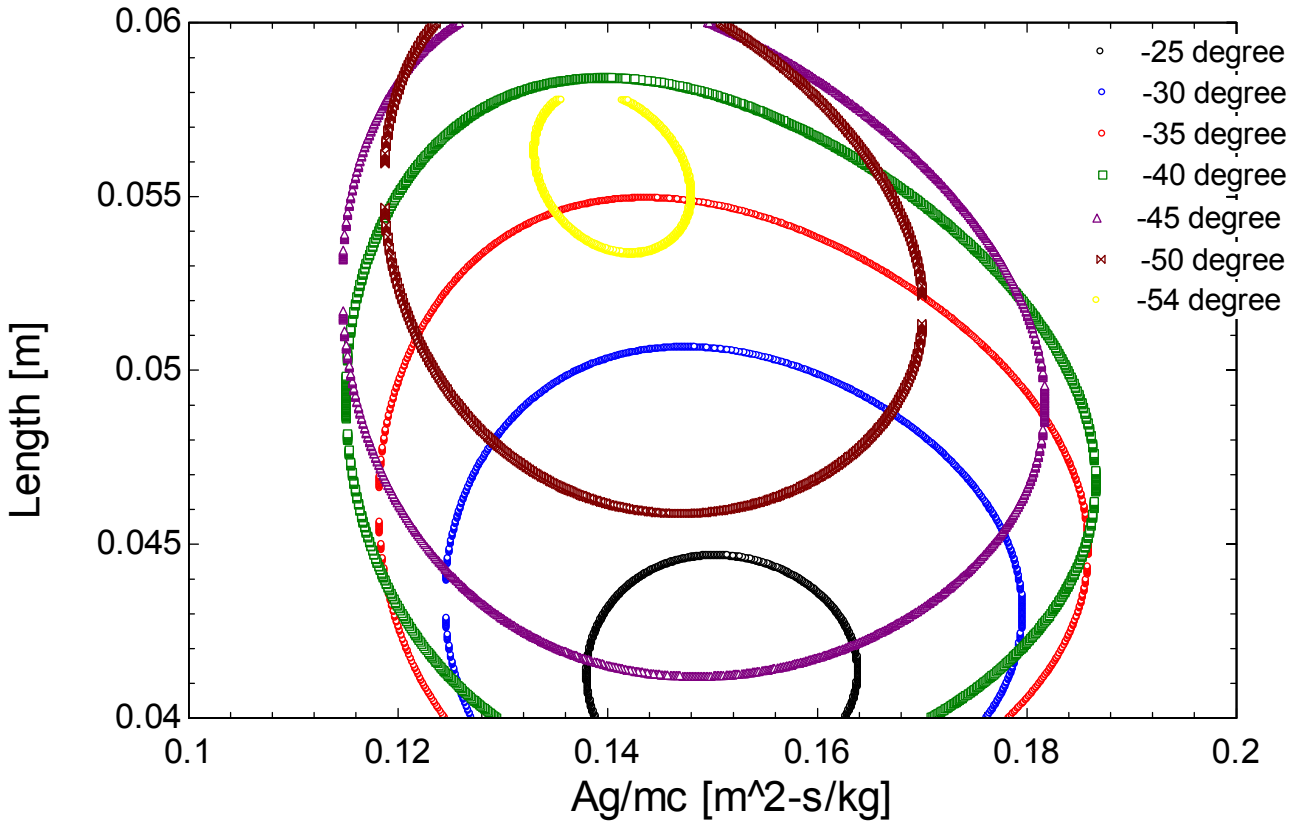

Figure 19 COP shell 2% off optimum contour view, Length vs. Ag/mc at various phase angles

Figure 19 shows a contour view of the COP shell associated with 2% off the optimal COP (the red shell in Figure 18) for various phase angles of Length versus Ag/mc. Each colored ring represents a different phase angle. And each ring shows the range of Length and Ag/mc with 2% or less off the optimal at a constant phase angle. The flexibility of Ag/mc and length are at most when phase angle is -40degree, and gradually become limited at higher and lower phase angles. The upper phase angle limit of getting 2% or less the optimal COP is approximately -25 degree, and the lower limit is approximately -54 degree.

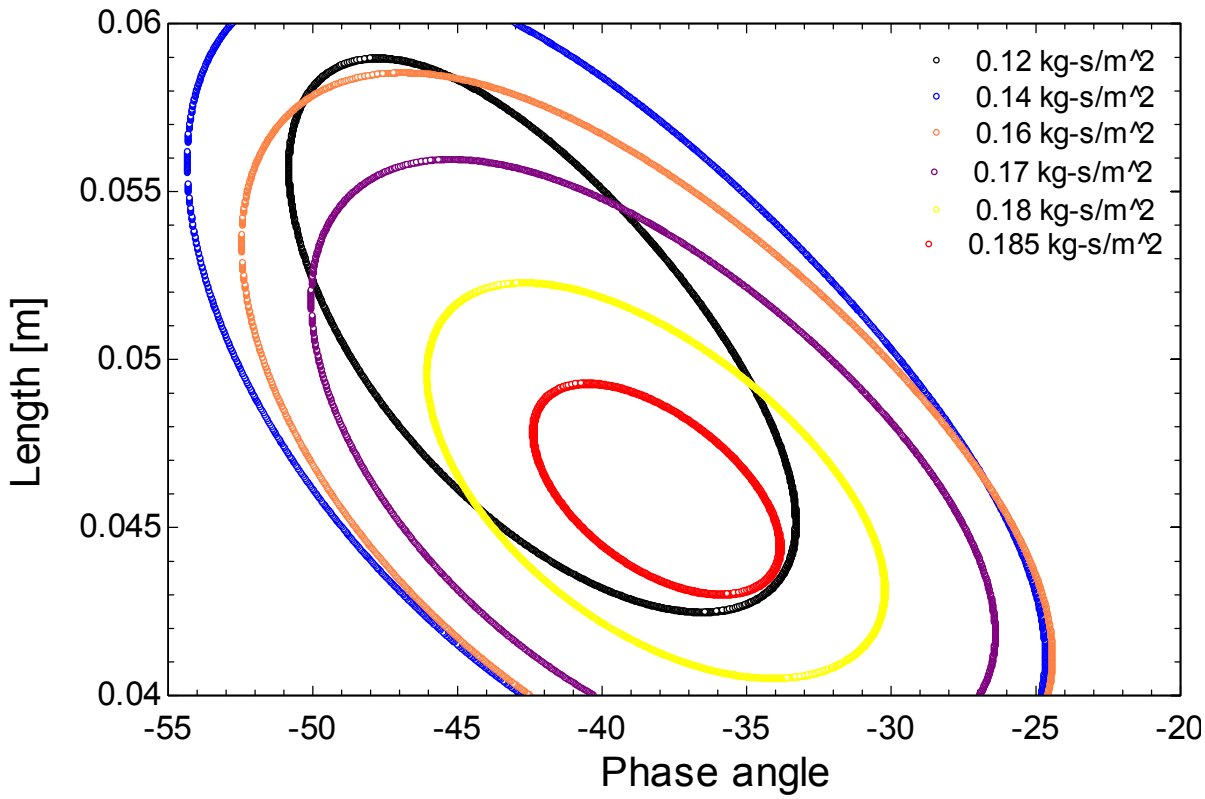

Figure 20 COP shell 2% off optimum contour view, Length vs. Phase angle for various Ag/mc

Figure 20 shows a contour view of the COP shell associated with 2% off the optimal COP (the red shell in Figure 18) for various Ag/mc values of Length versus Phase angle. Each colored ring represents a different value of Ag/mc. And each ring shows the range of Length and phase angle with 2% or less off the optimal at a constant Ag/mc. The flexibility of Length and phase angle are at most when Ag/mc is 0.14 kg-s/m^2, and gradually become limited at higher and lower Ag/mc values. The upper Ag/mc limit of getting 2% or less the optimal COP is approximately 0.185 kg-s/m^2, and the lower limit is approximately 0.12 kg-s/m^2.

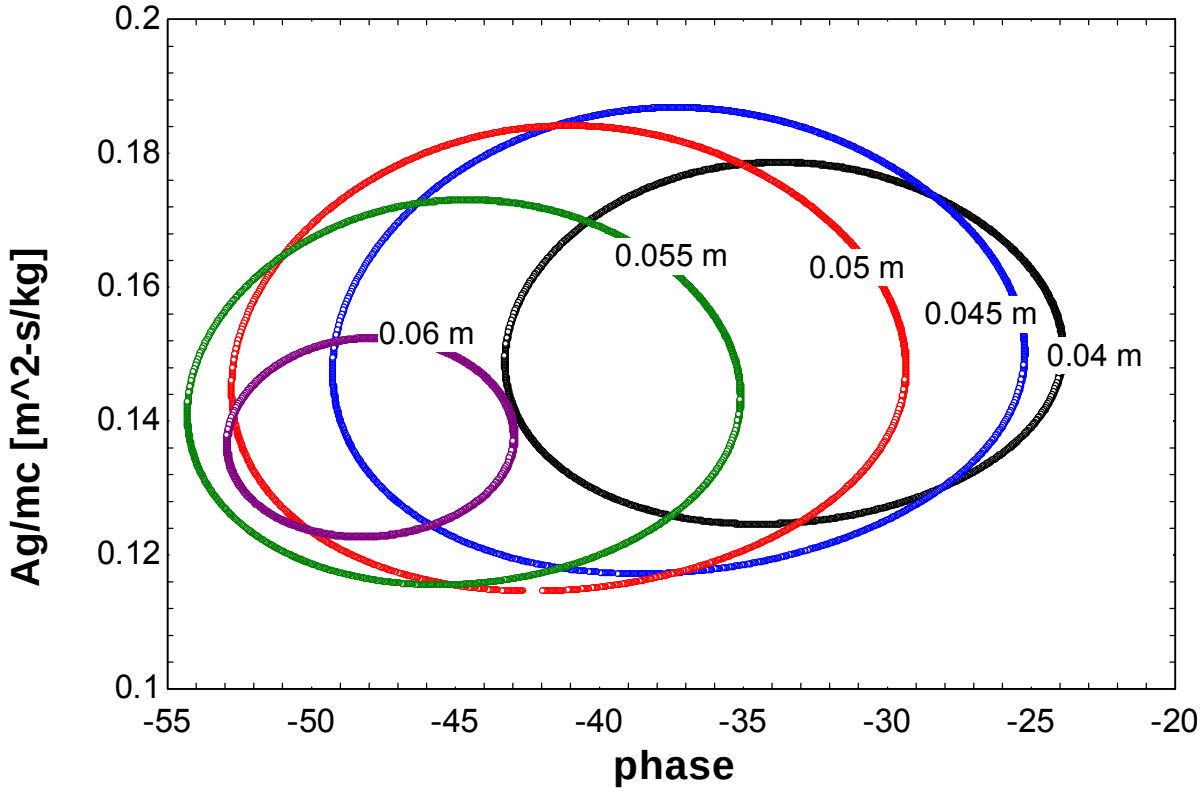

Figure 21 COP shell 2% off optimum contour view, Ag/mc vs phase angle at various lengths

