2020

Thermal-hydraulic Optimization of the Heat Exchange Between a Molten Salt Small Modular Reactor and a Super-critical Carbon Dioxide Power Cycle

James Sherwood
University of Central Florida

THERMAL-HYDRAULIC OPTIMIZATION OF THE HEAT EXCHANGE BETWEEEN A MOLTEN SALT SMALL MODULAR REACTOR AND A SUPERCRITICAL CARBON DIOXIDE POWER CYCLE

By

JAMES SHERWOOD

A thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mechanical and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring term
2020

ABSTRACT

The next generation of nuclear power sources, Gen. IV, will include an emphasis on small, modular reactor (SMR) designs, which will allow for standardized, factory based manufacturing and flexibility in the design of power plants by utilizing one or several modular reactor units in parallel. One of the reactor concepts being investigated is the Molten Salt Reactor concept (MSR), which utilizes a molten salt flow loop to cool the reactor and transfer heat to the power conversion cycle (PCS). Here, the use of a supercritical carbon dioxide (S-CO2) Brayton cycle is assumed for that PCS. The purpose of this thesis is to investigate the heat exchange between these two systems and to determine the suitability of a common heat exchanger concept, the shell-and-tube heat exchanger (STHE). A design algorithm was developed to determine the number of shells in series that are required to accommodate the heat duty and inlet/outlet fluid temperatures specified and to produce and thermal-hydraulically rate an efficient STHE design for the heat exchange system. A detailed discussion of heat exchanger analysis is presented, and the process of the algorithm is reported.

Dedicated to my late father
Jack Sherwood

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

List of Abbreviations

ASME	American Society of Mechanical Engineers
CSP	Concentrated Solar Power
EPRI	Electric Power Research Institute
FLiBe	$LiF - BeF_2$
HEX	Heat Exchanger
HTR	High Temperature Recuperator
LMTD	Log-Mean Temperature Difference
LTR	Low Temperature Recuperator
LWR	Light Water Reactor
MSR	Molten Salt Reactor
SMR	Small Modular Reactor
PCHE	Printed Circuit Heat Exchanger
PCS	Power Conversion System
RC	Recompression Cycle
S-CO_2	Supercritical Carbon Dioxide
STHE	Shell-and-Tube Heat Exchanger
TEMA	Tubular Exchanger Manufacturer's Association
T-s	Temperature – entropy

INTRODUCTION

Generation IV Nuclear Reactors

The next generation of nuclear power sources, Gen. IV, will include an emphasis on small, modular designs, which will allow for standardized, factory-based manufacturing and flexibility in the design of power plants by utilizing one or several modular reactor units in parallel. Small Modular Reactors (SMR's) are defined by their factory production capability. Rather than custom designing and/or in situ fabricating significant plant components, these reactor systems are intended to be highly standardized [1]. This is advantageous in assembly and maintenance and especially advantageous for certification, which can be a difficult obstacle to plant commissioning in some countries, such as the United States. Plants utilizing these reactors are envisioned with standalone units or in larger plants of multiple modular units. They may additionally be fit into such brownfield communities as in place of decommissioned coal-fired plants [1]. There are SMR's under development for all principal reactor types, however this proposal will concern itself with three of them [2].

The Molten Salt Reactor (MSR) is one of the Generation IV reactor concepts and is receiving interest principally for use with thorium or spent Light Water Reactor (LWR) fuel [3]. There are two categories in the MSR concept: the first is characterized by using a molten salt as the primary coolant; the second involves dissolving the nuclear fuel into the molten salt itself. The second necessitates an extra loop, denoted the intermediate loop, to separate radioactive material from non-radioactive [3]. This second design concept is particularly engaging, because it circumvents the need to manufacture solid fuel. In general, MSR's receive interest for their higher operating temperatures, yielding higher power cycle efficiency, and lower operating pressures, decreasing

risk of rupture failure. Figure 1 below shows an MSR design concept, from a U.S. Department of Energy report on Gen. IV reactor technology [4].

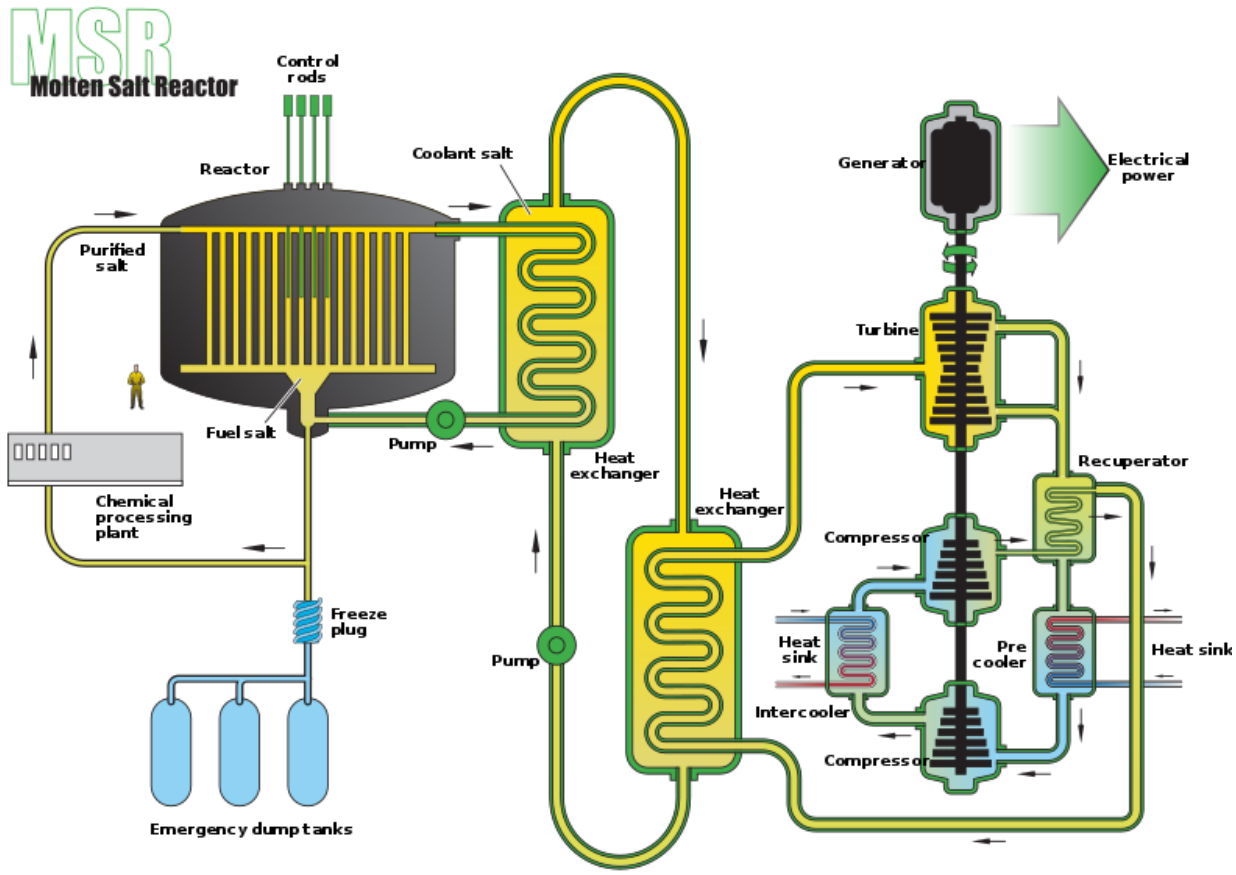


Figure 1: MSR Power Plant Design Concept [4]

Supercritical Carbon Dioxide

The supercritical carbon dioxide Brayton power cycle has received growing interest in the preceding decades for use in nuclear, concentrated solar power (CSP), geothermal, and other applications [5][6][7]. The attractive features of this cycle include its high efficiency and low turbomachinery capital. Supercriticality is defined for a fluid as being above the critical

temperature and pressure. Below the critical point these fluids transition from liquid to gas linearly; above, fluids exhibit some qualities of both liquids and gases. Significantly, carbon dioxide above the critical point has a near-liquid density. The cycle is considered supercritical because some/all processes (depending on the particular variation of the cycle) take place above the critical temperature and pressure, 31.1°C and 7.39 MPa, respectively. When the compression process occurs near and above the critical point, much smaller machinery to achieve the same work [8]. Figure 2 illustrates the scale of the reduction.

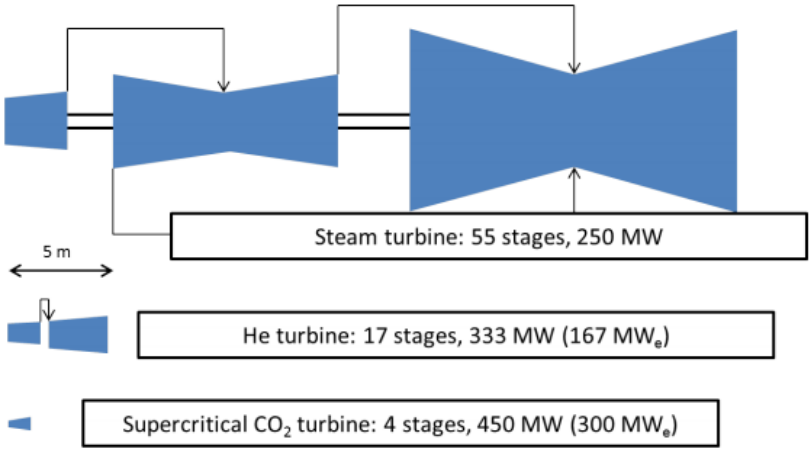


Figure 2. Turbines for S-CO2 Cycles [8]

Table 1: Re-compression S-CO_2 cycle – Parameters

Recompression cycle	600 MW case	30 MW case	Units
Pressure ratio	2.55		
Turbine inlet pressure	20		MPa
Turbine inlet temperature	550		ºC
Compressor inlet temperature	32		ºC
Mass flow rate	3176.3	127.052	kg/s
Cycle efficiency	37.62		%
Net power	232.3	9.3	MW

The cycle requires high turbine inlet temperatures to achieve the desired efficiency. For this reason, an important component of the S-CO_2 cycle is heat recuperation. The Recompression Cycle (RC) is a common variation of the S-CO_2 Power Conversion System (PCS), for which the cycle layout and T-s diagram are given in Figures 3 to 4 [9].

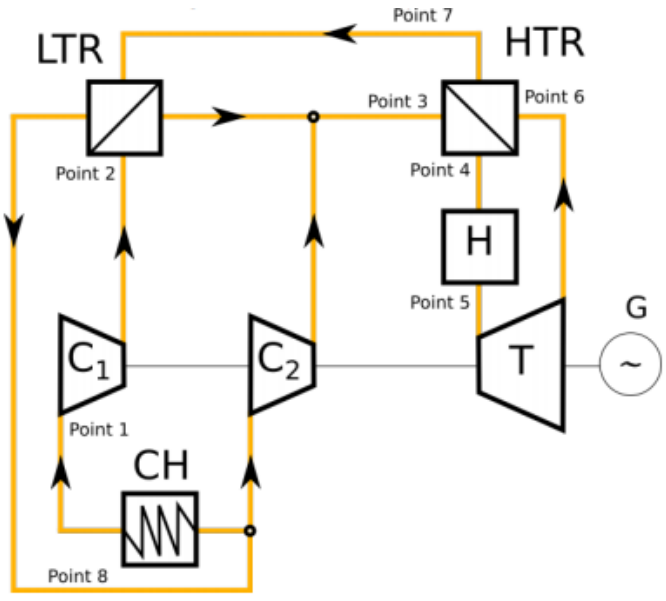


Figure 3: S-CO2 RC Cycle [9]

