

Integrated network-based thermofluid model of a once-through boiler at full- and part-load



Prepared by:

Kai-Yu Feng

FNGKAI002

Department of Mechanical Engineering
University of Cape Town

Supervisor:

Prof Pieter Rousseau

Dr Ryno Laubscher

February 2022

Submitted to the Department of Mechanical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Master of Science degree in Mechanical Engineering

Key Words: boiler modelling, thermofluid network modelling

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Abstract

The increased penetration of renewable energy sources in South Africa requires greater operational flexibility of existing coal-fired power plants (CFPPs). Operational flexibility implies that power plants need to operate intermittently or at low load for extended periods. Existing CFPPs are designed to operate at a steady baseload. Operating at these off-design conditions increase the risk of damaging the boiler's thick-walled components, leading to reduced life expectancy and/or failure.

Given that extensive experimental investigations on operating plants are impractical due to the risks, costs and complexity involved, there is a need for an integrated boiler model that has the necessary detail to study off-design and low load operations of coal-fired power plants. For that reason, a 1D quasi-steady-state thermofluid network model of a tower type once-through boiler was developed using the Flownex simulation environment.

The furnace model assumes complete, infinitely fast combustion with a specified value of unburned carbon and excess air. The radiation heat transfer in the furnace is modelled using the projected area approach (Gurvich/Blokh model) together with a high ash loading model. The gas-to-steam tube bank heat exchangers are discretised pass-by-pass, and the complex heat transfer phenomena in the heat exchangers and membrane water walls are represented by equivalent thermal networks.

The model results for the as-designed cases show that the ash deposition resistances suggested in the literature are not applicable for the case study boiler. For that reason, the proposed model calibration methodology was therefore applied at full-load operation (100%), and the results show good accuracy compared to real-plant data. The average error of the predicted heat exchanger heat loads is 2.0% and the maximum error is 5.2%.

The calibrated model was then validated by applying it to two part-load operational states in dry mode operation, as well as a wet mode (low-load) operational state. For the 81% load case, the average error in the heat exchanger duty is 2.0% and the maximum is 4.9%, while for the 63% load case, the average error is 3.6% and the maximum is 9.7%. For the low-load wet mode case at 35% load, the average error is 10.4% and the maximum is 18.1%. The cumulative heat transfer results for all the load cases correspond closely to the measured data, with the maximum error being 0.83% for the low load case.

These results suggest that the calibrated model can capture the heat distribution in the boiler with sufficient accuracy to allow suitable ash deposition resistances to be obtained from the calibration process. Furthermore, the metal temperatures predicted by the model are also shown to be sufficiently accurate, which means that it can be used to identify the heat exchanger tube passes or membrane water walls that may be at risk during operation.

Declaration

I, Kai-Yu Feng, hereby declare the work contained in this dissertation to be my own. All information which has been gained from various journal articles, textbooks or other sources has been referenced accordingly. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signed by candidate

Kai-Yu Feng

09/02/2022

09/02/2022

Acknowledgements

First and foremost, I would like to thank my supervisors, Professor Pieter Rousseau and Professor Ryno Laubscher, for their guidance. They have always provided me with insightful, knowledgeable and practical advice. Additionally, they have always helped and supported me whenever I am stuck with problems and pulled me out of rabbit holes countless times. I am sincerely grateful for having them as supervisors, and I could not complete this work without them.

Secondly, I would like to thank my industry mentor, Dr Gary De Klerk. Dr De Klerk has always provided me with the requested information and suggestions based on onsite experience. Additionally, he always has a fast response with short notice, which I really appreciate.

Thirdly, I would like to thank everyone in the ATProM research group, especially Mr Priyesh Gosai and Mr Brad Rowlin. They have always been kind and generously helped me whenever I asked.

Lastly, I would like to thank my mother, brother and friends for their emotional support whenever I feel despondent. They have always believed in me and supported me through difficult times; for that, I am truly grateful.

Table of Contents

List of Figures	vii
List of Tables.....	xi
List of Nomenclature.....	xii
1. Introduction	1
1.1 Background.....	1
1.2 Research question	1
1.3 Research objective	2
1.4 Dissertation Overview	2
2. Literature review.....	3
2.1 CFD modelling	3
2.2 One-dimensional thermofluid network modelling	5
2.3 Summary.....	12
3. Theory	13
3.1 Once-through boiler operation modes	14
3.2 Pulverised coal combustion	15
3.3 Gas side thermofluid modelling	20
3.4 Water/steam side thermofluid modelling	35
3.5 Heat transfer in solid materials.....	37
4. Model development	42
4.1 Boiler physical layout	42
4.2 Integrated boiler model layout	44
4.3 Component models	45
4.4 Model verification & validation.....	57
5. Results and Discussion	65
5.1 Plant measurements and data processing.....	65
5.2 Boiler simulation at full load conditions	68
5.3 Simulation of part-load conditions during dry mode operation.....	76
5.4 Simulation of low load operation during wet mode operation	88
5.5 Gas side pressure drop calculations at full load conditions.....	96

6. Conclusion and recommendations	98
7. List of References.....	100
Appendix A. Oxidations of combustion species.....	104
Appendix B. Correlations for Gurvich method.....	105
Appendix C. Furnace gas and particle emissivity with high ash loading.....	110
Appendix D. Furnace heat flux nonuniformity factor	114
Appendix E. Furnace-superheater interaction factor	116
Appendix F. Tube bank heat exchanger geometry	117
Appendix G. Sling tube geometry	122
Appendix H. Fin efficiency.....	127
Appendix I. Fluid property polynomials	130
Appendix J. Molar mass of ash.....	132
Appendix K. Furnace membrane wall geometry	133
Appendix L. Component model verification	135
L.1 Adiabatic flame	135
L.2 Furnace	138
L.3 Gas-to-steam tube bank heat exchanger	139
L.4 Rotary air heater	144
Appendix M. Designer specification boundary conditions and inputs.....	147
M.1 Fuel specification	147
M.2 Boiler model inputs and boundary conditions	148
M.3 Burner biases	149
M.4 Heat exchanger tube pressure loss parameters	150
Appendix N. Designed case simulation results	151
N.1 Full load case (100% MCR).....	151
N.2 Part load case (70% MCR).....	155
N.3 Error summary	159
Appendix O. Mass and energy balance at attemperator.....	160
Appendix P. Fuel specification	163
Appendix Q. Model boundary conditions and inputs extracted from plant data	167
Q.1 Full load case (100% MCR).....	167

Q.2 Dry mode part load cases (81% and 63% MCR)	169
Q.3 Wet mode part load case (35% MCR).....	171
Q.4 Burner biases	173
Q.5 Heat exchanger tube pressure loss parameters.....	173
Appendix R. Plant operation simulation results	174
R.1 Full load case (100% MCR)	174
R.2 Part load case 1 (81% MCR)	178
R.3 Part load case 2 (63% MCR)	182
R.4 Wet mode part load case (35% MCR)	186
R.5 Error summary.....	190
Appendix S. Case study boiler geometry	191
S.1 Boiler.....	191
S.2 Furnace	191
S.3 Furnace water wall (helical)	192
S.4 Burners	192
S.5 Superheaters	193
S.6 Reheaters.....	194
S.7 Economiser	195
S.8 Tubular air heater	196
S.9 Rotary air heater.....	197
Appendix T. Flownex boiler model layout	198
Appendix U. Flownex component model C# scripts	201

List of Figures

Figure 2.1 Temperature contours ($^{\circ}\text{C}$) measured by Acoustic Gas Temperature Measuring system and CFD simulation results for 200 MWe [13].....	3
Figure 2.2 Metal temperature contour (seen from the boiler front wall) of (a) Platen and (b) final superheater [4]	4
Figure 2.3 (a) Discretisation of the counter-flow heat exchanger and (b) Pulverised coal-fired boiler network [11].....	5
Figure 2.4 Cross-section through a typical PC boiler's membrane water wall geometry with isotherm contours at full load operational conditions [37].	8
Figure 2.5 Schematic of a typical tubular air heater [45]	11
Figure 2.6 Arrangement of rotary air heater [45]	11
Figure 3.1 General boiler heat transfer	13
Figure 3.2 Water flow path of the main steam cycle for (a) dry (Benson) and (b) wet mode operation	14
Figure 3.3 Water flow rate vs MCR	14
Figure 3.4 Pulverised coal particle transport during combustion	15
Figure 3.5 Zero-dimensional control volume of the combustion process.....	15
Figure 3.6 Finite control volume with one-dimensional flow	21
Figure 3.7 Illustration of a flow-through tube bundle [49].....	23
Figure 3.8 Zero-dimensional control volume of the flue gas inside the furnace	24
Figure 3.9 Direct radiation travel through the boiler	28
Figure 3.10 Mass and Energy Balance of heat exchangers.....	29
Figure 3.11 Mass and Energy Balance of tubular air heaters	33
Figure 3.12 1D cylindrical element	38
Figure 3.13 External slagging resistances for coal-oil fired superheaters/reheaters [33]	39
Figure 4.1 Case study boiler physical layout	42
Figure 4.2 Sling tube position	43
Figure 4.3 Sling tube screen layout.....	43

Figure 4.4 Process model diagram of the case study boiler	44
Figure 4.5 Adiabatic flame model (a) layout and (b) script logic diagram.....	45
Figure 4.6 Furnace model layout	46
Figure 4.7 Furnace heat transfer script logic diagram	46
Figure 4.8 Furnace membrane wall model layout	47
Figure 4.9 Heat exchanger discretisation	48
Figure 4.10 Heat exchanger external model layout.....	48
Figure 4.11 Heat exchanger model internal layout	49
Figure 4.12 Heat transfer equivalent thermal resistance series	50
Figure 4.13 Heat transfer series model	50
Figure 4.14 Heat exchanger script logic.....	51
Figure 4.15 Tubular air heater model layout	52
Figure 4.16 Tubular air heater heat transfer series model.....	53
Figure 4.17 Tubular air heater script logic	53
Figure 4.18 Rotary air heater model layout.....	54
Figure 4.19 Rotary air heater schematics [34]	55
Figure 4.20 Rotary air heater script logic flow	56
Figure 4.21 Boiler model Q'T diagram for simulation using designer specification at full load	57
Figure 4.22 Water/steam pressure result.....	58
Figure 4.23 Flue gas pressure results at 100% MCR	59
Figure 4.24 Flue gas pressure results at 70% MCR	59
Figure 4.25 Water/steam temperature results at 100% MCR.....	60
Figure 4.26 Water/steam temperature results at 70% MCR.....	61
Figure 4.27 Flue gas temperature results at 100% MCR	62
Figure 4.28 Flue gas temperature results at 70% MCR	62
Figure 4.29 Total boiler heat transfer results	63
Figure 4.30 Heat exchanger heat transfer results at 100% MCR.....	63

Figure 4.31 Heat exchanger heat transfer results at 70% MCR.....	64
Figure 5.1 Generator load reading for the full load case.....	65
Figure 5.2 Total boiler heat load	68
Figure 5.3 MS stream water/steam pressure results at 100% MCR.....	70
Figure 5.4 Water/steam temperature results at 100% MCR.....	71
Figure 5.5 Total heat transfer in each heat exchangers at 100% MCR.....	72
Figure 5.6 Q'T diagram of model results at 100% MCR	74
Figure 5.7 Metal temperature results at 100% MCR	75
Figure 5.8 Membrane wall tube and fin metal temperature results at 100% MCR	76
Figure 5.9 MS stream water/steam pressure results at 81% and 63% MCR	77
Figure 5.10 Water/steam temperature results at 81% MCR.....	78
Figure 5.11 Water/steam temperature results at 63% MCR	79
Figure 5.12 Total heat transfer in each heat exchangers at 81% MCR.....	80
Figure 5.13 Total heat transfer in each heat exchanger at 63% MCR	82
Figure 5.14 Q'T diagram of model results at 81% MCR	84
Figure 5.15 Q'T diagram of model results at 63% MCR	84
Figure 5.16 Metal temperature results at 81% MCR	85
Figure 5.17 Metal temperature results at 63% MCR	85
Figure 5.18 Membrane wall tube and fin metal temperature results at 81% MCR	87
Figure 5.19 Membrane wall tube and fin metal temperature results at 63% MCR	87
Figure 5.20 MS stream water/steam pressure results at 35% MCR.....	88
Figure 5.21 Water/steam temperature results at 35% MCR.....	90
Figure 5.22 Total heat transfer in each heat exchangers at 35% MCR.....	91
Figure 5.23 Cumulative heat transfer in MS and RS at 35% MCR	92
Figure 5.24 Q'T diagram of model results at 35% MCR	94
Figure 5.25 Metal temperature results at 35% MCR	94
Figure 5.26 Membrane wall tube and fin metal temperature results at 35% MCR	95

Figure 5.27 Air and flue gas pressure results at 100% MCR 96

List of Tables

Table 3.1 Coefficient c for rotary air heater rotational speed [31]	34
Table 3.2 External fouling resistances for superheaters and reheaters [33]	40
Table 5.1 Steady-state load operation periods.....	66
Table 5.2 Coal specifications used in the model (as-received basis).....	67
Table 5.3 Fouling/slugging resistance of the calibrated model	69
Table 5.4 RH2 exit steam pressure result at 100% MCR.....	70
Table 5.5 RH2 inlet and exit steam temperature results at 100% MCR	71
Table 5.6 Heat transfer results at 100% MCR.....	73
Table 5.7 Maximum metal temperatures of each heat exchanger at 100% MCR.....	75
Table 5.8 RC and SH3 steam pressure at 81% MCR.....	77
Table 5.9 RH2 exit steam pressure result at 81% and 63% MCR.....	77
Table 5.10 SH2 and RH2 exit steam temperature results at 81% MCR	78
Table 5.11 SH2 exit and RH2 inlet steam temperature results at 63% MCR.....	79
Table 5.12 Heat transfer results at 81% MCR.....	81
Table 5.13 Heat transfer results at 63% MCR	83
Table 5.14 Maximum metal temperature of each heat exchanger at 81% and 63% MCR	86
Table 5.15 RC and SH3 steam pressure at 35% MCR.....	89
Table 5.16 RH2 exit steam pressure result at 35% MCR.....	89
Table 5.17 SH1 exit and SH2 inlet steam temperature results at 35% MCR	90
Table 5.18 Heat transfer results at 35% MCR.....	92
Table 5.19 cumulative heat transfer results of MS and RS at 35% MCR	93
Table 5.20 Maximum metal temperatures of each heat exchanger at 35% MCR	95
Table 5.21 Out of confidence air and flue gas pressure results at 100% MCR.....	96
Table 5.22 Model results for flue gas pressure drop across the boiler at 100% MCR	97

List of Nomenclature

General symbols

Symbol	Description
A	Area [m^2]
A_{cond}	Conductive heat transfer area [m^2]
$A_{duct,open}$	Heat exchanger open duct area [m^2]
A_e	Cross-sectional area at the exit of one-dimensional control [m^2]
A_{ext}	Tube bundle external surface area [m^2]
$A_{ext,f}$	Finned tube bundle external surface area [m^2]
A_f	Finned tube bundle total fin area [m^2]
A_{flow}	Flow area [m^2]
$A_{flow,gaps}$	Flow area of rotary air heater gaps [m^2]
$A_{furnace}$	Planar furnace area [m^2]
$A_{furnace,exit plane}$	Furnace exit plane area [m^2]
A_{HT}	Solid elements' total heat transfer area in rotary air heater [m^2]
A_i	Cross-sectional area at the inlet of one-dimensional control [m^2] or Area of different furnace wall sections [m^2]
A_{int}	Tube bundle internal surface area [m^2]
A_p	Particle area [m]
$A_{surface}$	Surface area of the surfaces in contact with combustion products [m^2]
A_u	Finned tube bundle total unfinned area [m^2]
A_{WW}	Furnace water wall surface area [m^2]
$A_{WW,i}$	Surface area of specific section of the furnace water wall [m^2]
A'	Flame modification factor coefficient [-]
Adj	Fuel-dependent adjustment factor [-]
AF	Air-fuel ratio [$\frac{kg}{kg\ fuel}$]
AF_{theo}	Theoretical air-fuel ratio [$\frac{kg}{kg\ fuel}$]
a	Tube bundle transverse pitch to outer diameter ratio [-]
b	Tube bundle longitudinal pitch to outer diameter ratio [-]
b_{WW}	Furnace membrane wall tube axial pitch [m]
CI	Confidence interval/band [$^{\circ}C, MPa, kPa, MW$]
c	Tube bundle diagonal pitch to outer diameter ratio [-] or Rotational speed adjustment factor [-]

$c_{p,ash}$	Specific heat capacity of ash $[\frac{kJ}{kg \cdot K}]$
$c_{p,fuel}$	Specific heat capacity of fuel $[\frac{kJ}{kg \cdot K}]$
$D_{duct,0}$	Rotary air heater duct inner diameter $[m]$
$D_{duct,N}$	Rotary air heater duct outer diameter $[m]$
D_h	Hydraulic diameter $[m]$
$D_{h,rot}$	Rotary air heater compartment hydraulic diameter $[m]$
d	Diameter $[m]$
d_{fa}	Ash particle diameter $[m]$
d_i	Tube inner diameter $[m]$
d_o	Tube outer diameter $[m]$
HHV_{carbon}	High heating value for pure carbon $[\frac{J}{kg}]$
HHV_{fuel}	High heating value of fuel $[\frac{J}{kg}]$
f	Frictional loss factor $[\frac{1}{m}]$
$f_{discharge}$	Compartment discharge frequency $[-]$
$f_{rad\ loss}$	Radiation loss fraction $[-]$
g	Gravitational acceleration $[\frac{m}{s^2}]$
H	Height $[m]$
$H_{furnace}$	Furnace height $[m]$
$H_{r,eff}$	Effective flame height $[m]$
$H_{row,i}$	Height of each burner $[m]$
H_{WW}	Furnace wall sectional heights $[m]$
h	Specific enthalpy $\frac{J}{kg}$ or Heat transfer coefficient $[\frac{W}{m^2 \cdot K}]$
h_{air}	Specific enthalpy of air $[\frac{J}{kg}]$
h_{BA}	Specific enthalpy of bottom ash $[\frac{J}{kg}]$
$h_{combine,ext}$	Combined heat transfer coefficient for external flow over the tube bundles $[\frac{W}{m^2 \cdot K}]$
$h_{combine,int}$	Combined heat transfer coefficient for flow inside the tubes $[\frac{W}{m^2 \cdot K}]$
$h_{conv,ext}$	Heat transfer coefficient for forced convection of external flow over the tube bundles $[\frac{W}{m^2 \cdot K}]$
$h_{conv,int}$	Heat transfer coefficient for forced convection of flow inside tube bundles $[\frac{W}{m^2 \cdot K}]$
$h_{conv,vertical\ WW}$	Convective heat transfer coefficient to the vertical water wall $[\frac{W}{m^2 \cdot K}]$
$h_{fg,exit}$	Specific enthalpy of flue gas exiting the combustion zone $[\frac{J}{kg}]$
h_{flame}	Specific enthalpy of flame $[\frac{J}{kg}]$

h_{fuel}	Specific enthalpy of fuel [$\frac{J}{kg}$]
$h_{G+P rad}$	Gas and particle radiation heat transfer coefficient [$\frac{W}{m^2 \cdot K}$]
$h'_{G+P rad}$	Gas and particle radiation heat transfer coefficient with pre-cavity correction [$\frac{W}{m^2 \cdot K}$]
$h'_{G+P rad, ext}$	Gas and particle radiation heat transfer coefficient to the tube bundle outer wall [$\frac{W}{m^2 \cdot K}$]
$h'_{G+P rad, int}$	Gas and particle radiation heat transfer coefficient to the tube bundle inner wall [$\frac{W}{m^2 \cdot K}$]
$h'_{G+P rad, vertical WW}$	Gas and particle radiation heat transfer coefficient [$\frac{W}{m^2 \cdot K}$]
h_{gas}	Specific enthalpy of the gas in two-phase mixture [$\frac{J}{kg}$]
h_i	Specific enthalpy of each combustion products gas constituent [$\frac{J}{kg}$]
h_{liq}	Specific enthalpy of the liquid in two-phase mixture [$\frac{J}{kg}$]
$h_{products, exit}$	Specific enthalpy of combustion products at the exit [$\frac{J}{kg}$]
$h_{products, in}$	Specific enthalpy of combustion products at the inlet [$\frac{J}{kg}$]
$h'_{combine, int}$	Combined heat transfer coefficient for flow inside rotary air heater [$\frac{W}{m^2 \cdot K}$]
$h'_{conv, int}$	Heat transfer coefficient for forced convection of flow inside rotary air heater [$\frac{W}{m^2 \cdot K}$]
$h'_{products, in}$	Specific enthalpy of the combustion products without the bottom ash [$\frac{J}{kg}$]
h_{mix}	Specific enthalpy of the gas mixture [$\frac{J}{kg}$]
$\bar{h}_{mix}$	Specific enthalpy of the two-phase mixture [$\frac{J}{kg}$]
$\bar{h}_{mix, oe}$	Specific stagnation enthalpy of two-phase mixture at the exit one-dimensional control volume [$\frac{J}{kg}$]
$\bar{h}_{mix, oi}$	Specific stagnation enthalpy of two-phase mixture at the inlet one-dimensional control volume [$\frac{J}{kg}$]
h_o	Specific stagnation enthalpy [$\frac{J}{kg}$]
$h_{o, e}$	Specific stagnation enthalpy of fluid at the exit one-dimensional control volume [$\frac{J}{kg}$]
$h_{o, i}$	Specific stagnation enthalpy of fluid at the inlet one-dimensional control volume [$\frac{J}{kg}$]
$h_{vertical WW}$	Heat transfer coefficient to the vertical water wall [$\frac{W}{m^2 \cdot K}$]
K	Secondary loss coefficient [-]
K_{coke}	Coke absorption coefficient [$\frac{1}{m}$]
$K_{emi, G}$	Gas absorption coefficient for tri-atomic gases [$\frac{1}{m}$]
K_{ext}	Secondary loss coefficient of gas flow through the tube bundle [-]
K_{gap}	Gap leakage discharge coefficient [-]
k	Thermal conductivity [$\frac{W}{m \cdot K}$]

$L_{c,0}$	Width of the rotary air heater compartments closest to the centre [m]
$L_{c,N}$	Width of the rotary air heater compartments furthest to the centre [m]
L_P	Ash loading [$\frac{kg}{m^3}$]
l	Longitudinal Length of control volume [m]
l_b	Total depth of the heat exchanger cavity [m]
l_{bundle}	Heat transfer characteristic length of the tube bundle [m]
$l_{products}$	Heat exchanger products control volume length [m^2]
l_{tube}	Length of the tube [m^2]
l_R	Depth of heat exchanger pre-cavity [m]
l_{rot}	Rotary air heater length [m]
M	Molar mass [$\frac{kg}{mole}$] or Flame modification factor [-] Mach number [-]
M_{air}	Molar mass of air [$\frac{kg}{mole}$]
M_{G+P}	Molar mass of the gas and particle mixture [$\frac{kg}{mole}$]
M_{H_2O}	Molar mass of water [$\frac{kg}{mole}$]
M_i	Molar mass of each flue gas constituent [$\frac{kg}{mole}$]
M_j	Molar mass of each specie in fuel [$\frac{kg}{mole}$]
$\dot{m}$	Mass flow rate of ash [$\frac{kg}{s}$]
$\dot{m}_{air}$	Mass flow rate of supply air [$\frac{kg}{s}$]
$\dot{m}_{air,leak}$	Mass flow rate of rotary air leakage [$\frac{kg}{s}$]
$\dot{m}_{air,leak,carryover}$	Carryover leakage [$\frac{kg}{s}$]
$\dot{m}_{air,leak,gap}$	Gap leakage [$\frac{kg}{s}$]
$\dot{m}_{ash}$	Mass flow rate [$\frac{kg}{s}$]
$\dot{m}_{BA}$	Mass flow rate of bottom ash [$\frac{kg}{s}$]
$\dot{m}_{exit}$	Mass flow rate of fluid at the exit of one-dimensional control volume [$\frac{kg}{s}$]
$\dot{m}_{fg,exit}$	Mass flow rate of flue gas [$\frac{kg}{s}$]
$\dot{m}_{fuel}$	Mass flow rate of fuel [$\frac{kg}{s}$]
$\dot{m}_{fuel \text{ per row}}$	Fuel consumption per burner in a row [$\frac{kg}{s}$]
$\dot{m}_{G+P}$	Gas and particle mass flow rate [$\frac{kg}{s}$]
$\dot{m}_{gas}$	Mass flow rate of gas in two-phase mixture [$\frac{kg}{s}$]
$\dot{m}_i$	Mass flow rate of each flue gas constituent [$\frac{kg}{s}$]

$\dot{m}_{in}$	Mass flow rate of fluid at the inlet one-dimensional control volume [$\frac{kg}{s}$]
$\dot{m}_j$	Mass flow rate of individual species in fuel [$\frac{kg}{s}$]
$\dot{m}_{products}$	Mass flow rate of combustion products [$\frac{kg}{s}$]
$\dot{m}_{products,exit}$	Mass flow rate of combustion products at the exit [$\frac{kg}{s}$]
$\dot{m}_{products,in}$	Mass flow rate of combustion products at the inlet [$\frac{kg}{s}$]
$\dot{m}_{liq}$	Mass flow rate of liquid in two-phase mixture [$\frac{kg}{s}$]
$\dot{m}_{mix}$	Mass flow rate of two-phase mixture [$\frac{kg}{s}$]
$\dot{m}_{UC}$	Mass flow rate of unburnt carbon [$\frac{kg}{s}$]
N	Number of tubes [-]
N_l	Total number of tubes in the flow/longitudinal direction [-]
N_{MR}	Number of tubes in the flow/longitudinal direction resisting the flow [-]
Nu_{bundle}	Nusselt number for forced convection of external flow over the tube bundles [-]
Nu_{int}	Nusselt number for internal forced convection [-]
n	Molar amount [-] Measurement sample size [-]
$n_{air,th}$	Molar amount of the theoretical supply air [-]
n_c	Molar amount of carbon in fuel [-]
n_{CO_2}	Molar amount of carbon dioxide produced by oxidations [-]
n_{comp}	Number of compartments per radial plane [-]
n_{fw}	Total number of tubes in one tube bank pass [-]
n_H	Molar amount of hydrogen in fuel [-]
$n_{H_2O,air}$	Molar amount of moisture in air [-]
$n_{H_2O,fuel}$	Molar amount of moisture in fuel [-]
$n_{H_2O,product}$	Molar amount of moisture produced by oxidations [-]
$n_{H_2O,total}$	Molar amount of the total moisture in the combustion products [-]
n_i	Molar amount of each combustion products gas constituent [-]
n_j	Molar amount of each species in the fuel [-]
n_N	Molar amount of nitrogen in fuel [-]
n_{N_2}	Molar amount of nitrogen gas in air [-]
$n_{O_2,excess}$	Molar amount of excess oxygen [-]
$n_{O_{fuel}}$	Molar amount of oxygen in fuel [-]
n_s	Molar amount of sulphur in fuel [-]
n_{SO_2}	Molar amount of sulphur dioxide in fuel [-]
P_{wet}	Heat exchanger products wetted perimeter [m]
p	Static pressure of the fluid [Pa]

p_{G+P}	Fluid static pressure of the gas and particle flow [Pa]
p_i	Tri-atomic gas partial pressures [Pa]
p_o	Fluid stagnation pressure [Pa]
p_{oe}	Fluid stagnation pressure at the exit of one-dimensional control [Pa]
p_{oi}	Fluid stagnation pressure at the inlet of one-dimensional control [Pa]
Δp	Pressure difference [Pa]
Δp_{ext}	Pressure drop for the external fluid flow over the tube bundles [Pa]
Q_{abs}	Particle absorptivity [-]
Q_{scat}	Back-scattering efficiency [-]
$\dot{Q}$	Heat transfer [W]
$\dot{Q}_{back\ rad}$	Back panel radiation [W]
$\dot{Q}_{combined,ext}$	Total heat transfer of external flow over the tube bundles [W]
$\dot{Q}_{combined,int}$	Total heat transfer of internal flow over the tube bundles [W]
$\dot{Q}_{combustion}$	Heat released during combustion [W]
$\dot{Q}_{cond}$	Conductive heat transfer [W]
$\dot{Q}_{direct\ rad\ furnace\ exit}$	Radiation heat transfer through the exit plane of the furnace [W]
$\dot{Q}_{direct\ rad\ bypass,next}$	Bypassed direct radiation to following surface [W]
$\dot{Q}_{direct\ rad\ bypass,previous}$	Bypassed direct radiation from previous surface [W]
$\dot{Q}_{G+P\ rad}$	Gas and particle radiation of combustion products [W]
$\dot{Q}_{Losses}$	Miscellaneous heat losses [W]
$\dot{Q}_{rad\ total}$	Total radiative heat transfer in the furnace [W]
$\dot{Q}_{rad\ to\ WW}$	Radiation heat transfer to the furnace water wall/evaporator [W]
$\dot{Q}_{steam}$	Steam heat load [MW]
$\dot{Q}_{verticle\ WW}$	Heat transfer to the heat exchanger vertical water wall [W]
$\dot{Q}_{WW,i}$	Radiation projected on a specific section of the furnace water wall [W]
$\dot{Q}_{UC}$	Heat loss due to unburnt carbon [W]
$\dot{q}_{furnace}$	Uniform radiative heat flux projected onto furnace surfaces [$\frac{W}{m^2}$]
$\dot{q}_{WW}$	Uniform radiative heat flux projected onto the furnace water walls [$\frac{W}{m^2}$]
R	Radius [m^2]
R_{th}	Thermal resistance [$\frac{K}{W}$]
$R_{slagging,ext}$	Thermal resistance of slagging [$\frac{K}{W}$]
$R_{fouling,ext}$	Thermal resistance of fouling [$\frac{m^2 K}{W}$]
$R_{th,parallel}$	Thermal resistance of materials in parallel [$\frac{m^2 K}{W}$]
$R_{th,series}$	Thermal resistance of materials in series [$\frac{K}{W}$]
Re	Reynold's number [-]