Figure 21 shows a contour view of the COP shell associated with 2% off the optimal COP (the red shell in Figure 18) for various lengths of Ag/mc versus Phase angle. Each colored ring represents a different length. And each ring shows the range of Ag/mc and phase angle with 2% or less off the optimal at a constant Ag/mc. The flexibility of Ag/mc and phase angle are at most between 0.045 m and 0.050 m, and gradually become limited at higher and lower lengths. The upper and lower limit of length calculated is 0.060 m and 0.040 m respectively.

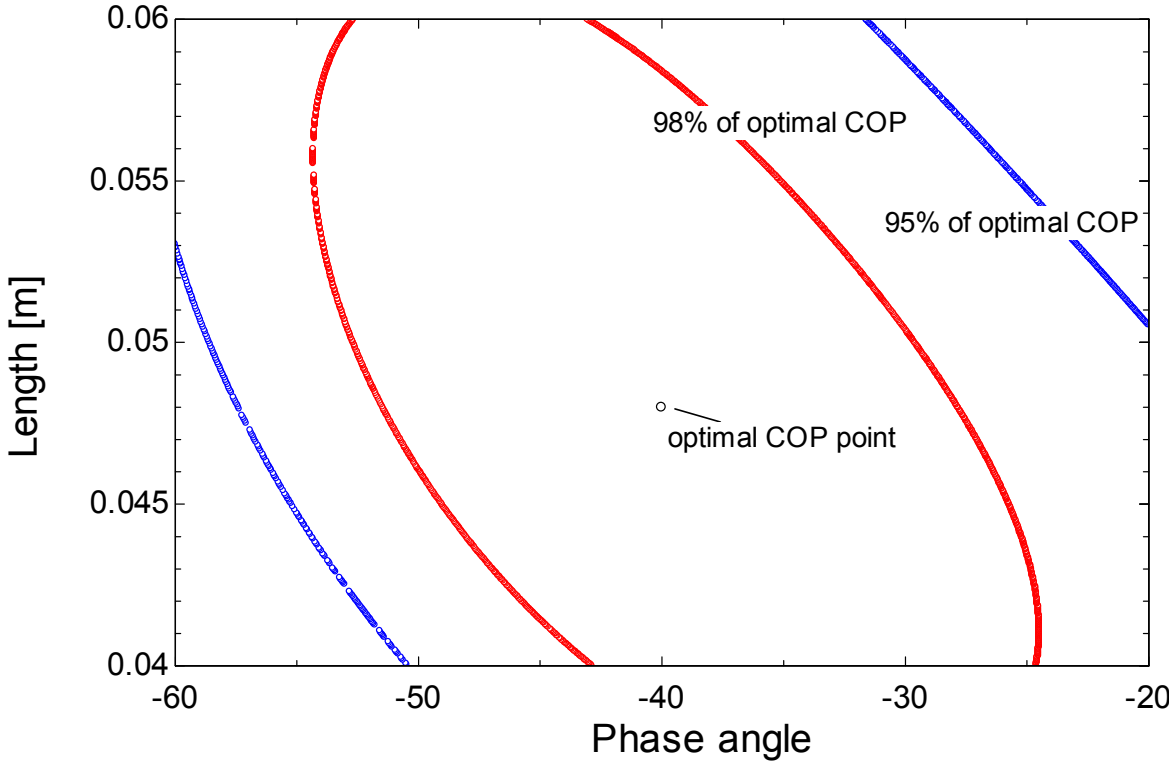

Figure 22 98% and 95% of optimal COP, Phase angle vs. Length for optimal Ag/mc value

Figure 22 shows the cross sectional view of the shell in Figure 18 at optimal Ag/mc value, 0.141 kg-s/m^2. The blue and red ring is the boundary of phase angle and length with the performance 5% and 2% off the optimal COP respectively. The discontinuity in the two rings is due to limited range of phase angle and length being calculated.

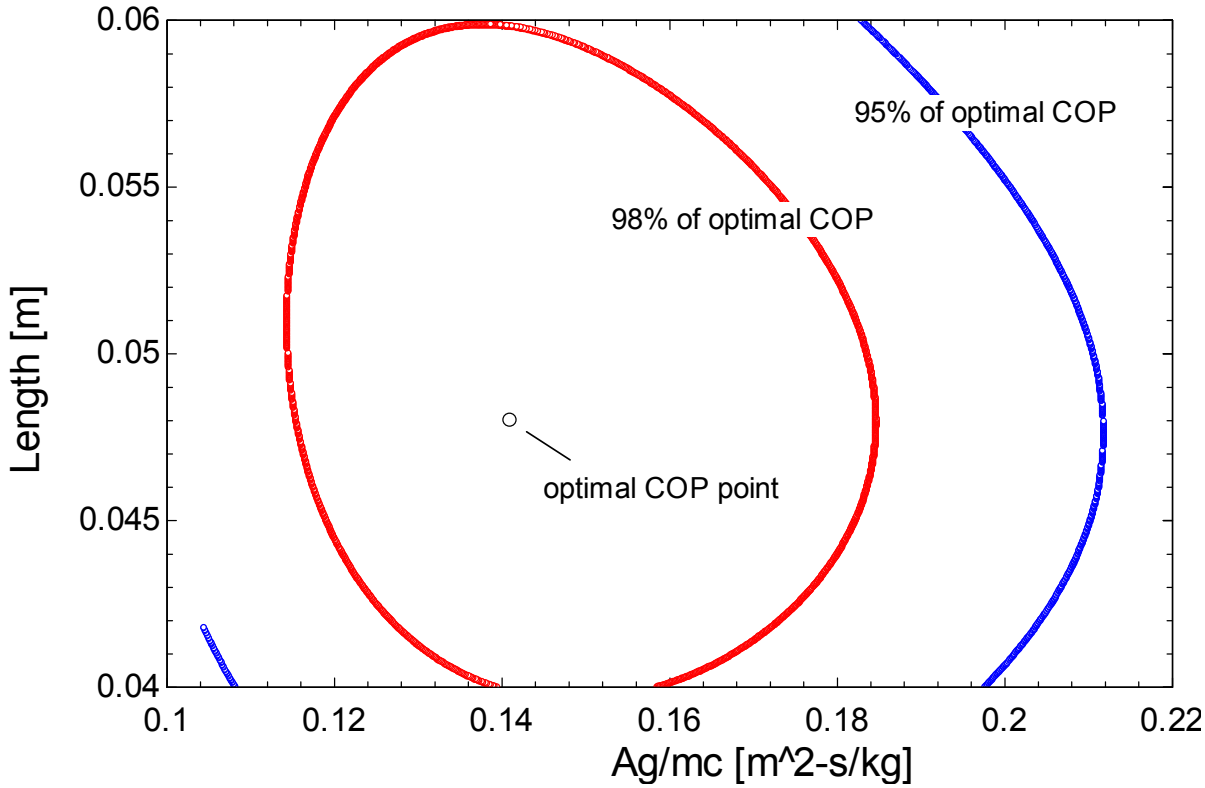

Figure 23 98% and 95% of optimal COP, Length vs. Ag/mc for optimal phase angle

Figure 23Figure 22 shows the cross sectional view of the shell in Figure 18 at optimal phase angle, -40 degree. The blue and red ring is the boundary of Ag/mc and length with the performance 5% and 2% off the optimal COP respectively. The

discontinuity in the two rings is due to limited range of Ag/mc and length being calculated.