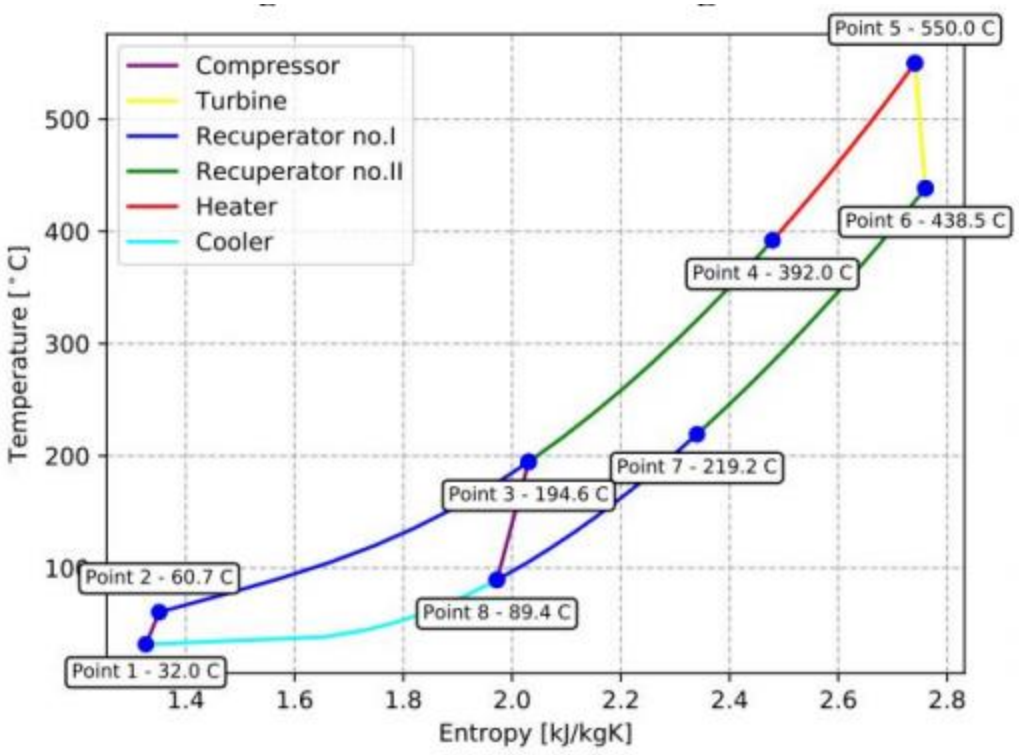


Figure 4: S-CO2 RC T-s Diagram

This cycle includes a High Temperature Recuperator (HTR) and a Low Temperature Recuperator (LTR), which is due to the fact that the specific heat of the cold side is approximately two times greater than that of the hot side. The basic parameters from optimization of the Re-compression S-CO_2 cycle are shown in Table 1. The results are for the reference case and SMR – 30 MW_t reactor. Splitting the flow between the two Recuperators reduces waste heat and thereby improves the thermal efficiency [10]. The box labeled CH in Figure 2 stands for Cooler/Chiller. There is much research activity in CO_2 heat exchange, as well as industry innovation. An example is Printed Circuit Heat Exchangers (PCHE's), in which channels of varying geometry are chemically etched into metals plates, which are subsequently diffusion bonded. Although these heat exchangers can have extraordinary surface area density, Heatric advertises on the order of 1300 m^2/m^3 [11], some

disadvantages include potential blockage effects and the requirement for extreme fluid cleanliness. Printed circular Heat exchanger (PCHE's) are also highly capital intensive compared to more traditional design concepts, like shell-and-tube (STHE).

THEORY

Heat Exchangers

Baffles

A very detailed handbook of calculations and design notes for baffled STHE's is given by Taborek [12]. Baffling a HEX results in several sections of near perfect crossflow between the baffles, assuming they are spaced appropriately, and parallel or counterflow in the baffle windows. This flow pattern is depicted below in Figure 5, courtesy of Taborek [12].

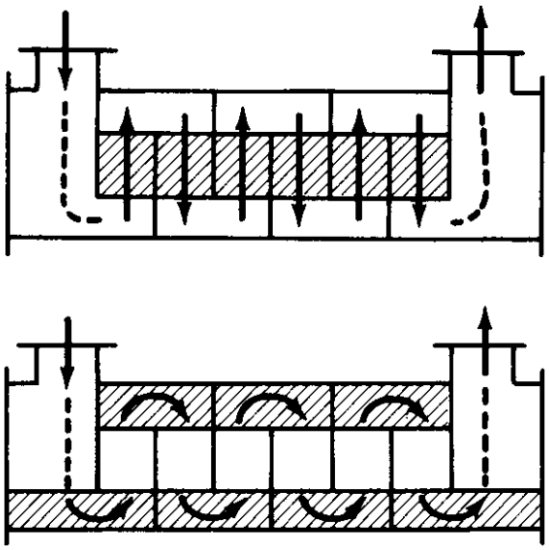
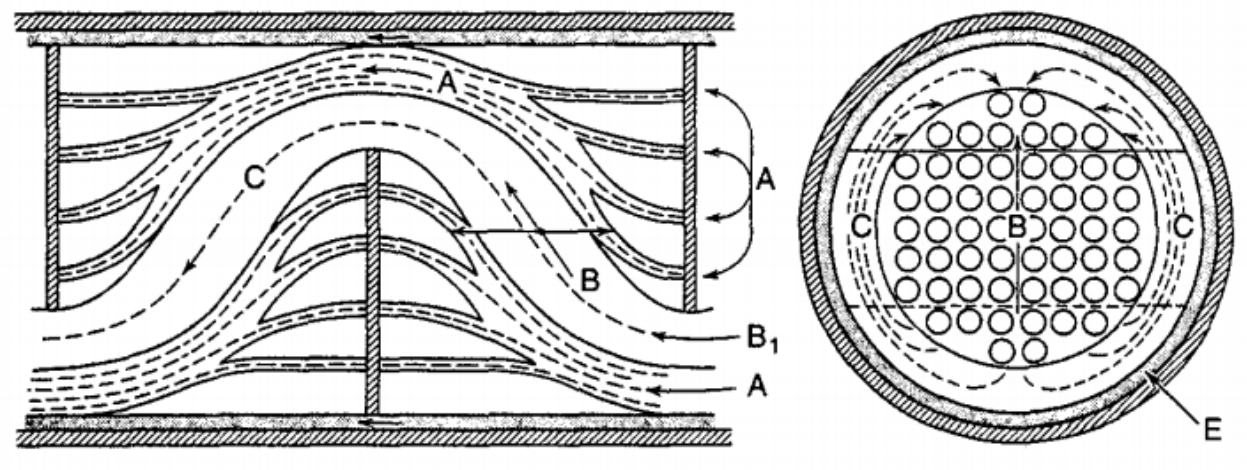
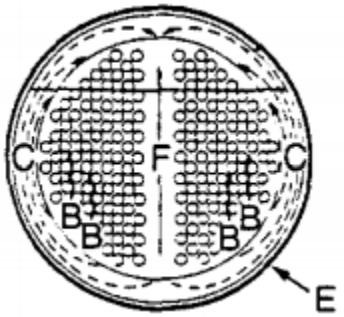


Figure 5: Baffled HEX Crossflow (Top) and Window flow (Bottom) [12]

Multiple flow regions exist around the baffles, and they each contribute to the rating calculations differently. Figure 6 below illustrates those flow regions, courtesy of Serna [13].



(a)



(b)

Figure 6: Baffle Flow Regions [13]

A = tube baffle leakage

B = crossflow

C = crossflow bypass

E = shell-baffle leakage

F = tube pass partition bypass

Inlet/outlet spacings are greater than or equal to the central baffle spacing to accommodate the flow developing through the inlet/outlet [14]. TEMA advises that baffle spacing be kept less than or equal to 10% of the shell diameter [15]. Baffle cut has been shown to be ideal between 20-45%, where baffle cut B_c is defined according to Equation 1. L_b is the length of the baffle.

$$B_c = (L_b/D_s)*100 \quad (\%) \quad (1)$$

Taborek graphically defines several geometric variables according to Figure 7 below. These are useful in certain calculations related to STHE rating, which will be detailed in the next section of this chapter.

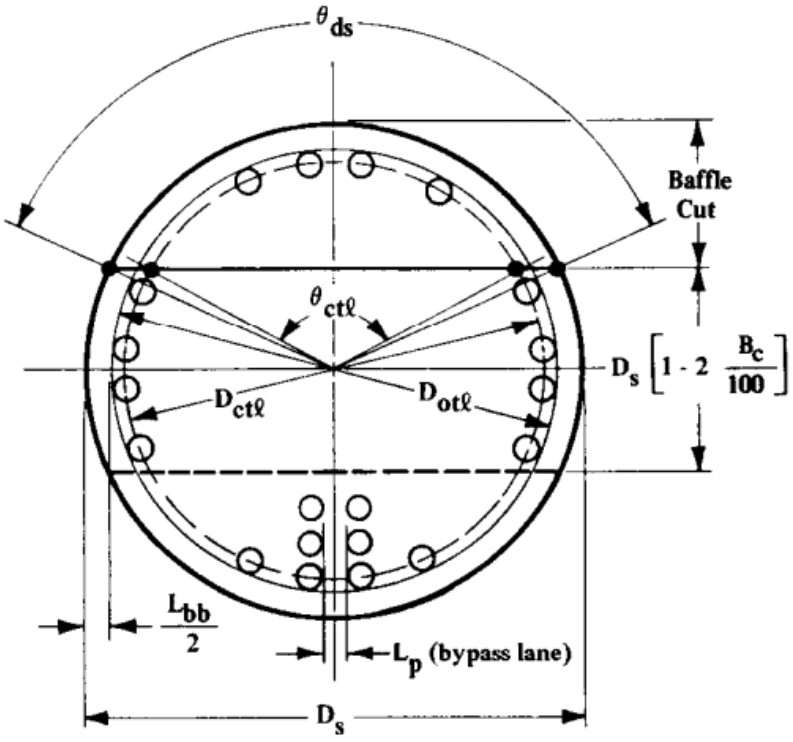


Figure 7: Inside Shell Diameter Geometrical Definitions [12]

D_s = inner shell diameter

D_{otl} = tube bundle-circumscribed circle

D_{ctl} = outermost tube center circle

L_{bb} = shell-to-tube bundle bypass clearance

Θ_{ctl} = tube bundle-circumscribed circle centri-angle

Tubes

There are two general options for tube layout, either square or triangular; this layout can additionally be rotated 45^o from the horizontal. The triangular layout yields the greatest tube density, and it therefore assumed here. Kakac advises that the triangular layout is default [14]. For simplicity, a 0^o from the horizontal orientation is assumed. This layout is depicted in Figure 8, for the region between baffles when crossflow is achieved. Here $\Theta_{tp} = 30^o$.

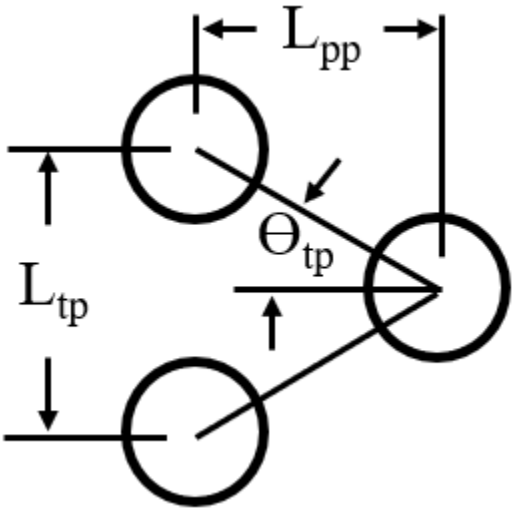


Figure 8: Triangular Tube Layout in Crossflow [12]

Tube thickness is guided by the 2010 ASME Boiler and Pressure Vessel Code [16]. Here the tube is treated as a cylindrical pressure vessel and should be designed to uphold integrity even without the outside pressure of the shell side fluid. Equation 2 defines the requisite thickness t_t for the tubes [16]:

$$t_t = (P_t * D_t) / ((2 * \tau_{allow}) + P_t) + 0.005 * D_t \quad \text{(mm)} \qquad (2)$$

where P_t is tube side operating pressure, D_t is the tube diameter, and τ_{allow} is the tube material allowable stress.