$Re_{products}$	Reynold's number of combustion products [-]
r_v	Tube bundle velocity ratio [-]
S	Mean bean length [m]
$S_{furnace}$	Mean bean length of furnace [m]
$S_{tube,bundle}$	Mean bean length of tube bundle [m]
s	Tube bundle pitch [m]
s_d	Tube bundle diagonal pitch [m]
s_l	Tube bundle longitudinal pitch [m]
s_t	Tube bundle transverse pitch [m]
T	Temperature [$^{\circ}C$]
T_{BA}	Temperature of bottom ash [$^{\circ}C$]
T_{ext}	Heat exchanger ash deposition surface temperature [$^{\circ}C$]
T_{fuel}	Temperature of pulverised coal fed into the burner [$^{\circ}C$]
T_{flame}	Adiabatic flame temperature [$^{\circ}C$]
$T_{products,cavity}$	Combustion product temperature in the upstream cavity [K]
T_{G+P}	Gas and particle flow temperature [$^{\circ}C$]
$\bar{T}_{H_2O}$	Mean water/steam temperature in the boiler tubes [$^{\circ}C$]
T_{int}	Tube bundle internal surface temperature [$^{\circ}C$]
T_o	Fluid stagnation temperature [$^{\circ}C$]
$T_{products,exit}$	Temperature of combustion products at the exit [$^{\circ}C$]
$T_{products,in}$	Temperature of combustion products at the inlet [$^{\circ}C$]
$\bar{T}_{products}$	Mean product temperature in the heat exchanger [$^{\circ}C$]
$T_{surface}$	Temperature of the surfaces in contact with combustion products [$^{\circ}C$]
U_0^T	Tangential velocity of the rotary air heater compartments closest to the centre [$\frac{m}{s}$]
U_N^T	Tangential velocity of the rotary air heater compartments furthest to the centre [$\frac{m}{s}$]
u	Specific internal energy of fluid [$\frac{J}{kg}$]
u_{gas}	Specific internal energy of gas in two-phase mixture [$\frac{J}{kg}$]
u_{liq}	Specific internal energy of liquid in two-phase mixture [$\frac{J}{kg}$]
$\bar{u}_{mix}$	Specific internal energy of two-phase mixture [$\frac{J}{kg}$]
V	Volume [m^3]
$V_{furnace}$	Volume of the furnace [m^3]
V_{flow}	Volume of the flow region in the rotary air heater [m^3]
$\dot{V}$	Volume flow rate [$\frac{m^3}{s}$]
$\bar{V}_{rotor}$	Rotary air heater compartment volume [m^3]

$\overline{VC}$	Mean overall heat capacity per kilogram fuel [$\frac{J}{kg \cdot K}$]
v	Flow velocity [$\frac{m}{s}$]
v_{xe}	Flow velocity of fluid at the exit of one-dimensional control volume [$\frac{m}{s}$]
v_{xi}	Flow velocity of fluid at the inlet one-dimensional control volume [$\frac{m}{s}$]
v_{int}	Mean interstitial velocity of gas flow through the tube bundle [$\frac{m}{s}$]
$\dot{W}$	Rate of fluid work [W]
w	Mass ratio [$\frac{kg}{kg}$]
w_{ash}	Mass ratio of ash in combustion products [$\frac{kg}{kg \text{ products}}$]
w_{fg}	Mass ratio of flue gas in combustion products [$\frac{kg}{per \text{ kg products}}$]
$w_{furnace}$	Furnace height [m]
w_{UC}	Mass ratio of unburnt carbon in combustion products [$\frac{kg}{kg \text{ products}}$]
X	Molar fraction [$\frac{mole}{mole}$]
$X_{H_2O,air}$	Molar fraction of moisture in air [$\frac{mole}{mole \text{ air}}$]
X_i	Molar fraction of each combustion products gas constituent [$\frac{mole}{mole \text{ products}}$]
X_r	Relative height of the highest temperature zone in the furnace [-]
ΔX	Flame core position correction factor [-]
x_{air}	Heat transfer area fractions of air rotary air heater [-]
x_{ang}	Angular coefficient of furnace wall [-]
$x_{products}$	Heat transfer area fractions of air rotary air heater [-]
$\bar{x}$	Two-phase mixture weight fraction [-] Measurement mean [$^{\circ}C, MPa, kPa, MW$]
Y	Mass fraction [$\frac{kg}{kg}$]
Y_{ash}	Mass fraction of ash in fuel [$\frac{kg}{kg \text{ fuel}}$]
Y_i	Mass fraction of each combustion products gas constituent [$\frac{kg}{kg \text{ products}}$]
Y_j	Mass fraction of individual species in fuel [$\frac{kg}{kg \text{ fuel}}$]
Y_{UC}	Mass fraction of unburnt carbon in fuel [$\frac{kg}{kg \text{ fuel}}$]
z	Elevation [m]
z_e	Elevation of fluid at the exit of one-dimensional control volume [m]
z_i	Elevation of fluid at the inlet of one-dimensional control volume [m]

Greek symbols

Symbol	Description
α_{air}	Excess air ratio [-]
β	Interaction factor between the heat exchanger and the furnace [-]
β_{G+P}	Calculation parameter for gas and particle emissivity [-]
$\beta_{platten}$	Interaction factor between the platen superheater and the furnace [-]
γ	Radiation scattering factor [-]
ζ	Drag coefficient [-]
η_f	Fin efficiency [-]
η_k	Heat flux nonuniformity factor [-]
$\eta_{k,furnace,exit}$	Heat flux nonuniformity factor for the furnace exit plane [-]
$\eta_{k,i}$	Heat flux nonuniformity factor on a specific section of the furnace water wall, taken from the relative height of the section's mid-point [-]
η_o	Extended heat transfers surface overall efficiency [-]
ϵ	Emissivity [-]
ϵ_{ash}	Emissivity of ash [-]
$\epsilon_{furnace}$	Furnace emissivity [-]
ϵ_G	Effective of tri-atomic gases [-]
ϵ_{G+P}	Effective gas-particle emissivity of the products throughout the boiler [-]
$\epsilon_{G+P,f}$	Effective gas-particle emissivity in the furnace [-]
$\epsilon_{surface}$	Surfaces emissivity [-]
ν	Specific volume of fluid [$\frac{m^3}{kg}$]
ν_{gas}	Specific volume of gas in two-phase mixture [$\frac{m^3}{kg}$]
ν_{liq}	Specific volume of liquid in two-phase mixture [$\frac{m^3}{kg}$]
$\bar{\nu}_{mix}$	Specific volume of two-phase mixture [$\frac{m^3}{kg}$]
ξ	Furnace surface fouling factor [-]
ξ_{sweep}	Non-uniform sweeping factor [-]
ρ	Density [$\frac{kg}{m^3}$]
$\bar{\rho}$	Average density of the fluid [$\frac{kg}{m^3}$]
ρ_{air}	Density of air [$\frac{kg}{m^3}$]
ρ_{fa}	Particle density of ash [$\frac{kg}{m^3}$]
ρ_e	Density of fluid at the exit one-dimensional control volume [$\frac{kg}{m^3}$]
ρ_{gas}	Density of gas in two-phase mixture [$\frac{kg}{m^3}$]

ρ_i	Density of fluid at the inlet one-dimensional control volume $[\frac{kg}{m^3}]$
ρ_{liq}	Density of liquid in two-phase mixture $[\frac{kg}{m^3}]$
ρ_{mix}	Density of two-phase mixture $[\frac{kg}{m^3}]$
$\rho_{mix,e}$	Density of two-phase mixture at the exit one-dimensional control volume $[\frac{kg}{m^3}]$
$\rho_{mix,i}$	Density of two-phase mixture at the inlet one-dimensional control volume $[\frac{kg}{m^3}]$
σ	Standard deviation [$^{\circ}C, MPa, kPa, MW$]
σ_o	Stefan-Boltzmann's constant $[\frac{W}{m^2 \cdot K^4}]$
ϕ_e	Angular coefficient of direct radiation absorbed from the inlet plane to the outlet plane of a heat exchanger [-]
ϕ_f	Empirical multiplier for the frictional component [-]
ϕ_{G+P}	Optical thickness of the gas-particulate dispersion [-]
ψ	Furnace wall efficiency factor [-]
$\bar{\psi}$	Average furnace wall efficiency factor [-]
ω	Specific humidity $[\frac{kg}{kg \text{ air}}]$

Acronyms and Abbreviations

ABMA: American Boiler Manufacturer Association
AFT: Adiabatic flame temperature
AH: Air heater
ASME: American Society of Mechanical Engineering
CAD: Computer-aided design
PCFB: Pulverised Coal Fired Boiler
CFD: Computational Fluid Dynamics
CFPP: Coal-Fired Power Plants
CI: Confidence interval
Eco: Economiser
FEGT: furnace exit gas temperature
FET: Furnace Exit temperature
FG: Flue Gas
HPT: High-pressure turbine
HTR: Heater
IPT: intermediate-pressure turbine
MCR: Maximum Continuous Rating
MS: Main Steam
NIST: National Institute of Standards and Technology
PA: Primary air
RC: Ring collector
RE: Reheater
RH: Re-heater
RS: Re-heat Steam
rpm: revolution per minute
SA: Secondary air
SH: Superheater
WW: Water wall

1. Introduction

1.1 Background

Since 2011, renewable energy sources have been introduced to the South African grid to mitigate greenhouse gas emissions [1]. However, renewable energy sources such as wind and solar are inherently intermittent. Thus, to integrate more renewable energy sources into the national grid, there is a need for existing coal-fired power plants (CFPPs) to improve their operational flexibility.

The three main features that characterise operational flexibility for a power plant are operational bandwidth, ramp rate and start-up time. A flexible power plant should operate with large bandwidth, a fast ramp rate and a short start-up time [2].

In South Africa, the design and operating philosophies of existing CFPPs are biased towards baseload operation; this, therefore, limits the allowable operating bandwidth. Operating these CFPPs under loads outside the designed range substantially decreases their lifetime [2]. The primary concern is that thick-walled components are highly susceptible to thermal stress, which is exacerbated during flexible operations.

A study has shown that superheater tubes tend to overheat during low load operation [3]. Superheater tube failures can contribute up to 40% of emergency shutdowns [3][4]. This vulnerability highlights a need to study boilers under low load operations to improve plant flexibility. This especially applies to once-through boilers, since the majority of Eskom CFPPs use once-through type boilers [5].

Experimental investigations on operating plants are impractical due to the risks, costs and complexity involved. Scaled experimental studies [6]–[9] mainly aim to study the specific properties of once-through boiler components, which does not provide insight into the boiler performance on a system level.

A practical solution is a numerical tool to simulate the heat transfer phenomena in the boiler under various loads. This model needs to provide the necessary details for boiler operators to investigate specific components at risk and the overall boiler system performance. The practical applicability of such tools requires a balance between accuracy, computational cost, and component resolution.

1.2 Research question

The aim of the study is the development of a thermofluid network-based modelling methodology for a tower-type once-through boiler that can be used to provide insight into the boiler performance

on a system level at full- and part-load. The resultant model needs to have acceptable accuracy, low computational cost, and requisite component resolution for the boiler operator to gain insight into the boiler operation under off-design conditions.

The boiler tube surface fouling and slagging are complex phenomena to model and are highly influential in the heat transfer in the boiler. The degree of fouling is inter alia related to the ash content in the fuel. The existing literature has limited information on fouling and slagging for high ash coal used in South African CFPPs. This lack of information poses a challenge in modelling the heat transfer in the boiler.

1.3 Research objective

The objectives of the project are:

- Develop an integrated thermofluid process model of a subcritical tower-type once-through boiler in quasi-steady-state operation.
- Verify the integrated boiler model.
- Validate the boiler model using the design specifications.
- Calibrate the boiler model using plant data at 100% Maximum Continuous Rating (MCR).
- Apply the calibrated boiler model to different part-load operating conditions (81%, 63% and 35% MCR).

The model does not include any control features associated with the transient operation since it only includes the steady-state boiler performance.

1.4 Dissertation Overview

In chapter two, the relevant literature on different boiler models is reviewed to select the appropriate modelling methodology. The review then goes into the literature on different boiler component models. Chapter three details all the relevant theories and mathematical modelling of each boiler component model. Chapter four discusses the model implementation and verification. Each boiler component was first verified individually and then integrated into the network model of the entire boiler. This integrated boiler model is then verified using the mass and energy balance of the system as well as the design specifications. Chapter five goes over the model calibration and validation. The calibration uses the plant running data at full load, and the calibrated model is then applied to various partial loads. After analysing the results, chapter six concludes the study.

2. Literature review

Pulverised coal-fired boiler (PCFB) modelling requires analysing the heat transfer, fluid flow, multi-stage chemical reactions, and other complex phenomena [10]. Typically, computational fluid dynamics (CFD) or one-dimensional thermofluid network models (process models) are used to simulate PCFB operations.

2.1 CFD modelling

CFD models discretise the fluid region into numerous two or three-dimensional finite volumes. The model then solves mass, momentum, and energy conservation equations together with fluid property relationships to determine each finite volume's unknown fluid properties such as velocity, temperature, and chemical composition. CFD models can provide detailed results at high accuracy [7] but are generally associated with high computational costs [11]. For that reason, CFD models are often used for detailed analysis of PCFB components with complex flow profiles such as furnaces, superheaters and reheaters.

Combustion and heat transfer in a PCFB furnace are complex processes. Due to this, modelling these processes in detail requires a sophisticated tool like CFD [8]. Modlinski et al. [12] model a front wall fired PCFB to study the heat distribution in the furnace. The boiler CFD model was applied to two different load cases, 200 MW (full-load) and 135 MW (67.5% load). The results show that the model predicted the average furnace exit gas temperature (FEGT) within 3% of the measured value at full load and 2.5% when operating at 67.5% load. The results also captured the non-uniform temperature distribution in the gas, as shown in Figure 2.1.

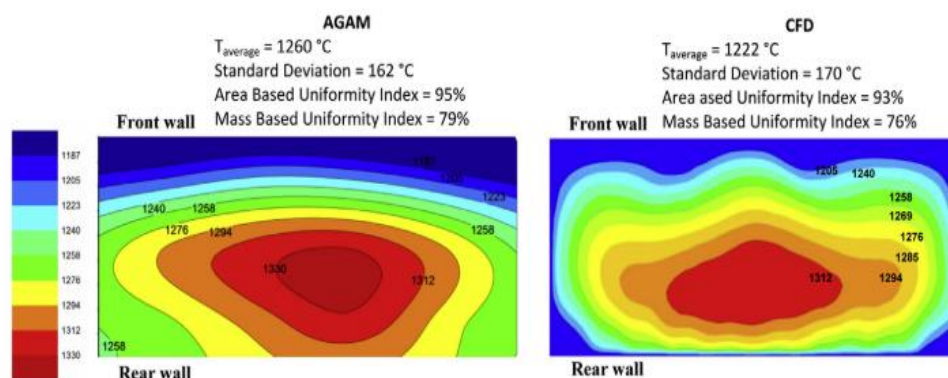


Figure 2.1 Temperature contours (°C) measured by Acoustic Gas Temperature Measuring system and CFD simulation results for 200 MWe [13]

CFD models are also used to study the heat transfer and temperature distribution in heat exchanger components. Laubscher and Rousseau [4] have modelled a 620 MW subcritical two-pass drum boiler using commercial software FLUENT to study heat exchanger elements. The focus of the study is on heat uptake of the heat exchanger elements during low-load operation. The model was applied at 99%, 81% and 60.5% load, and it was able to capture the metal temperature maldistribution, as shown in Figure 2.2. This temperature profile was validated using site data.

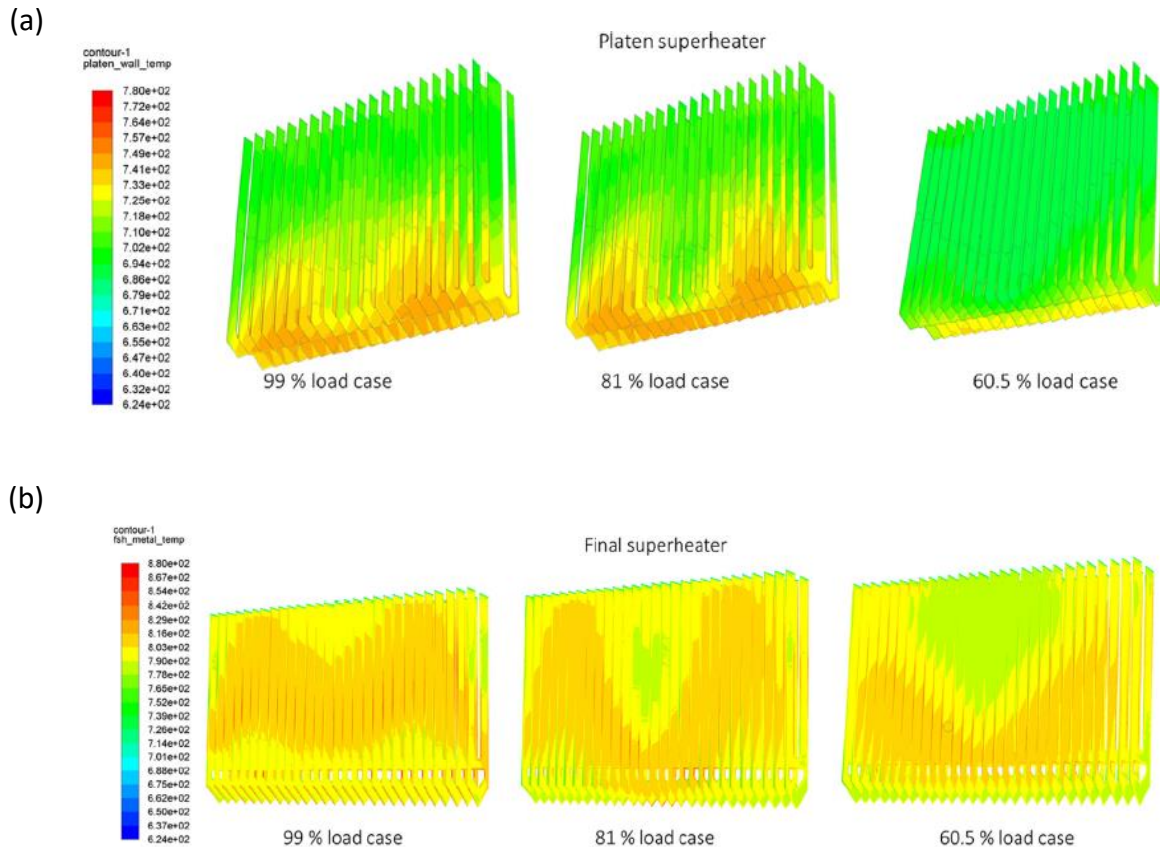


Figure 2.2 Metal temperature contour (seen from the boiler front wall) of (a) Platen and (b) final superheater [4]

For the 60.5% boiler load case, the error increased. Laubscher and Rousseau suggested that this could be attributed to heat transfer maldistribution caused by soot blowing operations. This process results in higher thermal resistance in the areas where the ash deposition layer has not been removed.

Chen et al. [14] have modelled a supercritical PCFB by coupling CFD and process models. The study used FLUENT to model combustion and flue gas flow. At the same time, the steam flow in the heat exchanger tubes was modelled by a process model. The results agree well with the plant measurements and showed that the tube's exit steam temperature is affected by the change in ash deposition temperature.

Laubscher and Rousseau [15] studied the influence of ash particle radiation in a PCFB that fires high ash fuels using two different models for the radiation heat transfer. The high-ash fuel-firing PCFB was studied using a CFD model of a second stage reheater. The model uses empirical data of ash absorption and scattering efficiencies. This correlation considered the effects of variable particle sizes on ash emissivity and scattering factors [16]. The results agree well with the plant measurements, but the radiation emission values are lower than those reported by other literature that assumes constant particle emissivity [17], [18].

The literature has shown that, when applied correctly, CFD models can accurately simulate thermofluid processes of PCFB components and provide insights into the detail flow and heat transfer processes.

2.2 One-dimensional thermofluid network modelling

Process models discretise each flow component in the PCFB as a one-dimensional finite control volume, as shown in Figure 2.3 (a). These flow components can then be linked into a network, as shown in Figure 2.3 (b).

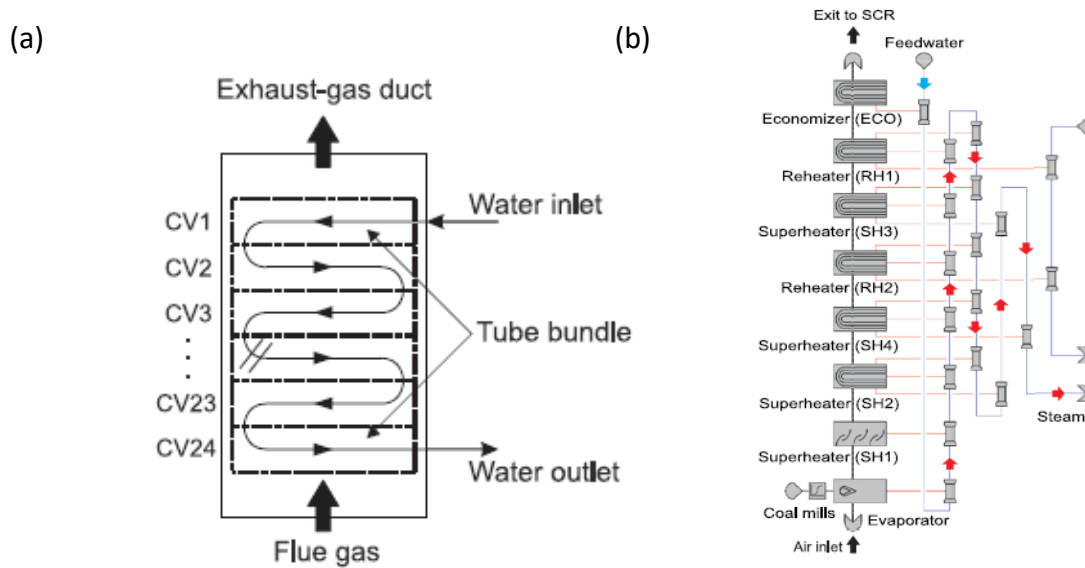


Figure 2.3 (a) Discretisation of the counter-flow heat exchanger and (b) Pulverised coal-fired boiler network [11]

The model then determines the unknown fluid properties at discrete locations by solving the mass, momentum and energy conservation equations [11] together with the fluid property relations and component characteristic equations. The simplicity of process models makes them a widely used tool to study the steady-state and dynamic behaviour of the PCFB on a system level.

Deng et al. [19] have presented a dynamic process model of a 600MW supercritical once-through boiler to study the response during CFPP start-up. The model was first validated under steady-state

and agreed with the design specifications. The validated model was then used to simulate two cold start-up processes, which agreed well with the designed value.

In the study done by Starkloff et al. [20], a detailed process model of a large scale CFPP with a once-through boiler was developed using commercial software APROS. The model simulated plant performance at off-design conditions ranging from 27% to 100% load. The model results show that the relative errors in temperature, pressure, and mass flow rate were less than 5% compared to site data. Based on this evidence, Starkloff et al. concluded that the process model can realistically simulate CFPP operation with minimal inputs. This makes them an adequate tool for research regarding innovative features that improve CFPP flexibility.

Ritcher, Oeljeklaus and Görner [21] applied this idea by investigating the effect of integrating a steam accumulator into a CFPP using a process model. The model was first validated using site data, and the results showed sufficient accuracy. The study then integrated the steam accumulator into the model to study plant response. The results showed that this retrofitting feature significantly improved the case study plant's flexibility.

Another use of the PCFB process model is to study plant controls appropriate for flexible operation. In a study done by Taler et al. [22], a dynamic model of a tower type boiler was developed. This study aimed to improve the control features of a new Polish supercritical power plant under construction. The results showed a sufficient agreement with the design specifications and captured the transient response of the boiler under rapid load change.

Niu et al. [23] developed a process model of a 600 MW supercritical once-through boiler-turbine unit. The model was developed to study control systems for low load operations (30~40%). The model simplified the components (superheaters, turbines) into collective modules and neglected transient effects in the flue gas. Temperature and pressure results showed mean relative errors within 5% compared to the plant measurements.

Wang et al. [24] developed a process model of a 300MW subcritical CFPP boiler-turbine unit to study the coordinate control systems for fast cut back. The model's pressure and motor power fluctuation at full load showed sufficient accuracy compared to the site data. However, the results are less accurate in the 50% load case. The additional finding of the study was that at 50% load, the drum water level is likely to be out of the safe operating range for fast cut back.

The literature has shown that process models can simulate the PCFBs' operations with sufficient accuracy when appropriate correlations are applied. These models can evaluate a PCFB system's cycle performance and help improve CFPP flexibility.

2.2.1 Combustion modelling

Modelling pulverised coal combustion is a complex task since it requires one to analyse multiple processes such as devolatilisation, char combustion and volatile combustion [25]. For that reason, researchers have been developing network models that can capture the chemical kinetics of pulverised coal combustion, as seen from the review by Sankar et al. [25]. Although important, the combustion process in system-level PCFB process models is usually simplified, as evident by the literature [11], [19], [26], [27]. The simplified combustion model assumes complete infinitely fast combustion with a specified value of unburned carbon and excess air, and determines the mass of each product species from the molecular balance equations of the chemical reaction. The heat released during the combustion process is determined using the fuel's heating value, an empirical coefficient. Hajebzadeh et al. [28] modelled the combustion process of a 320MW tangential fired PCFB using a similar approach, which is referred to as equilibrium combustion, and the results show good agreement compared to NASA CEA software [29].

2.2.2 Furnace modelling

Studies [30]–[32] have shown that heat transfer to the evaporator walls in furnaces is mainly through radiation, with the convective heat transfer contributing only about 5% of total heat. Traditionally, boiler designers used semi-empirical methods developed using tests carried out in pilot plants and utility boilers to predict the furnace exit gas temperature (FEGT).

One such approach is Hottel's method that calculates the overall radiation heat released from the flue gas using gas and evaporator wall temperatures [33]. This method is also known as the direct approach [34]. Although the direct approach is fundamentally correct, it is troublesome since it introduces several parameters that are hard to determine [34]. For that reason, PCFB process models often use the Gurvich method [35], also known as the projected approach, to calculate the heat transfer in the furnace.

In Hajebzadeh et al.'s [28] study, the Gurvich method was compared to the direct method. In the study, the FEGT calculated by Gurvich's method was generally higher than the FEGT calculated by the direct method. Hajebzadeh et al. then pointed out that many experimental studies have shown that the Gurvich method can predict FEGT with an accuracy of ± 30 K.