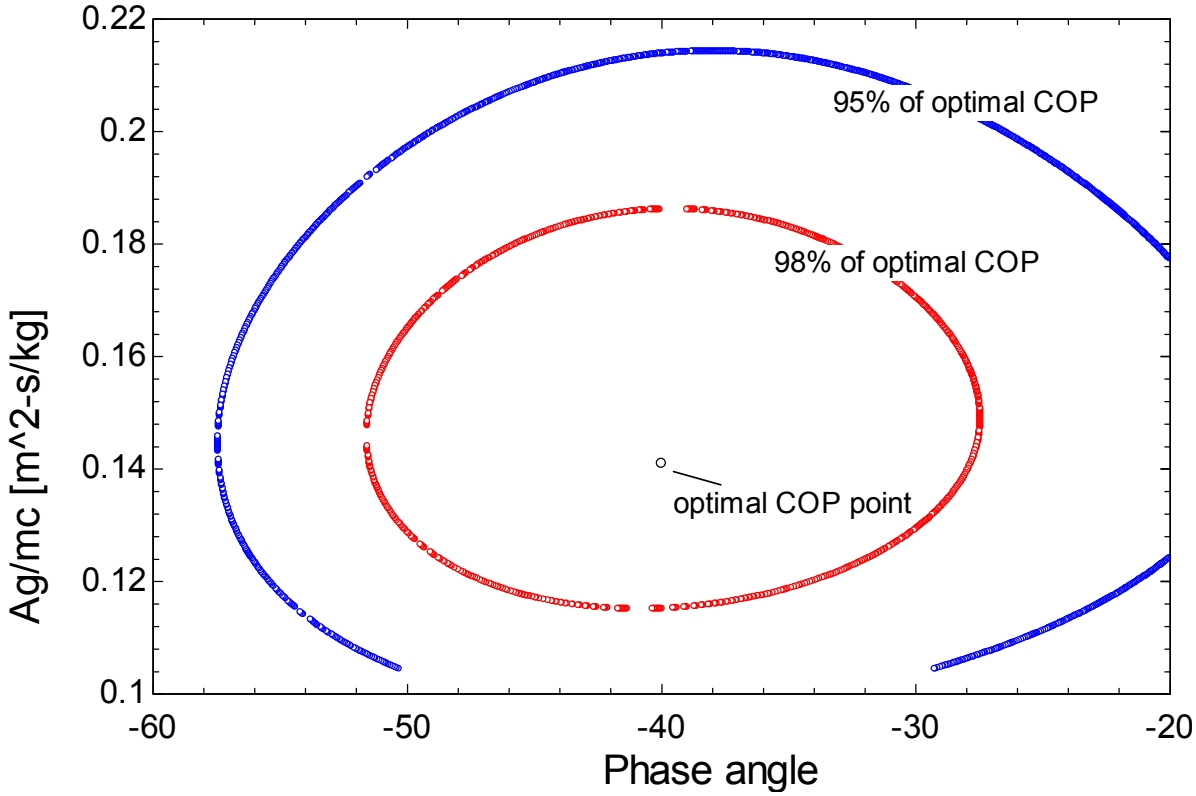

Figure 24 98% and 95% of optimal COP, Ag/mc vs. Phase angle for optimal phase angle

Figure 24Figure 22 shows the cross sectional view of the shell in Figure 18 at length, 0.048 m. The blue and red ring is the boundary of Ag/mc and phase angle with the performance 5% and 2% off the optimal COP respectively. The discontinuity in the two rings is due to limited range of Ag/mc and phase angle being calculated.

The shell shape above shows the flexibility in regenerator design. It could also provide an alternate operating scheme that would maintain the regenerator operating close to optimum efficiency, while changing of mass flow rate or phase angle accommodates the need of a different cooling power. The broad range of variables is not limited to 40Hz, 80K, but applicable to 35K, 60K and 80K at 30Hz to 300Hz. A third order polynomial equation with COP as a function of cold end temperature (Tc), frequency, length, phase angle (phase), Ag/mc (Agmc) is included in Appendix A. It is based on a linear regression of all data from 35K, 60K and 80K with a 0.999 coefficient of determination (R^2). The valid range of frequency for different cold end temperature is the range of frequency shown in Figure 12.

4.4. Convergence Issues for REGEN3.3 and Solutions

4.4.1. Description and Cause of Convergence Issues

As mentioned in Chapter 3, appropriate values for the mesh size and time step are 41 nodes and 300 steps respectively, and these provide a reasonable point of compromise for calculation time and calculation accuracy.

At higher frequencies (>200Hz) or with a large temperature difference between the warm and cold ends, i.e. between 300K and 35K, REGEN3.3 experiences difficulties obtaining a solution with 41 or more mesh points and 300 or higher time steps.

In such cases, error number 129 appears, indicating that an iteration failure has occured, and REGEN will halt without reaching the specified number of FINAL_CYC.

The output message in the REGEN3.3 rgpr file, indicates that the REGEN3.3 numerical solver has failed to solve the condition provided.

A higher number of mesh points may cause an increased scatter in the results because such a condition magnifies the conditions at the end of the regenerator (cold end), where the regenerator loss is determined. Fewer mesh points may lead to some averaging over a larger distance and cause less scatter. The error originates from the method used to calculate the cold end cooling power in REGEN3.3. The cooling power at cold end of the regenerator is calculated by assuming an isothermal expansion of the gas. Since the information for the gas at the cold end comes from the very last node at the cold end, and if that information is inaccurate, the inaccuracies can significantly impact the solution (or lack thereof). Round-off errors due to the fact that the enthalpy flow in each direction come very close to cancelling each other as the flow changes direction can also contribute to the problem. The remainder is the average enthalpy flow, which may have a few percent round-off errors. Those errors can be minimized by a careful selection of the midpoint temperature guess. If the final midpoint temperature is much different (more than about 0.03) than the initial value, there is more opportunity for the calculation not to converge or run into more round-off errors in the convergence [17].

4.4.2. Solutions to the Convergence Problem

The cause of the problem points to the solution of the problem, either by more accurately predicting the mid-point temperature ratio or by reducing the mesh

number. To have a better idea of the mid-point temperature ratio, some pre-calculations can be done using a smaller mesh number and a smaller number of final cycles, thus yielding a better prediction than random guessing. Then the updated mid-temperature ratio can be used as input. This could solve problems in some occasions or at least serve as a starting point for further maneuvers.

A smaller number of mesh points can also bypass the convergence error. As stated above, a mesh of 41 nodes is typically used. For a common 40mm long regenerator, a mesh of 41 nodes will maintain a grid spacing of 1mm. For time steps equal to or greater than 300, the same error message occasionally appears. This, too, can be avoided by reducing the number of time steps. Generally, if the error shows up with the usage of 300 time steps, using 100 or 200 time steps will solve the problem. With a mesh of 41 nodes and 100 time steps, REGEN3.3 is capable of giving satisfactory solution for general purpose.

The same error message can appear at higher frequencies, i.e. greater than 200Hz, or with a lower cold end temperature, i.e. 35K. At 80K, the error occurs frequently at frequencies greater than 150Hz; at 60K, it occurs frequently above 100Hz; at 35K, the error frequently shows up at frequencies higher than 60Hz.

As stated in chapter 2, solutions using 300 time steps are more accurate than with fewer time steps. Therefore it will be ideal to use this number of time steps for the performance mapping work. To achieve this goal, a ‘save and load’ function in REGEN3.3 can be used. By saving the solution calculated with a smaller number of

time steps, subsequent calculations can load the solution and use it as a starting point for a calculation with a greater number of time steps. By implementing the save and load process a few time at most, the solution at the desired number of time steps can be achieved. There are some limits to the use of this function. It only works when the boundary condition input is the exactly the same; and the mesh number is constant. For example, if the saved file is based on a 41-node mesh and 100 time steps, the load input can only use a 41-node mesh, but might have a different number of time steps, i.e. 300.

The “save and load” procedure can be implemented by using the following combined set of commands.

1. Change ‘user_save’ value input to 1 when doing calculations at a small number of time steps. The solution profile is saved in rgsav file, which will be loaded in a subsequent run.
2. Keep all the input variables the same, except set the ‘new_case’ value to 3, and ‘nrun_restart’ equal to the number of the saved file. The new solution will be calculated with a new initial guess. Also increase the number of time steps to either the final number of time steps desired or an intermediate number.
3. Sometimes step 2 needs to be repeated with the use of an intermediate number of time steps, and through two or more iterations, a solution with the desired accuracy at a certain number of time steps can be reached.

4.4.3. Limits of REGEN3.3

As stated above, REGEN3.3 is not capable of giving accurate solutions at certain boundary conditions. With cold and warm end temperatures of 80K and 300K respectively, and for frequencies above 300Hz, REGEN3.3 cannot provide solutions for 41 mesh points and 300 time steps. Lowering number of mesh points and time steps, REGEN3.3 is able to solve the same boundary condition. However, ENGBAL and EHTDIF is higher. At 80K, 300Hz, when using 41 meshes and 300 time steps per cycle, ENGBAL is around 1.5% of ENTFLX on average, EHTDIF is around 3.8% of ENTFLX on average. When using 41 meshes and 100 time steps, the two averages are 4.3% and 8.1% respectively. When using 41 meshes and 200 time steps, the two averages are 2.5% and 4.2% respectively. With cold and warm end temperatures of 60K and 300K respectively, and for frequencies at 150Hz, with 41 mesh points and 100 time steps per cycle, ENGBAL is around 6.8% of ENTFLX, EHTDIF is around 6.6% of ENTFLX. The two averages are 4.6% and 2.1% respectively with 41 mesh points and 200 time steps per cycle. With 41 mesh points and 300 times steps, the two averages are 1.3% and 4.4% respectively.

4.5. Case Comparison of REGEN3.3 Modeling Results with Experimental Data

A comparison of REGEN3.3 modeling results with experimental data in open literature is done. Two cases were calculated and compared to actual experimental data. The first case is a pulse tube cryocooler with cold end temperature at 80K, 635

meshes stainless steel matrix and 3 MPa average pressure working at 120Hz [22]. The second case is another pulse tube cryocooler with cold end temperature at 80K, 635 meshes stainless steel matrix and 5 MPa average pressure working at 150Hz [23]. Based on the physical conditions from these papers, the optimal sets of variables are predicated using REGEN3.3. For case 1, the actual cooling power is 3.35 W for case 1, and the predicted optimal cooling power is 4.08 W; the actual input PV work (after cooler) is 46.4 W, and the predicated optimal input power is 42.5 W. For case 2, the actual cooling power is 0.5 W, and the predicated optimal cooling power is 1.01 W; the actual swept volume for the compressor is 0.5 cm^3, and the predicated optimal input power is 10.3 W.

4.6. Scale Relationship Between Cross Sectional Area and Cooling Power

As stated in section 3.1, COP is only a function of Ag/mc instead of depending on both Ag and mc separately.