The effective tube length L_{ta} is given in Equation 3 in terms of the number of baffles N_b, the central baffle spacing L_b, and the inlet/outlet baffle spacings, L_{bi} and L_{bo}, respectively [12]. For simplicity, the inlet and outlet baffle spacings are taken as equal: that is, $L_{bi} = L_{bo}$.

$$L_{ta} = (N_b - 1)L_{bc} + (L_{bi} + L_{bo}) \quad \text{(mm)} \qquad (3)$$

The number of tubes is given by Equation 4, in which ψ_n is a corrective factor that accounts for the number of tube passes [12], C_l is a constant that accounts for the tube layout [12], and L_{tp} is the tube pitch [12]. ψ_n is a function of D_s and tube pass number N_{tp} [12]. For a triangular arrangement $C_l = 0.866$ [12].

$$N_t = (\pi/4C_l)(D_{ctl}/L_{tp})^2(1 - \psi_n) \qquad (4)$$

Heat Transfer

Heat Transfer Surface Area

The heat transfer surface area A_o is defined by the heat exchanger design equation [14]. Here Q is the heat duty of the HEX, LMTD is the log-mean temperature difference [14], and R_{dw} is the combined resistance of the tube wall and fouling factors [12].

$$A_o = (Q/F_T\,LMTD)(R_{dw} + (1/h_s) + (D_t/D_{ti}\,h_t)) \quad \text{(mm}^2\text{)} \qquad (5)$$

Temperature Correction Factor

F_T is a corrective factor that accounts for the flow not being in pure counterflow, as it is in a simple double-pipe HEX [14]. For STHE's F_T can decrease drastically with an improper design. F_T is commonly defined by the dimensionless parameters P [14] and R [14] with definitions given by Equations 6 and 7. Wales proposed the further parameter G, Equation 8 [17].

$$P = (T_{c,\,out} - T_{c,\,in})/(T_{h,\,in} - T_{c,\,in}) \quad (6)$$

$$R = (T_{h,\,out} - T_{h,\,in})/(T_{c,\,out} - T_{c,\,in}) \quad (7)$$

$$G = (T_{h,\,out} - T_{c,\,out})/(T_{h,\,in} - T_{c,\,in}) = 1 - P(1 + R) \quad (8)$$

Vengateson used G to perform modularity analysis on generic E and F shell STHE's, shown for the E shell type in Figure 8 [17]. He also provides equations for F_T for both shell types, below as Equations 9 and 10 [17].

$$F_{T,\,E} = \frac{\sqrt{1 + R^2}\ln((1 - P)/(1 - P\,R))}{(R - 1)\ln\left\{\frac{[2 - P(R + 1 - \sqrt{1 + R^2})]}{[2 - P(R + 1 + \sqrt{1 + R^2})]}\right\}} \quad \text{(E shell)} \quad (9)$$

$$F_{T,\,F} = \frac{\left[\frac{\sqrt{1 + R^2}}{2(R - 1)}\ln((1 - P)/(1 - P\,R))\right]}{\ln\left\{\frac{[2/P - 1 - R + (2/P)\sqrt{(1 - P)(1 - P\,R)} + \sqrt{1 + R^2}]}{[2/P - 1 - R + (2/P)\sqrt{(1 - P)(1 - P\,R)} - \sqrt{1 + R^2}]}\right\}} \quad \text{(F shell)} \quad (10)$$

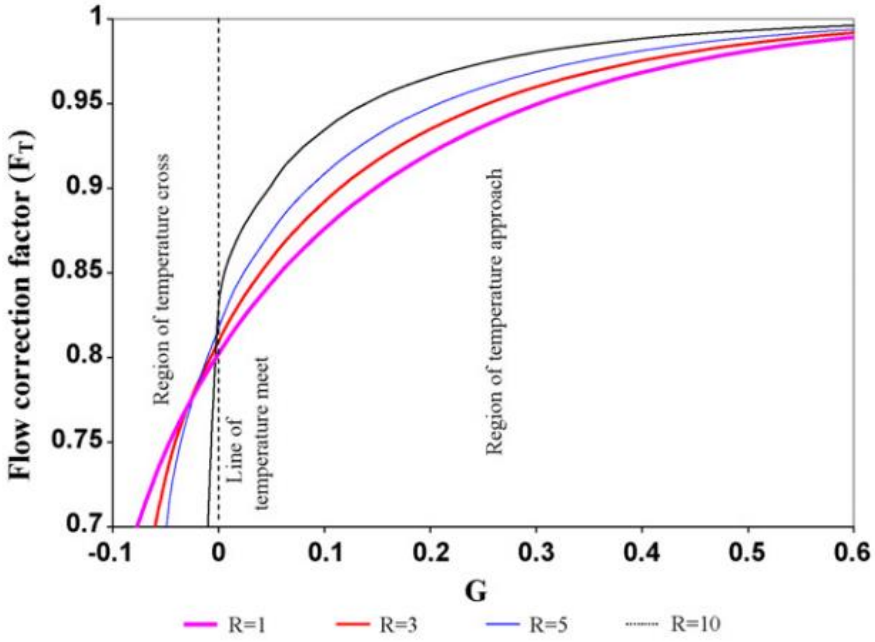


Figure 9: F_T Correction Factor for E Shell Based on R and G [17]

Shell Side Flow

The average shell side flow velocity v_s depends on the geometry of the shell side and the volumetric flow rate $Л_s$ [13]. Since the mass flow rate is fixed in order to achieve the desired heat duty, volumetric flow rate can be determined by Equation 11. Then Equation 12 yields the average flow velocity [13] This velocity will be used to calculate the heat transfer coefficient of the shell side.

$$Л_s = \dot{m}/\rho \qquad m^3/s \qquad (11)$$

$$v_s = Л_s/[10^{-6}L_b(L_{bb} + D_{ctl}/L_{tp}(L_{tp} - D_t))] \qquad (m/s) \qquad (12)$$

The shell side Reynolds number Re_s can be calculated using a formulation specific to baffled STHE's, expressed below in Equation 13; it is necessary to calculate a parameter S_m.

$$S_m = L_b(L_{bb} + ((D_{ctl}/L_{tp})(L_{tp} - D_t)) \quad m^2 \quad (13)$$

$$Re_s = (D_t/\mu_s)(\dot{m}/S_m) \quad (14)$$

Bell-Delaware Method

The Bell-Delaware method is the most established and reliable method for determining film heat transfer coefficients h_s and pressure drops ΔP_s on the shell side of an STHE [12] [13] [14]. Input parameters are averaged across the shell, and the outputted heat transfer coefficient is also a shell-side average [12]. The actual coefficient h_s is given by Equation 15 in terms of the ideal crossflow coefficient h_{si} [12]. S_{sb} is the shell-to-baffle leakage area [12], S_{tb} is the tube-baffle hold leakage area [12], S_m is the crossflow area at the bundle centerline [12], and r_{lm} [12] and r_s [12] are correlational parameters. C_{bh} is a corrective factor [12]; F_{sbp} is the ratio of bypass to cross-flow area [12].

$$h_s = h_{si}(J_c J_l J_b J_r J_s) = h_{si}(J_{tot}) \quad W/m^2\,K \quad (15)$$

$$J_c = 0.55 + 0.72[1 - 2((\Theta_{ctl}/360) - \sin(\Theta_{ctl})/2\pi)] \quad (16)$$

$$J_l = 0.44(1 - r_s) + [\,1 - 0.44(1 - r_s)]\exp(-2.2r_{lm}), \quad (17)$$

$$\text{Where } r_s = S_{sb} / (S_{sb} + S_{tb}), \quad (18)$$

$$\text{And } r_{lm} = (S_{sb} + S_{tb}) / S_m \quad (19)$$

$$J_b = \exp[-C_{bh} F_{sbp}(1 - (2r_{ss})^{1/3}], \quad (20)$$

$$\text{Where } C_{bh} = 1.25 \quad Re > 100 \quad (21)$$

$$\text{And } F_{sbp} = S_b / S_m \quad (22)$$

$$J_r = 1 \quad Re > 100 \quad (23)$$

$$J_s = [(N_b - 1) + (L_{bi} / L_{bc})^{1-n} + (L_{bo} / L_{bc})^{1-n}\,] / [(N_b - 1) + (L_{bi} / \quad (24)$$

$$L_{bc}) \quad + (L_{bo} / L_{bc})],$$

$$\text{Where } n = 0.6 \qquad Re > 100 \quad (25)$$

All of the J constants are non-dimensional corrective factors [12]. J_c corrects pure crossflow ideal heat transfer for the effects of baffle window flow [12]. J_l corrects for baffle leakage effects [12]. J_b corrects for tube bundle bypass effects [12]. J_r corrects for the effect of an adverse temperature gradient developing through the boundary layer during deep laminar flow (i.e. Re < 100), hence the value of 1 for non-deep laminar Reynolds numbers [12]. J_s corrects for the flow effects of unequal inlet/outlet baffle spacing [12]. The ideal crossflow heat transfer coefficient h_{si} is given by equation 26 [13]. ϕ_s is a corrective factor for the variation of viscosity at the wall temperature μ_{sw} from the value at the bulk temperature μ_s [12].

$$h_{si} = (\phi_s \, k_s \, j_{si} \, Re_s \, Pr_s^{1/3}) / D_t, \qquad W/m^2 K \quad (26)$$

$$\text{Where } \phi_s = (\mu_s / \mu_{sw})^{0.14} \quad (27)$$

The pressure drop on the shell side is given by Equation 28 [12]. ΔP_{bi} is the pressure drop in the baffle sections [12]; ΔP_{bi} is the pressure drop in the baffle window sections [12]. All of the R constants are again corrective factors [12]; the subscripts align with those of the J constants enumerated above. N_{tcc} is the number of effective tube rows crossed in one crossflow section [12]; N_{tcw} is the number of effective tube rows crossed within each window [12]. L_{pp} is the effective tube row distance in the flow direction [12].

$$\Delta P_s = [(N_b - 1)R_b \, R_l + (1 + N_{tcw} / N_{tcc})R_b \, R_s] \, \Delta P_{bi} + N_b \, \Delta P_{wi} \, R_l, \qquad Pa \quad (28)$$

$$\text{Where } N_{tcw} = (0.8 / L_{pp})[D_s \, (B_c / 100) - (D_s - D_{ctl}) / 2] \quad (29)$$

$$\text{And } N_{tcc} = (D_s / L_{pp})[1 - 2(B_c/100)] \quad (30)$$

$$R_l = \exp[-1.33(1 + r_s)(r_{lm})^p], \quad (31)$$

$$\text{Where } p = [-0.15(1 + r_s) + 0.81] \quad (32)$$

$$R_b = \exp[-C_{bp} F_{sbp}(1 - (2r_{ss})^{1/3}], \quad (33)$$

$$\text{Where } C_{bp} = 3.7 \qquad Re > 100 \quad (34)$$

$$R_s = (L_{bc} / L_{bo})^{2-n} + (L_{bc} / L_{bi})^{2-n}, \quad (35)$$

$$\text{Where } n = 0.2 \qquad Re > 100 \quad (36)$$

Compact Delaware Formulation

Serna et. al. derived a compacted formulation of the Bell Delaware method to find shell and tube side heat transfer coefficients and pressure drops as functions of flow velocity [13]. This formulation allows for the creation of a swift STHE design and/or rating algorithm. For the shell side, h and ΔP are given by Equations 37 and 38 [13]. Equations 39-52 [13] complete the compact formulation.

$$\Delta P_s = K_s A_o (h_s)^m \qquad Pa \quad (37)$$

$$h_s = K_{s1}(v_s)^{1-r\_h} \qquad W/m^2K \quad (38)$$

$$K_s = K_{s4} K_{s5} / (K_{s1})^m \quad (39)$$

$$K_{s5} = \{4C_l / D_t \jmath_s(1 - \psi_n)\}\{L_{tp} / \pi D_{ctl}\}\{D_s(N_b + 1)L_{bc} / [(N_b - 1)L_{bc} + L_{bi} + L_{bo}]\}\{L_{bb} + D_{ctl}[(L_{tp} - D_t) / L_{tp,\,eff}]\} \quad (40)$$

$$K_{s4} = K_{s2}(v_s)^{r\_p'} + K_{s3}(v_s)^{r\_p' - r\_p} \quad (41)$$

$$K_{s3} = \{R_l[(N_b - 1) / (N_b + 1)] + R_s[(N_{tcc} + N_{tcw}) / N_{tcc}(N_b + 1)]\} \{[1 - 2(B_c / 100][2c_p R_b \rho_s / \phi_s L_{pp}][\mu_s / D_t \rho_s]^{r\_p}\} \quad (42)$$

$$K_{s2} = (S_m / S_w)[(1 + 0.3N_{tcw})R_l N_b \rho_s / (N_b + 1)D_s] \quad (43)$$

$$K_{s1} = (\phi_s c_h k_s Pr_s^{1/3} / D_t)(D_t \rho_s / \mu_s)^{1 - r\_h} J_{tot} \quad (44)$$

$$m = (3 - r_p') / (1 - r_h) \quad (45)$$

$$r_p' = r_p / (K_{s2} / K_{s3}(v_s)^{-r\_p} + 1) \quad (46)$$

$$r_h = -a_2 \quad \text{Table 2} \quad (47)$$

$$r_p = -b_2 \quad \text{Table 2} \quad (48)$$

$$c_h = a_1(1.33D_t / L_{tp})^a \quad (49)$$

$$c_p = b_1(1.33D_t / L_{tp})^b \quad (50)$$

$$a = a_3 / (1 + 0.14(Re_s)^{a\_4}) \quad (51)$$

$$b = b_3 / (1 + 0.14(Re_s)^{b\_4}) \quad (52)$$

The a and b constants are correlational coefficients for the ideal tube bank factors, respectively; the definitions for these factors are built into the compressed formulation above and so are not included here. Values for a_1, a_2, a_3, a_4, b_1, b_2, b_3, and b_4 are given for varying shell side Re and for different tube layout angles below in Table 2 [12].