However, Zhang et al. [33] pointed out that the experimental data used in Gurvich's method is not suitable for modern-day PCFBs with large capacities. As a result, Zhang et al. proposed a modified Gurvich method that includes a correction for an uneven temperature field.

Zhang et al. [36] used this approach to model the furnace in a 1000 MW ultra-supercritical CFPP in eastern China. Zhang et al. point out that measuring the temperature directly in high-temperature areas such as the furnace poses many challenges. Thus, the model was developed to approximate the FEGT. The model was validated using the historical data of the case study boiler. The relative errors between the calculated and the designed FEGT were less than 3.5%. Although this method proposed by Zhang showed promising results, the current standard method for modelling furnace heat transfer is still the standard Gurchich method.

Another aspect to consider when modelling the furnace heat transfer is the influence of the participating media (flue gas and solid particles) on radiation. Although the standard Gurchich method accounts for the radiation influence of the participating media, it does not consider the radiation scattering effect due to high ash-loading.

For South African CFPPs, the high ash content in the fuel makes it essential to consider the radiation scattering effect. In the journal Heat Atlas [12], the correlations to model the radiation scattering effect due to different degrees of ash-loading are discussed in detail.

The evaporator of the furnace is generally a membrane wall with varying metal temperatures with heat transfer in different regions, as shown in Figure 2.4.

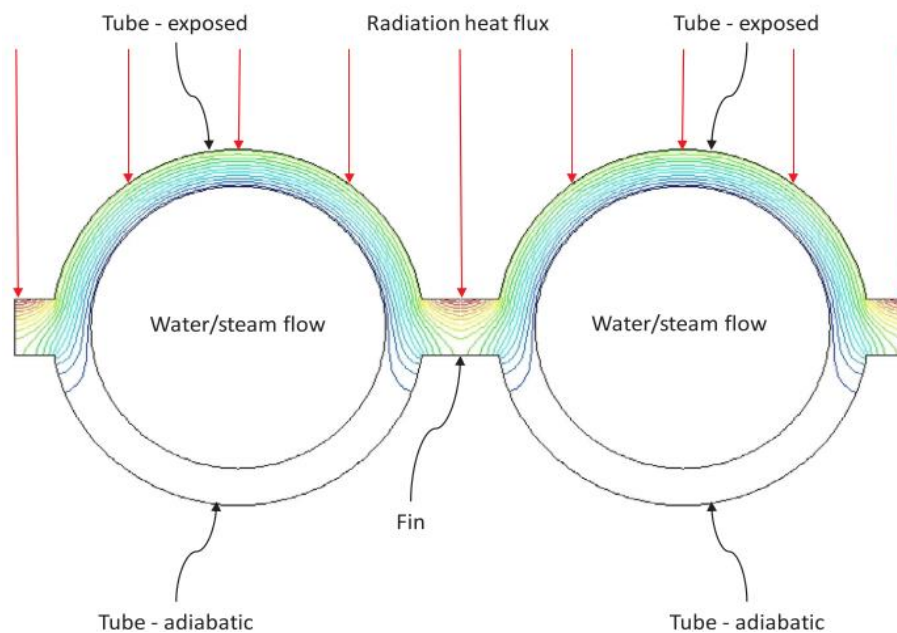


Figure 2.4 Cross-section through a typical PC boiler's membrane water wall geometry with isotherm contours at full load operational conditions [37].

In many Integrated PCFB process models, membrane walls are not modelled in detail and the heat transfer in the water wall is generally simplified.

Niu et al.'s [23] model lumped the economiser and the evaporator into one control volume and did not consider the conduction heat transfer within the membrane wall. In Richter et al.'s [21] model, the evaporator was modelled as a single control volume, and the results did not include the metal temperature of the water wall. Trojan [26] also used a similar approach to Richter et al. for modelling the evaporator.

However, localised overheating of the heat exchanger material for low load and dynamic operations becomes prominent. Therefore, it requires detailed analysis to accurately capture the heat transfer within the tube and fin material.

For that reason, Rousseau and Laubscher [37] have developed a thermal network model that can capture temperatures at different regions of the membrane wall. The model was verified using CFD and showed sufficient agreement in all cases presented, except for a very thick tube wall. The results showed that the model could be applied accurately over a full range of operating conditions and water wall geometries for sub and supercritical PCFBs.

2.2.3 Heat exchanger modelling

The superheaters and reheaters in PCFBs are all classified as gas-to-steam heat exchangers. These heat exchangers are tube banks in ducts that draw heat from combustion products (flue gas, ash and unburnt fuel) to different steam bodies in the boiler. All three modes (conduction, convection and radiation) of heat transfer appear in these heat exchangers and should be modelled accordingly.

The conductive and radiative heat transfer can be modelled using generic correlations seen in literature [34], [38]. However, convective heat transfer requires sophisticated empirical correlations. The internal forced convection in the steam flow can be modelled using conventional internal turbulent flow correlations such as Dittus–Boelter or Gnielinski.

In comparison, the flow pattern of the flue gas over the tube banks is more complex, which requires more refined correlations to model the convective heat transfer. Cengel's textbook [38] recommends Zukauskus' correlations to approximate the flue gas heat transfer coefficient over tube banks. However, implementing Zukauskus' correlations is laborious since the correlations change depending on the flow pattern.

Another well-known correlation published by Kuznetsov et al. [39] only involves one correlation for a range of flow patterns. This correlation was used by Modliński et al. [40] for developing a novel one-dimensional pipe flow model to predict the reheater tube metal temperatures. The model results were compared to site data and showed sufficient accuracy. However, Kuznetsov et al.'s correlation does not take the oblique angle of the flow into account, which is not appropriate for modelling heat exchangers set at certain angles. Additionally, Zukauskus' and Kuznetsov et al.'s

correlations generalise flue gas flow over tube banks as a single flow regime (laminar or turbulent) and neglect the complex flow pattern of the flue gas.

Gnielinski [41] published a correlation that accounts for different degrees of laminar and turbulent flow mixture in fluids. Gnielinski's correlation also accounts for tube bank arrangements (in-line or staggered), tube rows numbers, temperature-dependent property variations and the oblique angle of flow. The versatility of Gnielinski's correlation makes it an appropriate method to model the convective heat transfer of flue gas and is often recommended in literature [34], [42].

Another critical aspect to consider when modelling the heat exchangers in a PCFB is the slagging or fouling on the heating surfaces. These ash deposition effects introduce additional thermal resistances, which inhibits heat transfer.

Traditionally, fouling or slagging thermal resistance is determined empirically because ash deposition is hard to analyse using physics models. The empirical correlations of fouling or slagging thermal resistance can be found in literature [33], [43]. However, these values might not apply to the high ash coal used in South African PCFBs.

Park et al. [44] presented a practical way to study the ash deposition effect in an operating CFPP using a PCFB process model and plant data. The model determined the fouling or slagging thermal resistance by calibrating the computed heat transfer to the heat transfer obtained from plant data. The model was applied to a 500MW supercritical once-through PCFB and tested five different coal blends with an average ash content of 12%. The model results showed sufficient accuracy compared to the field-tested value at different operating conditions.

2.2.4 Air Heater modelling

In the existing South African PCFBs, tubular and rotary regenerative air heaters are used to increase boiler efficiency [34]. In a typical tubular air heater, the air is heated by flue gas flowing through many thin-walled tubes, as seen in Figure 2.5.

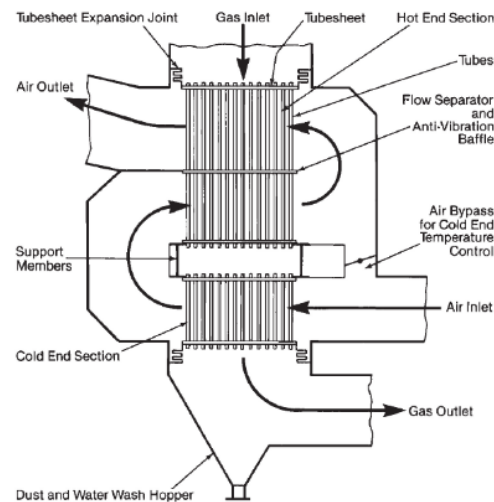


Figure 2.5 Schematic of a typical tubular air heater [45]

Tubular air heaters consist of tube banks, but the flow arrangement is inverted compared to gas-to-steam heat exchangers. Hence, the aforementioned convective heat transfer correlations (Gnielinski [41], Dittus-Boelter and Gnielinski [38]) can still be applied [26], [34].

In a rotary air heater, the heat is indirectly transferred from the flue gas to the air by periodically exposing a solid mass to a single fluid stream, as shown in Figure 2.6. The intermediate heat storage mass is typically constructed from tightly packed corrugated steel plates. This type of air heater usually suffers from significant cross-contamination between the flue gas and air streams.

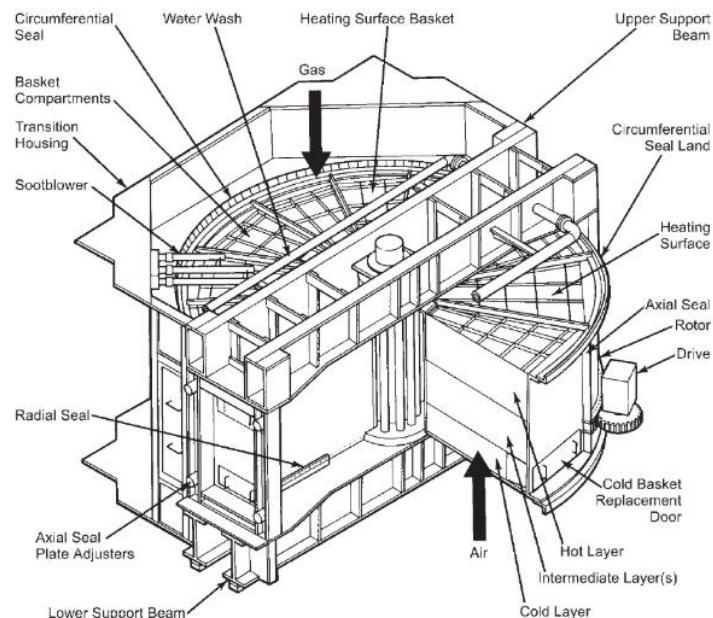


Figure 2.6 Arrangement of rotary air heater [45]

Hajebzadeh et al.[28] presented a method to model the heat transfer of a Ljungstrom type rotary air heater using sets of time-dependent partial differential equations. Although it is generally

significant, the model does not consider air leakage to the flue gas stream. The results from this rotary air heater model compared well with measurements with a maximum relative error of less than 6%.

Kakaç [31] presented a time-independent method to calculate the heat transfer in a rotary air heater. This method assumed the thermal inertia of the energy storage solid is negligible and simplified the model by reducing the number of equations. As mentioned before, a rotary air heater suffers from cross-contamination, which is an operational concern. Laubscher and Rousseau [34] have discussed the detailed analysis of air leakage based on geometry and operational parameters.

2.3 Summary

For boiler modelling, CFD models are generally used to study the details of complex fluid flow and heat transfer phenomena. In comparison, one-dimensional thermofluid network models are used to study the performance of the integrated PCFB system, which is in line with the purpose of the current study.

In an integrated boiler model, the main aim of combustion calculations is to determine the product temperature and species composition. Therefore, the combustion model does not need to include the details of chemical kinetics.

The Gurchich method is widely used in industry and research to calculate the radiative heat transfer in furnaces. Additionally, as South African CFPPs fire high ash coal, the radiation scattering effect of solid particles needs to be addressed in the furnace model. The furnace membrane walls can be modelled using a detailed thermal network model proposed by Rousseau and Lauscher [37] to provide more insights into the temperature profile of the membrane wall. Additionally, it can capture the heat transfer into the steam in more detail.

The steam side heat transfer in the heat exchangers can be approximated using the common correlations of internal pipe flow. However, the flue gas side heat transfer is more complicated and requires a more refined approach, such as the method derived by Gnielinski [41]. Literature suggests values ([33], [43]) for the generic fouling and slagging thermal resistance. However, these suggested values might not apply to the high ash coal utilised in South African CFPPs. Therefore, additional site data calibration is required to account for fouling and slagging.

The tubular air heater is a cross-flow tube bank heat exchanger which can be modelled similar to the gas-to-steam tube banks with a different flow arrangement. On the other hand, the heat transfer in the rotary air heater is indirect and requires its own analysis. Although there are sophisticated models, the simple model suggested in literature [31], [34] is sufficient for modelling the rotary air heater.

3. Theory

In a CFPP, electricity is generated through multiple stages of energy conversion. The primary energy source of the plant is the chemical potential energy of coal, which releases heat during combustion. The heat released during the combustion process is then transferred to the water inside the boiler and generates high-temperature steam. The thermal energy of the high-temperature steam gets further converted into mechanical energy by turbines and, in turn, converted to electrical energy through electrical generators.

The heat released during the combustion process is stored as thermal energy in the combustion products (flue gas, ash and unburnt carbon particles), and it is transferred to various fluid bodies in the boiler, as shown in Figure 3.1.

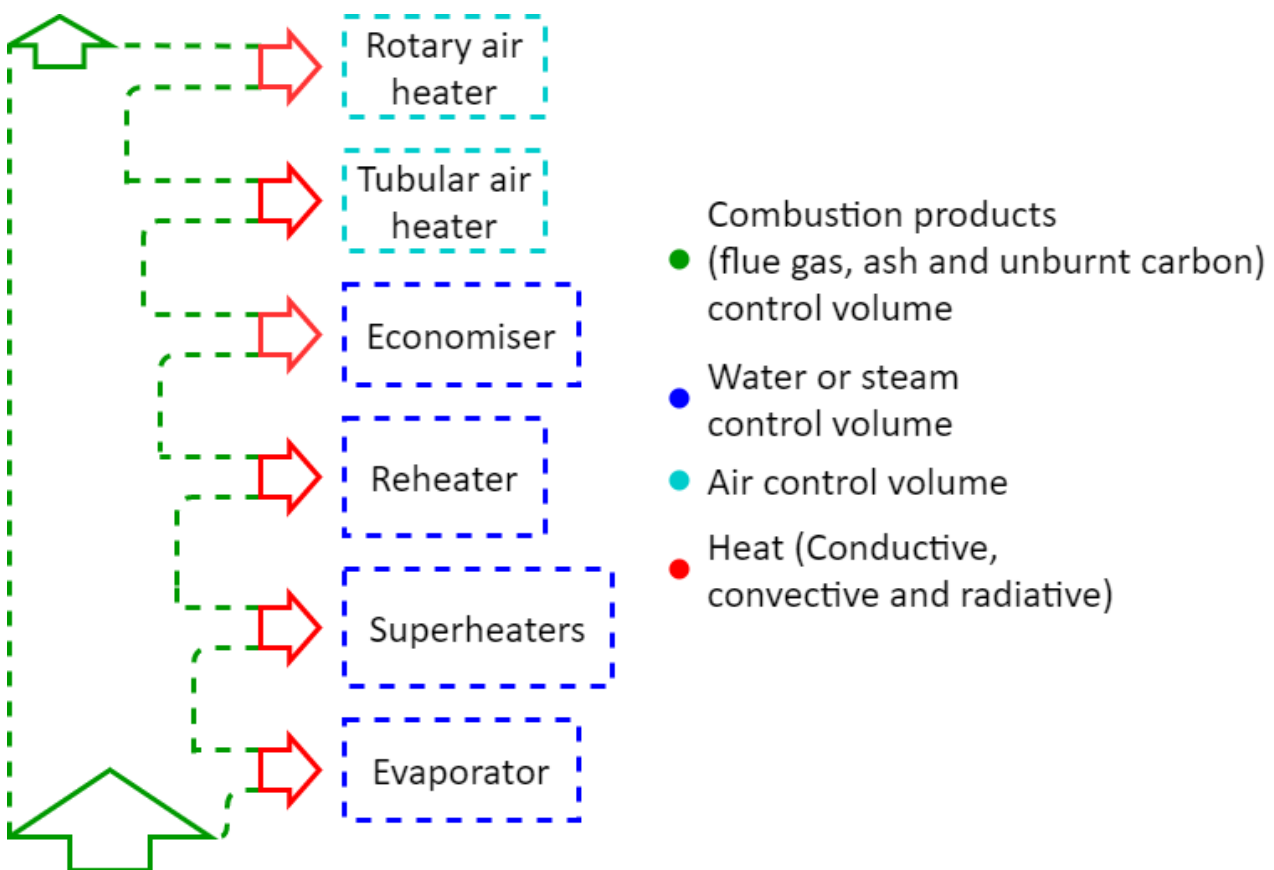


Figure 3.1 General boiler heat transfer

To model the various modes of heat transfer in a once-through boiler requires an understanding of the fundamental thermodynamic theory and operational requirements. This includes boiler operation modes, combustion theory, gas-mixture theory, two-phase flow phenomena, and multi-mode heat transfer. This theory is applied to develop the integrated once-through PCFB model in the project, which will be further discussed in subsequent sections.

3.1 Once-through boiler operation modes

Once-through boilers have two modes of operation, namely dry (Benson) and wet modes. The difference between these two modes of operation is their water circulation, as shown in Figure 3.2.

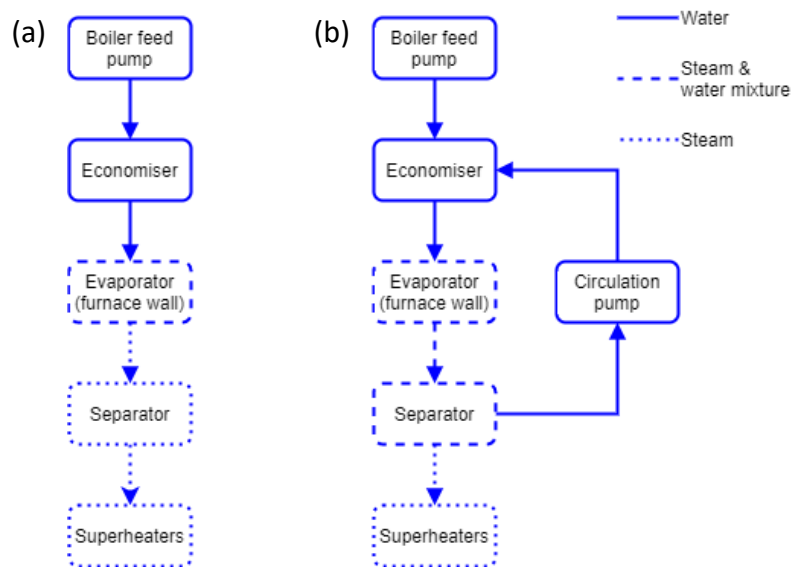


Figure 3.2 Water flow path of the main steam cycle for (a) dry (Benson) and (b) wet mode operation

Once-through boilers typically operate in dry mode during normal operation and only experience wet mode operation once the load drops below the once-through point, typically between 35-40% MCR. While operating in wet mode, the water flow rate in the evaporator is maintained at a constant point to avoid overheating the evaporator tube walls, as shown in Figure 3.3.

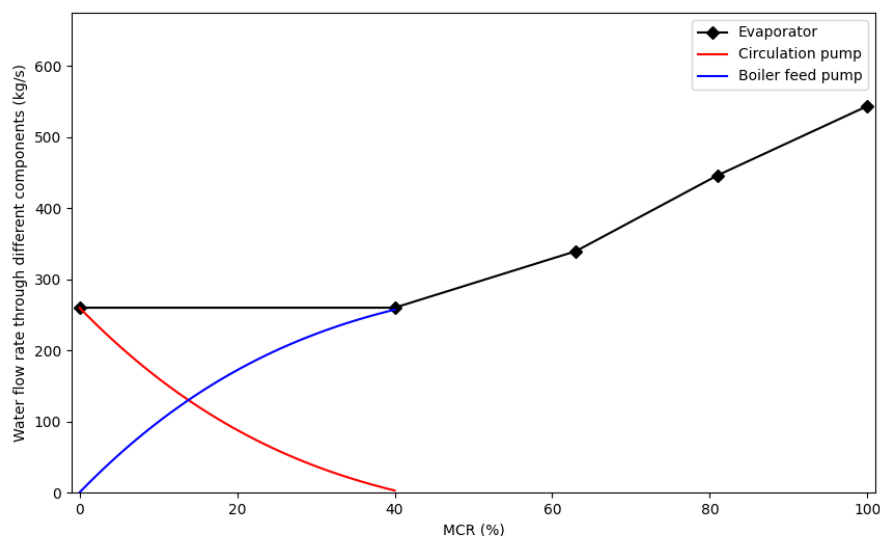


Figure 3.3 Water flow rate vs MCR

3.2 Pulverised coal combustion

Pulverised coal combustion consists of multiple exothermic chemical reactions, as shown schematically in Figure 3.4.

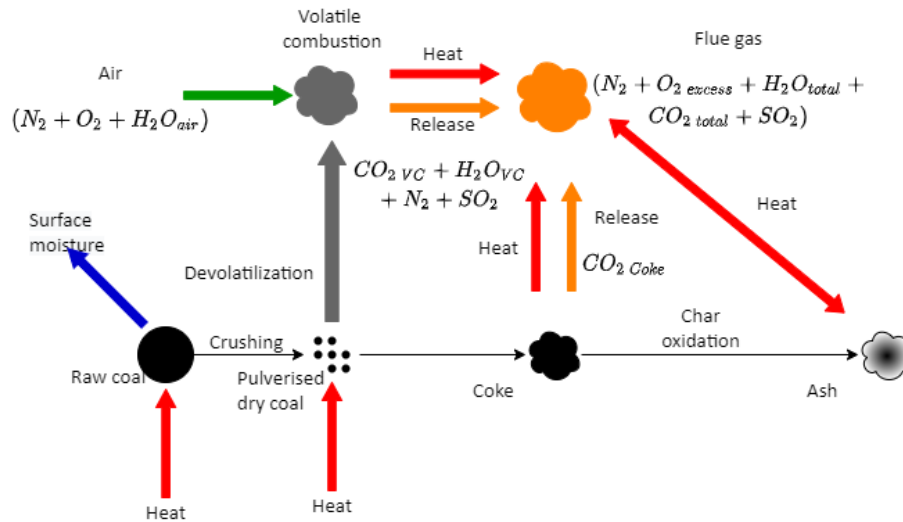


Figure 3.4 Pulverised coal particle transport during combustion

In practice, the supplied air is higher than the theoretically required amount to ensure sufficient mixing of oxygen and coal. However, the combustion process typically leaves a small fraction of carbon particles unburnt even with excess air supply. This phenomenon, therefore, needs to be considered for the analysis.

Combustion analysis focuses on the combustion zone's overall mass and energy balance. Therefore, the analysis is simplified to a zero-dimensional control volume of the flame, as shown in Figure 3.5.

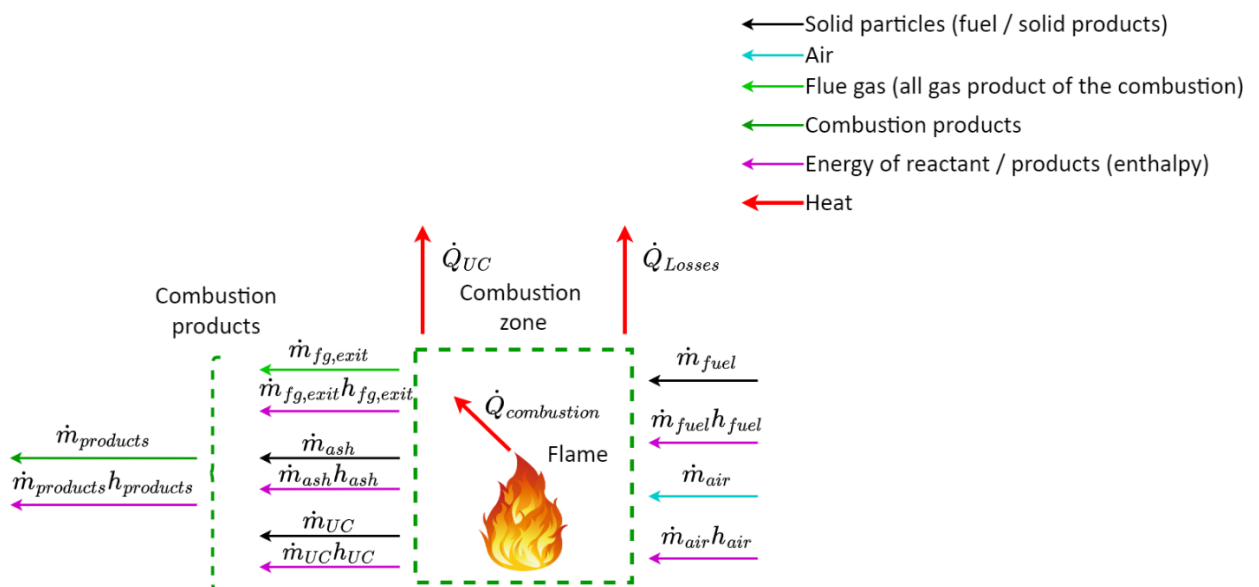
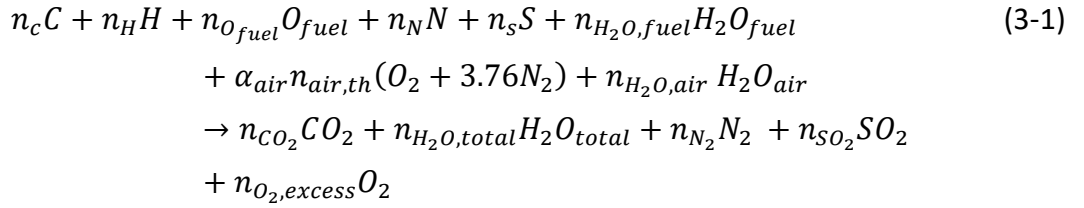


Figure 3.5 Zero-dimensional control volume of the combustion process

In this case, the chemical kinematics is not considered, and therefore the miscellaneous heat loss ($\dot{Q}_{losses}$) is insignificant. For that reason, the flame can be analysed as adiabatic.

3.2.1 Combustion products composition

The mass fraction of each gas product can be calculated using a straightforward combustion analysis. These mass fractions are essential for finding the temperature and fluid properties of the flue gas. The chemical equation for the combustion analysis is expressed as:



where the molar amount of each species in the fuel (n_j) can be expressed as follows:

$$n_j = \frac{\dot{m}_j}{M_j} = \frac{Y_j \dot{m}_{fuel}}{M_j} \quad (3-2)$$

where $\dot{m}_j$ is the mass flow rate of each chemical species in the fuel, $\dot{m}_{fuel}$ is the mass flow rate of the fuel, and M_j is the molar mass of each specie. The mass flow rate of each species in the fuel can be calculated using the mass fraction of the species (Y_j), given by the fuel specification.

The excess air ratio (α_{air}) for PCFBs typically range between 1.15 to 1.40.

The molar amount of the theoretical supply air ($n_{air, th}$) is determined using the chemical oxidation equations of stoichiometric combustion (see Appendix A).

This molar amount can be calculated using the following expression:

$$n_{air, th} = \frac{2 \cdot n_{CO_2} + n_{H_2O, product} + 2 \cdot n_{SO_2} - n_{O_{fuel}}}{2} \quad (3-3)$$

where the gas products of the oxidations can be calculated using the molar number ratios of the chemical reactions (see Appendix A):

$$n_{CO_2} = n_C \quad (3-4)$$

$$n_{H_2O, product} = \frac{1}{2} n_H \quad (3-5)$$

$$n_{SO_2} = n_S \quad (3-6)$$

The air moisture ($n_{H_2O,air}$) can be accounted for using the specific humidity (ω):

$$n_{H_2O,air} = \frac{\alpha_{air} n_{air,th} \omega}{X_{H_2O,air}} \quad (3-7)$$

where the molar fraction of moisture ($X_{H_2O,air}$) can be expressed as follows:

$$X_{H_2O,air} = \frac{M_{H_2O}}{M_{air}} \quad (3-8)$$

where M_{H_2O} is the molar mass of water and M_{air} is the molar mass of air.