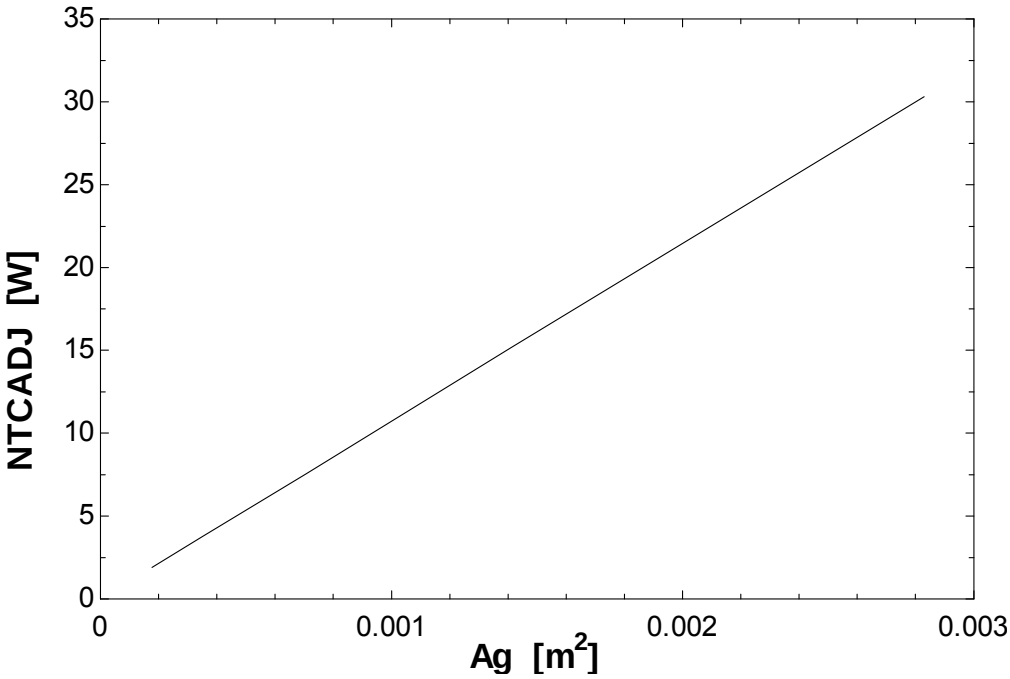

Figure 25 Cooling Power vs. Cross Sectional Area at 60K, 30Hz, 0.065m

The figure above shows with the optimal length (0.065m), phase angle (-45 degree) and Ag/mc (0.143 m^2-s/kg) at 60K, 30Hz, the cooling power is linearly proportional to the cross sectional area (Ag). And the COP for different cross sectional is almost constant at 9.268E-2 with a maximum 6E-4 variance. Therefore, if a different cooling power is required, a simple scale of cross sectional area will suffice, while maintaining the same COP.

4.7. Conclusion

A parametric study on the optimal choice of regenerator length, phase angle between cold end mass flow and pressure oscillations, and mass flow rate has been accomplished using REGEN 3.3. The operating conditions that were maintained

constant throughout this study include a mean pressure of 2MPa, a pressure ratio of 1.2,,and a 400 mesh stainless steel matrix material, operating in the frequency range from 30Hz to 300Hz. Maps of the optimal COP as a function of frequency, regenerator length, phase angle and mass flow rate have been generated. In addition to the maps, the study has revealed a significant tolerance when choosing length, phase angle and mass flow rate, within which there is little downgrading of the COP. The flexibility is shown in a shell shape 3-D plot with Length, Phase angle and Ag/mc as the three coordinates.

Over 10000 iterations through the loops were done in order to accomplish the mapping. Based on the map, without running into complicated optimization procedure, one can design a cryogenic regenerator with optimal performance, and satisfy the cooling power requirement with a simple scale of the cross sectional area. The author recommends using the optimal COP and corresponding variables figures to find the optimal variables at certain frequency of interest. Then a scale of cross sectional area can be used to adjust the cooling power in order to suffice the cooling power requirement.

References

[1] R. A. Ackermann, *Cryogenic Regenerative Heat Exchangers*. Springer, 1997.

[2] J. Herschel, "Making Ice," *The Athenaeum*, p. 22, 1850.

[3] W. E. Gifford and H. O. McMahon, "A New Low-Temperature Gas Expansion Cycle," in *Advances in Cryogenic Engineering SE - 44*, vol. 5, K. D. Timmerhaus, Ed. Springer US, 1960, pp. 368–372.

[4] R. Ackermann, K. Herd, and W. Chen, "Advanced Cryocooler Cooling for MRI Systems," in *Cryocoolers 10 SE - 101*, R. G. Ross Jr., Ed. Springer US, 2002, pp. 857–867.

[5] W. V Hassenzahl, D. W. Hazelton, B. K. Johnson, P. Komarek, M. Noe, and C. T. Reis, "Electric power applications of superconductivity," *Proceedings of the IEEE*, vol. 92, no. 10. pp. 1655–1674, 2004.

[6] R. Radebaugh, "Pulse tube cryocoolers for cooling infrared sensors," 2000, vol. 4130, pp. 363–379.

[7] R. G. Ross and R. F. Boyle, "An Overview of NASA Space Cryocooler Programs — 2006," pp. 1–10, 2006.

[8] B. A. Willemsen, "HTS filter subsystems for wireless telecommunications," *Applied Superconductivity, IEEE Transactions on*, vol. 11, no. 1. pp. 60–67, 2001.

[9] J. Gary, A. O. Gallagher, R. Radebaugh, Y. Huang, and E. Marquardt, "REGEN3 . 3 : USER MANUAL," no. April, 2008.

[10] D. Gedeon, *Sage User's Guide*. 2013, p. 423.

[11] B. Ward, J. Clark, and G. Swift, "Design Environment for Low-amplitude Thermoacoustic Energy Conversion DeltaEC Users Guide," 2012.

[12] A. Kashani and P. Roach, "An Optimization Program for Modeling Pulse Tube Coolers," in *Advances in Cryogenic Engineering SE - 240*, vol. 43, P. Kittel, Ed. Springer US, 1998, pp. 1903–1909.

[13] M. J. Cheadle, G. F. Nellis, S. a. Klein, and J. G. Weisend, "Micro-Scale Cfd Modeling of Oscillating Flow in a Regenerator," vol. 157, pp. 157–164, 2010.

[14] J. Shi, J. Pfotenhauer, and G. Nellis, "Dimensionless analysis for regenerator design," …, vol. 5, pp. 419–427, 2007.

[15] W. M. Kays and A. L. London, *Compact Heat Exchangers*. Krieger Publishing Company, 1984.

[16] R. Radebaugh, "Thermodynamics of regenerative refrigerators," *Gener. Low Temp. It's Appl.*, no. 3, 2003.

[17] R. Radebaugh, "Personal Communication." 2013.

[18] K. F. Alvin, S. M. Deland, B. M. Rutherford, and K. V Diegert, "Estimation of Total Uncertainty in Modeling and Simulation Sandia National laboratories," no. April, 2000.

[19] M. J. Cheadle, "FUNDAMENTAL CHALLENGES OF PULSE TUBE CRYOCOOLER," 2011.

[20] B. Taylor and C. Kuyatt, "NIST Technical Note 1297 1994 Edition Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results NIST Technical Note 1297 1994 Edition Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results," 1994.

[21] R. Radebaugh, "Regenerator Operation at Very High Frequencies for Microcryocoolers," *AIP Conf. Proc.*, vol. 823, no. 1, pp. 1919–1928, 2006.

[22] S. Vanapalli, M. Lewis, Z. Gan, and R. Radebaugh, "120 Hz pulse tube cryocooler for fast cooldown to 50 K," *Appl. Phys. Lett.*, vol. 90, no. 7, p. 072504, 2007.

[23] I. Garaway, Z. Gan, P. Bradley, A. Veprik, and R. Radebaugh, "Development of a miniature 150 Hz pulse tube cryocooler," *Cryocoolers*, pp. 105–113, 2009.

Appendix

A. Fitted Equations

Complete EES code of function for COP versus cold end temperature, frequency, length, phase angle and Ag/mc.

COP=1.23970280E+02-7.18757276E+00*Tc+1.29693036E-01*Tc^2-7.41055480E-04*Tc^3+3.26846344E-03*frequency-1.25217679E-05*frequency^2+1.45707816E-08*frequency^3+2.45220342E+00*Agmc-9.90820748E+00*Agmc^2+1.24580658E+01*Agmc^3+2.08710943E+01*Length-2.75551010E+02*Length^2+1.16781491E+03*Length^3+8.71052204E-04*Phase+3.55329386E-06*Phase^2+2.85104062E-07*Phase^3-5.19642079E-06*Tc*frequency+3.30600388E-08*Tc*frequency^2+2.41430363E-02*Tc*Agmc-8.61562254E-02*Tc*Agmc^2+4.29746439E-02*Tc*Length-3.84109126E-01*Tc*Length^2+3.24198122E-05*Tc*Phase+2.75795663E-07*Tc*Phase^2+7.68615136E-08*Tc^2*frequency-3.86305968E-10*Tc^2*frequency^2-1.10144645E-04*Tc^2*Agmc+5.21349306E-04*Tc^2*Agmc^2-4.83312627E-04*Tc^2*Length+4.09054319E-03*Tc^2*Length^2-1.73844674E-07*Tc^2*Phase-8.58438216E-10*Tc^2*Phase^2-1.08986468E-02*frequency*Agmc+4.11551993E-02*frequency*Agmc^2-8.88911092E-02*frequency*Length+5.47629162E-01*frequency*Length^2-2.50070461E-05*frequency*Phase-1.26658689E-07*frequency*Phase^2+5.40676338E-05*frequency^2*Agmc-4.52071888E-04*frequency^2*Agmc^2+1.32503493E-

04*frequency^2*Length-3.00641109E-04*frequency^2*Length^2+5.47009135E-08*frequency^2*Phase+3.20184342E-10*frequency^2*Phase^2-8.07379688E+01*Agmc*Length+6.28249985E+02*Agmc*Length^2-2.31567586E-02*Agmc*Phase-2.64856313E-04*Agmc*Phase^2+2.01020159E+02*Agmc^2*Length-1.44485083E+03*Agmc^2*Length^2+5.41335150E-02*Agmc^2*Phase+5.17846603E-04*Agmc^2*Phase^2+3.02790756E-02*Length*Phase+9.08427256E-04*Length*Phase^2-2.64856825E-01*Length^2*Phase-5.64844604E-03*Length^2*Phase^2