Table 2: Correlational coefficients for ideal tube bank factors [12]

Layout angle	Re_s	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4
30°	10^5-10^4	0.321	-0.388	1.45	0.519	0.372	-0.123	7.00	0.500
	10^4-10^3	0.321	-0.388			0.486	-0.152		
	10^3-10^2	0.593	-0.477			4.57	-0.476		
	10^2-10	1.360	-0.667			45.1	-0.973		
90°	10^5-10^4	0.370	-0.395	1.187	0.370	0.391	-0.148	6.30	0.378
	10^4-10^3	0.107	-0.266			0.0815	0.022		
	10^3-10^2	0.408	-0.460			6.090	-0.602		
	10^2-10	0.900	-0.631			32.10	-0.963		

Modularity

It may become necessary to connect multiple HEX shells in series in order to accomplish the design goal; managing temperature cross is the relevant example. In this case, the overall temperature profiles for the heat exchange system may be sectioned into individual shells, with each shell seeing incremental input/output temperatures for both fluids. A simple method of sectioning the

total heat exchange is to choose each shell to achieve temperature approach. This method may be visualized readily by Figure 10 below:

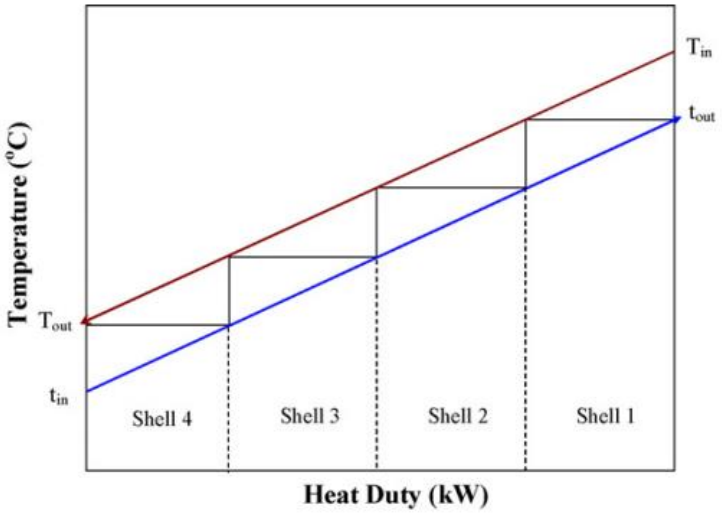


Figure 10: Graphical Method of Sectioning Temperature Profiles for Shells in Series [17]

However, while this method is suitable for preliminary estimations, a more robust analytical method was adopted here, based on equations 6-10. A desirable value of F_T is chosen, from which values for R and P are back-calculated. Figure 8 shows a steep region of F_T. For E shells and F shells, a value of F_T of 0.75 and 0.9, respectively, are desired to avoid this region [17]. F_T falling sharply may be thought of roughly as a stand-in for temperature efficiency, since the heat transfer area required tends to ∞ as F_T tends to $-\infty$ [17]. In order to determine the requisite number of shells N_s, the maximum possible value of P for the given R, P_{max}, the minimum possible value of G for the same, G_{min}, and a further dimensionless parameter X_P must be calculated. Vengateson provides expressions for these parameters, in Equations 53 - 59 for E and F shells below [17].

$$P_{max} = 2 / (R + 1 + \text{sqrt}(1 + R^2)) \quad \text{(E shell)} \quad (53)$$

$$P_{max} = \{[2(1 + R)(1 + R^2)] - 2*\text{sqrt}[(1 + R^2)(R^4 - 2R^3 + 3R^2 - 2R + 1)]\} / (4R^3 - R^2 + 4R) \quad \text{(F shell)} \quad (54)$$

$$G_{min} = 1 - (P_{max}(1 + R)) \quad (55)$$

$$Y = G - G_{min} \quad (56)$$

$$X_P = 1 - \{[Y((1 + R) + \text{sqrt}(1 + R^2))] / (2(1 + R)\} \quad (57)$$

$$N_s = P(1 - X_P*P_{max}) / (X_P*P_{max}(1 - P)) \quad R = 1 \quad (58)$$

$$N_s = \text{Ln}[(1 - R\,P) / (1 - P)] / \ln[(1 - R\,X_P\,P_{max}) / (1 - X_P\,P_{max})] \quad R \neq 1 \quad (59)$$

Once the number of shells has been calculated, the inlet/outlet temperatures seen by each shell may be determined by marching across the nodes of the temperature profiles. These temperatures are given recursively below, where N indexes the shell number, in Equations 60 – 65 [17]. It should be noted that, since the hot and cold streams enter from opposite ends of the HEX, $T_{h,\,i}$ occurs at the same node as $T_{c,\,o}$.

$$T_{h,\,i}[N] = T_{h,\,i} + P(N - 1)(T_{c,\,i}) / (1 - P) \quad R = 1 \quad (60)$$

$$T_{h,\,o}[N] = T_{h,\,i} + P(N)(T_{c,\,i}) / (1 - P) \quad R = 1 \quad (61)$$

$$T_{h,\,i}[N] = [T_{h,\,i} - [(R\,T_{c,\,i}) - T_{h,\,i}) / (R - 1)][(1 - (P\,R)) / (1 - P)]^{(N - 1)} + [((R\,T_{c,\,i}) - T_{h,\,i}) / (R - 1)] \quad R \neq 1 \quad (62)$$

$$T_{h,\,o}[N] = [T_{h,\,i} - [(R\,T_{c,\,i}) - T_{h,\,i}) / (R - 1)][(1 - (P\,R)) / (1 - P)]^{(N)} + [((R\,T_{c,\,i}) - T_{h,\,i}) / (R - 1)] \quad R \neq 1 \quad (63)$$

$$T_{c,\,i}[N] = ((P\,R - 1)\,T_{h,\,o}[N_s\text{-}N] + T_{h,\,o}[N_s\text{+}1\text{-}N]) / (P\,R) \quad (64)$$

$$T_{c,\,o}[N] = ((P\,R - 1)\,T_{h,\,i}[N_s\text{-}N] + T_{h,\,i}[N_s\text{+}1\text{-}N]) / (P\,R) \quad (65)$$

Finally, a decision had to be made as to how to design the shells that these inlet/outlet temperatures were to be passed into. It was desirable in approaching the design of a heat exchange system involving a series of shells to standardize the geometry of each shell, so as to avoid the cumbersome task of optimizing several distinct geometries and requiring a manufacturer to produce several distinct units. Four choices presented themselves: (1) to use a local method, involving precisely the outcome mentioned above, where each temperature bracket is passed into a design algorithm; (2) to use an upper-bound method, where the geometry optimized for the uppermost temperature bracket is replicated for each other shell and where the performance of each shell would be slightly different; (3) to use the midmost temperature bracket; and (4) to use the lowermost temperature bracket. These different temperature brackets are depicted visually below in Figure 10.

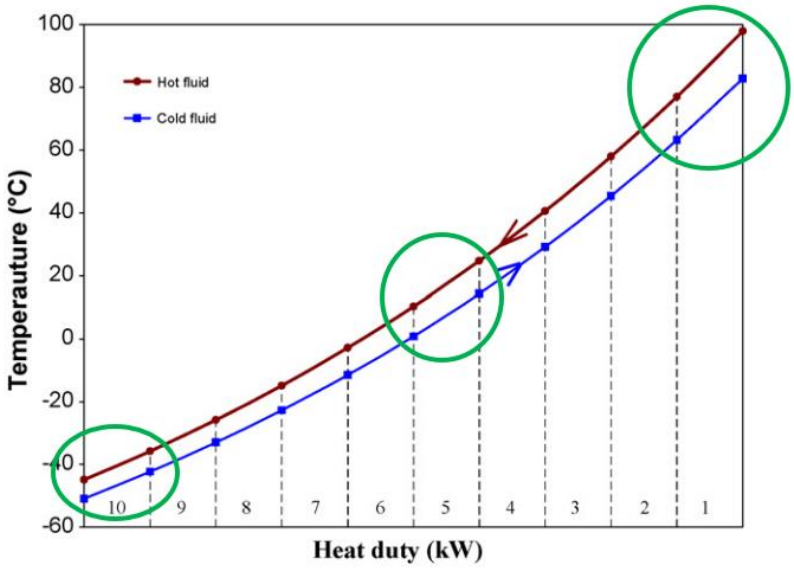


Figure 11: Upper/Mid/Lowermost Temperature Brackets for Shells in Series [17]

METHODOLOGY

Input parameters

An Electric Power Research Institute (EPRI) Technology Assessment of a Molten Salt Reactor Design [18] written in conjunction with FliBe Co. was used as a source for the operating parameters of the HEX and for the selection of a molten salt for the coolant. In this report, the heat exchanger for heat transfer to the power conversion system (PCS) is shown in Figure 12 as the Gas Heater, with LiF-BeF2 flowing in the hot side from 6 to 7 and CO2 flowing in the cold side from 19 to 8. The molar concentration ratio for this salt is 67-33% LiF-BeF_2, respectively [18]. The acronym FLiBe is used generally in research to represent this molten salt [19] [20] [21]; hereafter, FLiBe will refer to the molten salt, not the company. The technology assessment provides a good reference case for the design of an MSR power plant.