With the molar amount of air moisture ($n_{H_2O,air}$) determined, the total water vapour in the product can be calculated by summing all the moisture of the combustion process:

$$n_{H_2O,total} = n_{H_2O,fuel} + n_{H_2O,air} + n_{H_2O,product} \quad (3-9)$$

In the stoichiometric combustion analysis, the oxidation of nitrogen (N_2) is not considered; thus, the nitrogen in the product is simply the sum of all the nitrogen in the reactants:

$$n_{N_2} = n_N + 3.76 \alpha_{air} n_{air,th} \quad (3-10)$$

The molar amount of oxygen in the excess air ($n_{O_2,excess}$) can be expressed using the following equation:

$$n_{O_2,excess} = (\alpha_{air} - 1) n_{air,th} \quad (3-11)$$

with the molar amounts of each gas product known, the mass fraction of each flue gas constituent (Y_i) can be calculated as follows:

$$Y_i = \frac{\dot{m}_i}{\dot{m}_{fg,exit}} = \frac{\dot{m}_i}{\sum_i \dot{m}_i} \quad (3-12)$$

where $\dot{m}_i$ is the mass flow rate of each flue gas constituent, and it can be expressed as follows:

$$\dot{m}_i = n_i M_i \quad (3-13)$$

where n_i is the molar amount of each gas constituent and M_i is the molar mass of the gas.

The mass flow rate of the unburnt carbon ($\dot{m}_{UC}$) and the mass flow rate of the ash ($\dot{m}_{Ash}$) are calculated using their respective mass fractions:

$$\dot{m}_{UC} = Y_{UC} \dot{m}_{Fuel} \quad (3-14)$$

$$\dot{m}_{ash} = Y_{ash} \dot{m}_{Fuel} \quad (3-15)$$

where the mass fraction of ash in the fuel (Y_{ash}) is given by the fuel specification and the mass fraction of unburnt carbon in the fuel (Y_{UC}) is assumed to be a small fraction, typically 1%.

3.2.2 Adiabatic flame calculation

The adiabatic flame temperature (AFT) can be determined by analysing the energy conservation of the process. By analysing the energy conservation, the specific enthalpy of the flame (h_{flame}) can be determined using the following expression:

$$h_{flame} = h_{products} = \frac{\dot{Q}_{combustion} + \dot{m}_{fuel} h_{fuel} + \dot{m}_{air} h_{air} - \dot{Q}_{UC}}{\dot{m}_{products}} \quad (3-16)$$

In the combustion model, the specific enthalpy of the flame is treated as the specific enthalpy of the products ($h_{products}$).

The heat released during combustion ($\dot{Q}_{combustion}$) can be determined using the mass flow rate of the fuel, and the higher heating value of the fuel (HHV_{fuel}):

$$\dot{Q}_{combustion} = \dot{m}_{fuel} HHV_{fuel} \quad (3-17)$$

The higher heating value of the fuel is measured by burning 1kg of fuel in a bomb calorimeter and cooling the products inside to the initial fuel temperature. The standard reference temperature for the higher heating value is $25^{\circ}C$ and therefore, this reference temperature is used throughout the flue gas stream (air, fuel, flue gas, ash and unburnt carbon). For that reason, the specific sensible heats of the fuel can be determined using the following expression:

$$h_{fuel} = c_{p,fuel} (T_{fuel} - 25^{\circ}C) \quad (3-18)$$

where specific heat capacity of the fuel ($c_{p,fuel}$) is approximated as $730 \frac{kJ}{kg \cdot K}$ for coal [34]. The term T_{fuel} refers to the temperature of pulverised coal fed into the burner.

The heat loss due to unburnt carbon ($\dot{Q}_{UC}$) is also considered in the energy balance, and it can be determined using the following expression:

$$\dot{Q}_{UC} = \dot{m}_{UC} HHV_{carbon} \quad (3-19)$$

where the higher heating value for pure carbon (HHV_{carbon}) is taken as $33820 \frac{kJ}{kg}$ [46].

The mass flow rate of the combustion products ($\dot{m}_{products}$) is defined using the principle of mass conservation for the combustion zone shown in Figure 3.5:

$$\dot{m}_{products} = \dot{m}_{fuel} + \dot{m}_{air} \quad (3-20)$$

The supply air mass flow rate ($\dot{m}_{air}$) is typically calculated using the air-fuel ratio (AF):

$$\dot{m}_{air} = AF \cdot \dot{m}_{fuel} \quad (3-21)$$

where the air-fuel ratio is defined as follows:

$$AF = \alpha_{air} \cdot AF_{theo} \quad (3-22)$$

AF_{theo} is the theoretical air-fuel ratio, and it can be calculated as follows:

$$AF_{theo} = \frac{n_{air,th} M_{air}}{\dot{m}_{fuel}} \quad (3-23)$$

As shown in Figure 3.5, the mass flow rate of the products on a component basis is:

$$\dot{m}_{products} = \dot{m}_{fg,exit} + \dot{m}_{UC} + \dot{m}_{ash} = \dot{m}_{fuel} + \dot{m}_{air} \quad (3-24)$$

where the sensible heat of the products is defined as follows:

$$\begin{aligned} \dot{m}_{products} h_{flame} \\ = \dot{m}_{fg,exit} h_{fg,exit} + \dot{m}_{UC} c_{p,fuel} (T_{flame} - 25^\circ C) \\ + \dot{m}_{ash} c_{p,ash} (T_{flame} - 25^\circ C) \end{aligned} \quad (3-25)$$

The specific heat capacity of the ash ($c_{p,ash}$) is approximated as $730 \frac{kJ}{kg \cdot K}$. The enthalpy of the flue gas exiting the combustion zone ($h_{fg,exit}$) is a function of temperature, which will be further discussed in Section 3.3.1.

By rearranging Equation (3-25), The enthalpy of the flue gas exiting the combustion zone can be expressed as:

$$h_{products} = w_{fg}h_{fg,exit} + w_{UC}c_{p,fuel}(T_{flame} - 25^{\circ}C) + w_{ash}c_{p,ash}(T_{flame} - 25^{\circ}C) \quad (3-26)$$

where w_{fg} , w_{UC} , w_{ash} refers to the mass ratio of flue gas, unburnt carbon and ash inside the products.

With the energy relations mentioned earlier, the AFT can be solved using Equation (3-16) and Equation (3-26) together with the temperature-enthalpy correlation of the flue gas.

3.3 Gas side thermofluid modelling

Gas flows are the thermal energy carriers that heat other fluid bodies in the boiler. For that reason, it is essential to model the thermofluid flows of these gases to find the discrete heat transfer to each fluid body in the boiler.

3.3.1 Fluid properties

Fluid properties are essential inputs to the thermofluid conservation equations, and they can be obtained from an empirical database or calculated using suitable equations of state.

For a gas mixture, fluid properties are dependent on the mass (or mole) fractions of each constituent. The mass fractions can be determined using Equation (3-12), and the mole fractions (X_i) can be calculated as follows:

$$X_i = \frac{n_i}{\sum_i n_i} \quad (3-27)$$

where X_i is the mole fraction of each gas component and n_i is the number of moles.

These ratios are incorporated with real gas mixing methods [47] together with the real gas fluid properties of each gas constituent to find the fluid properties of the mixture. The real gas fluid properties are generally found in the National Institute of Standards and Technology (NIST) or the American Society of Mechanical Engineering (ASME) database.

However, to simplify the analysis, the specific enthalpy of the gas mixture h_{mix} is found using the ideal gas assumption. The ideal gas assumption is valid since the gas flows in the boiler operate at relatively low pressures (80~100 kPa) with high temperatures (200°C~2000°C), and therefore, the associated errors of using the ideal gas assumptions are insignificant.

As mentioned in Section 3.2.2, the reference temperature used to determine the specific enthalpy of the gas mixture (h_{mix}) is taken to be $25^\circ C$. Provided that the reference temperatures are the same for all the gas constituents, the enthalpy of an ideal gas mixture can be calculated as follows:

$$h_{mix} = \sum_i Y_i h_i \quad (3-28)$$

where specific enthalpies of each gas constituent (h_i) can be found in the NIST or ASME database.

3.3.2 Conservation equations

In the boiler system, fluid flows are modelled using the finite volume method (FVM). The flow regions are discretised as finite one-dimensional control volumes aligned with the flow direction, as seen in Figure 3.6.

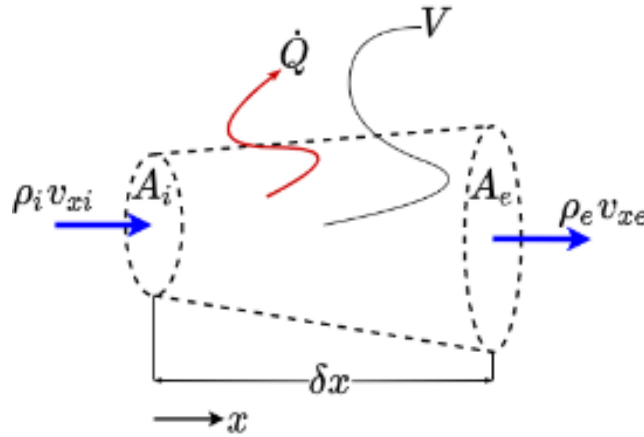


Figure 3.6 Finite control volume with one-dimensional flow

Note that the subscripts i and e in Figure 3.6 refer to the inlet and exit of the control volume, respectively.

For steady-state flow, the mass conservation can be expressed as follows [48]:

$$\sum \dot{m}_{exit} - \sum \dot{m}_{in} = \rho_e v_{xe} A_e - \rho_i v_{xi} A_i = 0 \quad (3-29)$$

For compressible flows, the momentum conservation can be expressed as follows [48]:

$$\frac{\bar{\rho} \bar{v}_x^2}{2 \bar{T}_o} (T_{oe} - T_{oi}) = -\frac{\bar{p}}{\bar{p}_o} (p_{oe} - p_{oi}) - \bar{\rho} g (z_e - z_i) - \left(\frac{f l}{D_h} + \Sigma K \right) \frac{\bar{\rho} |\bar{v}_x| \bar{v}_x}{2} \quad (3-30)$$

In Equation (3-30), the bars over the variables indicate the average value of the variable across the control volume. The stagnation pressure (p_o) and temperature (T_o) for compressible flow are functions of the Mach number (M), for which the expressions can be found in [48].

The last term on the right-hand side of Equation (3-30) refers to the friction force acting on the fluid. In this term, f is the frictional loss factor, a function of the flow's Reynolds number (Re). ΣK denotes to the sum of all the secondary loss coefficients, l refers to the length of the control volume and D_h is the hydraulic diameter.

For compressible flows with low velocities (a Mach number smaller than 0.3), the impact of compressibility effects on the momentum is negligible. Thus, the momentum conservation equation for compressible flow can be simplified to the incompressible form.

The energy conservation is expressed in terms of the specific stagnation enthalpy (h_o), and it can be expressed as follows [48]:

$$\begin{aligned} \Sigma \dot{m}_{exit}(h_{oe} + gz_e) - \Sigma \dot{m}_{in}(h_{oi} + gz_i) \\ = \rho_e v_{xe} A_e (h_{oe} + gz_e) - \rho_i v_{xi} A_i (h_{oi} + gz_i) = \Sigma \dot{Q} - \Sigma \dot{W} \end{aligned} \quad (3-31)$$

where the specific stagnation enthalpy is defined as follows:

$$h_o = h + \frac{1}{2} v_x^2 \quad (3-32)$$

Furthermore, specific enthalpy h is defined as follows:

$$h = u + pv \quad (3-33)$$

Here u is the specific internal energy, and v is the specific volume of the fluid.

The source term $\Sigma \dot{Q}$ refer to the sum of all heat sources and $\Sigma \dot{W}$ refers to the sum of all work rates of the fluid.

3.3.3 Momentum source terms

The gas flows in the boiler lose their momentum as they flow through obstacles such as the tube bundle and rotary air heater elements. The momentum losses translate to pressure drops in the fluid, typically accounted for by frictional and secondary loss terms. For the case of the tube bundle and air heater elements, the net pressure loss is accounted for by the secondary loss coefficient. It, therefore, does not need to introduce additional pressure loss through the frictional loss factor.

For the gas flow over the tube bundles, the pressure drop is due to friction and flow separation in the wake region behind the individual tubes [49], and it can be calculated as follows:

$$\Delta p_{ext} = \zeta N_{MR} \frac{\rho v_{int}^2}{2} \quad (3-34)$$

where ζ is the drag coefficient found using the empirical correlations presented by Gaddis [49]. These correlations are valid for Reynolds numbers ranging from 1 to 3×10^5 , and for tube bundles with more than five tubes in the flow/ longitudinal direction.

The term N_{MR} refers to the number of tubes in the flow/longitudinal direction resisting the flow and v_{int} is defined as the mean interstitial velocity of the gas flow through the tube bundle. The number of tubes resisting the flow and the mean interstitial velocity of the flow are related to the flow pattern of the gas, which is dependent on the pitches of the tube bundle (s_t, s_l, s_d), as shown in Figure 3.7.

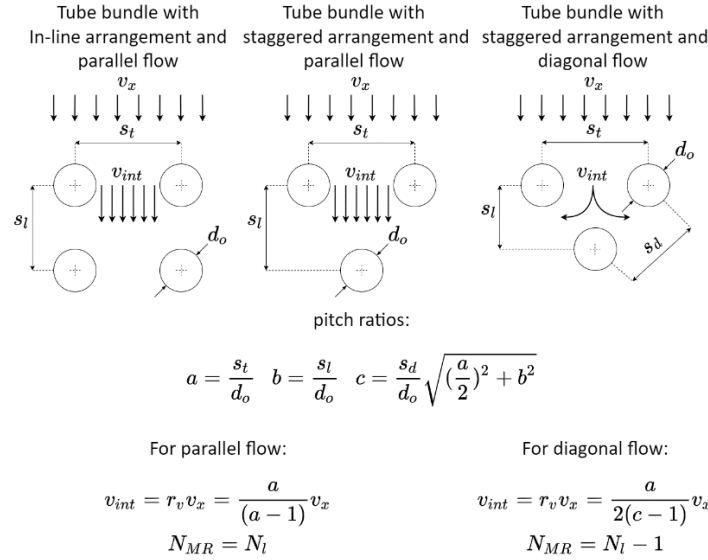


Figure 3.7 Illustration of a flow-through tube bundle [49]

N_l refers to the total number of tubes in the flow/longitudinal direction.

By applying Equation (3-30), Equation (3-34) and the correlations shown in Figure 3.7, the equivalent secondary loss coefficient (K_{ext}) can be determined as:

$$K_{ext} = \zeta N_{MR} r_v^2 \quad (3-35)$$

The pressure losses in the gas flow through the rotary air heater are complex to model. Therefore, the secondary loss coefficient associated with the pressure drop across the rotary air heater is found using plant data.

3.3.4 Energy source terms

As shown in Figure 3.1, the thermal energy of the combustion products is transferred as heat to various fluid bodies in the boiler. The heat is transferred via radiation and convection, which will be discussed in this section.

As the products flow through the furnace, they radiate heat to the surrounding surfaces. The common approach to model the radiation in the furnace is the zero-dimensional projected method. This approach is commonly used for various technical applications such as boiler sizing and dynamic simulations.

This approach solves the mass and energy conservation equations by considering the products in the PCFB furnace as a zero-dimensional control volume, shown schematically in Figure 3.8.

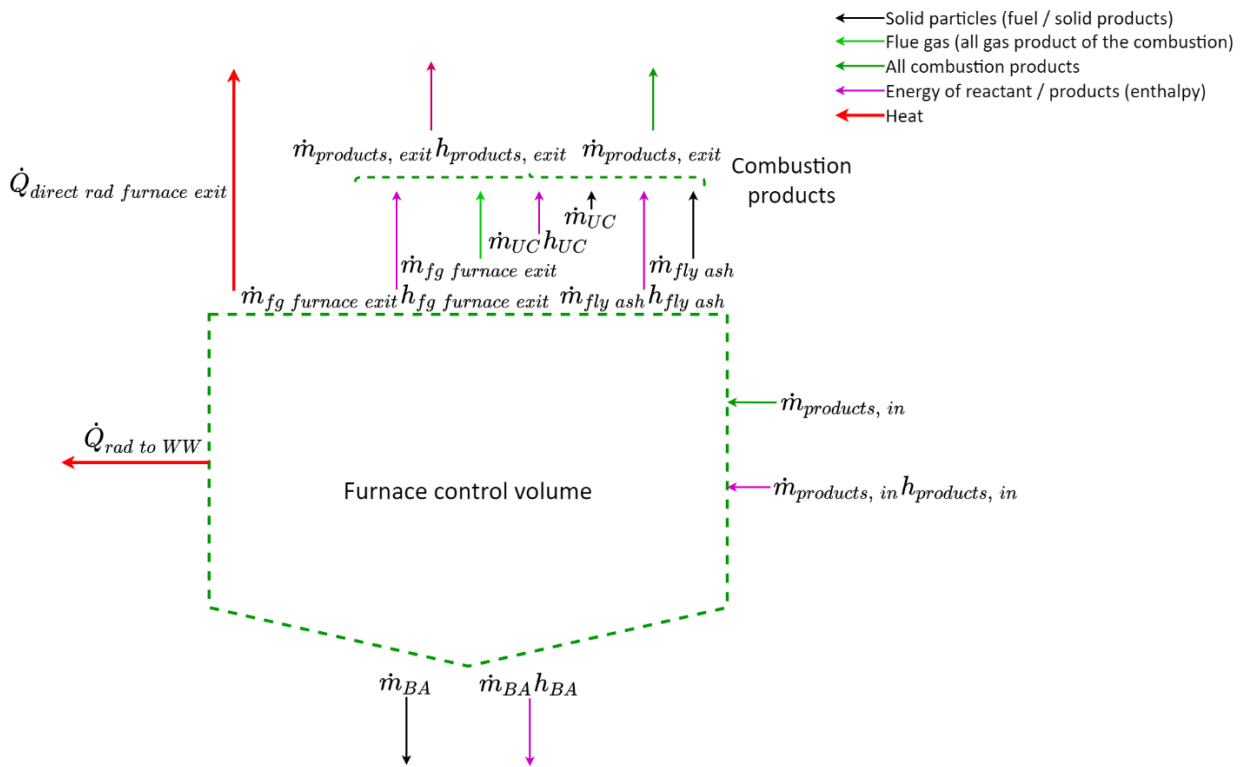


Figure 3.8 Zero-dimensional control volume of the flue gas inside the furnace

The total radiative heat transfer $\dot{Q}_{rad\ total}$ can be calculated by solving the energy conservation equation of the control volume:

$$\begin{aligned}\dot{Q}_{rad\ total} &= \dot{Q}_{rad\ to\ WW} + \dot{Q}_{direct\ rad\ furnace\ exit} \\ &= (\dot{m}_{products,exit} h_{products,exit} + \dot{m}_{BA} h_{BA}) \\ &\quad - \dot{m}_{products,in} h_{products,in}\end{aligned}\tag{3-36}$$

where BA refers to the bottom ash. Bottom ash is assumed to have the same thermal properties as ash in the products, and therefore its enthalpy can be calculated as follows:

$$h_{BA} = c_{p,ash}(T_{BA} - 25^{\circ}C) \quad (3-37)$$

For pulverised coal boilers, it is common practice to assume bottom ash to be 5% in weight of the overall ash and that it leaves the furnace at $800^{\circ}C$.

From the analysis mentioned in Section 3.2.2, the specific enthalpy of the products flowing in the furnace ($h_{products,in}$) is the same as the specific enthalpy of the flame h_{flame} .

The specific enthalpy of products exiting the furnace ($h_{products,exit}$) is a function of furnace exit temperature (FET), and it can be approximated using the direct or Gurvich method. The Gurvich method does not directly solve the radiation between the furnace wall surfaces and the products. Instead, it approximates the FET using empirical correlations, expressed as:

$$T_{products,exit} = \frac{T_{flame}}{M \left[\frac{\sigma_o \bar{\psi} A_{furnace} \epsilon_{furnace} T_{flame}^3}{(1-f_{rad\ loss}) \dot{m}_{fuel} \bar{V}C} \right]^{0.6} + 1} \quad (3-38)$$

where σ_o is Stefan-Boltzman's constant.

The term M is the flame modification factor that accounts for the temperature distribution along the height of the furnace, and it is found using empirical correlations [31] (see Appendix B for details).

The average furnace wall efficiency factor ($\bar{\psi}$) accounts for all the wall surfaces in the furnace and is calculated as follows:

$$\bar{\psi} = \frac{\sum_i \psi_i A_i}{\sum_i A_i} = \frac{\sum_i \psi_i A_i}{A_{furnace}} \quad (3-39)$$

where A_i is the area of different furnace wall sections, $A_{furnace}$ is the planar furnace area and ψ_i is the efficiency factor of different furnace wall surfaces, which are found using empirical correlations (see Appendix B for correlations).

The furnace effective emissivity ($\epsilon_{furnace}$) is influenced by tri-atomic gas, soot, coke particle and ash particle radiation. By considering these radiation effects, the furnace emissivity can be calculated as follows:

$$\epsilon_{furnace} = \frac{\epsilon_{G+P,f}}{\bar{\psi}(1 - \epsilon_{G+P,f}) + \epsilon_{G+P,f}} \quad (3-40)$$

where $\epsilon_{G+P,f}$ is the effective gas-particle emissivity in the furnace.

However, when the products contain a high amount of ash, radiation scattering by the suspended particles becomes significant, and therefore should be accounted for when calculating the effective gas-particle emissivity in the furnace.

The empirical correlations used to calculate the effective gas-particle emissivity in the furnace are presented by Brummel [16] and Ritcher et al. [50] (see Appendix C for details).

The term $f_{rad\ loss}$ is the radiation loss fraction, which accounts for the radiation loss through the boiler skin casings [34] (see Appendix B for details).

The term $\overline{VC}$ refers to the mean overall heat capacity per kilogram fuel, which is calculated as follows:

$$\overline{VC} = \left(\frac{h'_{products,in} - h_{products,exit}}{T_{flame} - T_{products,exit}} \right) \left(\frac{\dot{m}_{products,exit}}{\dot{m}_{fuel}} \right) \quad (3-41)$$

The specific enthalpy of the products flowing into the furnace without the bottom ash ($h'_{products,in}$) is expressed as:

$$h'_{products,in} = \left(\frac{\dot{m}_{products,exit} h_{flame} - \dot{m}_{BA} h_{BA}}{\dot{m}_{products,in} - \dot{m}_{BA}} \right) \quad (3-42)$$

The Gurvich method approximated FET can be used to find the specific enthalpy of products exiting the furnace, which is used to solve the total radiative heat transfer of Equation (3-36).

With the total radiative heat transfer known, the radiation to different surfaces in the furnace can be determined by analysing the projected radiation heat flux.

The average radiative heat flux projected onto furnace surfaces ($\dot{q}_{furnace}$) can be determined as follows:

$$\dot{q}_{furnace} = \frac{\dot{Q}_{rad\ total}}{A_{furnace}} \quad (3-43)$$

where the planar furnace area consists of the furnace water wall surface area (A_{WW}) and the furnace exit plane ($A_{furnace,exit\ plane}$).

However, the radiation heat flux is not uniformly distributed to the furnace wall, and therefore the heat flux is adjusted by a non-uniformity factor (η_k).

This factor accounts for the heat flux distribution along the height of the furnace and assumes a homogeneous heat flux distribution in the cross-sectional directions of the furnace (see Appendix D for correlations).

The direct radiation to the platen superheater (directly downstream of the furnace) is determined by the heat radiated through the exit plane of the furnace ($\dot{Q}_{direct\ rad\ furnace\ exit}$), and it can be calculated as follows:

$$\dot{Q}_{direct\ rad\ furnace\ exit} = \eta_{k,furnace,exit} \dot{q}_{furnace} A_{furnace,exit\ plane} \beta_{platen} \quad (3-44)$$

where $\eta_{k,furnace,exit}$ is the heat flux nonuniformity factor for the furnace exit plane.

The factor β_{platen} accounts for the radiation interaction between the platen superheater and the furnace, which can be found using an empirical function of FET (see Appendix E for correlation).

The results of Equation (3-44) can be applied to Equation (3-36) to solve the radiation to the furnace water wall/evaporator ($\dot{Q}_{rad\ to\ WW}$) which is not uniformly projected on the furnace water walls. Thus, finding the radiation projected on a specific section of the furnace water wall $\dot{Q}_{WW,i}$ also requires η_k :

$$\dot{Q}_{WW,i} = \eta_{k,i} \dot{q}_{WW} A_{WW,i} \quad (3-45)$$

where $\eta_{k,i}$ is the heat flux nonuniformity factor taken from the relative height of the section's midpoint and $A_{WW,i}$ is the surface area of the furnace water wall section.

The uniform radiative heat flux projected onto the furnace water walls ($\dot{q}_{WW}$) can be determined as follows:

$$\dot{q}_{WW} = \frac{\dot{Q}_{rad\ to\ WW}}{A_{WW}} \quad (3-46)$$

where A_{WW} is the total surface area of the entire water wall.

The heat radiated through the exit plane of the furnace, as shown in Equation (3-44), travels through the boiler and radiates heat to different heat exchanger external surfaces, as shown in Figure 3.9.

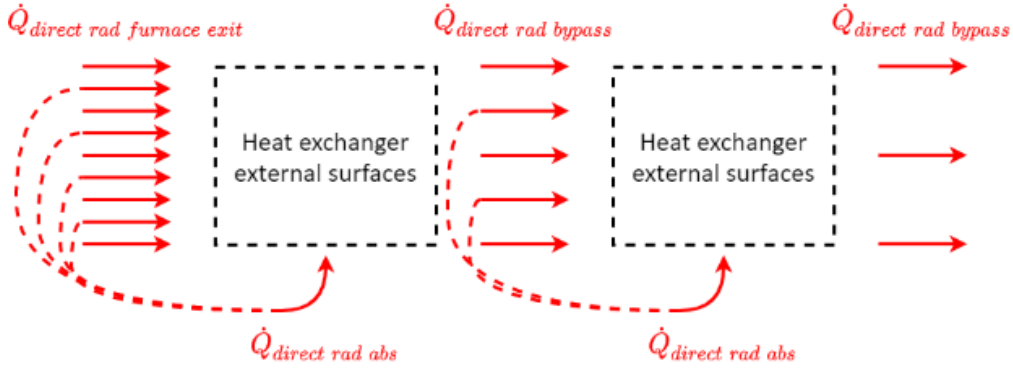


Figure 3.9 Direct radiation travel through the boiler

The method used to determine the direct radiation propagation through the heat exchangers can be found in Kakaç [31], and it is expressed as:

$$\dot{Q}_{direct\ rad\ bypass,next} = \frac{\dot{Q}_{direct\ rad\ bypass,previous}(1 - \epsilon_{G+P})\phi_e}{\beta} \quad (3-47)$$

where ϵ_{G+P} is the effective gas-particle emissivity of the products at each point in the boiler and ϕ_e is the angular coefficient of direct radiation absorbed from the inlet plane to the outlet plane of a heat exchanger. The method of determining the effective gas-particle emissivity of the products is the same as the method used to determine the effective gas-particle emissivity in the furnace, except that the effect of coke in the products is not considered (see Appendix C for details).

The value of ϕ_e can be found using the expression presented by Kakaç [31]:

$$\phi_e = \left[\left(\frac{l_b}{s_t} \right)^2 + 1 \right]^{\frac{1}{2}} - \frac{l_b}{s_t} \quad (3-48)$$

where l_b is the total depth of the heat exchanger cavity.

As shown in Figure 3.9, the direct radiation absorbed by the external tube surfaces ($\dot{Q}_{direct\ rad\ abs}$) is the difference between the direct radiation into the heat exchanger ($\dot{Q}_{direct\ rad\ bypass,previous}$) and the direction radiation bypassing the heat exchanger ($\dot{Q}_{direct\ rad\ bypass,next}$):

$$\dot{Q}_{direct\ rad\ abs} = \dot{Q}_{direct\ rad\ bypass,previous} - \dot{Q}_{direct\ rad\ bypass,next} \quad (3-49)$$

For the platen superheater, the direct radiation propagation through the heat exchangers is the heat radiated through the exit plane of the furnace. The term β is the interaction factor between the platen superheater and the furnace.

For all the other heat exchangers in the boiler, Equation (3-47) is directly applicable with the interaction factor equal to 1.

As shown in Figure 3.1, the products leaving the furnace would flow through the tube bank heat exchangers (superheater, reheater and economiser) and release heat to the water/steam bodies flowing inside the tubes in the heat exchanger duct. Additionally, the duct walls contain a vertical tube bank (vertical water wall), which also receives heat from the products stream.