EES code for Curve fitting

Optimal COP as a function of frequency and cold end temperature

COP=1.87500000E-01-1.60330745E-04*frequency-6.34708228E-04*Tc

Optimal Ag/mc and Length as a function of frequency and cold end temperature

```
$If PARAMETRICTABLE = '80K'
Agmc  = 1.00386*frequency^(-0.524077) "80K"
Length=0.203775*frequency^(-0.389682) "only for 80K"

{$elseif PARAMETRICTABLE = '60K'
Agmc = 0.830617*frequency^(-0.523687)
Length=0.248206*frequency^(-0.404045)}

$elseif PARAMETRICTABLE='35K'
Agmc=0.598768*frequency^(-0.569334)
Length=0.143396*frequency^(-0.24321)

$endif
```

Optimal Phase angle as a function of frequency and cold end temperature

phase =-6.68501446E+01+4.82374979E-01*Tc-6.32643468E-04*Tc^2-6.91590949E-02*frequency+4.63778937E-04*frequency^2-1.93921405E-03*Tc*frequency

Optimal COP density vs. Frequency as a function of frequency and cold end temperature

COP_density=6.37731042E-01-2.23726000E-02*Tc+5.19152683E-04*Tc^2-3.33810814E-03*frequency-3.29232592E-05*frequency^2+2.45012562E-04*Tc*frequency

Cooling power per area at optimal performance vs. frequency at various temperatures

Cooling =-1.36783069E+03+1.83803351E+02*Tc+6.94463471E-02*Tc^2+1.25959770E+01*frequency-1.08053778E-01*frequency^2+6.17245389E-01*Tc*frequency

B. Codes for Programms

1. C code for consecutively running Regen3.3

```
#include "stdafx.h"
#include  "stdlib.h"
#include  "string.h"
int _tmain(int argc, _TCHAR* argv[])
{
        FILE  *fp1, *fp2;
        char c;
        int i;
        int startnumber=0,number=0,current=0, endnumber=0;
        char s1[60],s2[8]=".txt",filename[28]="data";
```

```
printf_s("Input startnumber \n");
scanf_s("%d", &startnumber);
printf_s("Input end number of caculation \n");
scanf_s("%d", &endnumber);
printf_s("\n\n");
number=endnumber-startnumber+1;
for(i=0;i<number;i++){
	strcpy(s1, "");
	strcpy(s2, ".txt");
	strcpy(filename, "data");
	current=startnumber+i;
	sprintf(s1, "%d", current);
	strcat(filename, s1);
strcat(filename, s2);
printf("%s \n\n\n",filename);
if((fp1=fopen(filename,"r"))==NULL){
	printf("File open error! \n");
}

if((fp2=fopen("data.DAT","w"))==NULL){
	printf("Can not close the file! \n");
}
while(!feof(fp1)){
	c=fgetc(fp1);
	fputc(c,fp2);
}
fclose(fp1);
fclose(fp2);
system("d:\\SkyDrive\\Study\\Regen33\\Regen33programs\\REGEN33\
\rg33.exe"); /*subject to change based on the directory where rg33.exe is*/
printf("End of %d Caculation! \n\n\n", i+1);
}
printf("END OF ALL %d CACULATION! \n", i);
getchar();
getchar();
return 0;
```

2. Matlab code for generating massive data files

```
clear all
%----Generate Series of Data Files for Parametric Studies using Regen 3.3-
-
%==========================================================================
=
%===== Unchanging Constants
==============================================
```

```
%-----------------------------------------------------------------------
-
% pressure

Pr       = 1.3;                                  %pressure ratio    [-]
P0       = 3.50000E+6;                           %average pressure  [Pa]
pres_inc = 0.05*P0;                              %pressure increment,
default setup as 0.05 of average pressure[Pa]
pres_ini = P0;                                   %initial pressure, set
equal to average pressure [Pa]
P_am     = (Pr-1)/(Pr+1)*P0;                     %pressure amplitude, used
for analysis of phase relation,  imaginary part[Pa]

%-----------------------------------------------------------------------
-
% temperature

Th = 300;                                        %hot end temperature [K]

%-----------------------------------------------------------------------
-
% material & fluid properties

material      = 1;                               %material number,
stainless steel = 1 [-]
porosity      = 0.6014;                         %porosity of stainless
stell screen = 0.68584 [-]
hydra_diam    = 1.531E-05;                       %hydraulic diameter of
#400 meshes stainless steel screen with 0.68584 porosity [m]
R             = 2077.5;                          %gas constant for helium-4
[J/kg-K]

%-----------------------------------------------------------------------
-
% geometry

d        = 0.00902000000;                        %outer diameter of
regenerator [m]
rg_area  = pi/4*d^2;                             %cross section area of
regenerator [m^2]
geometry = 4;                                    %matrix geometry setup, 4
represent screens [-]
pi       = 3.1416;                               %pi,[-]
%-----------------------------------------------------------------------
-
% calculation control

decay_cyc       = 0.05;                          %e-folding time for
calculating the inflow temp at out- to in-flow reversal, default 0.05 [-]
eps_newton      = 1E-05;                         %error tolerance of
velocity for the Newtonian iterations , default 1e-6 [m/s]
htalp           = 4.000000E-02;                  %to get heat transfer
coefficient between gas and matrix, see Page 16, default 0.04[-]
```

```
cooling_mult    = 8.000000E-01;                 %adjustment factor for
isotheraml expasion net cooling at the cold end, default 1, use 0.8[-]
mat_cond_factor = 1.300000E-01;                 %factor used to reduce the
thermal conductivity of the matrix, default 1, use 0.13 [-]
mid_temp_ratio  = 0.45;                         %middle point temperature
ratio, default 0.5, might be a function of mc, default in Regen is 0.5 [-]
nplot           = 3;                            %output graph control

%==========================================================================
=
%----- Parametric Study Control-----------------------------------------
-
%==========================================================================
=
% calculation accuracy control

num_point_x = 41;                               %number of mesh points
along the matrix
num_steps   = 200;                              %number of time steps per
cycle
final_cycle = 60000;
% initial value setup

Tc        = 80;                                 %cold end temperature, [K]
f         = 120;                                %frequency, [Hz]
L_ini     = 3.0E-02;                            %initial length of the
series, [m]
theta_ini = -40;                                %initial phase angle of
one length, [deg]
theta_fin = -40;                                %final phase angle of one
length,   [deg]
%Ag_mc_ini = 1.6E-02;                            %initial Ag/mc value,
[m^2-s/kg], not in use
mc_ini    = 5.0e-4;                             %initial mc value,[kg/s]
n_file_ini= 6370;                               %initial file number

%-------------------------------------------------------------------------
-
% increment & file number setup

num_len     = 1;                                  %number of different
lengths to calculate, [-]
phase_inc   = 5;                                  %increment of phase
angle, [deg]
num_phase   = (theta_ini-theta_fin)/phase_inc + 1;%number of different
phases to calculate,  [-]
Ag_mc_incre = 1E-02;                              %increment of Ag_mc,
[m^2-s/kg]
mc_incre    = 6E-5;                             %increment of mc
len_inc     = 2E-03;                               %increment of length, [m]
n_file      = n_file_ini;                         %initialize file
number,[-]
num_file_pha= 8;                                  %number of calculations
for one phase angle, [-]
```

```
%-----------------------------------------------------------------------
-
% iteration loops
%----changed with temperature

Tr = (Th -
Tc)/log(Th/Tc);                                                  %regenerator
mean temperature, [K]

%----changed with frequency

omega =
2*pi*f;                                                             %angle
velocity, [rad/s]

%----changed with length

%mid_temp_ratio =
zeros(num_file_pha);                                     %possible
mid_temp_ratio array,[-]

% user save and load control

use_save     =
1;                                                          %%user save option,
1 means want to save the file for further calculations, comment if not
used
nrun_restart =
6320;                                                      %%user load file
number, input the first user file number want to load, comment if not used
newcase      =
3;                                                         %default is 1,
start new calculation, 3 is used to load a file with rgsav.

for  i =1:num_len
    len  = (i-1)*len_inc +
L_ini;                                          %length of the regenerator, [m]
    V_rg = rg_area *
len;                                                 %volume of regenerator,
[m^3]
    m_img=
P_am*V_rg*omega/(R*Tr);                                          %acoustic
power normalized mass flow rate, [kg/s]

    %----changed with phase

    for j = 1:num_phase
        theta = -(j-1)*phase_inc +
theta_ini;                            %phase angle between mc and pressure
wave, [deg]

        %----changed with mc
```

```
        for k = 1:num_file_pha
                %mc = rg_area/(Ag_mc_incre*(k-1) +
Ag_mc_ini)          %cold end mass flow rate, [kg/s]
                mc = mc_ini + mc_incre*(k-
1);                          %cold end mass flow rate,[kg/s]
                mh = (mc^2 + m_img^2 +
2*mc*m_img*cos((theta+90)))^0.5;%hot end mass flow rate, [kg/s]

           %----mid_temp setup

                %mid_temp_ratio = -0.187*log(mc) -
0.6622;                 %mid Tempearture ratio , [-]

           %----file print out-------------------------------------------
-
           file_num  =
int2str(n_file);                                %file number to string
           %file_name = [ 'data' file_num
'.txt' ];                     %current file name

           %file name for iteration control
           file_name = ['data' '.dat'];

           file = fopen(file_name,
'wt');                                %open file

                fprintf(file, ' &inp      nrun=%d \n' ,           n_file);
                fprintf(file, ' final_cycle=    %d \n' ,
final_cycle);
                fprintf(file, ' geometry=       %d\n' ,
geometry);

                fprintf(file, ' gas_temp_cold= %f\n',            Tc);
                fprintf(file, ' gas_temp_hot= %f\n',             Th);
                fprintf(file, ' herz= %f\n',                     f);

                fprintf(file, ' hydra_diam= %11.9f\n',
hydra_diam);
                fprintf(file, ' mass_flux_cold= %.8f\n',           mc);
                fprintf(file, ' mass_flux_hot= %f\n',            mh);

                fprintf(file, ' porosity= %f\n',                  porosity);
                fprintf(file, ' pres_initial= %f\n',             P0);
                fprintf(file, ' rg_area= %9.7f\n',               rg_area);
                fprintf(file, ' rg_length= %f\n',                len);
                fprintf(file, ' material=      %d\n' ,
material);
                fprintf(file, ' pres_ratio= %f\n',               Pr);
                fprintf(file, ' pres_phase= %f\n',               theta);
                fprintf(file, ' ave_pres= %f\n',                 P0);
                fprintf(file, ' decay_cyc= %f\n',
decay_cyc);
```

```
                fprintf(file, ' eps_newton= %f\n',
eps_newton);
                fprintf(file, ' htalp= %f\n',                    htalp);
                fprintf(file, ' num_points_x=      %d\n' ,
num_point_x);
                fprintf(file, ' num_steps_cyc=     %d\n' ,
num_steps);

                fprintf(file, ' use_save=     %d\n' ,
use_save);

                fprintf(file, ' cooling_mult= %f\n',
cooling_mult);

                fprintf(file,' newcas=     %d\n',               newcase);
                fprintf(file,' nrun_restart=     %d\n',
nrun_restart);

                fprintf(file, ' output_inc=    %d \n' ,
final_cycle);
                fprintf(file, ' nplot     =    %d\n' ,            nplot);
                fprintf(file, ' mat_cond_factor= %f\n',
mat_cond_factor);
                fprintf(file, ' mid_temp_ratio= %f\n',
mid_temp_ratio);
                fprintf(file, '/\n');

                fclose(file);
        %-------end of txt. file print out--------------------------------
-

        %for iteration control
        %-------call regen programm----------------------------------------
-
        s = system ('rg33');

        n_file       = n_file + 1;                                  %next
file number, [-]
        nrun_restart = nrun_restart + 1;                            %next
load file number, [-]
        end
    end
end
```