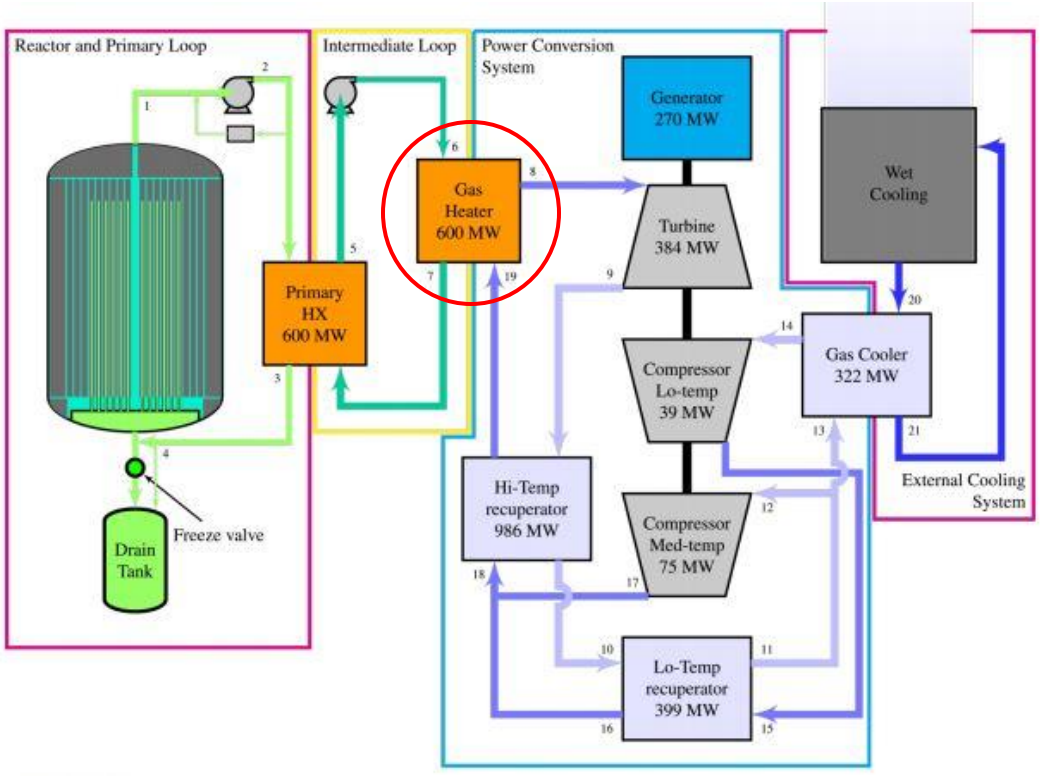


Figure 12: Schematic of FLiBe Co. MSR Power Plant [18]

CoolProp software was used to calculate the thermophysical and transport properties of CO_2. It was compared to NIST REFPROP, and the results showed very good agreement [22] [23]. At each iteration, CoolProp was called in the code to calculate density, thermal conductivity, dynamic viscosity, and the Prandtl number for CO_2. For calculating heat transfer, it should be noted that, since the specific heat of CO_2 can vary dramatically over small temperature changes, heat transfer was found by taking the difference in enthalpy, with enthalpies calculated by CoolProp. This avoids the drop in accuracy that using specific heats might entail.

There does not exist any thermophysical property library for FLiBe in a software platform like CoolProp or NIST REFPROP [22] [23], so a molten salt property function was written in Python programming language [24] with correlations for the necessary properties chosen from the literature for FLiBe [25]. A 2013 Molten Salt Thermophysical Property Database by Idaho

National Laboratory gives correlations for density, thermal conductivity, and dynamic viscosity of FLiBe, as determined experimentally by Ignat'ev et al. [20], Williams et al. [20], and Allen [20]. These are enumerated in Table 3 below. Specific heats for molten salts are difficult to measure experimentally; a 2006 assessment of molten salts for Gen. IV reactors by Oak Ridge National Laboratory states that, "The variation of [specific] heat capacity with temperature is small and is typically neglected during preliminary calculations. The temperature variation was not resolved within the accuracy of most previous measurements [19]." Nevertheless, a correlation for specific heat was found by Williams et. al. [19] and was used here. All of the properties found using these correlations (Equations 66 – 70) showed good agreement with those used by FliBe Co. in the Technology Assessment [18]. Sample calculations for those correlations at 700°C are provided in Table 3. MM is the molar mass of FLiBe.

$$\rho_{FLiBe} = 2518 - 0.406\,T \text{ , if } T \leq 923 \text{ K} \quad (kg/m^3) \quad (66)$$

$$\rho_{FLiBe} = 2763.7 - 0.0687\,T \text{ , if } T > 923 \text{ K} \quad (67)$$

$$\mu_{FLiBe} = 0.000116 \exp(3755 / T) \quad (Pa.s) \quad (68)$$

$$k_{FLiBe} = .0005\,T + (32 / MM) - 0.34 \quad (W/m.K) \quad (69)$$

$$C_{P,\,FLiBe} = 976.78044 + (1.06344\,T) \quad (J/K) \quad (70)$$

Table 3: Sample FLiBe Properties at 700°C [18]

ρ (kg/m3)	2696.8
μ (Pa.s)	0.0054983
k (W/m.K)	1.11627
C_P (J/K)	2011.7

Molten salts have corrosive qualities, for which multiple alloy solutions have been investigated. Nickel alloys, such as Hastelloy-N [26] [28] and Inconel 625 [21], have shown good corrosion resistance, including up to temperatures of 700°C [21], and have for this reason received research interest for MSR applications. The thermal conductivities for these alloys were included in the property function described above. The value of thermal conductivity for a given tube material was passed into the main, and the results were compared for each. Haynes International [26] provides thermal conductivity data for Hastelloy-N, which are shown in Table 4, as well as for Inconel 625, in Table 5. A correlation for the thermal conductivity of stainless steel is given by Wiley [27], below as Equation 71.

Table 4: Hastelloy-N Thermal Conductivity Data [26]

$T \leq 473$	K	k = 14.4	W/m.K
$473 < T \leq 573$		16.5	
$573 < T \leq 673$		18.0	
$673 < T \leq 773$		20.3	
$T > 773$		23.6	

Table 5: Inconel 625 Thermal Conductivity Data [26]

$T \leq 373$	K	k = 9.8	W/m.K
$373 < T \leq 473$		10.9	
$473 < T \leq 573$		12.5	
$573 < T \leq 673$		13.9	
$673 < T \leq 773$		15.3	
$773 < T \leq 873$		16.9	
$873 < T \leq 973$		18.3	
$973 < T \leq 1073$		19.8	
$1073 < T \leq 1173$		21.5	
$1173 < T \leq 1273$		23.4	
$T > 1273$		25.6	

$$k = 14.6 + 1.27*10^{-2}*T \quad \text{(W/m.K)} \quad (71)$$

Design Algorithm

A design algorithm was created to take the input parameters described above and output a complete STHE system design, including the number of shells in series, the geometry of each shell, its heat transfer rating, and the pressure drops across both sides. This algorithm was developed based on the example provided by Serna, based on the compact formulation of the Delaware method [13].

Step 1: Define Input Parameters

The required input parameters included the inlet/outlet fluid temperatures, fluid property banks or correlations, mass flowrates, operating pressures, allowable pressure drops. It was also necessary to specify a few geometrical input parameters. All of the requisite input parameters are listed below in Table 6.

Table 6: STHE System Design Algorithm Required Input Parameters

Parameter	Variable	Units
Shell side inlet temperature	T_{si}	K
Shell side outlet temperature	T_{so}	K
Tube side inlet temperature	T_{ti}	K
Tube side outlet temperature	T_{to}	K
Shell side mass flow rate	$\dot{m}_s$	kg/s
Tube side mass flow rate	$\dot{m}_t$	kg/s
Shell side operating pressure	P_s	kPa
Tube side operating pressure	P_t	kPa
Shell side allowable pressure drop	ΔP_s	kPa
Tube side allowable pressure drop	ΔP_t	kPa
Shell side fouling factor	R_{ds}	K/m^2W
Tube side fouling factor	R_{dt}	K/m^2W
Tube outer diameter	D_t	mm
Tube pitch layout code (square, $\Theta_{tp} = 90^o$, or triangular, $\Theta_{tp} = 30^o$)	C_{tp}	
Tube pitch length	L_{tp}	mm
Tube material code	C_{mat}	
Shell design concept (E or F) code	C_{shell}	
Temperature bracket (local, upper, mid, or lowermost) code	C_{brack}	

Step 2: Determine number of shells required

The overall R for this system was calculated using Equation 7. The number of shells N_s was then determined from Equation 58 or 59.

Step 3: Determine inlet/outlet temperatures for each shell

Equations 60 – 65 were used to section the temperature profiles and find the temperatures that each shell would see. Based on the temperature bracket code chosen, the corresponding sectioned inlet/outlet temperatures for both fluid streams were then passed into the design loop, Steps 4 – 8.

Step 4: Guess initial values for K_s, K_t, n, and m

The Kern method provided the initial guess for K_s [13], which is given by Equation 72. Here g is the gravitational constant and has a value assumed as 9.807 m/s^2 [13].

$$K_s = [67.062 C_l / 1000^{3.406} g] [(L_{tp} - D_t) / D_t] [L_{tp} D_e^{1.109} \mu_s^{1.297} / \text{Л}_s \rho_s^2 k_s^{3.406} C_{P,s}^{1.703}], \quad \text{(W/m.K)} \quad (72)$$

$$\text{Where } D_e = 4[(L_{tp}^2) - ((\pi(D_t^2)) / 4)] / \pi D_t \qquad C_{tp} = \text{square} \quad (73)$$

$$\text{And } D_e = 4[L_{tp}^2 \text{sqrt}(3) - (\pi D_t^2 / 8)] / \pi D_t \qquad C_{tp} = \text{triangular} \quad (74)$$

Step 5: Numerically solve for h_t

Serna provides the following nonlinear equation in terms of h_t [13]:

$$h_t - \{[\Delta P_t F_T LMTD / K_T Q] / [(K_s \Delta P_t / K_t \Delta P_s h_t^n)^{1/m} + R_{dw} + D_t / h_t(D_t - t_t)]\}^{1/n}, = 0 \quad (75)$$

$$\text{Where } R_{dw} = R_{ds} + (10^3 D_t / 2k) \ln(D_t / (D_t - t_t) + R_{dt} (D_t / (D_t - t_t) \quad (76)$$

Step 6: Solve for h_s and A_o

Serna also provides the following expression for h_s [13], and A_o is determined from Equation 77 [13]:

$$h_s = [K_t \, \Delta P_s \, h_t^n / K_s \, \Delta P_t]^{1 / m} \tag{77}$$

Step 7: Determine characteristic fluid flow velocities

The initial guess for v_s is provided by Equation 78. Each other time v_s and v_t are calculated, Equations 79 and 80 are used.

$$v_s = (\mu_s^{1.3/6} \, D_e^{0.45} \, h_s / 36 k_s^{2/3} \, C_{P,\,s}^{1/3} \, \rho_s^{0.55})^{1 / 0.55} \tag{78}$$

$$v_s = (h_s / K_{s1})^{1 / (1 - r\_h)} \tag{79}$$

$$v_t = (\mu_t^{7/15} \, D_{ti}^{1/5} \, h_t / 2.3 k_t^{2/3} \, C_{P,\,t}^{1/3} \, \rho_t^{4/5})^{1 / 0.55} \tag{80}$$

Step 8: Extract geometrical design parameters

Once all of the above are calculated, the geometrical parameters that fully define the STHE design and performance can be calculated using the definitions provided in the Theory chapter. Equations 39 and 48 are then used to calculate values for K_s, K_t, m, and n for the next iteration, Steps 4 – 8. The design algorithm was coded in Python programming language [24]. The entire design loop is presented in the Appendix.

RESULTS

In the version of the loop included in the Appendix, an additional input code C_{val} is defined to check for comparison to the values reported by Serna during the validation of the compact formulation and the algorithm presented in that paper. The input values for the validation exercise performed by Serna against data reported by Thomas et. al. are shown below in Table 7 [13]. The results for these values are included below in Table 8 [13]; two designs were reported.

Table 7: Validation Input Values for Serna Algorithm [13]

Parameter	Shell side	Tube side	Units
Flowrate	43.6	45.377	kg/s
Density	820	993	kg/m^3
Specific heat capacity	2170	4170	J/kg.K
Viscosity	2.45	0.682	Pa.s (10^{-3})
Thermal conductivity	0.128	0.63	W/m.K
Inlet temperature	114	26	^oC
Outlet temperature	66	50	^oC
Allowable pressure drop	11.346	10.13	kPa
Fouling factor	0.0007	0.00015	K/m^2W
Tube wall thermal conductivity	0		
Heat duty	4541.4		kW
Geometry			
Outer tube diameter	19.1		mm
Inner tube diameter	16.6		mm
Tube layout	90		deg
Tube pitch	25.4		mm
Number of tube passes	4		
Shell to baffle clearance	5.72		mm
Tube to baffle clearance	0.794		mm
Shell to tube bundle clearance	12.7		mm

Table 8: Results for Validation of Serna Algorithm [13]

	Design 1	Design 2	Thomas et. al.	Units
Geometry				
Shell diameter	1015.71	1014.75	1070	mm
Total tube flow length	3872.97	3852.14	4480	mm
Baffle cut	22.73	20.8	20	%
Central baffle spacing	342.9	342.94	375	mm
Inlet baffle spacing	342.9	342.94	375	mm
Outlet baffle spacing	342.9	342.94	375	mm
Number of baffles	10.2945	10	12	
Number of tubes	1091	1089	1195	
Number of tube passes	4	4	4	
Installed area	253.544	251.717	349.922	m^2
Performance				
Required area	253.352	251.496	262.825	m^2
Shell side Re	36608	33768	33544	
Shell side pressure drop	11.346	11.346	11.346	kPa
Tube side pressure drop	7.151	7.136	7.65	kPa
Shell side heat transfer coefficient	692.091	701.05	657.84	W/m^2K
Tube side heat transfer coefficient	3775.964	3782.149	3510.477	W/m^2K
Overall heat transfer coefficient	381.29	384.13	367.58	W/m^2K
Ratio of baffle crossflow area to baffle window area	1.0004	1.1528	1.2282	

It was attempted to adopt the input values reported in Table 7 for the algorithm designed in this thesis and validate its performance against the results reported in Table 8. However, despite extensive debugging efforts, the best results achieved were not deemed acceptable for validation

of the design loop for Steps 4 – 8 of the STHE system design algorithm developed here. The validation results are reported below in Table 9, where the percentage deviations from the validation values are the values of interest.