By applying the concepts mentioned above, the heat transfer between the control volumes inside the heat exchangers can be represented using the diagram shown in Figure 3.10.

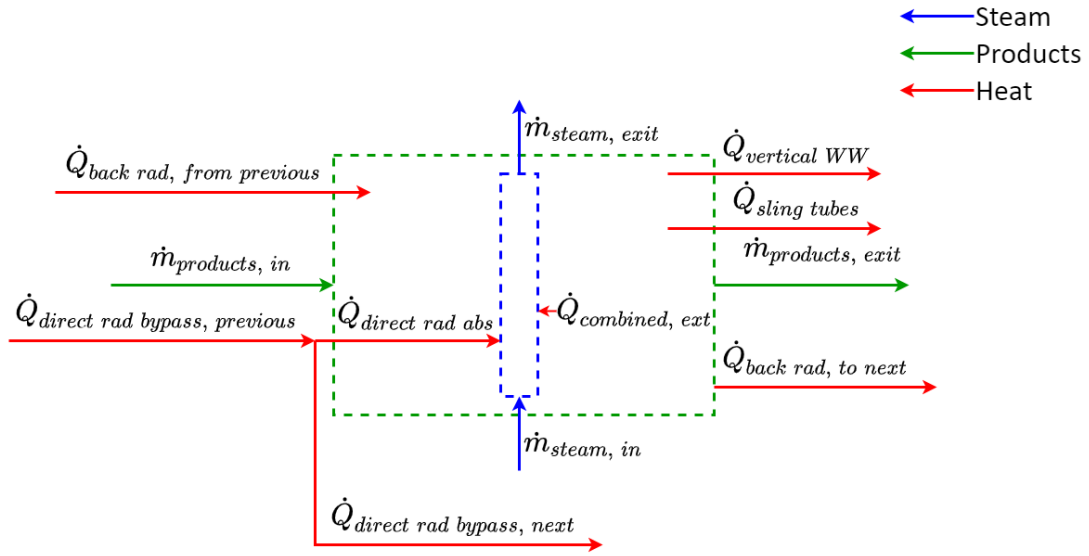


Figure 3.10 Mass and Energy Balance of heat exchangers

The back panel radiation from the upstream body ($\dot{Q}_{back\ rad}$) is calculated as follows:

$$\dot{Q}_{back\ rad} = \frac{\sigma_o \epsilon_{G+P} A_{products,exit\ plane} \bar{T}_{products}^4}{2} \quad (3-50)$$

where $\bar{T}_{products}$ is the mean product temperature in the heat exchanger and $A_{products,exit\ plane}$ is the exit plane of the products in the heat exchanger (see Appendix F for geometry).

The products' gas and particle radiation ($\dot{Q}_{G+P\ rad}$) uses the grey body assumption, and it can be determined using the expression presented by Prabir et al. [30]:

$$\dot{Q}_{G+P\ rad} = \frac{\epsilon_{surface} + 1}{2} \sigma_o \epsilon_{G+P} A_{surface} (\bar{T}_{products}^4 - T_{surface}^4) \quad (3-51)$$

where $\epsilon_{surface}$ is the emissivity and $T_{surface}$ is the temperature of the surfaces in contact with the products.

Since these surfaces are fouled or slagged, the surfaces' emissivity ($\epsilon_{surface}$) is equal to the emissivity of the ash (ϵ_{ash}):

$$\epsilon_{surface} = \epsilon_{ash} \quad (3-52)$$

where

$$\epsilon_{ash} = 0.68 \text{ for slag screen}$$

$$\epsilon_{ash} = 0.8 \text{ for other boiler tube surfaces} \quad (3-53)$$

The products' gas and particle radiation can be accounted for using an equivalent gas and particle radiation heat transfer coefficient ($h_{G+P rad}$), presented by Prabir et al. [30]:

$$h_{G+P rad} = \frac{\epsilon_{ash} + 1}{2} \sigma_o \epsilon_{G+P} \bar{T}_{products}^3 \left(\frac{1 - \left(\frac{T_{surface}}{\bar{T}_{products}} \right)^4}{1 - \left(\frac{T_{surface}}{\bar{T}_{products}} \right)} \right) \quad (3-54)$$

For the heat exchangers that have an upstream cavity, the gas and particle radiation heat transfer coefficient need to be corrected by using an empirical correlation presented by Prabir et al. [30]:

$$h'_{G+P rad} = h_{G+P rad} \left(1 + Adj \left(\frac{T_{products,cavity}}{1000} \right)^{0.25} \left(\frac{l_R}{l_b} \right)^{0.07} \right) \quad (3-55)$$

where $T_{products,cavity}$ (in degrees Kelvin) is the product temperature in the upstream cavity. The fuel-dependent adjustment factor (Adj) is an empirical constant presented by Prabir et al. [30]. The term l_R refers to the depth of the cavity, which is equal to zero for the platen superheater.

The heat transfer to the heat exchanger vertical water wall ($\dot{Q}_{vertical WW}$) can be expressed as follows:

$$\dot{Q}_{vertical WW} = h_{vertical WW} A_{vertical WW} (\bar{T}_{products} - T_{vertical WW}) \quad (3-56)$$

where $A_{vertical WW}$ is the surface area of the vertical water (heat exchanger duct wall, see Appendix F for heat exchanger geometry).

The heat transfer coefficient to the vertical water wall ($h_{vertical WW}$) comprises the gas and particle radiation heat transfer coefficient ($h'_{G+P rad, vertical WW}$) and the convective heat transfer coefficient to the vertical water wall ($h_{conv, vertical WW}$):

$$h_{vertical WW} = h'_{G+P rad, vertical WW} + h_{conv, vertical WW} \quad (3-57)$$

The gas and particle radiation heat transfer coefficient to the vertical water wall is calculated using Equations (3-54) and (3-55). In this case, the surface temperature is the vertical water wall's ash deposition layer temperature ($T_{vertical\ WW}$). The convective heat transfer coefficient to the vertical water wall is calculated using a flat plate assumption. The correlations for external forced convection of flow over flat plates are presented by Cengel [38].

The method of determining the heat transfer to the heat exchanger sling tubes ($\dot{Q}_{sling\ tubes}$) is the same as the heat transfer to the heat exchanger vertical water wall:

$$\dot{Q}_{sling\ tubes} = h_{sling\ tubes} A_{sling\ tubes} (\bar{T}_{products} - T_{sling\ tubes}) \quad (3-58)$$

where $A_{sling\ tubes}$ is the surface area of the sling tube wall (see Appendix G for sling tube geometry).

The heat transfer coefficient to the sling tube wall ($h_{sling\ tubes}$) also comprises the gas and particle radiation heat transfer coefficient ($h'_{G+P\ rad, sling\ tubes}$) and the convective heat transfer coefficient to the sling tube wall ($h_{conv, sling\ tubes}$):

$$h_{sling\ tubes} = h'_{G+P\ rad, sling\ tubes} + h_{conv, sling\ tubes} \quad (3-59)$$

The gas and particle radiation heat transfer coefficient to the sling tube wall is calculated using Equations (3-54) and (3-55). In this case, the surface temperature is the sling tube wall's ash deposition layer temperature ($T_{sling\ tubes}$). The convective heat transfer coefficient to the sling tube wall is also calculated using the flat plate assumption-based correlations presented by Cengel [38].

The total heat transfer from the products to the heat exchanger wall ($\dot{Q}_{combined, ext}$) is expressed as follows:

$$\dot{Q}_{combined, ext} = h_{combine, ext} A_{ext} (\bar{T}_{products} - T_{ext}) \quad (3-60)$$

where A_{ext} is the surface area (see Appendix F for heat exchanger geometry) and T_{ext} is the temperature of the ash deposition layer on the heat exchanger tube's external surfaces.

The combined heat transfer coefficient for external flow over the tube bundles ($h_{combine, ext}$) comprises the gas and particle radiation heat transfer coefficient ($h'_{G+P\ rad, ext}$) and the convection heat transfer coefficient of external flow over the tube bundles ($h_{conv, ext}$):

$$h_{combine, ext} = h'_{G+P\ rad, ext} + h_{conv, ext} \quad (3-61)$$

The gas and particle radiation heat transfer coefficient to the outer tube walls is calculated using Equations (3-54) and (3-55). In this case, the surface temperature is the tube bundle outer wall's ash deposition layer temperature.

The heat transfer coefficient for forced convection of external flow over the tube bundles is determined using the Nusselt number of the tube bundle (Nu_{bundle}):

$$h_{conv,ext} = \frac{Nu_{bundle} k}{l_{bundle}} \quad (3-62)$$

where k is fluid's thermal conductivity. The heat transfer characteristic length of the tube bundle (l_{bundle}) is defined as follows:

$$l_{bundle} = \frac{\pi d_o}{2} \quad (3-63)$$

where d_o refers to the outer tube diameter.

The Nusselt number of the tube bundle is an empirical function of Reynolds number ($Re_{products}$), for which the correlations are presented by Gnielinski [41].

For the heat exchangers that have finned tubes, the total heat transfer from the products to the heat exchanger wall can be expressed as:

$$\dot{Q}_{combined,ext} = \eta_o h_{combine,ext} A_{ext,f} (\bar{T}_{products} - T_{ext}) \quad (3-64)$$

where $A_{ext,f}$ is the total external surface area of the finned tubes.

The total external surface area of the finned tubes can be calculated by summing the total fin area (A_f) and the unfinned tube area (A_u):

$$A_{ext,f} = A_f + A_u \quad (3-65)$$

where the total fin area and the unfinned tube area are calculated based on the fin geometry (see Appendix H for fin geometry).

The extended heat transfer surface overall efficiency (η_o) is dependent on the finned tube areas, which can be described using the following expression:

$$\eta_o = [1 - (1 - \eta_f) \frac{A_f}{A_{ext,f}}] \quad (3-66)$$

The fin efficiency (η_f) in Equation (3-66) can be determined using the correlations presented by Schmidt [51] (see Appendix H for fin efficiency).

As the products leave the last heat exchanger (typically the economiser), they ducted to the air heaters to heat the air feeding into the boiler, thereby increasing the thermal efficiency. The infeed air is split into two branches, referred to as primary (PA) and secondary air (SA).

The primary air is heated in the tubular air heater and fed into the mills to transport the pulverised coal to the burner. The tubular air heater also has a tube bank arrangement, in which the air flowing over the tube bank is heated by the products flowing in the tubes, as shown in Figure 3.11.

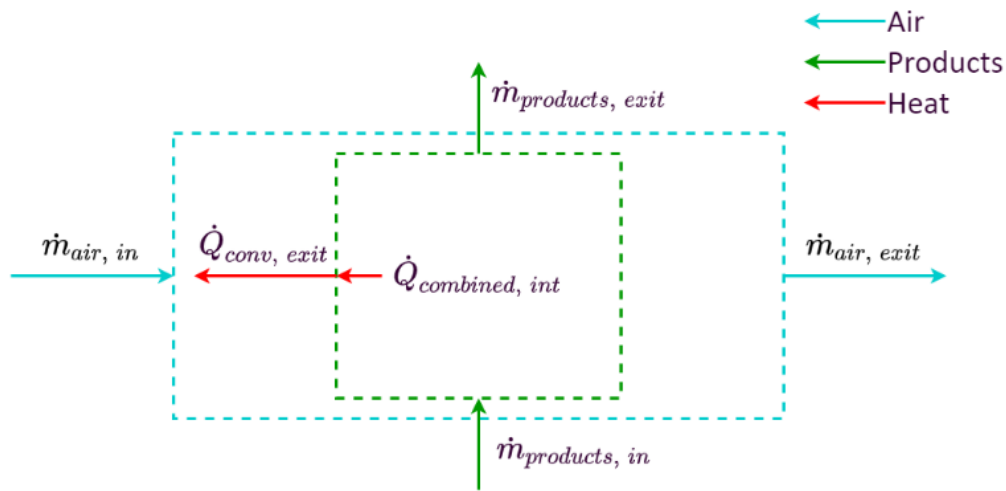


Figure 3.11 Mass and Energy Balance of tubular air heaters

The forced convection of external airflow over the tube bundles ($\dot{Q}_{conv,ext}$) can be determined using the heat transfer coefficient presented in Equation (3-62).

The total heat transfer from products to air ($\dot{Q}_{combined,int}$) is expressed as follows:

$$\dot{Q}_{combined,int} = h_{combine,int} A_{int} (\bar{T}_{products} - T_{int}) \quad (3-67)$$

where A_{int} is the surface area (see Appendix F for heat exchanger geometry) and T_{int} is the temperature of the air heater tube's internal surfaces.

The combined heat transfer coefficient for flow inside the tubes ($h_{combine,int}$) can be expressed as follows:

$$h_{combine,int} = h'_{G+P rad,int} + h_{conv,int} \quad (3-68)$$

In Equation (3-68), the heat transfer coefficient for internal forced convection ($h_{conv,int}$) is determined using the Nusselt number for internal flow (Nu_{int}):

$$h_{conv,int} = \frac{Nu_{int}k}{D_h} \quad (3-69)$$

For internal flows, the hydraulic diameter is the inner tube diameter (d_i). The Nusselt number for internal flow can be found using the Dittus-Boelter or Gnielinski correlations [38].

The secondary air is heated in the rotary air heaters and fed directly to the burner as it leaves the air heater. In the rotary air heaters, the secondary air is indirectly heated from the products through the solid elements, for which the heat transfer is time dependent. However, the modelling method proposed by Kakaç [31] ignores the thermal inertia of the solid elements and simplifies the heat transfer to time-independent convection between the gases.

The heat transfer coefficient of internal forced convection can be determined using Equation (3-69), with the hydraulic diameter ($D_{h,rot}$) found using the following equation:

$$D_{h,rot} = 4 \frac{V_{flow}}{A_{HT}} \quad (3-70)$$

where A_{HT} refers to the solid elements' total heat transfer area. The volume of the flow region in the rotary air heater V_{flow} is taken as half of the rotary air heater duct volume.

However, due to operational concerns, these heat transfer coefficients need to be adjusted using empirical correlations [31]:

$$\begin{aligned} h'_{combine,int} &= \xi_{sweep} c x_{products} h_{combine,int} \text{ for products} \\ h'_{conv,int} &= \xi_{sweep} c x_{air} h_{conv,int} \text{ for air} \end{aligned} \quad (3-71)$$

where ξ_{sweep} is the non-uniform sweeping factor, which ranges between 0.8 to 0.9.

The terms $x_{products}$ and x_{air} refer to the heat transfer area fractions of each gas flow in the rotary air heater. The term c is a constant that is related to the air heater's rotational speed n_{AH} (rpm) as shown in Table 3.1.

Table 3.1 Coefficient c for rotary air heater rotational speed [31]

Air heater rotational speed n_{AH} (rpm)	Coefficient c
0.5	0.85
1	0.97
1.5	1

3.4 Water/steam side thermofluid modelling

In a subcritical PCFB, the working fluid exists in three different states, namely water, steam and two-phase mixture. The modelling methodology is more straightforward for single-phase flow such as water and steam, whereas two-phase flow is more complex to model.

One of the approaches for modelling two-phase flow is the homogeneous mixture model. This type of model assumes that the liquid and gas phases are distributed evenly over the cross-sectional area of the flow path. It is assumed that the pressure, temperature and velocity of the phases are equal. Furthermore, the model assumes no distinct flow regimes/patterns; hence the single-phase constitutive equations and correlations can be implemented for the mixture [48].

3.4.1 Fluid properties

The fluid properties for single-phase flow can be found directly from the existing databases such as NIST or ASME. Whereas in the two-phase flow, the fluid properties are more implicit.

In a homogeneous mixture model, the effective fluid properties of the liquid-gas mixture can be obtained by the weighted average value of each phase. The weight fraction or quality ($\bar{x}$) is defined as:

$$\bar{x} = \frac{\dot{m}_{gas}}{\dot{m}_{mix}} = \frac{\dot{m}_{gas}}{\dot{m}_{gas} + \dot{m}_{liq}} = \frac{\bar{h}_{mix} - h_{liq}}{h_{gas} - h_{liq}} \quad (3-72)$$

where $\bar{h}_{mix}$ is the average specific enthalpy of the mixture, solved by the conservation equations. The specific enthalpy of the liquid (h_{liq}) and gas (h_{liq}) can be found from the NIST or ASME databases.

3.4.2 Conservation equations

For single-phase flow, the conservation equations are the same as those mentioned in Section 3.3.2.

For the two-phase flow, the mass conservation using the homogeneous mixture model can be expressed as follows [48]:

$$\sum \dot{m}_{exit} - \sum \dot{m}_{in} = \rho_{mix,e} v_{xe} A_e - \rho_{mix,i} v_{xi} A_i = 0 \quad (3-73)$$

The term ρ_{mix} in Equation (3-73) refers to the liquid-gas mixture density, which is defined as follows [48]:

$$\rho_{mix} = \frac{(1 - \bar{x})}{\rho_{liq}} + \frac{\bar{x}}{\rho_{gas}} \quad (3-74)$$

The fluid velocity (v) in Equation (3-73) refers to the two-phase mixture flow speed, which for the homogeneous mixture model, is assumed to be identical to the gas and liquid flow speed.

The momentum conservation can be expressed as follows [47], [48]:

$$\begin{aligned} & \frac{(\rho_{mix,i}v_{xi} + \rho_{mix,e}v_{xe})(v_{xe} - v_{xi}) - (\rho_{mix,e}v_{xe}^2 - \rho_{mix,i}v_{xi}^2)}{2} \\ & = -(p_{oe} - p_{oi}) - \bar{\rho}_{mix}g(z_e - z_i) - \phi_f \left(\frac{fl}{D_h} + \Sigma K \right) \frac{\bar{\rho}_{mix}|\bar{v}_x|\bar{v}_x}{2} \end{aligned} \quad (3-75)$$

where ϕ_f is an empirical multiplier for the frictional component. The multiplier is added to the single-phase pressure drop equations to incorporate the effects of the two-phase mixture. The frictional multiplier is calculated using the Lockhart-Martinelli method described in [48].

Energy conservation can be expressed as follows [48]:

$$\begin{aligned} & \Sigma \dot{m}_{exit}(h_{oe} + gz_e) - \Sigma \dot{m}_{in}(h_{oi} + gz_i) \\ & = \rho_{mix,e}v_{xe}A_e(\bar{h}_{mix,oe} + gz_e) - \rho_{mix,i}v_{xi}A_i(\bar{h}_{mix,oi} + gz_i) \\ & = \Sigma \dot{Q} + \Sigma \dot{W} \end{aligned} \quad (3-76)$$

where the specific stagnation enthalpy of the mixture ($\bar{h}_{mix,o}$) is defined as follows:

$$\bar{h}_{mix,o} = \bar{h}_{mix} + \frac{1}{2}v_x^2 \quad (3-77)$$

The specific enthalpy of the mixture $\bar{h}_{mix}$ is defined as follows:

$$\bar{h}_{mix} = \bar{u}_{mix} + p\bar{v}_{mix} \quad (3-78)$$

where the mixture fluid properties in Equation (3-78) is given by:

$$\begin{aligned} \bar{u}_{mix} &= (1 - \bar{x})u_{liq} + \bar{x}u_{gas} \\ \bar{v}_{mix} &= (1 - \bar{x})v_{liq} + \bar{x}v_{gas} \end{aligned} \quad (3-79)$$

3.4.3 Momentum source terms

Besides gravitational potential, the water, steam, or mixture flowing inside the boiler tubes lose their pressure due to frictional and secondary losses.

The frictional pressure losses in the boiler tubes are accounted for using the frictional loss factor, typically found using empirical correlations such as Darcy Weisbach [48].

The secondary pressure losses in the boiler tubes are caused by sudden expansion and contraction, which are accounted for by the secondary loss coefficients. For a complex tube network, calculating the secondary loss coefficients for different components is intricate; therefore, the secondary loss coefficients of boiler tubes are determined using design specifications or plant measurements.

3.4.4 Energy source terms

The heat transfer of the water/steam inside boiler tubes is purely convective ($\dot{Q}_{conv,int}$), and can be expressed as follows:

$$\dot{Q}_{conv,int} = h_{conv,int} A_{int} (\bar{T}_{H_2O} - T_{int}) \quad (3-80)$$

where A_{int} is the surface area and T_{int} is the temperature of the boiler tube internal surfaces. The term $\bar{T}_{H_2O}$ refers to the boiler tube's mean water, steam, or mixture temperature.

The internal heat transfer coefficient ($h_{conv,int}$) of the flow inside boiler tubes can be determined using Equation (3-69), and the associated Nusselt number can be found using the Dittus-Boelter or Gnielinski correlations [38]. Additionally, the heat transfer correlation of Sato and Matsumara is also applied in addition to the correlations mentioned above since flow boiling heat transfer is more complex than single-phase heat transfer [48].

3.5 Heat transfer in solid materials

The heat transfer mode within a solid is purely conductive, and it can be modelled using Fourier's law of heat transfer [38]:

$$\dot{Q}_{cond} = -k A_{cond} \frac{\partial T}{\partial x} \quad (3-81)$$

3.5.1 Boiler tube walls

For modelling conductive heat transfer in the boiler tube walls, Fourier's law of heat transfer is applied to a one-dimensional cylindrical element, shown in Figure 3.12.

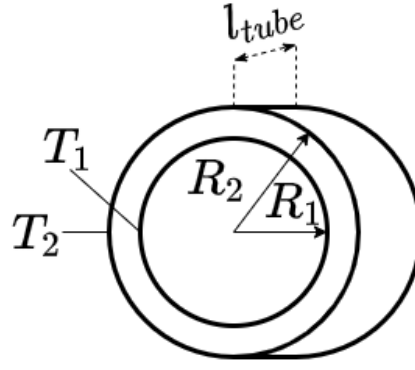


Figure 3.12 1D cylindrical element

l_{tube} refers to the length of the tube, R refers to the radius of the tube, and T refers to the surface temperature. The conductive heat transfer is therefore expressed as follows [38]:

$$\dot{Q}_{cond} = -k \left(\frac{2\pi l_{tube}}{\ln \left(\frac{R_2}{R_1} \right)} \right) (T_2 - T_1) \quad (3-82)$$

The conductive heat transfer through different surfaces is often described as a thermal network, similar to an electric circuit. This analogy treats the solid body as thermal resistance, heat transfer rate as thermal current, and the temperature difference as thermal potential. Based on this analogy, the thermal resistance of a tube wall section (R_{th}) can be expressed as [38]:

$$R_{th} = \frac{\Delta T}{\dot{Q}_{cond}} = \frac{\ln \left(\frac{R_2}{R_1} \right)}{2\pi k l_{tube}} \quad (3-83)$$

The electric circuit analogy also works for cases where different layers of solid bodies are arranged in series ($R_{th,series}$) or parallel ($R_{th,parallel}$) [38]:

$$R_{th,series} = \sum_{i=1}^n R_{th,i} \quad (3-84)$$

and

$$\frac{1}{R_{th,parallel}} = \sum_{i=1}^n \frac{1}{R_{th,i}} \quad (3-85)$$

These heat transfer equations are sufficient for modelling the conduction in boiler tubes, except the furnace membrane walls, where the set-up is more complex. For that reason, the membrane walls can be modelled using a detailed thermal resistance network developed by Rousseau and Laubscher [37]. The membrane wall model can capture temperatures at different wall regions, and this model takes the radiative heat flux projected onto the wall as its heat input.

3.5.2 Ash deposition

In a thermal network, ash deposition such as fouling and slagging can be modelled as a separate conductive thermal resistance between the tube wall and the products. The thermal resistance of the ash deposition layer is typically accounted for by using empirical data since modelling ash deposition is a complex phenomenon that depends on various factors.

For the case where the heat exchanger tubes are finned, the extended heat transfer overall efficiency is also applied to the thermal resistance of the ash deposition layer.

For PCFBs, the slagging resistance of the platen superheater is generally highest amongst the heat exchangers, and it can be found using correlations shown in Figure 3.13.

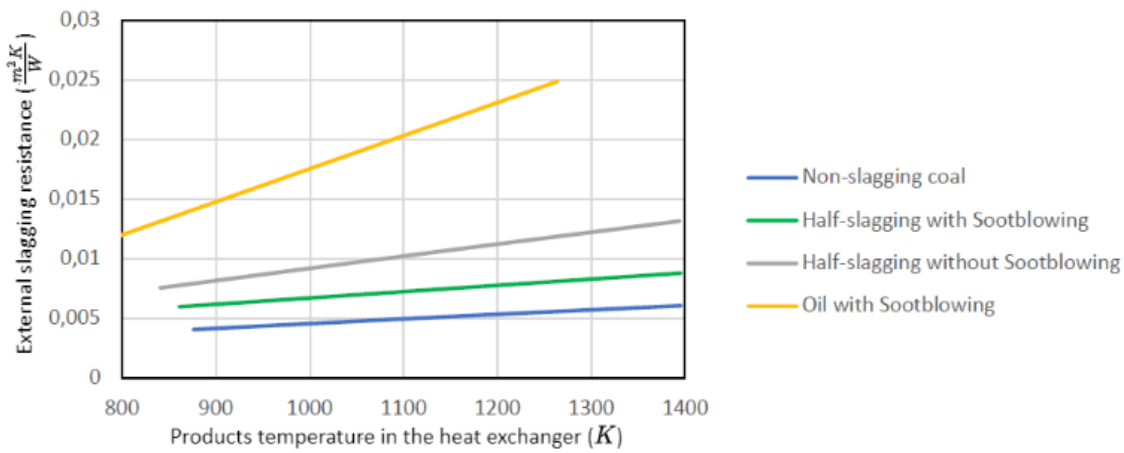


Figure 3.13 External slagging resistances for coal-oil fired superheaters/reheaters [33]

The linear function used to find the external slagging resistance ($R_{slagging,ext}$) is fitted from the data shown in Figure 3.13 are:

$$R_{slagging,ext} = 3.85 \times 10^{-6} \cdot T_{products} + 7.46885 \times 10^{-4}$$

for non-slugging coal

$$R_{slagging,ext} = 5.27497 \times 10^{-6} \cdot T_{products} + 1.47684 \times 10^{-3}$$

for half-slugging coal with soot blowing

(3-86)

$$R_{slagging,ext} = 1.01224 \times 10^{-5} \cdot T_{products} + 8.99667 \times 10^{-4}$$

for half-slugging coal without soot blowing

$$R_{slagging,ext} = 2.77299 \times 10^{-5} \cdot T_{products} + 1.01451 \times 10^{-2}$$

for oil with soot blowing

For convective superheaters and reheaters, the external fouling resistance ($R_{fouling,ext}$) is typically assumed to be constant, shown in Table 3.2.

Table 3.2 External fouling resistances for superheaters and reheaters [33]

Superheater/reheater configuration & fuel	Fouling resistance [$\frac{m^2 K}{W}$]
In-line + staggered firing oil	0.0026
In-line firing solid fuel	0.0043

Superheaters and reheaters usually have in-line configurations. The main reason for this is that the ash particle impaction rate in staggered tube banks is higher than in the case of in-line banks. The higher impaction rate and the higher product temperature would cause accelerated ash deposition on the heat exchanger tube surfaces in the solid fuel firing boiler.

The theories discussed in this chapter are applied to develop the integrated thermofluid network model of the case study boiler using the commercial one-dimensional thermofluid simulation software Flownex. The software simulates a one-dimensional thermofluid system by numerically solving the conservation equations presented in Sections 3.3.2 and 3.4.2 together with the fluid properties obtained from NIST or ASME databases for each control volume in the network.

For the gas mixture streams (air and products), Flownex utilise the real gas mixing methods mentioned in Section 3.3.1 to obtain the gas mixture's fluid properties.

As mentioned in Section 3.3.1, the enthalpies of the gas mixtures are determined using ideal gas assumptions and the mixing method mentioned in Section 3.3.1. For the boiler model, the ideal gas enthalpies of each product constituent are obtained using polynomials fitted from the data in the NIST database at the reference temperature of 25 degrees Celsius (see Appendix I for polynomials).

Additionally, Flownex's built-in database does not contain fluid properties of sulphur dioxide; therefore, the fluid properties of sulphur dioxide are also obtained using polynomials fitted from the ideal gas data in the NIST database at the reference temperature of 25 degrees Celsius (see Appendix I for polynomials).