3. Python code for reading massive output files

```
#variables needed to be read
#-------------------------------------
#from the input variable

#cold end temperature
```

```
GAS_TEMP_COLD     = "0"
GAS_TEMP_COLD_pos = -1

#regenerator length
RG_LENGTH         = "0"
RG_LENGTH_pos     = -1

#frequency of the mass flux
HERZ              = "0"
HERZ_pos          = -1

#cold end mass flux
MASS_FLUX_COLD    = "0"
MASS_FLUX_COLD_pos= -1

#phase between mass and pressure
PRES_PHASE        = "0"
PRES_PHASE_pos    = -1

#number of mesh points
NUM_POINTS_X      = "0"
NUM_POINTS_X_pos  = -1

#number of steps per cycle
NUM_STEPS_CYC     = "0"
NUM_STEPS_CYC_pos = -1

#duration of cycle
FINAL_CYCLE       = "0"
FINAL_CYCLE_pos   = -1

#---------------------------------------
#from output

#maximum CFL number over cycle
CFLMAX     = "0"
CFLMAX_pos = -1

#enthalpy plus heat variation across the regenerator
EHTDIF     = "0"
EHTDIF_pos = -1

#enthalpy plus heat flux at cold side of regenerator
EHTFLX     = "0"
EHTFLX_pos = -1
```

```
#energy inbalance for the gas over the cycle
ENGBAL     = "0"
ENGBAL_pos = -1

#normed matrix temperature at mid point
GTPNRM     = "0"
GTPNRM_pos = -1

#second law efficiency
EFFIC      = "0"
EFFIC_pos  = -1

#specific gas cross sectional area
GASARS     = "0"
GASARS_pos = -1

#gas volume in regenerator
GASVO      = "0"
GASVO_pos  = -1

#ineffectiveness
INEFCT     = "0"
INEFCT_pos = -1

#COP
NTACOP     = "0"
NTACOP_pos = -1

#adjusted net refrigeration power
NTCADJ     = "0"
NTCADJ_pos = -1

#portion of enthalpy flux due to pressurization
PRLOSS     = "0"
PRLOSS_pos = -1

#pv work done by gas at warm end
PVWK0T     = "0"
PVWK0T_pos = -1

#pv work by pressure drop at warm end
PVWKPR     = "0"
PVWKPR_pos = -1
```

```
#regenerator loss [W], loss due to ineffectiveness
RGLOSS     = "0"
RGLOSS_pos = -1

#the integral average of the enthalpy flux at the right side of the regenerator
over one cycle
ENTFLX     = "0"
ENTFLX_pos = -1

#-----END of data Variable setup-----------------------------------------------------------
-------

#-----------------Control variables-----------------------------------------------------------
--
num_file_1 = 0
num_file_2 = 0
num_file   = 0
file_nam   = "0"
rgpr       = "rgpr"
line       = "0"

#position of the actual number for two types of data
p11        = int(17)
p12        = int(27)
p21        = int(8)
p22        = int(17)

num_file_1_in = input("input start file number \n")
num_file_2_in = input("input end file number \n")

#integr number of files
num_file_1 = int(num_file_1_in)
num_file_2 = int(num_file_2_in)

num_file   = num_file_2 - num_file_1 + 1

#open txt file to be written
f_write = open("data read.txt", "w")

#loop through all output files
for i in range(0, num_file):
  file_num   = i + num_file_1
  file_nam   = rgpr + str(file_num)
  string     = str(file_num)
#reset variables
```

```
  CFLMAX     = "0"
  EHTDIF     = "0"
  EHTFLX     = "0"
  ENGBAL     = "0"
  GTPNRM     = "0"
  EFFIC      = "0"
  GASARS     = "0"
  GASVO      = "0"
  INEFCT     = "0"
  NTACOP     = "0"
  NTCADJ     = "0"
  PRLOSS     = "0"
  PVWK0T     = "0"
  PVWKPR     = "0"
  RGLOSS     = "0"
  ENTFLX     = "0"

#open the file
  try:
    file_regen = open(file_nam)
    for line in file_regen:
        GAS_TEMP_COLD_pos = line.find("GAS_TEMP_COLD")
        RG_LENGTH_pos     = line.find("RG_LENGTH")
        HERZ_pos          = line.find("HERZ")
        MASS_FLUX_COLD_pos= line.find("MASS_FLUX_COLD")
        PRES_PHASE_pos    = line.find("PRES_PHASE")
        NUM_POINTS_X_pos  = line.find("NUM_POINTS_X")
        NUM_STEPS_CYC_pos = line.find("NUM_STEPS_CYC")
        FINAL_CYCLE_pos   = line.find("FINAL_CYCLE")
        CFLMAX_pos        = line.find("CFLMAX")
        EHTDIF_pos        = line.find("EHTDIF")
        EHTFLX_pos        = line.find("EHTFLX")
        ENGBAL_pos        = line.find("ENGBAL")
        GTPNRM_pos        = line.find("GTPNRM")
        EFFIC_pos         = line.find("EFFIC")
        GASARS_pos        = line.find("GASARS")
        GASVO_pos         = line.find("GASVO")
        INEFCT_pos        = line.find("INEFCT")
        NTACOP_pos        = line.find("NTACOP")
        NTCADJ_pos        = line.find("NTCADJ")
        PRLOSS_pos        = line.find("PRLOSS")
        PVWK0T_pos        = line.find("PVWK0T")
        PVWKPR_pos        = line.find("PVWKPR")
        RGLOSS_pos        = line.find("RGLOSS")
```

```
        ENTFLX_pos        = line.find("ENTFLX")
        #reading input
        if GAS_TEMP_COLD_pos != -1:
           GAS_TEMP_COLD =
line[GAS_TEMP_COLD_pos+p11:GAS_TEMP_COLD_pos+p12].rstrip()
        elif RG_LENGTH_pos   != -1:
              RG_LENGTH     = line[RG_LENGTH_pos +
p11:RG_LENGTH_pos + p12].rstrip()
        elif HERZ_pos        != -1:
                 HERZ          = line[HERZ_pos + p11:HERZ_pos + p12].rstrip()
        elif MASS_FLUX_COLD_pos!= -1:
                    MASS_FLUX_COLD=
line[MASS_FLUX_COLD_pos+p11:MASS_FLUX_COLD_pos+p12].rstrip()
        elif PRES_PHASE_pos    != -1:
                        PRES_PHASE    = line[PRES_PHASE_pos+p11-
1:PRES_PHASE_pos+p12].rstrip()
        #be careful with the position of numbers after this
        elif NUM_POINTS_X_pos  != -1:
                           NUM_POINTS_X  =
line[NUM_POINTS_X_pos+p11:NUM_POINTS_X_pos+p11+3].rstrip()
        elif NUM_STEPS_CYC_pos != -1:
                              NUM_STEPS_CYC =
line[NUM_STEPS_CYC_pos+p11:NUM_STEPS_CYC_pos+p11+3].rstrip()
        elif FINAL_CYCLE_pos   != -1:
                                 FINAL_CYCLE   =
line[FINAL_CYCLE_pos+p11:FINAL_CYCLE_pos+p11+5].rstrip()
                                 #changed to position type 2 as start reading output
        elif CFLMAX_pos        != -1:
                                   CFLMAX        =
line[CFLMAX_pos+p21:CFLMAX_pos+p22].rstrip()
        elif EHTDIF_pos        != -1:
                                       EHTDIF        =
line[EHTDIF_pos+p21:EHTDIF_pos+p22].rstrip()
        elif EHTFLX_pos        != -1:
                                           EHTFLX        =
line[EHTFLX_pos+p21:EHTFLX_pos+p22].rstrip()
        elif ENGBAL_pos        != -1:
                                               ENGBAL        =
line[ENGBAL_pos+p21:ENGBAL_pos+p22].rstrip()
        elif GTPNRM_pos        != -1:
                                                   GTPNRM        =
line[GTPNRM_pos+p21:GTPNRM_pos+p22].rstrip()
        elif EFFIC_pos         != -1:
                                                       EFFIC         =
line[EFFIC_pos+p21:EFFIC_pos+p22].rstrip()
```

```
        elif GASARS_pos        != -1:
                                                GASARS        =
line[GASARS_pos+p21:GASARS_pos+p22].rstrip()
        elif GASVO_pos         != -1:
                                                GASVO         =
line[GASVO_pos+p21:GASVO_pos+p22].rstrip()
        elif INEFCT_pos        != -1:
                                                INEFCT        =
line[INEFCT_pos+p21:INEFCT_pos+p22].rstrip()
        elif NTACOP_pos        != -1:
                                                NTACOP        =
line[NTACOP_pos+p21:NTACOP_pos+p22].rstrip()
        elif NTCADJ_pos        != -1:
                                                NTCADJ        =
line[NTCADJ_pos+p21:NTCADJ_pos+p22].rstrip()
        elif PRLOSS_pos        != -1:
                                                PRLOSS        =
line[PRLOSS_pos+p21:PRLOSS_pos+p22].rstrip()
        elif PVWK0T_pos        != -1:
                                                PVWK0T        =
line[PVWK0T_pos+p21:PVWK0T_pos+p22].rstrip()
        elif PVWKPR_pos        != -1:
                                                PVWKPR        =
line[PVWKPR_pos+p21:PVWKPR_pos+p22].rstrip()
        elif RGLOSS_pos        != -1:
                                                RGLOSS
= line[RGLOSS_pos+p21:RGLOSS_pos+p22].rstrip()
        elif ENTFLX_pos        != -1:
                                                ENTFLX
= line[ENTFLX_pos+p21:ENTFLX_pos+p22].rstrip()


    check = int(NUM_STEPS_CYC) #check if number of steps per cycle is
greater than 300
    #COP vs. mc,L,phase
    if NTACOP =="0":
      string = ""
    elif check <280:
      string = ""
    else:
      #string = NTACOP+" "+GASARS+" "+RG_LENGTH+"
"+PRES_PHASE+"\n"
      string = str(file_num) +" "+ GAS_TEMP_COLD+" "+RG_LENGTH+"
"+HERZ+" "+MASS_FLUX_COLD+" "+PRES_PHASE+"
"+NUM_POINTS_X+" "+NUM_STEPS_CYC+" "+FINAL_CYCLE+"
```

```
        "+CFLMAX+" "+EHTDIF+" "+EHTFLX+" "+ENGBAL+" "+GTPNRM+"
        "+EFFIC+" "+GASARS+" "+GASVO+" "+INEFCT+" "+NTACOP+"
        "+NTCADJ+" "+PRLOSS+" "+PVWK0T+" "+PVWKPR+" "+RGLOSS+"
        "+ENTFLX+" "+"\n"
            f_write.write(string)
            file_regen.close()
          except FileNotFoundError:
                print ("no file")
                #string = string + " " + "\n"
                #f_write.write(string)
        f_write.close()
end = input("finish reading")
```