Table 9: STHE System Design Algorithm Required Input Parameters

	Serna	Calculated	Percent Error
Shell diameter	1015.71 mm	2092.5 mm	106%
Total tube length	3872.97 mm	1671.8 mm	56.8%
Baffle spacing	342.9 mm	18.3 mm	94.6%
Number of baffles	10	90	88.9%
Number of tubes	1091	5927	443%
Installed area	253.5 m^2	0.594 m^2	99.7%
Shell side Re	36608	35.1	99.9%

CONCLUSION

An overview of S-CO_2 cycles and MSR's for power production was presented. The primary contributions of these two systems to the content of this thesis was to provide an operational context for which the HEX design investigated would be applied. MSR's and S-CO_2 cycles operate under particular conditions; these involve high temperatures, very high operating pressures, and corrosive fluids. Understanding the impact of the context on the design of a HEX system was intended to supply the intellectual merit of this thesis.

However, the design algorithm never reached the implementation stage, due to the inability to achieve satisfactory performance using known input parameters and known results. Proper validation of the main loop of the algorithm was necessary before any of the intended results could be reported. Were validation to be achieved, the intended primary results were to include:

- Plots of A_o and U as functions of a) molten salt temperatures, b) CO_2 temperatures, c) operating pressures, and d) heat duties
- Plots of the number of shells for E and F design as functions of a) molten salt temperatures, b) CO_2 temperatures, c) operating pressures, and d) heat duties
- A table of ideal operating conditions for an E and F shell HEX series
- Concluding trends and recommendations based on these results as to the suitability of STHE's for MSR/CO_2 applications
    - High A_o will be used as a corollary for high capital expense
    - Low U will be used as a corollary for lower efficiency

Future research begins with the proper validation of the main loop of the design algorithm. Unit inconsistency is a possible source of error. The compact formulations provided by Serna did not include dimensions, which made it difficult to verify that results were reasonable. Deriving a similar formulation from scratch would allow for performing dimensional analysis at each step; this formulation should take only standard SI units, which was not the case with the formulation used here or by the Taborek correlations.

An additional possible error may be due to the flow of the equations solved in the main loop of the algorithm. A redesigned algorithm should solve each equation sequentially, such that there is no possibility for multiple branches or equation solving paths.

Beyond the desired results listed above, there are additional realms of investigation that future researchers may pursue. The ultimate goal is to create a representative model for the shell side that takes into account two aspects of S-CO_2 flow: 1) the effects of buoyancy and 2) the effects of the rapid change of fluid properties near the critical point. CFD simulations of the CO_2 flow on the shell side were intended for this thesis; however, without meaningful numerical data to compare them against, none were included.