The solid particles in the products, namely, ash and unburnt carbon, are modelled as pseudo-gases in the model. These solid particles are typically small fractions of the product, and therefore their effects on most of the products' fluid properties are negligible. For that reason, most of their pseudo-gas fluid properties are simplified to be the same as nitrogen (more than 65% of products). However, solid particles' density, molar mass and enthalpy are not insignificant; thus, these three fluid properties of the solid particles' pseudo-gases are considered.

The ash and unburnt carbon's densities are assumed to be the same, which is a constant of $2200 \frac{kg}{m^3}$ [34]. The molar mass of ash is dependent on its chemical composition. For high ash coal, the chemical compositions of the ash can be found in the article by Vassilev and Vassileva [52] (See Appendix J for ash molar mass).

The enthalpies of the solid particles are also referenced from the same temperature, and they are determined using the specific heat capacities presented in Section 3.2.2.

The integrated thermofluid network model of the case study boiler comprises multiple components, which their layouts and details are discussed in the subsequent chapter.

4. Model development

By applying the theory presented in Chapter 3, the integrated process model of the case study boiler was developed in Flownex.

The boiler model is discretised into different component models, with each component having its own representative one-dimensional thermofluid network. Additionally, the vertical water wall tubes of the boiler are discretised according to the number of heat exchangers, with one increment for each heat exchangers duct and one increment for the pre-cavity duct.

Each component model is developed by applying the theory discussed in Chapter 3, in which the details of the models are discussed in this chapter.

4.1 Boiler physical layout

The case study boiler is a tower type once-through boiler with a physical layout shown in Figure 4.1.

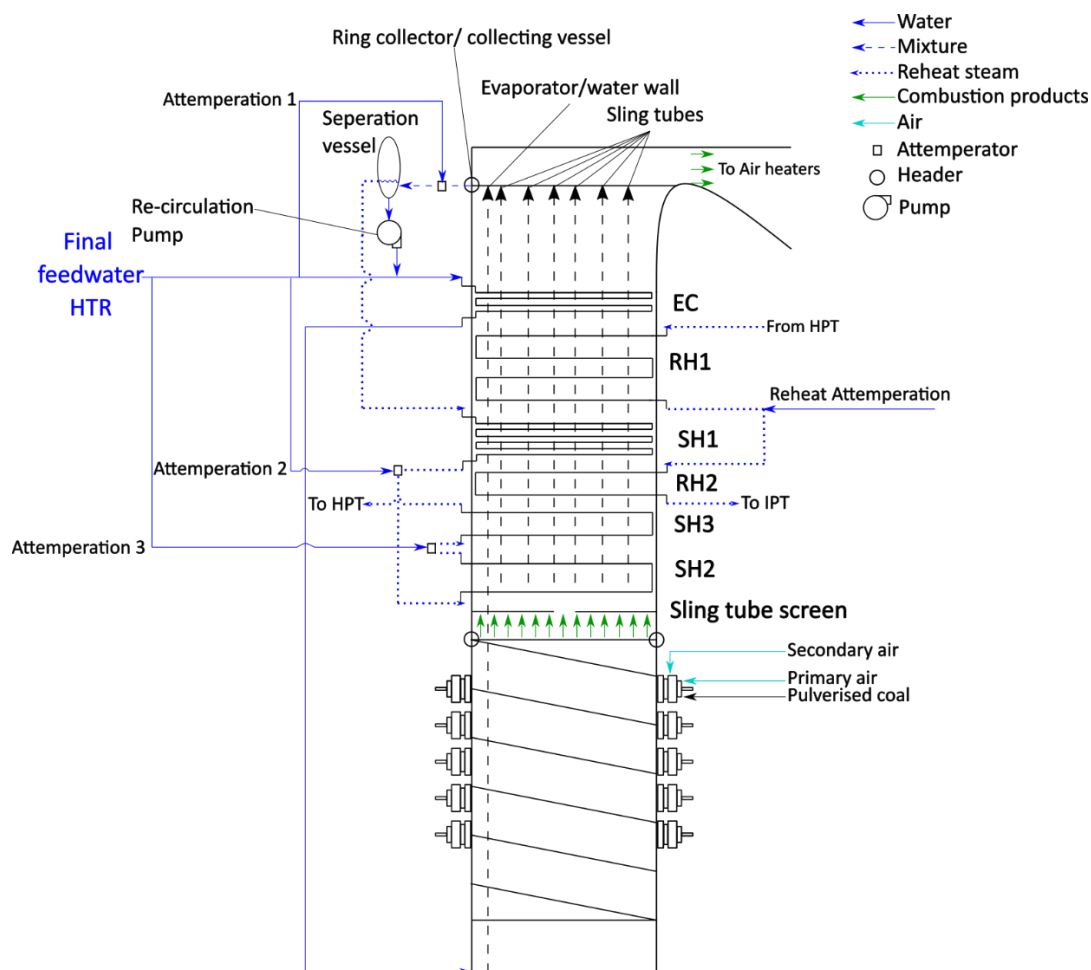


Figure 4.1 Case study boiler physical layout

As shown in Figure 4.1, the PA and SA leaving the air heaters are fed into the burners together with the pulverised coal for combustion. After combustion, the products flow straight up the boiler and heat the water/steam as the products flow past each heat exchanger. The products leaving the boiler are then fed into the air heaters to heat the infeed PA and SA.

The water/steam path flow inside the boiler follows the path shown in Figure 3.2, and the steam leaving the final superheater (SH3) is fed into the high-pressure turbine. The steam leaving the high-pressure turbine is fed back into the boiler for reheat and the steam leaving the final reheater (RH2) is fed into the Intermediate-pressure turbine.

The furnace wall forms part of the evaporator, which comprises helical membrane tubes that coil up to the furnace exit headers (see Appendix K furnace wall geometry). The steam or mixture leaving these headers branches into the sling tubes and the vertical water wall, which gets further heated by the flue gas stream in the heat exchanger duct.

In the case study boiler, the sling tubes form a staggered heat exchanger screen at the furnace exit with a layout shown in Figure 4.2 and Figure 4.3.

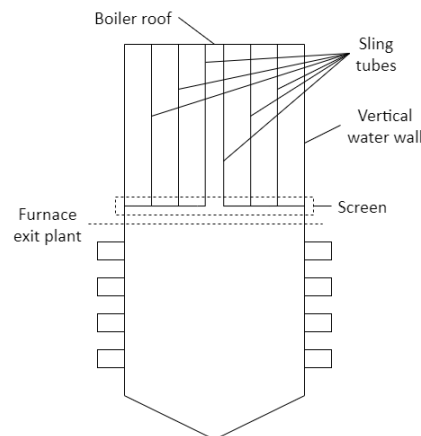


Figure 4.2 Sling tube position

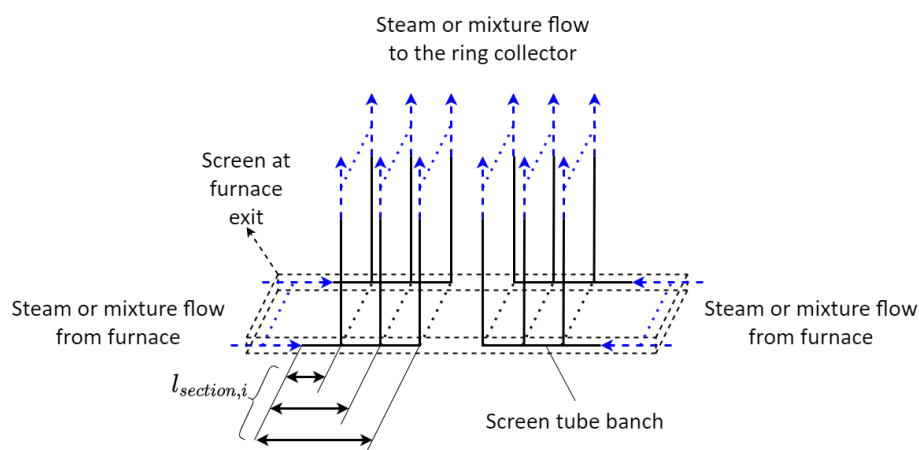


Figure 4.3 Sling tube screen layout

Once steam or mixture inside the sling tubes and vertical water wall reaches the boiler roof, it gets fed into the ring collector and separated by the separation vessel if the boiler operates in wet mode. In dry mode, the steam simply flows through the separation vessel to the superheaters.

4.2 Integrated boiler model layout

The developed integrated boiler model is based on the physical boiler layout shown in Figure 4.1, which has a model layout shown in Figure 4.4.

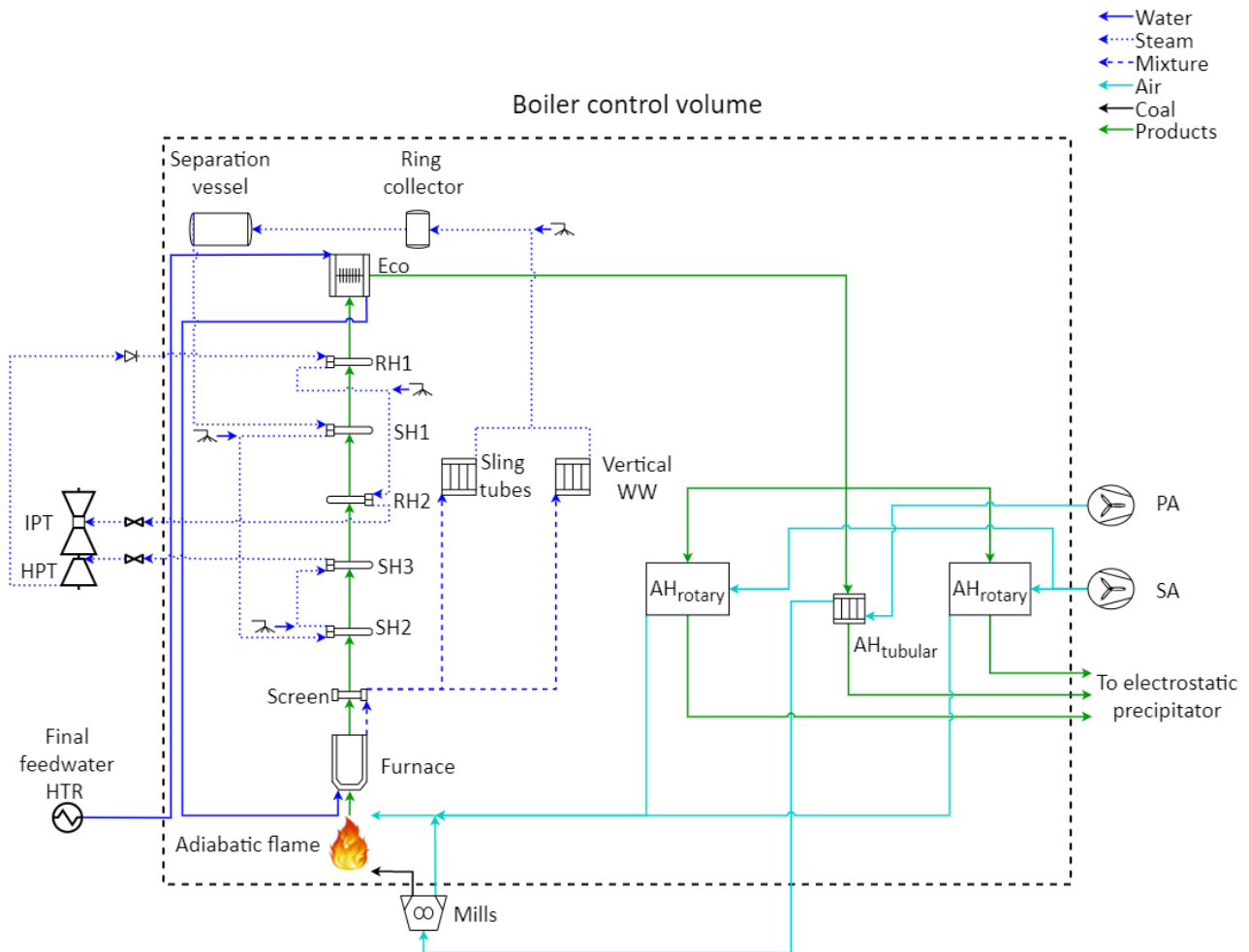


Figure 4.4 Process model diagram of the case study boiler

As mentioned in Chapter 1, the model focuses on the boiler's steady-state performance; therefore, any control features, such as the furnace pressure control by the induced draft fan, or CFPP components outside the control volume shown in Figure 4.4 are not directly modelled. Instead, their influence on the modelled components are included using boundary conditions, obtained from design specifications or plant measurements.

4.3 Component models

4.3.1 Adiabatic flame

For the boiler model, the adiabatic flame model performs the pulverised coal combustion analysis presented in Section 3.2 and has a layout shown in Figure 4.5 (a).

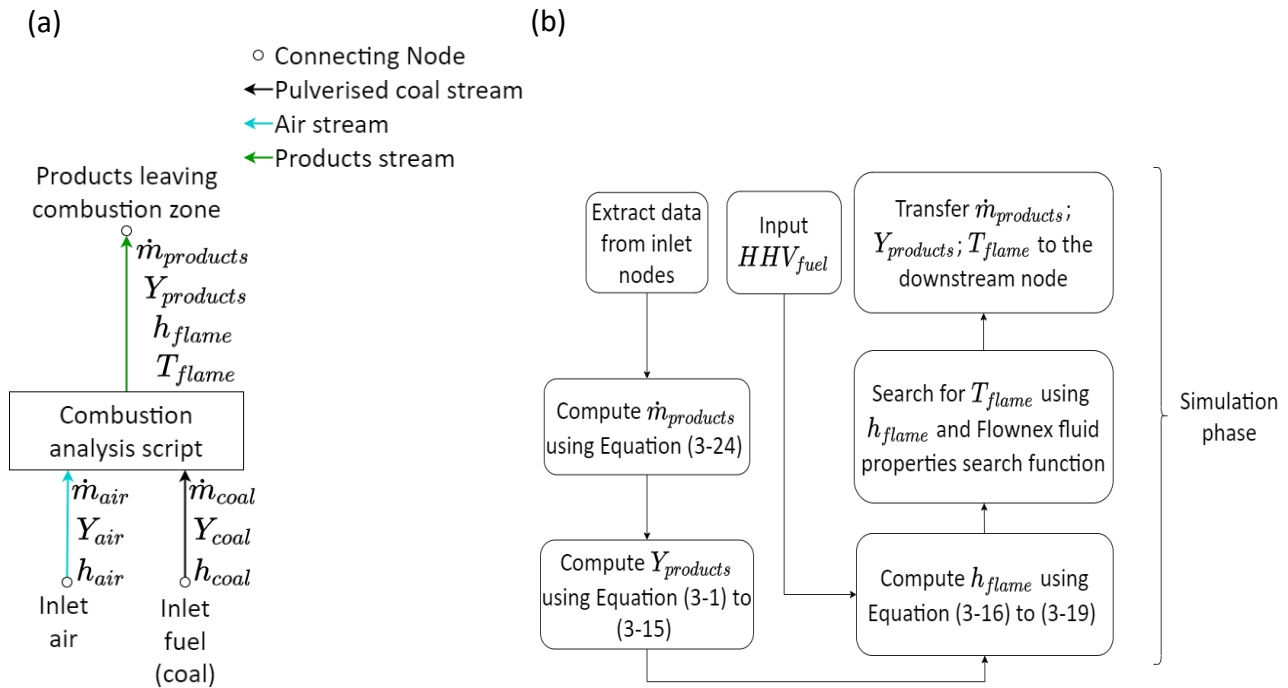


Figure 4.5 Adiabatic flame model (a) layout and (b) script logic diagram

The inlet mixed air feeding into the model in Figure 4.5 (a) is considered a gas mixture of nitrogen, oxygen and moisture, modelled using the method discussed in Section 3.3.1.

As mentioned in Section 3.2, the combustion analysis performed by the model only considers the flame control volume's mass and energy balance; therefore, only the parameters shown in Figure 4.5 (a) are extracted from the inlet coal node. For that reason, the pulverised coal is modelled as a pseudo-gaseous mixture of air and solid fuel. In the model, the combustion analysis is performed by a custom script component, which follows the logic shown in Figure 4.5 (b). The output data of the script gets transferred to the downstream products node, which feeds into the furnace component.

4.3.2 Furnace

As shown in Figure 4.4, the products leaving the adiabatic flame model flow into the furnace, where the component has a layout shown in Figure 4.6.

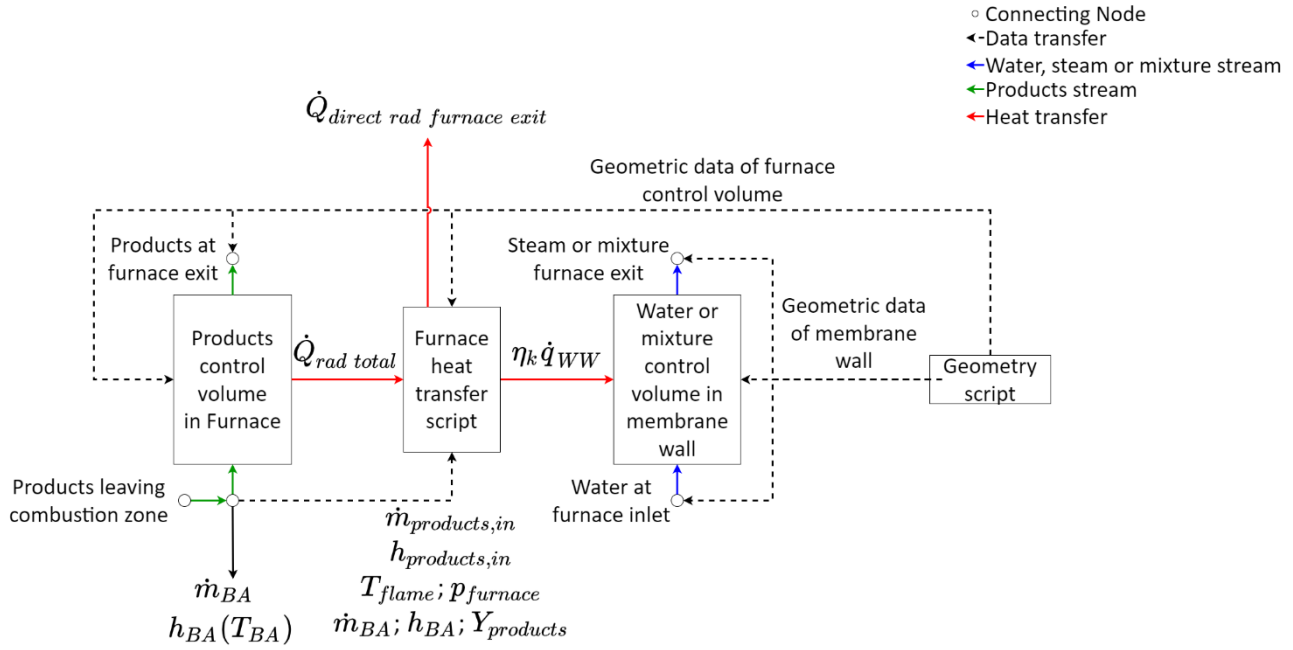


Figure 4.6 Furnace model layout

As shown in Figure 4.6, the radiation heat transfer in the furnace is computed by a custom script component with the logic shown in Figure 4.7.

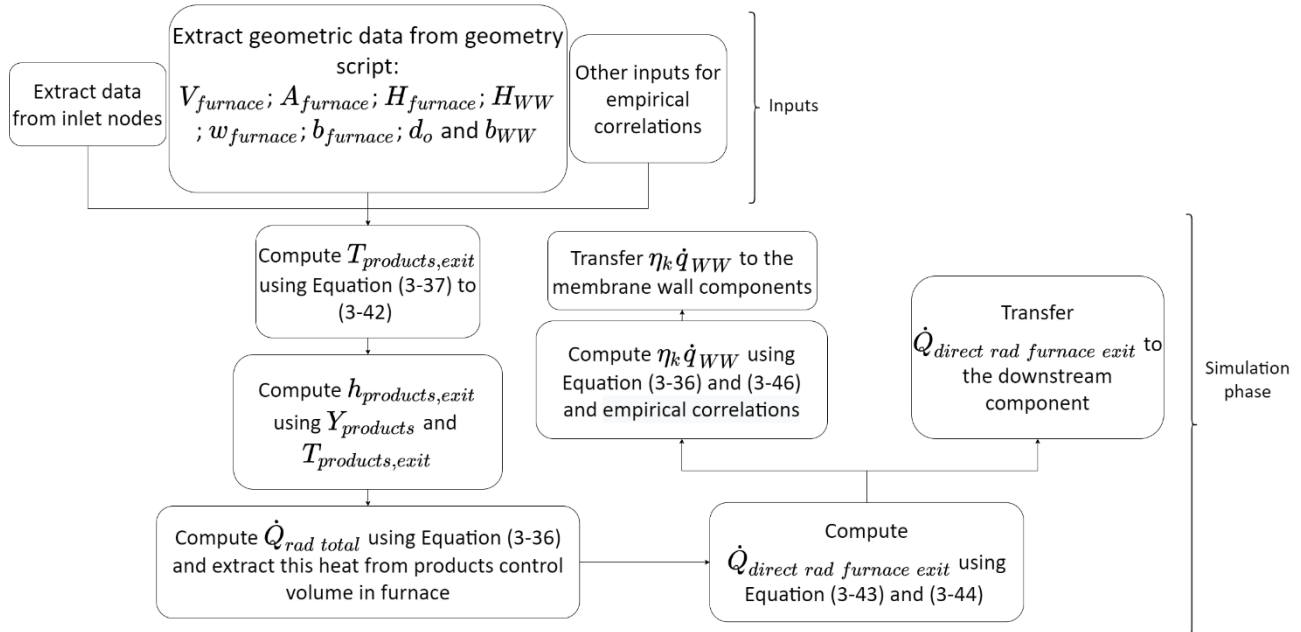


Figure 4.7 Furnace heat transfer script logic diagram

The geometric data such as height ($H_{furnace}$), width ($w_{furnace}$) and depth ($b_{furnace}$) of the furnace shown in Figure 4.7 can be obtained from the technical drawings of the boiler. The outer diameter (d_o) and axial pitch (b_{WW}) of the furnace wall tubes, are obtained from the design specifications (see Appendix K for the membrane wall geometry).

The volume ($V_{furnace}$) and the planar area of the furnace can also be calculated using the technical drawing of the boiler, which for the model, these parameters are computed using computer-aided design (CAD) software Onshape [53]. The term H_{WW} refers to the furnace wall section heights, which is an array of heights computed based on the discretisation.

As shown in Figure 4.6, the radiation heat flux to each section of the wall ($\eta_k \dot{q}_{WW}$) computed by the script is transferred to the membrane wall model, which has a layout shown in Figure 4.8.

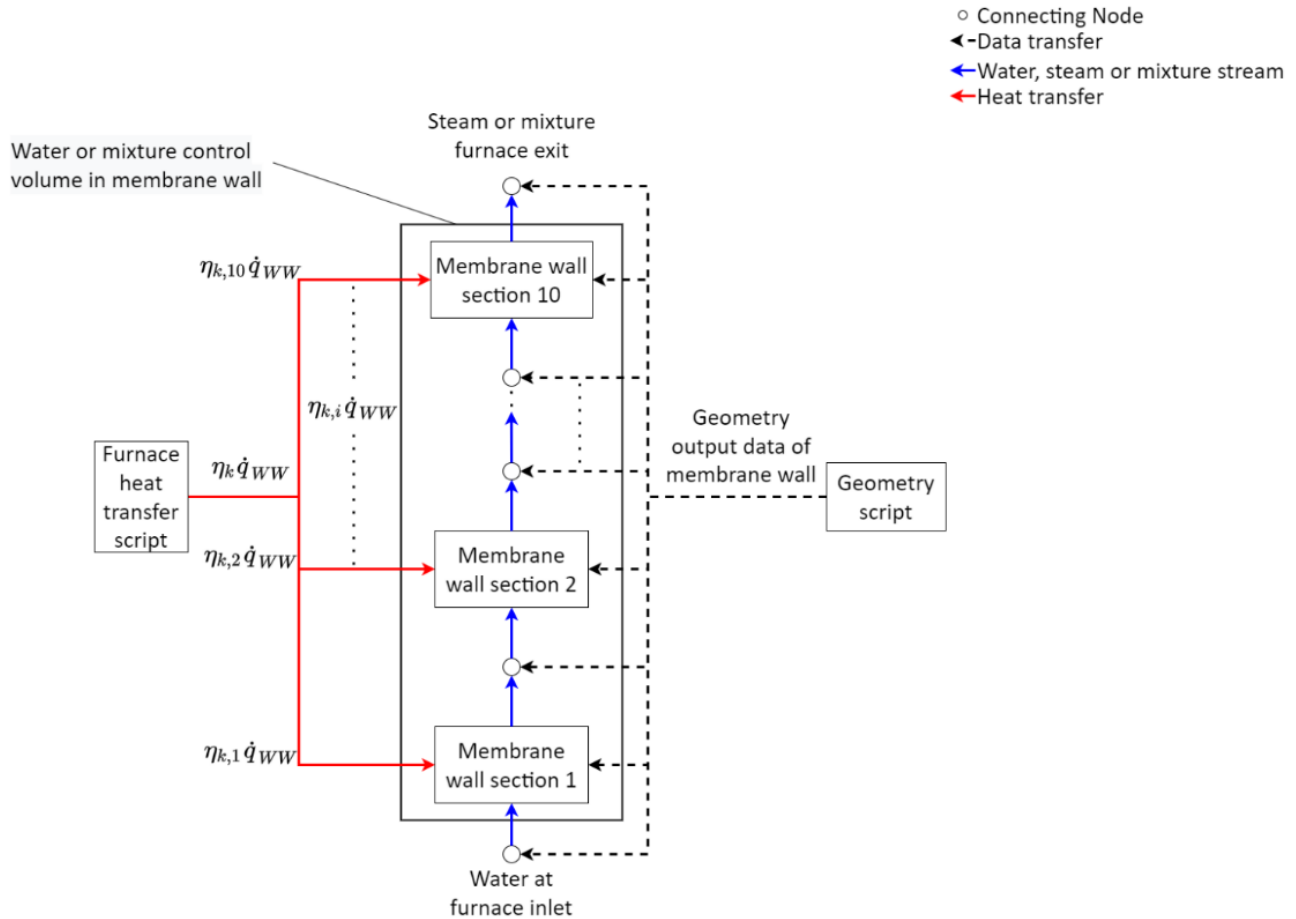


Figure 4.8 Furnace membrane wall model layout

In Figure 4.8, the heat transfer in the membrane wall is modelled using the approach discussed in Section 3.5.1. The heat transfer in the water or mixture inside the membrane wall tubes is modelled using the Gnielinski correlation [38] together with Sato and Matsumara's correlation [48], as discussed in Section 3.4.4.

4.3.3 Superheaters, reheaters & economiser

The tube bank heat exchangers that are downstream of the furnace, such as the superheater (SH), the reheater (RH) and the economiser (Eco), are all modelled using the same approach discussed in

Chapter 3. These heat exchangers are discretised into multiple control volumes, each containing a single pass of the tube bank, as shown in Figure 4.9.

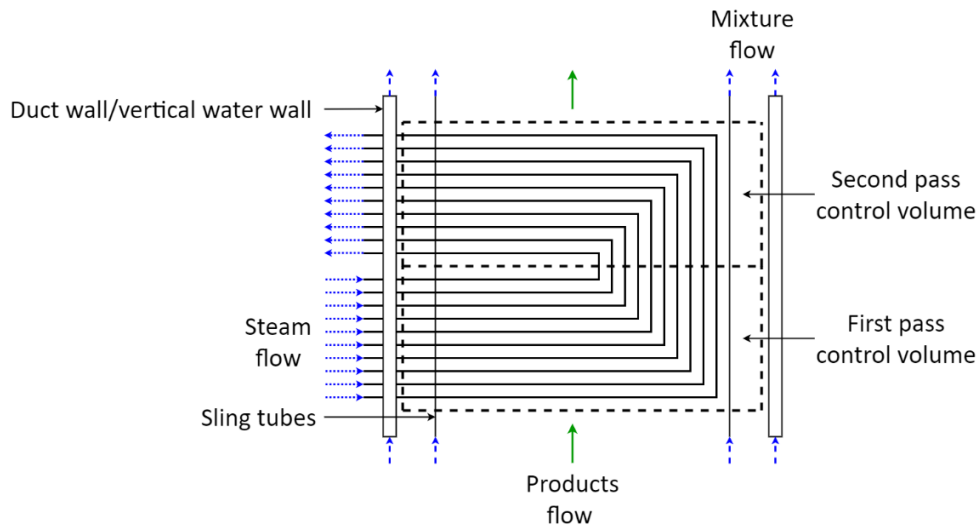


Figure 4.9 Heat exchanger discretisation

Each control volume shown in Figure 4.9 can be modelled as a thermofluid network shown in Figure 4.10.