4. Matlab and python code for making the shell

1. Python code

```
        #variables needed to be read
        #-------------------------------------
        #from the input variable

        #cold end temperature
        GAS_TEMP_COLD     = "0"
        GAS_TEMP_COLD_pos = -1

        #regenerator length
        RG_LENGTH         = "0"
        RG_LENGTH_pos     = -1

        #frequency of the mass flux
        HERZ              = "0"
        HERZ_pos          = -1

        #cold end mass flux
        MASS_FLUX_COLD    = "0"
        MASS_FLUX_COLD_pos= -1

        #phase between mass and pressure
        PRES_PHASE        = "0"
        PRES_PHASE_pos    = -1

        #number of mesh points
        NUM_POINTS_X      = "0"
```

```
NUM_POINTS_X_pos  = -1

#number of steps per cycle
NUM_STEPS_CYC     = "0"
NUM_STEPS_CYC_pos = -1

#duration of cycle
FINAL_CYCLE       = "0"
FINAL_CYCLE_pos   = -1

#----------------------------------------
#from output

#maximum CFL number over cycle
CFLMAX     = "0"
CFLMAX_pos = -1

#enthalpy plus heat variation across the regenerator
EHTDIF     = "0"
EHTDIF_pos = -1

#enthalpy plus heat flux at cold side of regenerator
EHTFLX     = "0"
EHTFLX_pos = -1

#energy inbalance for the gas over the cycle
ENGBAL     = "0"
ENGBAL_pos = -1

#normed matrix temperature at mid point
GTPNRM     = "0"
GTPNRM_pos = -1

#second law efficiency
EFFIC      = "0"
EFFIC_pos  = -1

#specific gas cross sectional area
GASARS     = "0"
GASARS_pos = -1

#gas volume in regenerator
GASVO      = "0"
GASVO_pos  = -1
```

```
#ineffectiveness
INEFCT     = "0"
INEFCT_pos = -1

#COP
NTACOP     = "0"
NTACOP_pos = -1

#adjusted net refrigeration power
NTCADJ     = "0"
NTCADJ_pos = -1

#portion of enthalpy flux due to pressurization
PRLOSS     = "0"
PRLOSS_pos = -1

#pv work done by gas at warm end
PVWK0T     = "0"
PVWK0T_pos = -1

#pv work by pressure drop at warm end
PVWKPR     = "0"
PVWKPR_pos = -1

#regenerator loss [W], loss due to ineffectiveness
RGLOSS     = "0"
RGLOSS_pos = -1

#the integral average of the enthalpy flux at the right side of the regenerator
over one cycle
ENTFLX     = "0"
ENTFLX_pos = -1

#-----END of data Variable setup-----------------------------------------------------------
-------

#-----------------Control variables-----------------------------------------------------------
--
num_file_1 = 0
num_file_2 = 0
num_file   = 0
file_nam   = "0"
rgpr       = "rgpr"
line       = "0"
```

```
#position of the actual number for two types of data
p11        = int(17)
p12        = int(27)
p21        = int(8)
p22        = int(17)

num_file_1_in = input("input start file number \n")
num_file_2_in = input("input end file number \n")

#integr number of files
num_file_1 = int(num_file_1_in)
num_file_2 = int(num_file_2_in)

num_file   = num_file_2 - num_file_1 + 1

#open txt file to be written
f_write = open("data read.txt", "w")

#loop through all output files
for i in range(0, num_file):
  file_num   = i + num_file_1
  file_nam   = rgpr + str(file_num)
  string     = str(file_num)
#reset variables

  CFLMAX     = "0"
  EHTDIF     = "0"
  EHTFLX     = "0"
  ENGBAL     = "0"
  GTPNRM     = "0"
  EFFIC      = "0"
  GASARS     = "0"
  GASVO      = "0"
  INEFCT     = "0"
  NTACOP     = "0"
  NTCADJ     = "0"
  PRLOSS     = "0"
  PVWK0T     = "0"
  PVWKPR     = "0"
  RGLOSS     = "0"
  ENTFLX     = "0"

#open the file
  try:
    file_regen = open(file_nam)
```

```
    for line in file_regen:
        GAS_TEMP_COLD_pos = line.find("GAS_TEMP_COLD")
        RG_LENGTH_pos     = line.find("RG_LENGTH")
        HERZ_pos          = line.find("HERZ")
        MASS_FLUX_COLD_pos= line.find("MASS_FLUX_COLD")
        PRES_PHASE_pos    = line.find("PRES_PHASE")
        NUM_POINTS_X_pos  = line.find("NUM_POINTS_X")
        NUM_STEPS_CYC_pos = line.find("NUM_STEPS_CYC")
        FINAL_CYCLE_pos   = line.find("FINAL_CYCLE")
        CFLMAX_pos        = line.find("CFLMAX")
        EHTDIF_pos        = line.find("EHTDIF")
        EHTFLX_pos        = line.find("EHTFLX")
        ENGBAL_pos        = line.find("ENGBAL")
        GTPNRM_pos        = line.find("GTPNRM")
        EFFIC_pos         = line.find("EFFIC")
        GASARS_pos        = line.find("GASARS")
        GASVO_pos         = line.find("GASVO")
        INEFCT_pos        = line.find("INEFCT")
        NTACOP_pos        = line.find("NTACOP")
        NTCADJ_pos        = line.find("NTCADJ")
        PRLOSS_pos        = line.find("PRLOSS")
        PVWK0T_pos        = line.find("PVWK0T")
        PVWKPR_pos        = line.find("PVWKPR")
        RGLOSS_pos        = line.find("RGLOSS")
        ENTFLX_pos        = line.find("ENTFLX")
        #reading input
        if GAS_TEMP_COLD_pos != -1:
           GAS_TEMP_COLD =
line[GAS_TEMP_COLD_pos+p11:GAS_TEMP_COLD_pos+p12].rstrip()
        elif RG_LENGTH_pos   != -1:
              RG_LENGTH     = line[RG_LENGTH_pos +
p11:RG_LENGTH_pos + p12].rstrip()
        elif HERZ_pos        != -1:
                 HERZ         = line[HERZ_pos + p11:HERZ_pos + p12].rstrip()
        elif MASS_FLUX_COLD_pos!= -1:
                    MASS_FLUX_COLD=
line[MASS_FLUX_COLD_pos+p11:MASS_FLUX_COLD_pos+p12].rstrip()
        elif PRES_PHASE_pos    != -1:
                        PRES_PHASE    = line[PRES_PHASE_pos+p11-
1:PRES_PHASE_pos+p12].rstrip()
        #be careful with the position of numbers after this
        elif NUM_POINTS_X_pos  != -1:
                            NUM_POINTS_X  =
line[NUM_POINTS_X_pos+p11:NUM_POINTS_X_pos+p11+3].rstrip()
        elif NUM_STEPS_CYC_pos != -1:
```

```
                                NUM_STEPS_CYC =
line[NUM_STEPS_CYC_pos+p11:NUM_STEPS_CYC_pos+p11+3].rstrip()
        elif FINAL_CYCLE_pos   != -1:
                                FINAL_CYCLE   =
line[FINAL_CYCLE_pos+p11:FINAL_CYCLE_pos+p11+5].rstrip()
                                #changed to position type 2 as start reading output
        elif CFLMAX_pos        != -1:
                                CFLMAX        =
line[CFLMAX_pos+p21:CFLMAX_pos+p22].rstrip()
        elif EHTDIF_pos        != -1:
                                EHTDIF        =
line[EHTDIF_pos+p21:EHTDIF_pos+p22].rstrip()
        elif EHTFLX_pos        != -1:
                                EHTFLX        =
line[EHTFLX_pos+p21:EHTFLX_pos+p22].rstrip()
        elif ENGBAL_pos        != -1:
                                ENGBAL        =
line[ENGBAL_pos+p21:ENGBAL_pos+p22].rstrip()
        elif GTPNRM_pos        != -1:
                                GTPNRM        =
line[GTPNRM_pos+p21:GTPNRM_pos+p22].rstrip()
        elif EFFIC_pos         != -1:
                                EFFIC         =
line[EFFIC_pos+p21:EFFIC_pos+p22].rstrip()
        elif GASARS_pos        != -1:
                                GASARS        =
line[GASARS_pos+p21:GASARS_pos+p22].rstrip()
        elif GASVO_pos         != -1:
                                GASVO         =
line[GASVO_pos+p21:GASVO_pos+p22].rstrip()
        elif INEFCT_pos        != -1:
                                INEFCT        =
line[INEFCT_pos+p21:INEFCT_pos+p22].rstrip()
        elif NTACOP_pos        != -1:
                                NTACOP        =
line[NTACOP_pos+p21:NTACOP_pos+p22].rstrip()
        elif NTCADJ_pos        != -1:
                                NTCADJ        =
line[NTCADJ_pos+p21:NTCADJ_pos+p22].rstrip()
        elif PRLOSS_pos        != -1:
                                PRLOSS        =
line[PRLOSS_pos+p21:PRLOSS_pos+p22].rstrip()
        elif PVWK0T_pos        != -1:
                                PVWK0T        =
line[PVWK0T_pos+p21:PVWK0T_pos+p22].rstrip()
```

```
        elif PVWKPR_pos       != -1:
                                                                PVWKPR       =
line[PVWKPR_pos+p21:PVWKPR_pos+p22].rstrip()
        elif RGLOSS_pos       != -1:
                                                                  RGLOSS
= line[RGLOSS_pos+p21:RGLOSS_pos+p22].rstrip()
        elif ENTFLX_pos       != -1:
                                                                    ENTFLX
= line[ENTFLX_pos+p21:ENTFLX_pos+p22].rstrip()


    check = int(NUM_STEPS_CYC) #check if number of steps per cycle is
greater than 300
    #COP vs. mc,L,phase
    if NTACOP =="0":
      string = ""
    elif check <280:
      string = ""
    else:
      string = NTACOP+" "+GASARS+" "+RG_LENGTH+"
"+PRES_PHASE+"\n"
      #string = str(file_num) +" "+ GAS_TEMP_COLD+" "+RG_LENGTH+"
"+HERZ+" "+MASS_FLUX_COLD+" "+PRES_PHASE+"
"+NUM_POINTS_X+" "+NUM_STEPS_CYC+" "+FINAL_CYCLE+"
"+CFLMAX+" "+EHTDIF+" "+EHTFLX+" "+ENGBAL+" "+GTPNRM+"
"+EFFIC+" "+GASARS+" "+GASVO+" "+INEFCT+" "+NTACOP+"
"+NTCADJ+" "+PRLOSS+" "+PVWK0T+" "+PVWKPR+" "+RGLOSS+"
"+ENTFLX+" "+"\n"
    f_write.write(string)
    file_regen.close()
  except FileNotFoundError:
        print ("no file")
        #string = string + " " + "\n"
        #f_write.write(string)
f_write.close()
end = input("finish reading")
```