APPENDIX – DESIGN ALGORITHM CODE FOR STEPS 4 - 8

```
import numpy as np
from scipy.optimize import newton as newt
from scipy.optimize import brentq as br
from mpmath import *
import Properties_MS as ms
import a_values as av
import k_values as kv
from CoolProp import PropsSI
mp.dps = 30
mp.pretty = True


def STHE_Opt(Mdot_s, Mdot_t, medium_t, mat, T_si, T_so, T_ti, T_to, P_si, P_ti, R_ds, R_dt, deltaP_sallow,
        deltaP_tallow, L_tp, shell, D_t, B_c, C_tpl, C_val, lcount):

    print('Reached step 0')

    # preliminary definitions
    T_sa = (T_si + T_so) / 2
    T_ta = (T_ti + T_to) / 2
    T_sw = (T_sa + T_ta) / 2
    T_tw = (T_sa + T_ta) / 2
    g_c = 9.807  # m/s^2
    # Pr_t is multiplied by a factor of 10**-3 to account for the ctp unit of viscosity
    Pr_t = ((mu_t*Cp_t)/k_t)*(10**(-3))
    print('Pr_t =', Pr_t)

    # choose shell configuration, E or F shell
    # report if assignment is misspelled
    if shell == "E":
        N_tp = 2
    elif shell == "F":
        N_tp = 4
    else:
        print('improper shell assignment')
        N_tp = nan

    # choose which side is hot and cold for LMTD calculation
    # report if temperatures are equal
    if T_si > T_ti:
        C_LMTD = "Shell hot"
        T_hi = T_si
        T_ho = T_so
        T_ci = T_ti
        T_co = T_to
    elif T_si < T_ti:
        C_LMTD = "Tube hot"
        T_hi = T_ti
        T_ho = T_to
        T_ci = T_si
        T_co = T_so
```

```
else:
    print('improper input temperatures')
    C_LMTD, T_hi, T_ho, T_ci, T_co = nan, nan, nan, nan, nan
print(C_LMTD)

# choose tube pitch layout
# check if improper code assignment
if C_tpl == 'square':
    C_l = 1
    D_e = (4*((L_tp**2) - ((np.math.pi*(D_t**2)) / 4))) / (np.math.pi*D_t)
elif C_tpl == 'triangular':
    C_l = 0.866
    D_e = (4*(((L_tp**2)*np.math.sqrt(3)) - ((np.math.pi*(D_t**2)) / 8))) / ((np.math.pi*D_t) / 2)
else:
    print('improper tube layout assignment')
    C_l = nan
    D_e = nan
print('C_l =', C_l, '\nD_e =', D_e)

H_si = ms.H_s(T_si)
H_so = ms.H_s(T_so)
Qdot = Mdot_s*(H_si - H_so)
tau_tallow = kv.tau_tallow(mat)
mu_s = ms.mu_s(T_sa)
rho_s = ms.rho_s(T_sa)
k_s = ms.k_s(T_sa)
Cp_s = ms.Cp_s(T_sa)
k = kv.k(mat, T_sw, T_tw)
print('k = ', k)
k_t = PropsSI('Conductivity', 'T', T_ta, 'P', P_ti, medium_t)
rho_t = PropsSI('rho', 'T', T_ta, 'P', P_ti, medium_t)
Pr_t = PropsSI('Prandtl', 'T', T_ta, 'P', P_ti, medium_t)
mu_t = PropsSI('V', 'T', T_ta, 'P', P_ti, medium_t)
mu_tw = PropsSI('V', 'T', T_tw, 'P', P_ti, medium_t)
Cp_t = PropsSI('C', 'T', T_ta, 'P', P_ti, medium_t)
t_t = ((P_ti * D_t) / ((2 * tau_tallow) + P_ti)) + (0.005 * D_t)
D_ti = D_t - (2*t_t)
phi_t = (mu_t / mu_tw)**0.14
Q_s = Mdot_s / rho_s
Q_t = Mdot_t / rho_t
print('Q_s =', Q_s, '\nQ_t', Q_t)
R_dw = R_ds + (((0.001*D_t) / (2*k))*np.math.log(D_t / D_ti)) + ((D_t / D_ti)*R_dt)
print('R_dw =', R_dw)
F_TP = (T_to - T_ti) / (T_si - T_ti)
F_TR = (T_si - T_so) / (T_to - T_ti)
F_TS = (((F_TR**2) + 1)**0.5) / (F_TR - 1)
F_TW = (1 - (F_TP*F_TR)) / (1 - F_TP)
F_T = (F_TS*np.math.log(F_TW)) / np.math.log((1 + F_TW - F_TS + (F_TS*F_TW)) / (1 + F_TW + F_TS -
    (F_TS*F_TW)))

K_s = np.zeros(lcount+1)
K_t = np.zeros(lcount+1)
n = np.zeros(lcount+1)
m = np.zeros(lcount+1)
```

```
h_t = np.zeros(lcount)
h_s = np.zeros(lcount)
v_s = np.zeros(lcount)
v_t = np.zeros(lcount)
L_ta = np.zeros(lcount)
N_t = np.zeros(lcount)
D_ctl = np.zeros(lcount)
D_s = np.zeros(lcount)
L_bc = np.zeros(lcount)
L_ts = np.zeros(lcount)
L_ti = np.zeros(lcount)
N_b = np.zeros(lcount)
L_bi = np.zeros(lcount)
L_bo = np.zeros(lcount)
A_a = np.zeros(lcount)
A_o = np.zeros(lcount)
L_sb = np.zeros(lcount)
D_otl = np.zeros(lcount)
theta_ctl = np.zeros(lcount)
F_w = np.zeros(lcount)
S_sb = np.zeros(lcount)
S_tb = np.zeros(lcount)
S_m = np.zeros(lcount)
mdot_s = np.zeros(lcount)
Re_s = np.zeros(lcount)
r_s = np.zeros(lcount)
r_lm = np.zeros(lcount)
J_l = np.zeros(lcount)
S_b = np.zeros(lcount)
F_sbp = np.zeros(lcount)
N_tcc = np.zeros(lcount)
r_ss = np.zeros(lcount)
J_b = np.zeros(lcount)
F_c = np.zeros(lcount)
J_c = np.zeros(lcount)
J_s = np.zeros(lcount)
J_tot = np.zeros(lcount)
p = np.zeros(lcount)
R_l = np.zeros(lcount)
R_s = np.zeros(lcount)
C_bp = np.zeros(lcount)
R_b = np.zeros(lcount)
a1 = np.zeros(lcount)
a2 = np.zeros(lcount)
a3 = np.zeros(lcount)
a4 = np.zeros(lcount)
r_h = np.zeros(lcount)
a = np.zeros(lcount)
c_h = np.zeros(lcount)
b1 = np.zeros(lcount)
b2 = np.zeros(lcount)
b3 = np.zeros(lcount)
b4 = np.zeros(lcount)
r_p = np.zeros(lcount)
```

```
b = np.zeros(lcount)
c_p = np.zeros(lcount)
j_si = np.zeros(lcount)
f_si = np.zeros(lcount)
S_wg = np.zeros(lcount)
S_wt = np.zeros(lcount)
S_w = np.zeros(lcount)
S_mw = np.zeros(lcount)
N_tcw = np.zeros(lcount)
K_s1 = np.zeros(lcount)
K_s2 = np.zeros(lcount)
K_s3 = np.zeros(lcount)
r_pprime = np.zeros(lcount)
K_s4 = np.zeros(lcount)
K_s5 = np.zeros(lcount)
K_t2 = np.zeros(lcount)
K_t3 = np.zeros(lcount)
r_prime = np.zeros(lcount)
K_t4 = np.zeros(lcount)
gg1 = np.zeros(lcount)
gg2 = np.zeros(lcount)
gg3 = np.zeros(lcount)
gg4 = np.zeros(lcount)
U = np.zeros(lcount)
for l in range(0, lcount):
    if l == 0:
        print('\n\n\n\n*********For the', l+1, '- st iteration*********')
    elif l == 1:
        print('\n\n\n\n*********For the', l+1, '- nd iteration*********')
    elif l == 2:
        print('\n\n\n\n*********For the', l+1, '- rd iteration*********')
    else:
        print('\n\n\n\n*********For the', l+1, '- th iteration*********')

    # store values to check later for convergence
    if l == 0:
        print('reached step 1')
        # STEP 1: initial guesses for K_t, K_s, n, and m
        m[l] = 5.109
        n[l] = 3.5
        K_s[l] = ((67.062 * C_l) / (g_c * (1000 ** 3.406))) * ((L_tp - D_t) / D_t) * \
                 ((L_tp * (D_e ** 1.109) * (mu_s ** 1.297)) / (Q_s * (rho_s ** 2) * (k_s ** 3.406) * (Cp_s ** 1.703)))
        K_t[l] = (((phi_t ** 4.5) * ((D_ti / 1000) ** 0.5) * ((mu_t / 1000) ** (11 / 6))) /
                  ((0.023 ** 2.5) * g_c * Q_t * (rho_t ** 2) * (k_t ** (7 / 3)) * (Cp_t ** (7 / 6)))) * (D_ti / D_t)

    print('reached step 2')
    print('K_s:', K_s[l], '\nK_t:', K_t[l], '\nm:', m[l], '\nn:', n[l])
    # STEP 2: calculate h_t
    # use Newton-Raphson method

    # preliminary definitions of function-reducing variable blocks
    deltaT1 = T_hi - T_co
    deltaT2 = T_ho - T_ci
    LMTD = (deltaT1 - deltaT2) / np.math.log(deltaT1 / deltaT2)
```

```
print('for calculating LMTD:\n   deltaT1 =', deltaT1, 'deltaT2 =', deltaT2, '\n   LMTD =', LMTD)

def Fh_tfunc(x):
   print('for calculating h_t using newt:\nx =', x)
   print('deltaP_tallow =', deltaP_tallow, ', F_T =', F_T, ', LMTD =', LMTD, ', K_t =', K_t[l], ', Qdot =',
        Qdot, ', K_s =', K_s[l], ', deltaP_sallow =', deltaP_sallow, ', n =', n[l], ', m =', m[l])
   return x - ((((deltaP_tallow * F_T * LMTD) / (K_t[l] * Qdot)) / ((((K_s[l] * deltaP_tallow) /
      ((K_t[l] * deltaP_sallow) * (x ** n[l]))) ** (1 / m[l])) + R_dw + (D_t / (D_ti * x)))) ** (1 / n[l]))
print('Fh_tfunc: [a, b] =', Fh_tfunc(1), ',', Fh_tfunc(1000*h_t[l-1]))
if C_val == 'N':
   if l == 0:
      h_t[l] = newt(Fh_tfunc, x0=3000, tol=0.1, maxiter=1000)
   else:
      h_t[l] = newt(Fh_tfunc, x0=h_t[l-1], tol=0.1, maxiter=1000)
      h_tu = 1000*h_t[l-1]
      h_t[l] = br(Fh_tfunc, a=1, b=h_tu, xtol=0.1, maxiter=10000)
elif C_val == 'Y':
   h_t[l] = 3775.9
else:
   print('improper C_val assignment')
   h_t[l] = nan
print('h_t =', h_t[l])

# STEP 3: calculate h_s and A_o
if C_val == 'N':
   h_s[l] = ((K_t[l]*deltaP_sallow*(h_t[l]**n[l])) / (K_s[l]*deltaP_tallow))**(1/m[l])
   A_o[l] = (Qdot / (F_T*LMTD))*((1 / h_s[l]) + R_dw + (D_t / (D_ti*h_t[l])))
elif C_val == 'Y':
   h_s[l] = 692
   A_o[l] = 253.352
else:
   h_s[l] = nan
   A_o[l] = nan
print('h_s =', h_s[l])
print('A_o =', A_o[l])

# STEP 4: extract geometrical parameters
if l == 0:
   v_s[l] = ((h_s[l]*(mu_s**(1.3/6))*(D_e**0.45)) / (36*(k_s**(2/3)*(Cp_s**(1/3))*(rho_s**0.55))))**(1 /
        0.55)
else:
   v_s[l] = (h_s[l] / K_s1[l-1])**(1/(1-r_h[l-1]))
   v_t[l] = ((h_t[l] * (D_ti ** (1 / 5)) * (mu_t ** (7 / 15))) / (
      2.3 * (k_t ** (2 / 3)) * (rho_t ** (4 / 5)) * (Cp_t ** (1 / 3)))) ** (1 / 0.8)
print('v_t =', v_t[l])
print('v_s =', v_s[l])

if C_val == 'N':
   N_t[l] = ((10**6)*N_tp*Q_t) / (np.math.pi*v_t[l]*((D_ti**2) / 4))
   L_ta[l] = (A_o[l] / (np.math.pi*D_t*N_t[l]))*(10**6)
elif C_val == 'Y':
   N_t[l] = 1091
   L_ta[l] = 3872.9
else:
```

```
    N_t[l] = nan
    L_ta[l] = nan
print('N_t =', N_t[l])
print('L_ta =', L_ta[l])
"""D_ctl[l] = L_tp*np.math.sqrt((4*N_t[l]*C_l) / np.math.pi)
print('D_ctl =', D_ctl[l])
Si_n[l] = 1 - ((4*N_t[l]*C_l*(L_tp**2)) / ((D_ctl[l]**2)*np.math.pi))
print('Si_n =', Si_n[l])"""
if N_tp == 2:
    Si_n = 0.08
elif N_tp == 4:
    Si_n = 0.135
else:
    print('N_tp improperly assigned for Si_n estimation')
    Si_n = nan

D_ctl[l] = L_tp*np.math.sqrt((4*N_t[l]*C_l) / (np.math.pi*(1 - Si_n)))
print('D_ctl =', D_ctl[l])

L_bb = 20
if C_val == 'N':
    D_s[l] = D_ctl[l] + L_bb + D_t
    L_bc[l] = ((10**6)*Q_s) / (v_s[l]*(L_bb + (D_ctl[l] / (L_tp*(L_tp - D_t)))))
    # L_bc[l] = ((10**6)*Q_s) / (v_s[l]*(L_bb + ((D_ctl[l] / L_tp)*(L_tp - D_t))))
elif C_val == 'Y':
    D_s[l] = 1015
    L_bc[l] = 342.9
else:
    D_s[l] = nan
    L_bc[l] = nan
print('D_s =', D_s[l])
print('L_bc =', L_bc[l])
L_ts[l] = 0.5*D_s[l]*np.math.sqrt(P_si / tau_tallow)
"""print('for calculating L_ti:\n   L_ta =', L_ta[l], ', L_ts =', L_ts[l])
L_ti[l] = L_ta[l] - (2*L_ts[l])
print('for calculating N_b:\n   L_ti =', L_ti[l], ', L_b =', L_bc[l])
if l < lcount - 1:
    N_b[l] = (L_ti[l] / L_bc[l]) - 1
else:
    N_b[l] = int((L_ti[l] / L_bc[l]) - 1)
L_bi[l] = (L_ta[l] - (L_bc[l]*(N_b[l] - 1))) / 2
L_bo[l] = L_bi[l]"""
if C_val == 'N':
    N_b[l] = ((L_ta[l] - (2*L_bc[l])) / L_bc[l]) + 1
elif C_val == 'Y':
    N_b[l] = 10.29
else:
    N_b[l] = nan
print("N_b =", N_b[l])
L_bi[l] = L_bc[l]
print("L_bi =", L_bi[l])
L_bo[l] = L_bc[l]
A_a[l] = L_ta[l]*D_t*np.math.pi*N_t[l]*(10**(-6))
print("A_a =", A_a[l])
```

```
# ideal tube-bank correlations
L_tb = av.L_tb(D_t)
print('L_tb =', L_tb)
L_sb[l] = 1.6 + (0.004*D_s[l])
print("L_sb =", L_sb[l])
D_otl[l] = D_s[l] - L_bb
print("D_otl =", D_otl[l])
theta_ds = 2*np.math.acos(1 - (B_c / 50))*(180 / np.math.pi)
print("theta_ds =", theta_ds)
theta_ctl[l] = 2*np.math.acos((D_s[l] / D_ctl[l])*(1 - (B_c / 50)))*(180 / np.math.pi)
print("theta_ctl =", theta_ctl[l])
F_w[l] = theta_ctl[l] / 360
print("F_w =", F_w[l])
S_sb[l] = 0.00436*D_s[l]*L_sb[l]*(360 - theta_ds)
print("S_sb =", S_sb[l])
S_tb[l] = ((np.math.pi / 4)*(((D_t + L_tb)**2) - (D_t**2)))*N_t[l]*(1 - F_w[l])
print("S_tb =", S_tb[l])
S_m[l] = L_bc[l]*(L_bb + ((D_ctl[l] / L_tp)*(L_tp - D_t)))
print("S_m =", S_m[l])
mdot_s[l] = (Mdot_s / S_m[l])*(10**6)
print("mdot_s =", mdot_s[l])
if C_val == 'N':
   Re_s[l] = (D_t*mdot_s[l]) / mu_s
elif C_val == 'Y':
   Re_s[l] = 36608
else:
   Re_s[l] = nan
print("Re_s =", Re_s[l])
r_s[l] = S_sb[l] / (S_sb[l] + S_tb[l])
print("r_s =", r_s[l])
r_lm[l] = (S_sb[l] + S_tb[l]) / S_m[l]
print("r_lm =", r_lm[l])
J_l[l] = (0.44*(1 - r_s[l])) + ((1 - (0.44*(1 - r_s[l])))*exp(-2.2*r_lm[l]))
print("J_l =", J_l[l])
S_b[l] = L_bc[l]*((D_s[l] - D_otl[l]) - D_t)
print("S_b =", S_b[l])
F_sbp[l] = S_b[l] / S_m[l]
print("F_sbp =", F_sbp[l])
C_bh = 1.25
L_pp = 0.866*L_tp
print('L_pp =', L_pp)
N_ss = av.N_ss(L_bb)
print("N_ss =", N_ss)
N_tcc[l] = (D_s[l] / L_pp)*(1 - (B_c / 50))
print("N_tcc =", N_tcc[l])
r_ss[l] = N_ss / N_tcc[l]
print("r_ss =", r_ss[l])
J_b[l] = np.math.exp(-C_bh*F_sbp[l]*(1 - ((2*r_ss[l])**(1/3))))
print("J_b =", J_b[l])
F_c[l] = 1 - (2*F_w[l])
print("F_c =", F_c[l])
J_c[l] = 0.55 + (0.72*F_c[l])
print("J_c =", J_c[l])
```

```
J_s[l] = ((N_b[l] - 1) + (L_bi[l]**0.4) + (L_bo[l]**0.4)) / ((N_b[l] - 1) + L_bo[l] + L_bi[l])
print("J_s =", J_s[l])
J_tot[l] = J_b[l]*J_c[l]*J_l[l]*J_s[l]
print("J_tot =", J_tot[l])
p[l] = (-0.15*(1 + r_s[l])) + 0.81
print("p =", p[l])
R_l[l] = np.exp(-1.33*(1 + r_s[l])*(r_lm[l]**p[l]))
print("R_l =", R_l[l])
R_s[l] = ((L_bc[l] / L_bo[l])**1.8) + ((L_bc[l] / L_bi[l])**1.8)
print("R_s =", R_s[l])
C_bp[l] = av.C_bp(Re_s[l])
print("C_bp =", C_bp[l])
R_b[l] = np.math.exp(-C_bp[l] * F_sbp[l] * (1 - ((2*r_ss[l])**(1/3))))
print("R_b =", R_b[l])
a1[l], a2[l], a3[l], a4[l] = av.avalue(Re_s[l])
r_h[l] = -a2[l]
print("r_h =", r_h[l])
a[l] = a3[l] / (1 + (0.14*(Re_s[l]**a4[l])))
print('a =', a)
c_h[l] = a1[l]*((1.33 / (L_tp / D_t))**a[l])
print("c_h =", c_h[l])
b1[l], b2[l], b3[l], b4[l] = av.bvalue(Re_s[l])
r_p[l] = -b2[l]
print("r_p =", r_p[l])
b[l] = b3[l] / (1 + (0.14*(Re_s[l]**b4[l])))
print('b =', b)
c_p[l] = b1[l]*((1.33 / (L_tp / D_t))**b[l])
print("c_p =", c_p[l])
j_si[l] = c_h[l]*((mu_s / (D_t*rho_s))**r_h[l])*(v_s[l]**(-r_h[l]))
print("j_si =", j_si[l])
f_si[l] = c_p[l]*((mu_s / (D_t*rho_s))**r_p[l])*(v_s[l]**(-r_p[l]))
print("f_si =", f_si[l])
if C_val == 'N':
    U[l] = Qdot / (F_T*A_o[l]*LMTD)
elif C_val == 'Y':
    U[l] = 381.29
else:
    U[l] = nan
print("U =", U[l])

# STEP 5: calculate new values for compact parameters for next iteration
T_sw = T_sa - (Qdot / (h_s*N_t*np.math.pi*D_t*L_ta))
T_tw = T_ta + (Qdot / (h_t*N_t*np.math.pi*D_ti*L_ta))
mu_sw = ms.mu_s(T_sw)
mu_tw = PropsSI('V', 'T', T_tw, 'P', P_ti, medium_t)
phi_s = (mu_s / mu_sw)**0.14
phi_t = (mu_t / mu_tw)**0.14
S_wg[l] = (np.math.pi / 4)*(D_s[l]**2)*((theta_ds / 360) - ((np.math.sin(theta_ds)) / (2*np.math.pi)))
print("S_wg =", S_wg[l])
S_wt[l] = N_t[l]*F_w[l]*((np.math.pi / 4)*(D_t**2))
print("S_wt =", S_wt[l])
S_w[l] = S_wg[l] - S_wt[l]
print("S_w =", S_w[l])
S_mw[l] = S_m[l] / S_w[l]
```

```
    print('S_mw =', S_mw[l])
    N_tcw[l] = (0.8 / (0.866*L_tp))*((D_s[l]*(B_c / 100)) - ((D_s[l] - D_ctl[l]) / 2))
    print("N_tcw =", N_tcw[l])
    Pr_s = ((mu_s*Cp_s) / k_s)*(10**(-3))
    print("Pr_s =", Pr_s)

    K_s1[l] = (((phi_s*c_h[l]*k_s*(Pr_s**(1/3))) / (0.001*D_t))*(((D_t*rho_s) / mu_s)**(1 - r_h[l])))*J_tot[l]
    print("K_s1 =", K_s1[l])
    K_s2[l] = 1 * (((1 + (0.3 * N_tcw[l])) * (R_l[l] * N_b[l] * rho_s)) / ((N_b[l] + 1) * D_s[l]))
    print("K_s2 =", K_s2[l])
    K_s3[l] = ((R_l[l]*((N_b[l] - 1) / (N_b[l] + 1))) + (R_s[l]*((N_tcc[l] + N_tcw[l]) / (N_tcc[l]*(N_b[l] + 1))))) \
            *(1 - (B_c / 50))*((2*c_p[l]*R_b[l]*rho_s) / (phi_s*L_pp))*((mu_s / (D_t*rho_s))**r_p[l])
    print("K_s3 =", K_s3[l])
    r_pprime[l] = r_p[l] / (((K_s2[l] / K_s3[l])*(v_s[l]**(-r_p[l]))) + 1)
    print("r_pprime =", r_pprime[l])
    K_s4[l] = (K_s2[l]*(v_s[l]**r_pprime[l])) + (K_s3[l]*(v_s[l]**(r_pprime[l] - r_p[l])))
    print("K_s4 =", K_s4[l])
    K_s5[l] = ((4*C_l) / (D_t*Q_s*(1 - Si_n)))*((L_tp / (np.math.pi*D_ctl[l]))**2)*(
          (D_s[l]*L_bc[l]*(N_b[l] + 1)) / (L_bi[l] + L_bo[l] + (L_bc[l]*(N_b[l] - 1))))*(L_bb + (D_ctl[l]*
                                                                                  ((L_tp - D_t) / L_tp)))
    print("K_s5 =", K_s5[l])
    K_indiv = 2.5
    K_t1 = ((0.023*k_t*(Pr_t**(1/3))) / ((10**(-3))*D_ti))*(((rho_t*D_ti) / mu_t)**0.8)
    print("K_t1 =", K_t1)
    K_t2[l] = K_indiv*(D_ti / L_ta[l])
    print("K_t2 =", K_t2[l])
    K_t3[l] = 0.184*phi_t*(1 + ((2*L_ts[l]) / L_ta[l]))*((mu_t / (D_ti*rho_t))**0.2)
    print("K_t3 =", K_t3[l])
    r_prime[l] = 0.2 / (((K_t2[l] / K_t3[l])*(v_t[l]**(-0.2))) + 1)
    print("r_prime =", r_prime[l])
    K_t4[l] = (K_t2[l]*(v_t[l]**r_prime[l])) + (K_t3[l]*(v_t[l]**(r_prime[l] - 0.2)))
    print("K_t4 =", K_t4[l])
    K_t5 = (D_ti*rho_t) / (4*Q_t*D_t)
    print("K_t5 =", K_t5)

    if l == lcount-1:

        print('\nShell diameter (mm):', D_s, '\n\nTotal flow length of the tubes (mm):', L_ta, '\n\nBaffle cut (%):'
            , B_c, '\n\nCentral baffle spacing (mm):', L_bc, '\n\nInlet/Outlet baffle spacing (mm):', L_bi,
            '\n\nNumber of baffles:', N_b, '\n\nNumber of tubes:', N_t, '\n\nNumber of tube passes:', N_tp,
            '\n\nInstalled area (m^2):', A_a, '\n\nRequired area (m^2):', A_o, '\n\nShell side Re:', Re_s,
            '\n\nh_s (W/m^2.K):', h_s, '\n\nh_t (W/m^2.K):', h_t, '\n\nU (W/m^2.K)', U, '\n\nS_m/S_w:', S_mw)

    # STEP 6: Define new tear values
    if l < lcount:
        m[l+1] = (3 - r_pprime[l]) / (1 - r_h[l])
        K_s[l+1] = (K_s4[l]*K_s5[l]) / (K_s1[l]**m[l+1])
        n[l+1] = (3 - r_prime[l]) / 0.8
        K_t[l+1] = ((K_t5*K_t4[l]) / (2*g_c))*((1 / K_t1)**n[l+1])
        print('\ncheck proper array assignment:\nK_s:\n', K_s, '\nm:\n', m, '\nK_t:\n', K_t, '\nn:\n', n)

return N_t, D_s, L_bc, N_b, L_bi, L_bo, h_s, h_t, A_o, U
```