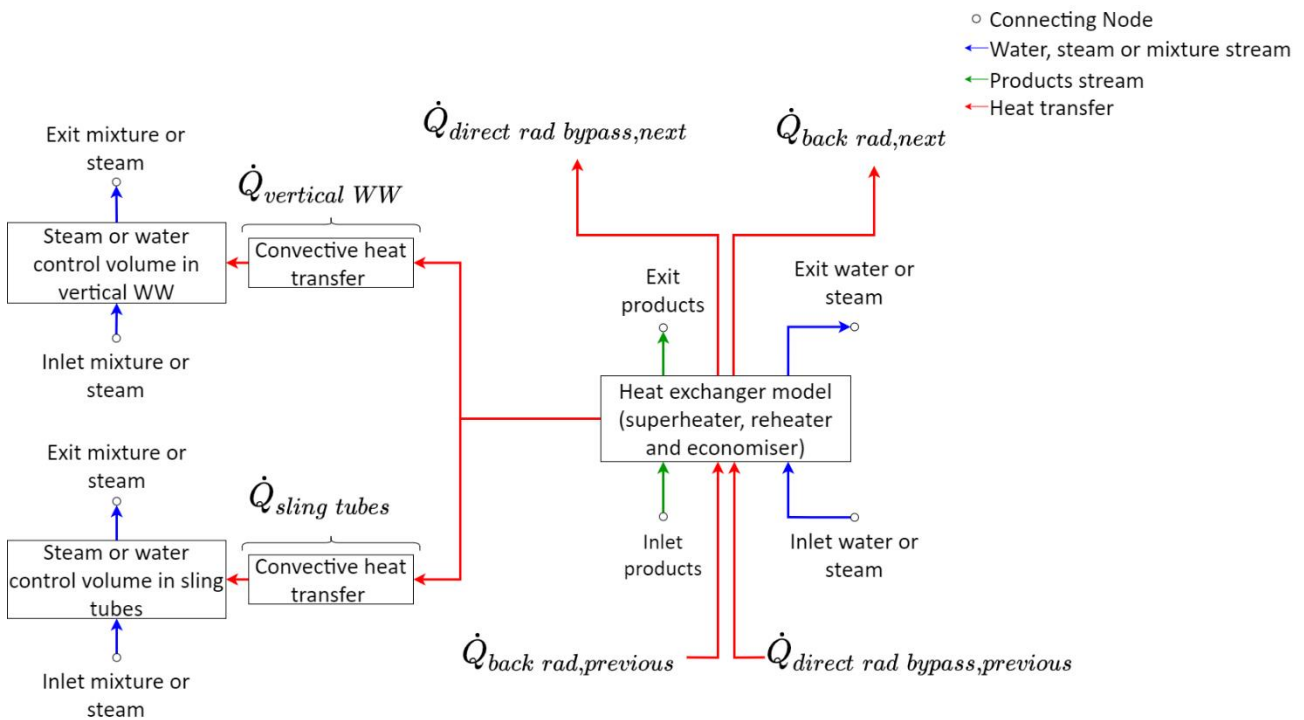


Figure 4.10 Heat exchanger external model layout

As shown in Figure 4.10, the vertical water wall and the sling tubes are not considered part of the heat exchanger component model since these tubes are part of the evaporator. For that reason, the heat transfers to these tubes are modelled separately as external heat transfer components to the

heat exchanger model and computed using the approach discussed in Section 3.3.4. The heat exchanger model shown in the figure represents the control volume of a single pass, which has a layout shown in Figure 4.11.

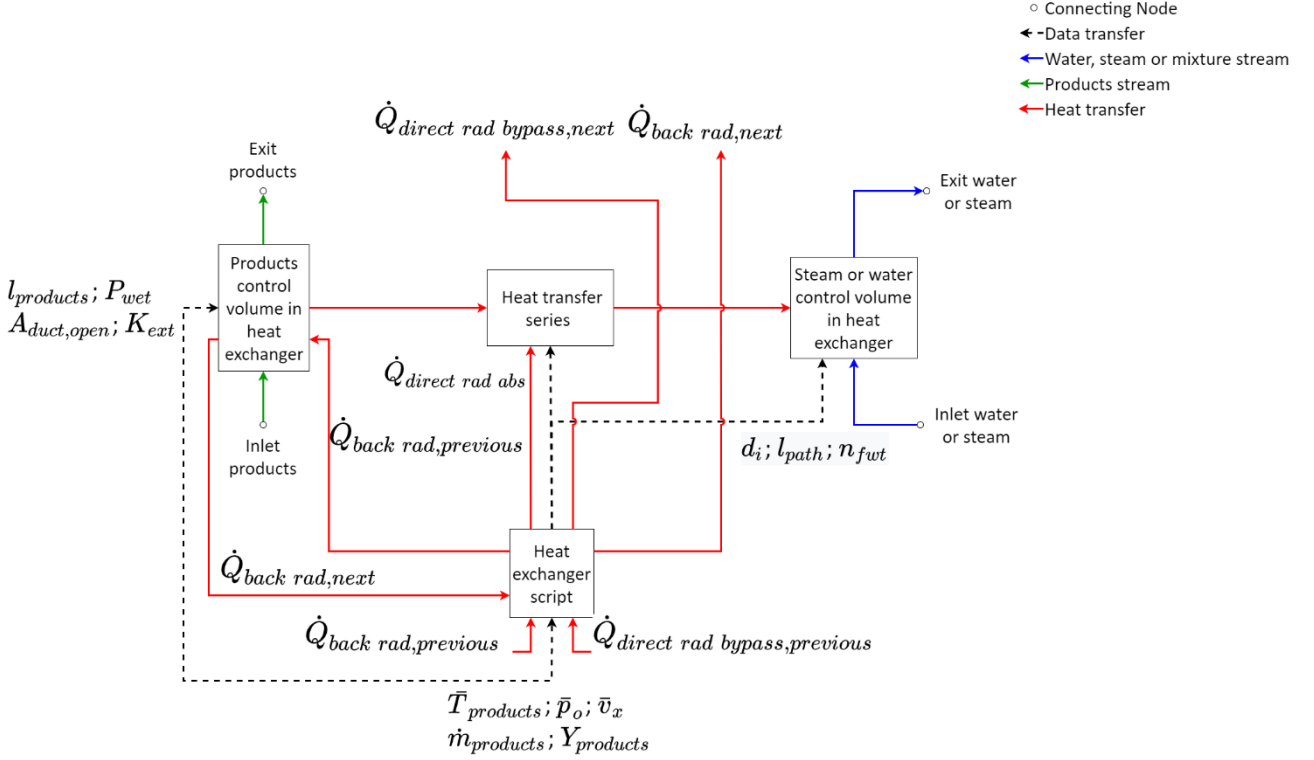


Figure 4.11 Heat exchanger model internal layout

As shown in Figure 4.11, the back panel and bypassed direct radiation from the upstream component are transferred to the heat exchanger script as inputs. The script processes the radiation inputs using the approach discussed in Section 3.3.4, in which the outputs are transferred to the corresponding components.

The products control volume in the heat exchanger is described by the open duct area ($A_{duct,open}$), length of the control volume ($l_{products}$) and the wetted perimeter (P_{wet} , see Appendix F for products control volume geometry). Whereas the water or steam control volume in the heat exchanger is described by the length (l_{path}) and inner diameter of the tubes together with the total number of tubes in one pass (n_{fwt} , see Appendix F for tube bank geometry).

Additionally, the model accounts for the pressure loss in the products stream using the secondary loss coefficient, computed by the script using the method discussed in 3.3.3.

The heat transfer between the two control volumes in the heat exchanger is modelled using a series of heat transfer components, represented by a thermal resistance network shown in Figure 4.12.

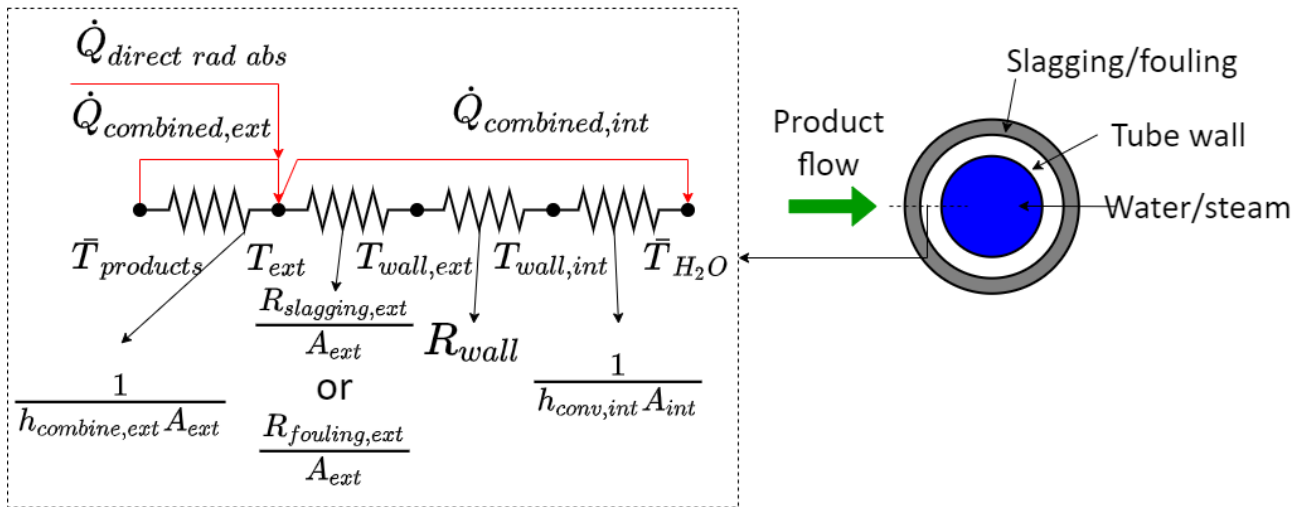


Figure 4.12 Heat transfer equivalent thermal resistance series

As shown in Figure 4.11, the heat transfer series in the heat exchanger model is based on the thermal resistance series shown in Figure 4.12, which has a layout shown in Figure 4.13.

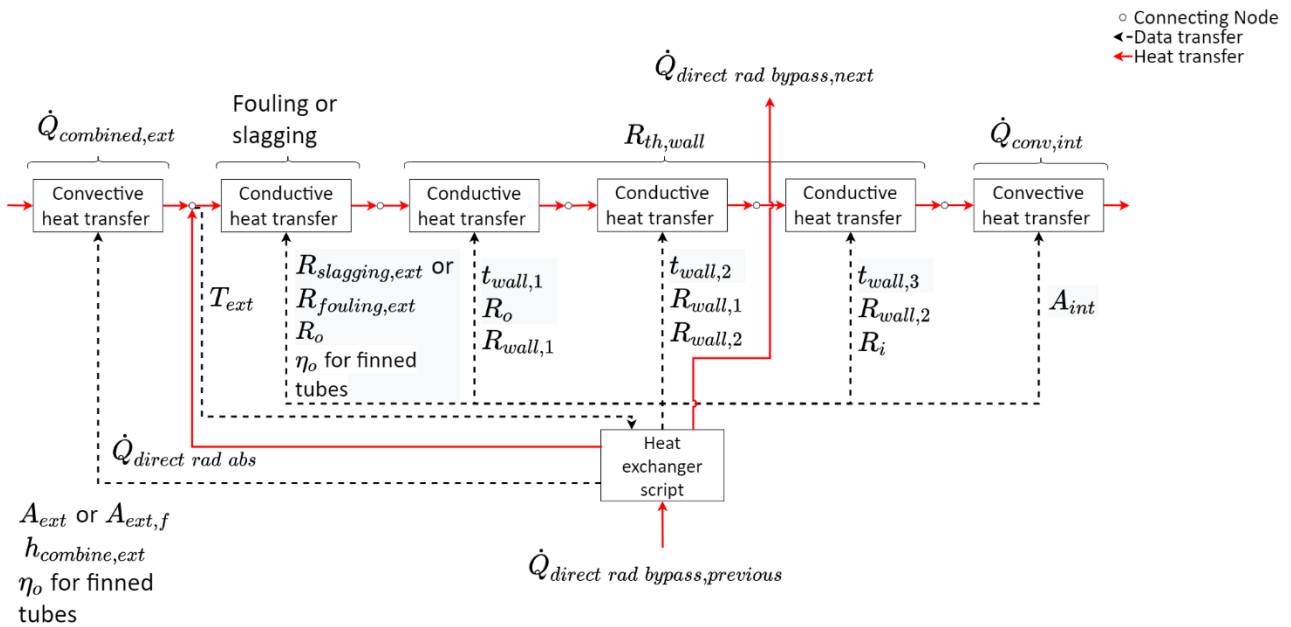


Figure 4.13 Heat transfer series model

As mentioned before, the heat exchanger component is developed to have the capability to model all types of heat exchangers in the case study boiler, including the tube banks with finned tubes. For the economiser, rectangular finned tubes are used; therefore, the modelling approach for the heat transfer to the finned tubes discussed in Section 3.3.4 is applied to the tube external surfaces.

The direct radiation absorbed by the tube external surfaces is computed by the custom script and directly transferred to the external surface node, as shown in Figure 4.13. Additionally, the heat

transfer in the water or steam inside the tubes is modelled using the Gnielinski correlation [38], as discussed in Section 3.4.4.

As mentioned before, the script computed all the geometric and thermofluid parameters, which has a logic shown in Figure 4.14.

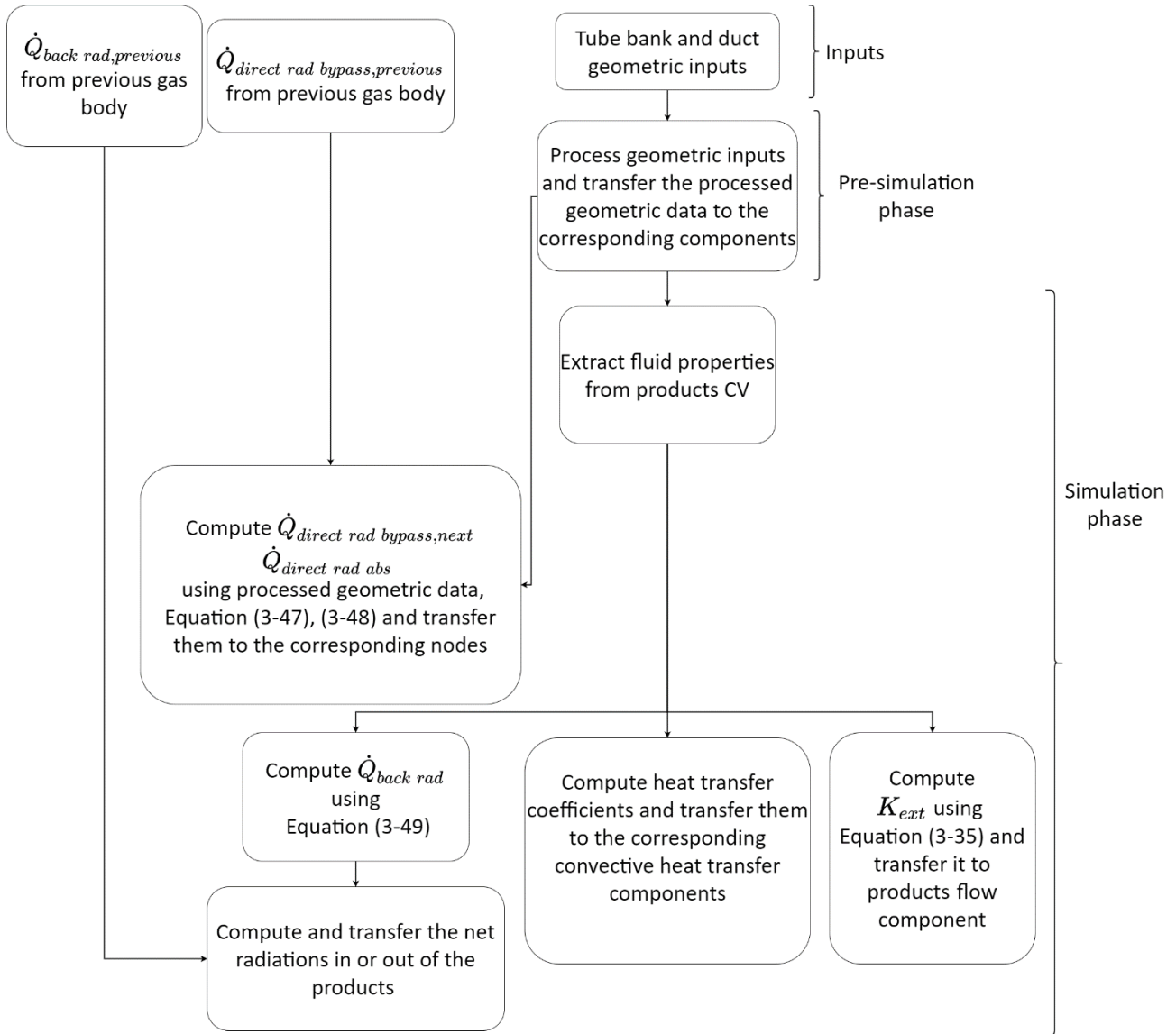


Figure 4.14 Heat exchanger script logic

As shown in Figure 4.14, the geometric data related parameters are computed once during the pre-simulation phase, outside the solver loops. The script only computes the thermal or fluid property-dependent parameters during the simulation phase since these properties are constantly changing until the simulation converges.

4.3.4 Sling tube screen

As shown in Figure 4.2 and Figure 4.3, the sling tubes at the furnace exit form a horizontal screen, which can be treated as a single row tube bank (see Appendix G for sling tube screen geometry). For that reason, the sling tube screen component is modelled as a heat exchanger presented in 4.3.3 with one row of tubes.

4.3.5 Air heaters

As discussed in Section 3.3.4, the boiler infeed air is split into two branches, with the primary air heated by the tubular air heater and the secondary air heated by the rotary air heaters. The tubular air heater is also a tube bank heat exchanger but with an arrangement different to the heat exchanger discussed in Section 4.3.3. For that reason, the tubular air heater component is modelled using the same approach discussed in Section 4.3.3, with a model layout shown in Figure 4.15.

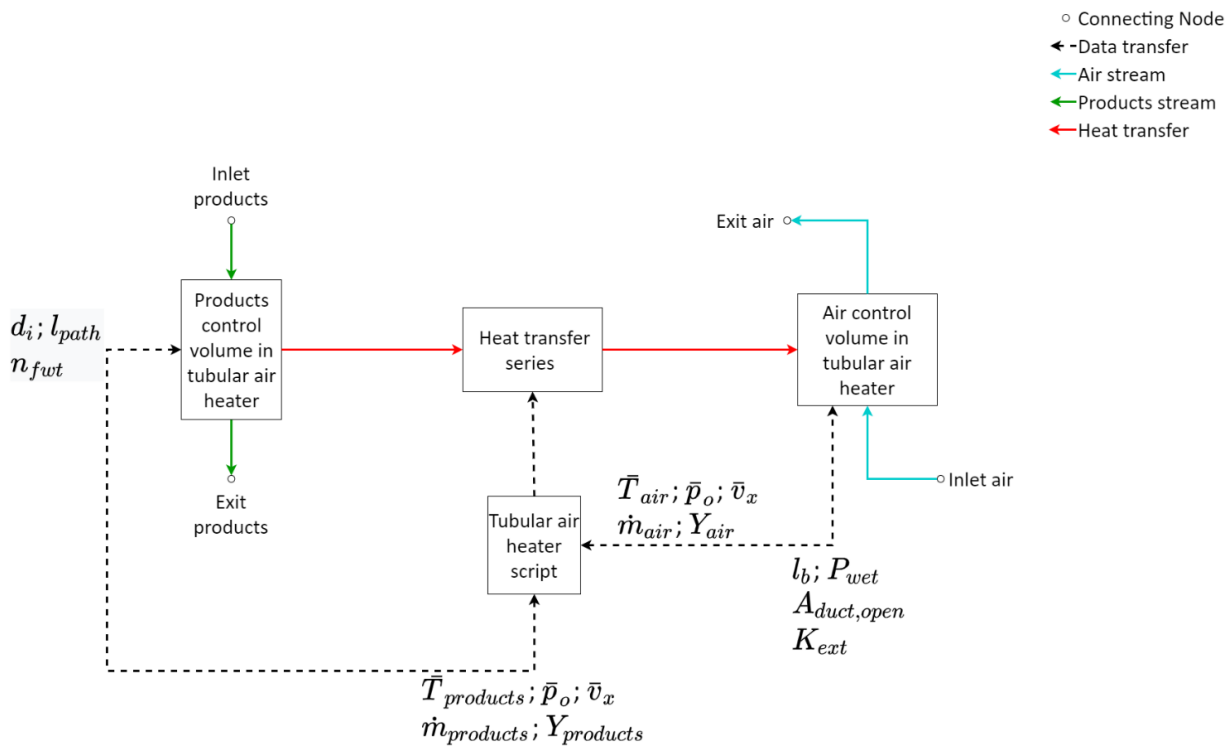


Figure 4.15 Tubular air heater model layout

As shown in Figure 4.15, the products control volume in the component represents the flow of the products inside the tubes, which transfers heat to the air flowing over the tube bank.

As shown in Figure 4.4, the air heaters are not located in the same path as the steam or water tube banks; therefore, the direct radiation cannot be projected to the air heaters' outer tube surfaces.

The layout of the heat transfer network in the component is the same as in Figure 4.13, except the model no longer transfers the direct radiation as a heat source term to the tube surface node, as shown in Figure 4.16.

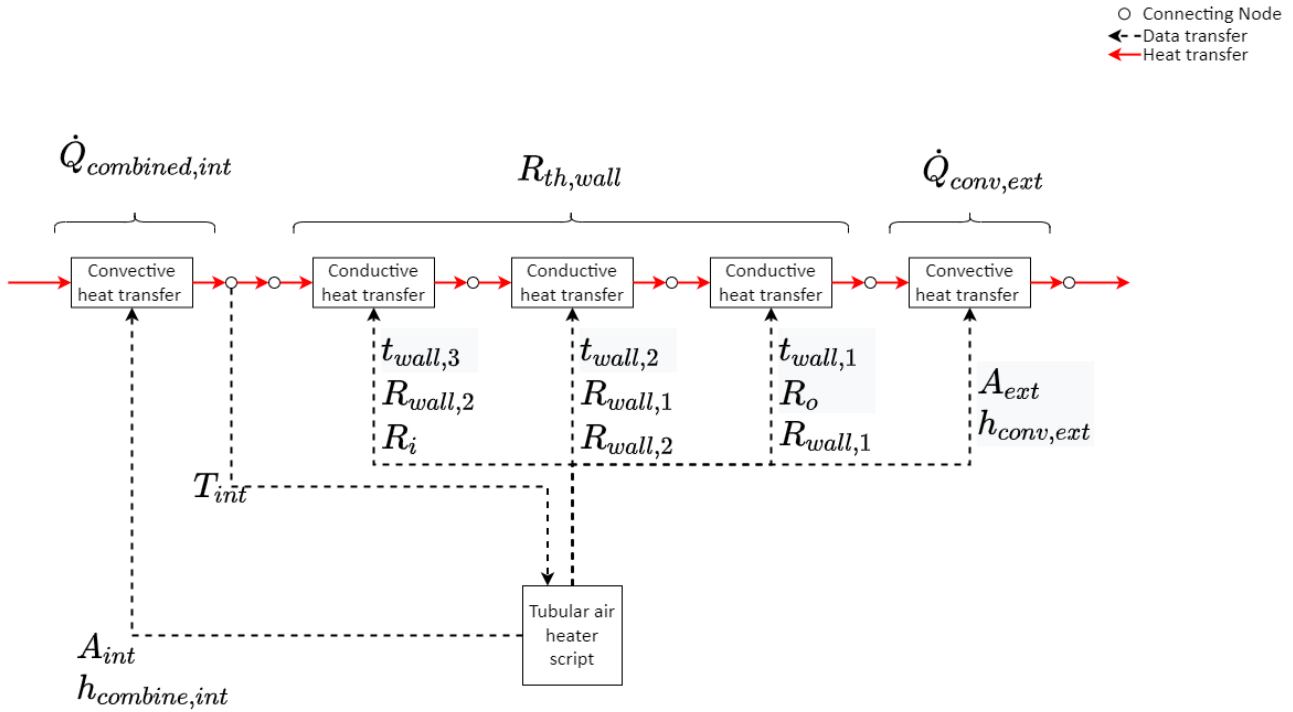


Figure 4.16 Tubular air heater heat transfer series model

The tubular air heater script follows the same logic shown in Figure 4.14, except the script no longer processes the radiation terms, as shown in Figure 4.17.

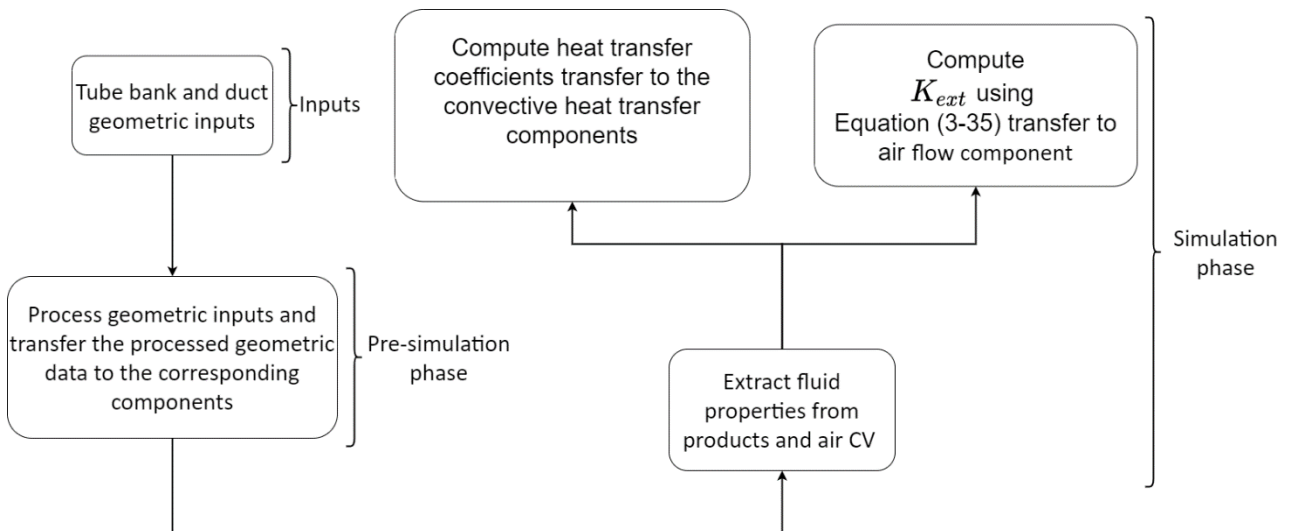


Figure 4.17 Tubular air heater script logic

The rotary air heaters are counter-flow regenerative heat exchangers modelled using the correlations discussed in Section 3.3.4, and the component has a layout shown in Figure 4.18.