2. Matlab code

```
clear all
%use the curve fitting to find the optimal shell

[a,agc_data,len_data,pha_data,cop_data] = curve_fitting_v1;
cop_fitting                             =
guess_function(a,agc_data,pha_data,len_data);
```

```
temp    = size(len_data);
%compare curve fitting data to original data
x_plot  = linspace(1,temp(1,1),temp(1,1));
plot(x_plot, cop_data, 'color','k');
hold on
plot(x_plot, cop_fitting,'-.r');
hold off

%matrix setup of length, phase and agmc
len_max = max(len_data);         %maximal length of interest, [m]
len_min = min(len_data);         %minimal length of interest,[m]
num_len = 100;                   %number of different lengths of interest

pha_max = max(pha_data);          %maximal phase angle of interest, [deg]
pha_min = min(pha_data);          %minimal phase angle of interest, [deg]
num_pha = 100;           %number of different phases of interest

agc_max = max(agc_data);        %max specific cross sectional area,
[m2*s/kg]
agc_min = min(agc_data);          %max specific cross sectional area,
[m2*s/kg]
num_agc = 100;           %number of different agmc of interest

len = linspace(len_min,len_max,num_len);
pha = linspace(pha_min,pha_max,num_pha);
agc = linspace(agc_min,agc_max,num_agc);


cop = fitting(a,agc,pha,len);


cop_opt = max(cop(:));
toler_s = 0.98;
toler_l = 0.95;
err     = 1e-4;

nn      = 1;
cop_s   = cop_opt*toler_s;
cop_l   = cop_opt*toler_l;

% %to find the coordinates for the COP of interest
% for i = 1:num_len
%     for j = 1:num_pha
%         for k = 1:num_agc
%             compare = cop(i,j,k)-cop_exp;
%             indi    = compare;
%             if indi >0
%                 if compare<err
%                 len_plot(nn) = len(i);
%                 pha_plot(nn) = pha(j);
%                 agc_plot(nn) = agc(k);
%                 nn           = nn+1;
%                 end
```

```
%             else
%             end
%         end
%     end
% end
figure
p = patch(isosurface(agc,pha,len,cop,cop_s));
set(p,'FaceColor','black','EdgeColor','red');
view(3);
hold on
p = patch(isosurface(agc,pha,len,cop,cop_opt*0.95));
set(p,'FaceColor','none','EdgeColor','blue');
view(3);
hold off




%[f,v] = isosurface(agc,pha,len,cop,cop_exp);
%
% pha_3d = zeros(nn,nn);
% agc_3d = zeros(nn,nn);
% for i = 1:nn
%     pha_3d(i,:) = len_plot(i);
%     agc_3d(:,i) = agc_plot(i);
% end
% len_3d = zeros(nn,nn,nn);
% for i = 1:nn
%
%             len_3d(i,i,i) = len_plot(i);
% end









function [ y ] = guess_function( a,x1,x2,x3 )
    m = size(x1);
    n = m(1,1);
    y = zeros(n,1);
    %temp = 0;
    %x1 = agmc, x2= pha, x3 = len
    for i = 1:n
    part_len_1 = a(1)*x3(i)^2+a(2)*x3(i)+a(3);
    part_len_2 = a(19)*x3(i)^2+a(20)*x3(i)+a(21);
    %part_len_2 = a(14)*x3(i)+a(15);
    %part_len = a(7)*exp(a(8)*x3(i)+a(9));
```

```
    temp     = x2(i)-part_len_1;
    part_pha_1 = a(4)*temp^2 + a(5)*temp + a(6);
    part_pha_2 = a(7)*temp^2 + a(8)*temp + a(9);
    part_pha_3 = a(10)*temp^2+ a(11)*temp + a(12);
    part_pha_4 = a(13)*part_len_2*temp^2 + a(14)*part_len_2*temp + a(15);
    %part_agm = a(4) + a(5)*x1(i) + a(6)*temp +
a(7)*x1(i)^2+a(8)*temp*x1(i)+a(9)*temp^2 + a(10)*x1(i)^3 +
a(11)*x1(i)^2*temp + a(12)*x1(i)*temp^2 + a(13)*temp^3;
    part_agm =
a(16)*part_pha_1*x1(i)^3+a(17)*part_pha_2*x1(i)^2+a(18)*part_pha_3*x1(i)+p
art_pha_4;
    y(i)     = part_agm;
    end

end


function [a,agmc,len,pha,COP] = curve_fitting_v1()
%learning fminsearch

%read data from data read.txt file to data_in matrix
data_in   =  importdata('300Hz_data_for_fitting.xlsx');

%size of data_in matrix
num_input = size(data_in);

%number of data sets
num_ag    = num_input(1);

%give input datas to specific vector
COP       = data_in(:,1);
agmc      = data_in(:,2);
len       = data_in(:,3);
pha       = data_in(:,4);

%initial guess of the factors
n_factor     = 21;
factor_guess = ones(n_factor,1)*10;

%start iteration count with 0
iter = 0;
iter_max = 100;

%curve fitting function handle
hand = @(factor)curve_fitting(agmc,pha,len,COP,num_ag,factor);
fval = 1;

%change fminsearch max iteration options
fun_num = 1e12;  %max number of iterations
options = optimset('MaxFunEvals', fun_num);
%start curve fitting iteration
while iter <iter_max && fval >1e-5
```

```
    [a,fval]    = fminsearch(hand,factor_guess,options); %lsqnonlin or
fminsearch or fminunc
    %update factor guess, iterate to get a new factor
    factor_guess= a;
    iter        = iter + 1;
end

% %matlab native nonlinear fitting
% %matrix setup
% y    = COP;
% beta0= linspace(1,1,10);
% b    = linspace(1,1,10);
% x(:,1) = agmc;
% x(:,2) = len;
% x(:,3) = pha;
% %setup model function
% model_fun =
@(b,x)(b(1)*x(:,1).^3+b(2)*x(:,1).^2+b(3)*x(:,1)+b(4)*x(:,2).^3+b(5)*x(:,2
).^2+b(6)*x(:,2)+b(7)*x(:,3).^3+b(8)*x(:,3).^2+b(9)*x(:,3)+b(10));
% %do nonlinear fitting
% [beta,R,J,CovB,MSE,ErrorModelInfo] = nlinfit(x,y,model_fun,beta0);

done = 'done'
end
```