LIST OF REFERENCES

[1] "Small Nuclear Power Reactors", World Nuclear Association [Online]. Available:https://www.world-nuclear.org/information-library/nuclear-fuel-cycle/nuclear-powerreactors/small-nuclear-power-reactors.aspx. [Accessed: Nov- 2019].

[2] Advances in Small Modular Reactor Technology Developments, International Atomic Energy Agency, Austria, ed. 2018.

[3] Dolan, T. (2017). Molten salt reactors and thorium energy. 1st ed. Duxford, United Kingdom: Woodhead Publishing is an imprint of Elsevier.

[4] A Technology Roadmap for Generation IV Nuclear Energy Systems. p. 34. U.S. DOE Nuclear Energy Research Advisory Committee and the Generation IV International Forum. 2002.

[5] Angelino G., Carbon Dioxide Condensation Cycles for Power Production., 1968: ASME Paper No. 68-GT-23.

[6] Kacludis A., Lyons S., Nadav D., Zdankiewicz E., Waste Heat to Power (WH2P) Applications Using a Supercritical CO2 - Based Power Cycle., 2012: Echogen Power Systems LLC: Presented at Power-Gen International 2012.

[7] Zhiwen M, Turchi C.S., Advanced Supercritical Carbon Dioxide Power Cycle Configurations for Use in Concentrating Solar Power Systems., 2011: NREL/CP5500-50787.

[8] Dostal V., Driscoll M.J., Hejzlar P., Supercritical Carbon Dioxide Cycle for Next Generation Nuclear Reactors, 2004: MIT-ANP-TR-100.

[9] Vesely L., Dostal V., Kapat J., Vasu S., Martin S., Techno-Economic Evaluation of the Effect of Impurities on the Performance of Supercritical CO2 Cycles., Proceedings of the ASME Turbo Expo 2019: Turbomachinery Technical Conference and Exposition. Volume 9: Oil and Gas Applications; Supercritical CO2 Power Cycles; Wind Energy. Phoenix, Arizona, USA. June 17–21, 2019. V009T38A014. ASME. https://doi.org/10.1115/GT2019- 90704

[10] Ahn Y., Bae S.J., Kim M., Cho S.K., Maik S., Lee J.I., Cha J.E., “Review of Supercritical CO2 Power cycle Technology and Current Status of Research and Development,” Nuclear Engineering and Technology, vol. 47, no. October, pp. 647-661, 2015.

[11] "Characteristics of Diffusion Bonded Heat Exchangers Heatric", Heatric [Online]. Available: https://www.heatric.com/heatexchangers/features/characteristics/. [Accessed: Nov2019].

[12] Taborek, J., 1983, Shell-and-tube exchangers: single-phase flow, in Schlunder, E.U. (ed). Heat Exchangers Design Handbook, Vol. 3, Section 3.3 (Hemisphere Publishing Corp., Washington, DC, USA).

[13] Serna, M., and Jimenez, A., “A Compact Formulation of the Bell-Delaware Method for Heat Exchanger Design and Optimization,” Chemical Engineering Research and Design, 83(A5), pp. 539-550.

[14] Kakac, S., 2002, “Heat Exchangers Selection, Rating, and Thermal Design,” CRC Press LLC, Coral Gables, pp. 283-341.

[15] TEMA. Standards of the Tubular Heat Exchanger Manufacturers Association, 10th ed.; Tubular Exchanger Manufacturers Association: New York, 2019.

[16] American Society of Mechanical Engineers. (1900). *ASME boiler and pressure vessel code*. New York: American Society of Mechanical Engineers, Boiler and Pressure Vessel Committee.

[17] Vengateson, U., 2010, “Design of Multiple Shell and Tube Heat Exchangers in Series: E Shell and F Shell,” Chemical Engineering Research and Design, 88, pp. 725-736.

[18] A. Sowder, “Program on technology innovation: Technology assessment of a molten salt reactor design. The Liquid-Fluoride Thorium Reactor (LFTR),” 2015.

[19] Williams, D., Toth, L., and Clarno, K., 2006, ASSESSMENT OF CANDIDATE MOLTEN SALT COOLANTS FOR THE ADVANCED HIGHTEMPERATURE REACTOR (AHTR), Oak Ridge National Laboratory, Nuclear Science and Technology Division, Oak Ridge, Tennessee.

[20] Sohal, M., Ebner, M., Sabharwall, P., and Sharpe, P., 2013, Engineering Database of Liquid Salt Thermophysical and Thermochemical Properties, Idaho National Laboratory, Idaho Falls, Idaho.

[21] Kondo, M., Takuya, N., Muroga, T., Sagara, A., Noda, N., Xu, Q., Ninomiya, D., Masaru, N., Suzuki, A., and Terai, T., 2017, "High Performance Corrosion Resistance of Nickel-Based Alloys in Molten Salt Flibe", Fusion Science and Technology, 56(1).

[22] Lemmon E.W., Huber M.L., McLinden M.O., NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, 2013: Version 9.1.

[23] Bell I.H., Wronski J., Quoilin S., Lemort V., Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp., Industrial & Engineering Chemistry Research, vol. 53, 2014, doi 10.1021/ie4033999.

[24] Python Core Team (2015). Python: A dynamic, open source programming language. Python Software Foundation. URL https://www.python.org/.

[25] Serrano-López, R., Fradera, J., and Cuesta-López, S., 2013, "Molten salts database for energy applications", Chemical Engineering and Processing: Process Intensification, 73, pp. 87-102.

[26] "Hastelloy N Alloy", Haynesintl.com [Online]. Available: http://haynesintl.com/docs/default-source/pdfs/new-alloy-brochures/corrosion-resistantalloys/brochures/n-brochure.pdf?sfvrsn=18. [Accessed: Nov- 2019].

[27] Franssen, J.M., and Real, P.V., "Thermal Data for Carbon Steel and Stainless Steel Sections," Fire Design of Steel Structures, 2012 ECCS

[28] Koger J.W. 1973, Evaluation of Hastelloy N Alloys After Nine Years Exposure to Both a Molten Fluoride Salt and Air at Temperatures from 700 to 560 Degrees C. Oak Ridge National Laboratory, Nuclear Science and Technology Division, Oak Ridge, Tennessee.