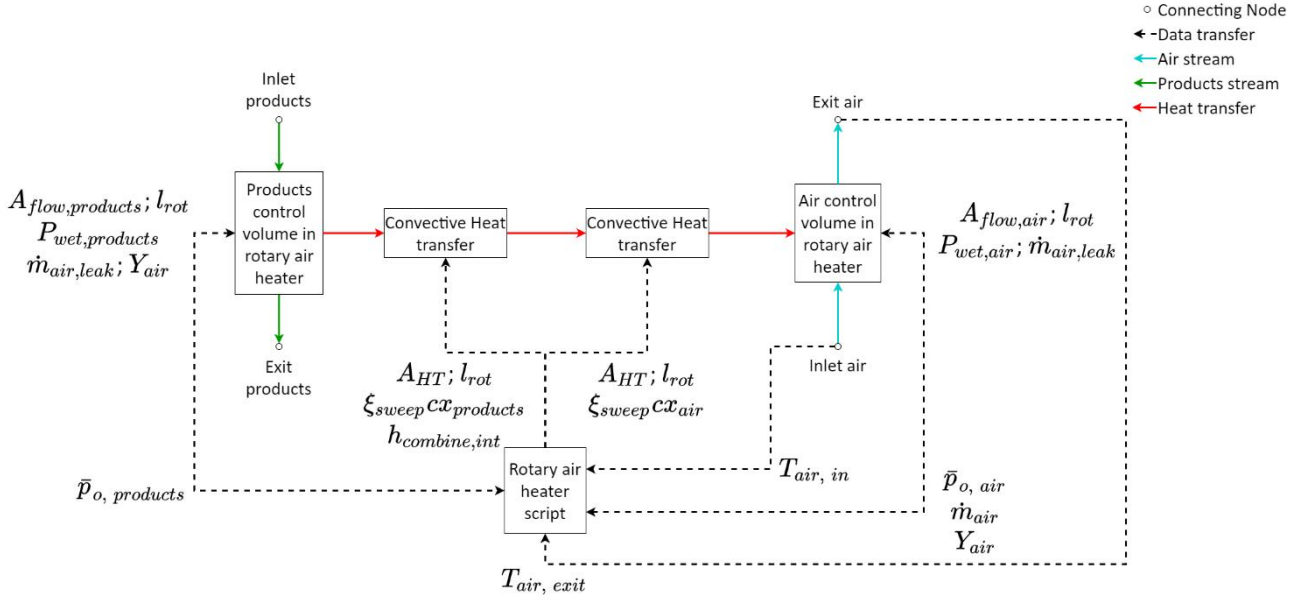


Figure 4.18 Rotary air heater model layout

As shown in Figure 4.18, the products and air control volume in the component are described by the flow area (A_{flow}), length of the air heater (l_{rot}) and the wetted perimeter (P_{wet}), which has the hydraulic diameter discussed in Section 3.3.4. The geometric data, including the heat transfer area (A_{HT}) are the inputs to the model, which can be obtained from the drawings and the design specifications.

Additionally, the rotary air heater suffers from cross-contamination between the gases, for which the mass flow rate of air leakage ($\dot{m}_{air,leak}$) can be calculated using the air heater geometry. In a rotary air heater, the two major sources of leakage are gap leakage ($\dot{m}_{air,leak,gap}$) and carryover leakage ($\dot{m}_{air,leak,carryover}$), which the overall air leakage is the sum of the two:

$$\dot{m}_{air,leak} = \dot{m}_{air,leak,gap} + \dot{m}_{air,leak,carryover} \quad (4-1)$$

Gap leakage refers to the high-pressure air leak into the low-pressure products through the gaps between the rotating and stationary parts, which can be calculated using the following equation [34]:

$$\dot{m}_{air,leak,gap} = K_{gap} A_{flow,gaps} (2\Delta p \rho_{air})^{0.5} \quad (4-2)$$

In Equation (4-2), the term $A_{flow,gaps}$ refers to the flow area of the gaps found in the plant data and K_{gap} refers to the discharge coefficient ranging from 0.4 to 1.0. The pressure difference between the gases (Δp) and the air density (ρ_{air}) are fluid properties obtained from the control volumes, as shown in Figure 4.18.

Carryover leakage refers to the air carried from the rotor heating compartment into the products stream, which can be calculated using the following equation [34]:

$$\dot{m}_{air,leak,carryover} = \rho_{air} \bar{V}_{rotor} n_{comp} f_{discharge} \quad (4-3)$$

where n_{comp} refers to the number of compartments in each radial plane and $\bar{V}_{rotor}$ refers to the average rotor compartment volume, as shown in Figure 4.19.

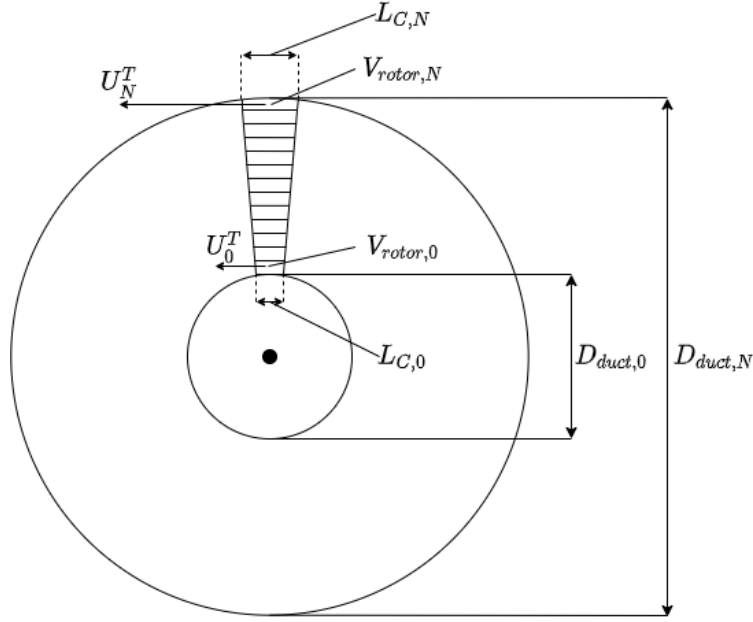


Figure 4.19 Rotary air heater schematics [34]

As shown in Figure 4.19, the compartments are situated between the outer and inner rim of the air heater; therefore, the volume of the air heater duct can be determined using the hollow cylinder volume equation:

$$V_{flow} = \frac{\pi(D_{duct,0}^2 - D_{duct,N}^2)l_{rot}}{4} \quad (4-4)$$

where the term $D_{duct,0}$ refers to the inner radius and $D_{duct,N}$ refers to the outer radius of the air heater duct.

The average compartment volume can be approximated by considering the compartment as a tube with a diameter equal to the hydraulic diameter of the compartment and the length equal to the length of the air heater. The assumptions simplify the geometry to a cylinder, for which the volume can be calculated using the following equation:

$$\bar{V}_{rotor} = \frac{\pi D_{h,rot}^2 l_{rot}}{4} \quad (4-5)$$

The average compartment discharge frequency ($f_{discharge}$) can be calculated using the tangential velocities (U_0^T, U_N^T) and the widths ($L_{c,0}, L_{c,N}$) of the compartments shown in Figure 4.19:

$$f_{discharge} = \sqrt{0.5 \left(\left[\frac{U_0^T}{L_{c,0}} \right]^2 + \left[\frac{U_N^T}{L_{c,N}} \right]^2 \right)} \quad (4-6)$$

where the subscript 0 refers to the compartments that are closest to the centre of the rotor and N refers to the compartments that are furthest from the centre.

As mentioned in Section 3.3.4, the rotational speed of the air heater is typically between 0.5 to 1.5 rpm, and the tangential velocity of the compartments can be calculated using the compartment's radial position together with the rotational speed:

$$U_0^T = n_{AH} \cdot \left(\frac{2\pi}{60} \right) \cdot \left(\frac{D_{duct,0}}{2} \right) \quad (4-7)$$

$$U_N^T = n_{AH} \cdot \left(\frac{2\pi}{60} \right) \cdot \left(\frac{D_{duct,N}}{2} \right) \quad (4-8)$$

As shown in Figure 4.18, the air leakage in the air heater is computed by the script, which has a logic shown in Figure 4.20.

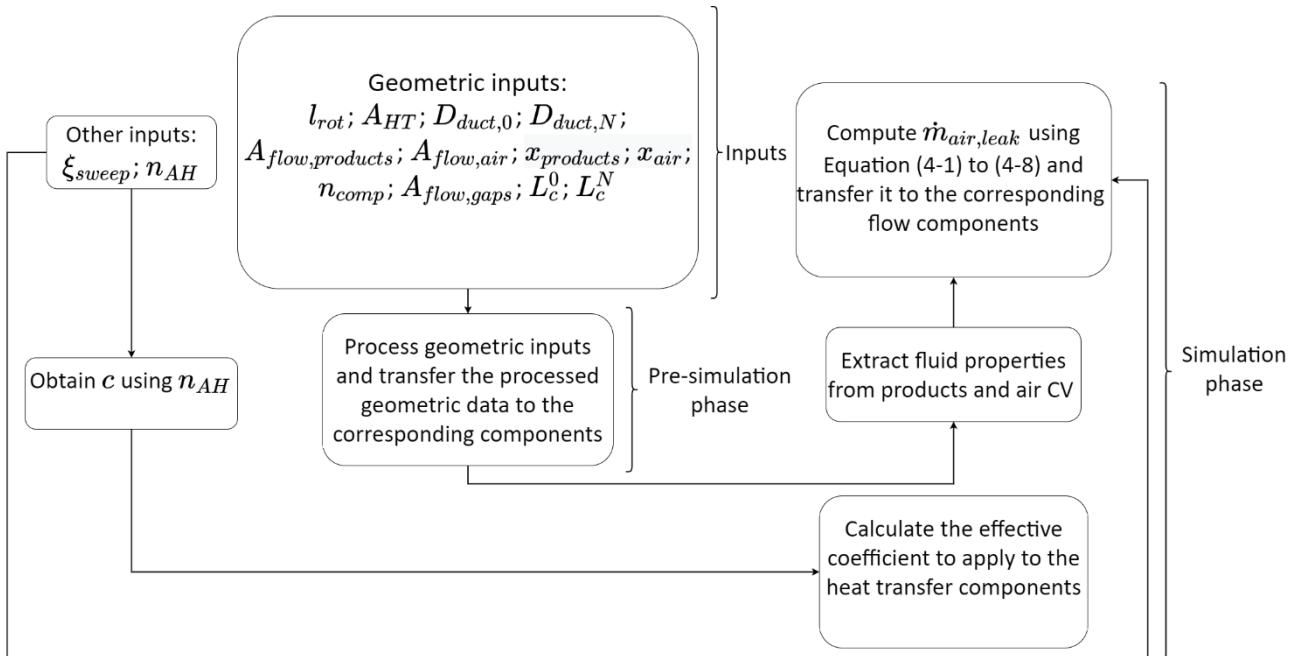


Figure 4.20 Rotary air heater script logic flow

4.4 Model verification & validation

Before assembling the component models presented in Section 4.3 to create the integrated boiler model, each component was verified against their corresponding process model developed using Python (see Appendix L for verification results). After verifying all the component models, the integrated model of the case study boiler was created by assembling the components in the layout shown in Figure 4.4. The integrated boiler model was verified using the design specifications, which prescribes inputs and boundary conditions of the boiler at full and part load (see Appendix M for boundary conditions and inputs).

The full load (100% maximum continuous rating of boiler unit) simulation results show that the boiler model achieves energy balance, presented using a Q'T diagram in Figure 4.21.

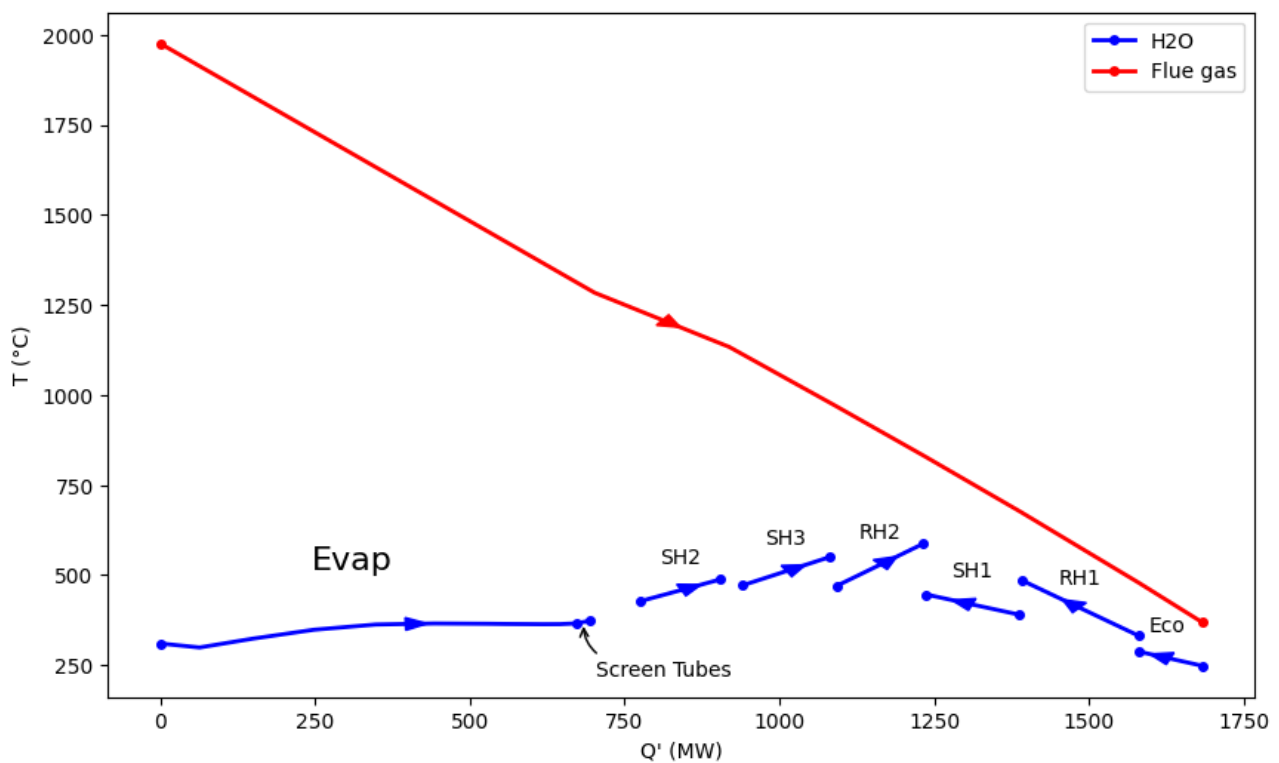


Figure 4.21 Boiler model Q'T diagram for simulation using designer specification at full load

The horizontal axis of the Q'T diagram shows the cumulative heat transfer between fluids in the boiler heat exchangers. As shown in Figure 4.21, the overall heat loss in the flue gas/products equals the heat gain in the water/steam, thus indicating that the system does achieve energy balance. The horizontal gaps in the water/steam streams are the heat transfer to the vertical water wall and the sling tubes. The heat transfer in these tubes is not shown in the diagram because the heat transfer into each tube segment is small and scatters throughout the boiler. For that reason, these heat transfer terms are difficult to label and show on the Q'T diagram. However, these heat transfer

terms need to be included in the overall mass and energy balance on the system; thus, resulting in the horizontal gaps in the diagram.

The fully verified model was first validated using the designer specifications, which prescribes boiler performance at 100% and 70% MCR. The results of the validation exercise are presented in the subsequent sections.

4.4.1 Validation using designer specification: pressure results

As discussed in Section 3.4.3, each boiler component's water/steam pressure loss coefficients are computed using the validation data at 100% MCR (see Appendix M for pressure loss related parameters). This set of pressure loss coefficients is then applied to the 70% MCR simulation, for which the results show sufficient accuracy compared to the validation data, as shown in Figure 4.22.

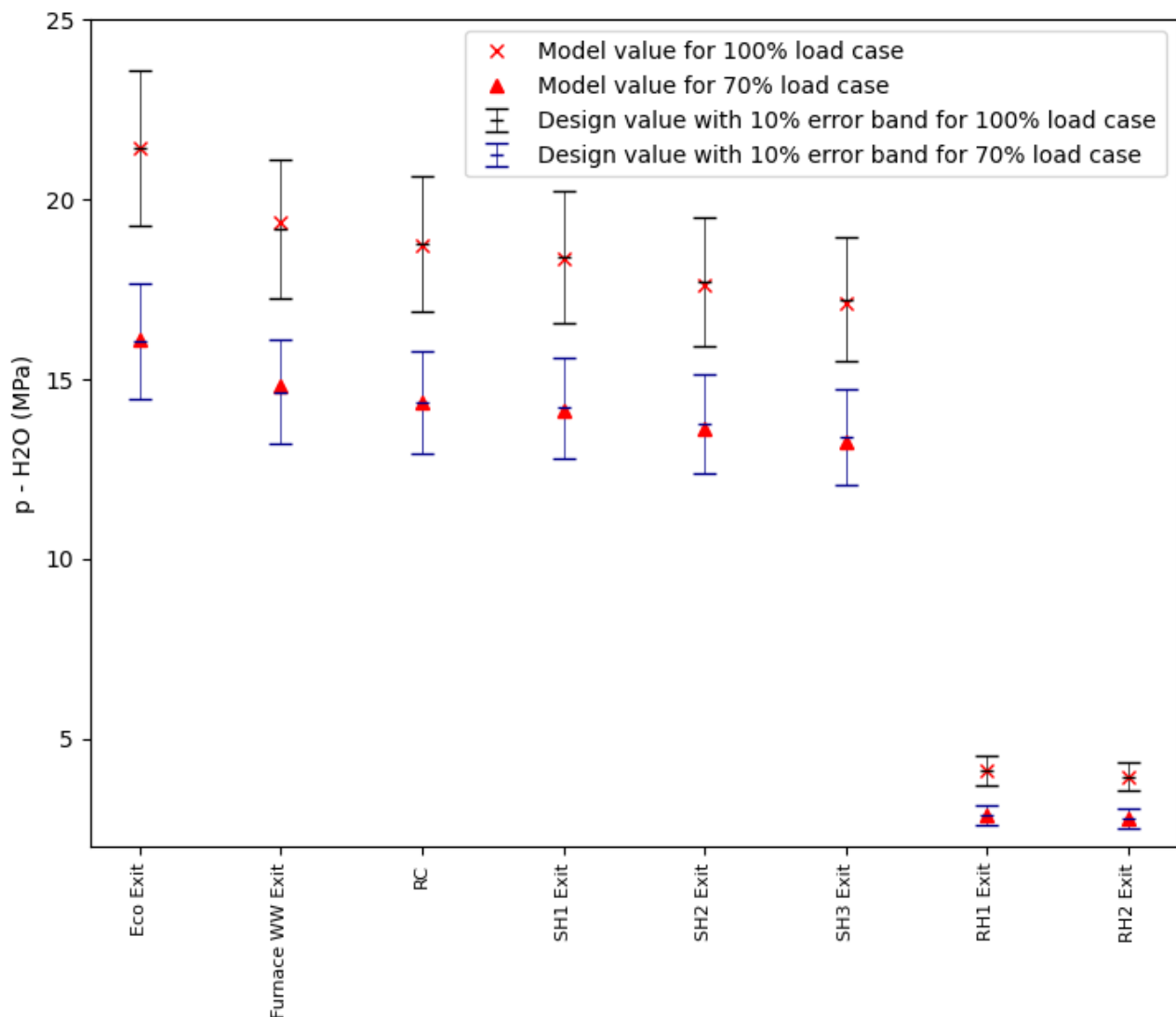


Figure 4.22 Water/steam pressure result

The flue gas pressure losses computed by the model also show sufficient accuracy, as shown in Figure 4.23 and Figure 4.24.

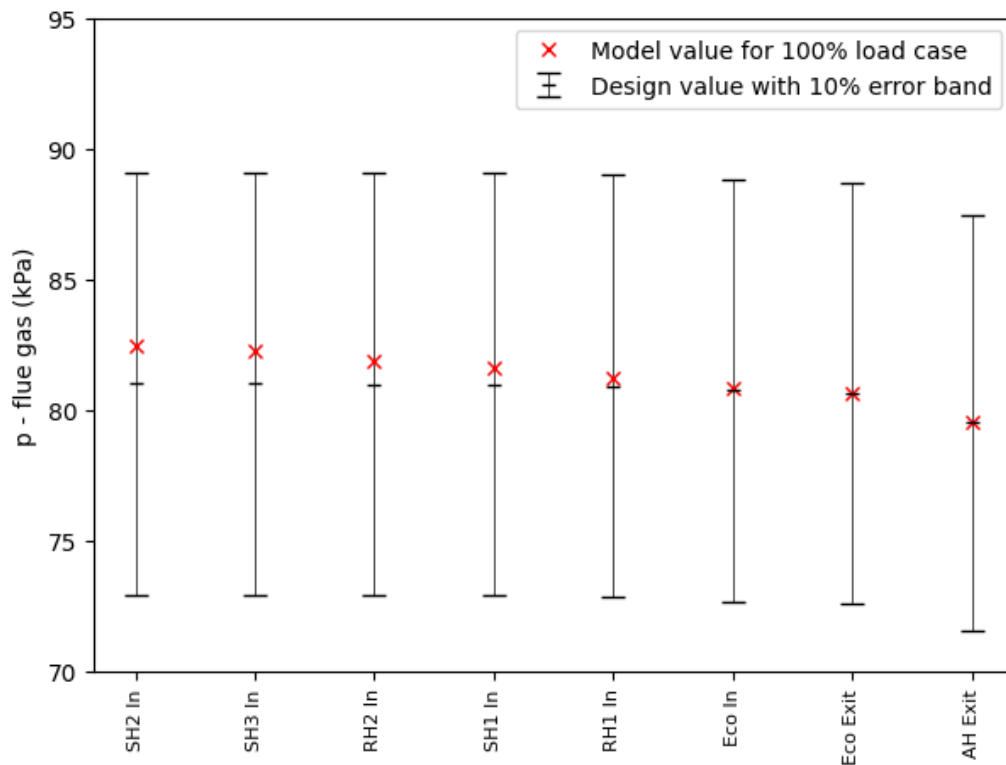


Figure 4.23 Flue gas pressure results at 100% MCR

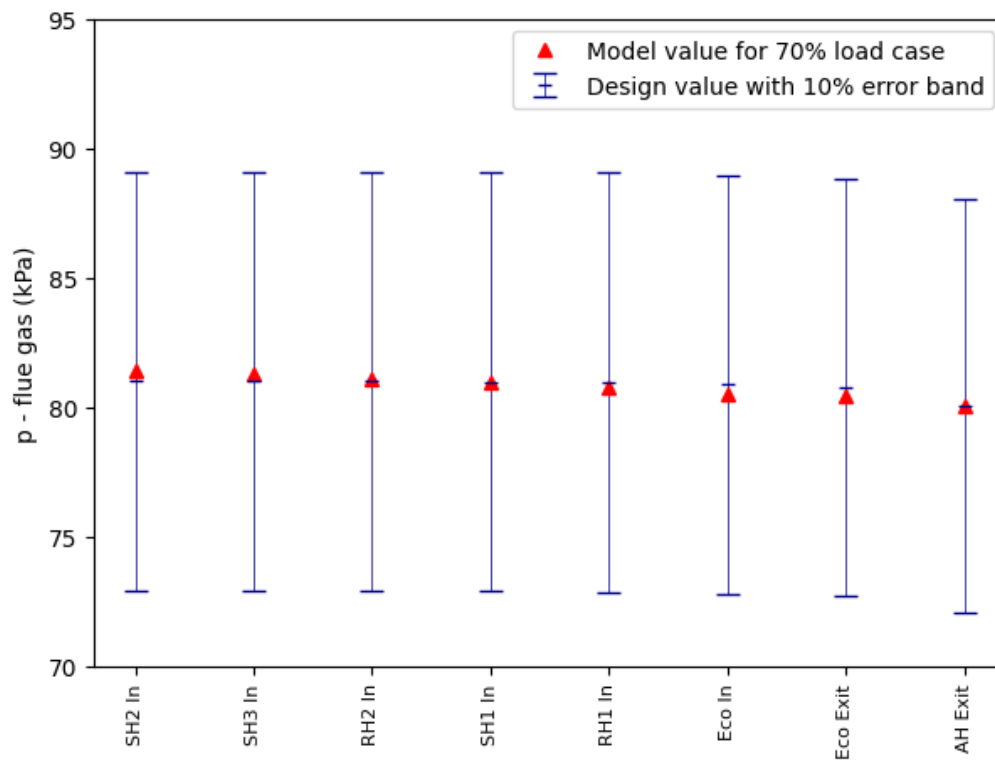


Figure 4.24 Flue gas pressure results at 70% MCR

These pressure results indicate that the methods used to model the pressure losses are valid and provide sensible values.

4.4.2 Validation using designer specification: temperature results

For the 100% MCR simulations, the model's water/steam temperatures are within a 10% error band compared to the designer specification data, as shown in Figure 4.25.

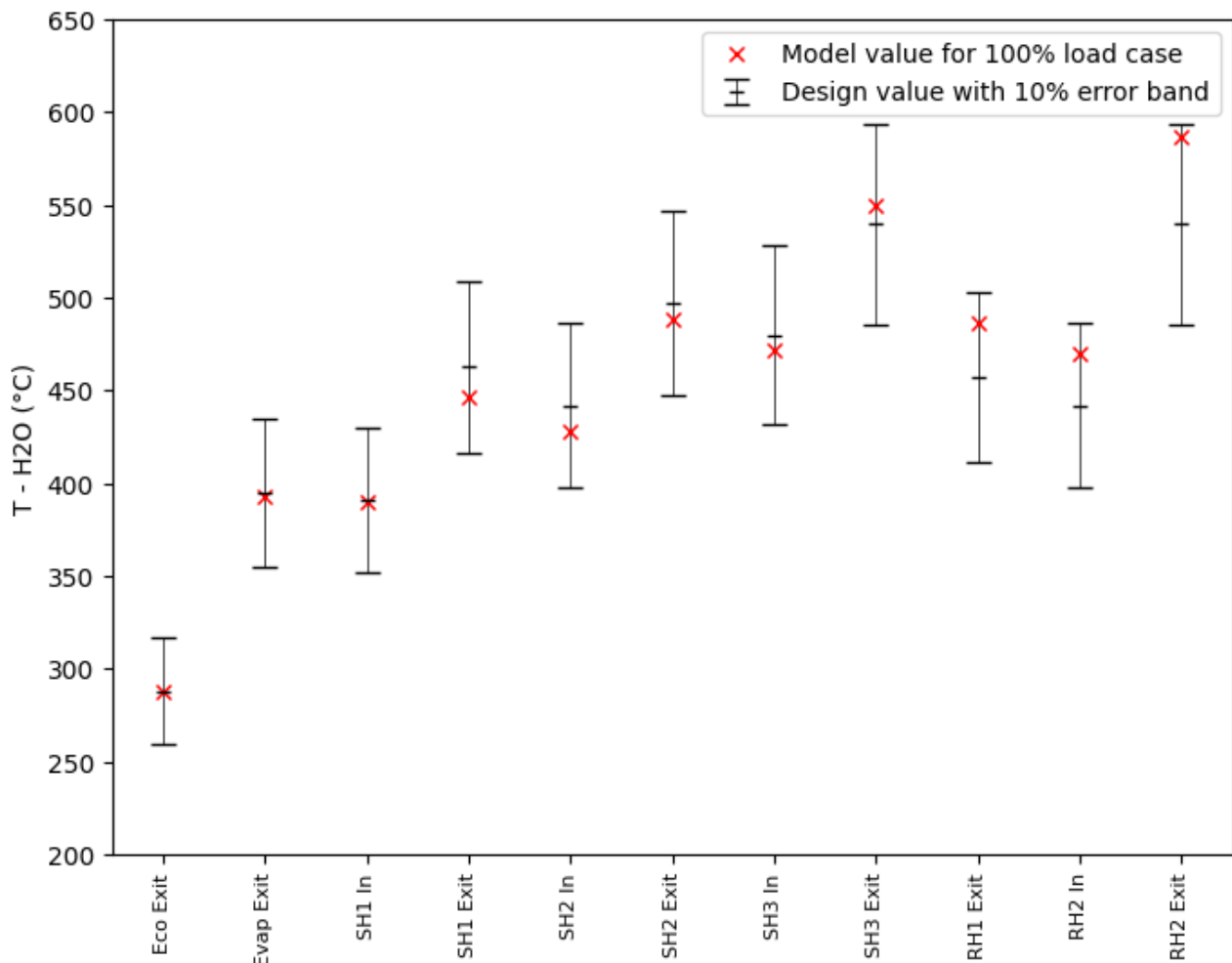


Figure 4.25 Water/steam temperature results at 100% MCR

Although the temperatures are all within the 10% error band, the errors in the steam temperatures are significant, as observed in Figure 4.25. The model underpredicts the main steam temperatures at SH inlets and exits but overpredicts the main steam (MS) exit temperature.

As for the re-heat steam (RS), the model overpredicts the steam temperatures at all RH exits. These RS temperature over-predictions are also found in the temperature results of the 70% MCR simulations shown in Figure 4.26.

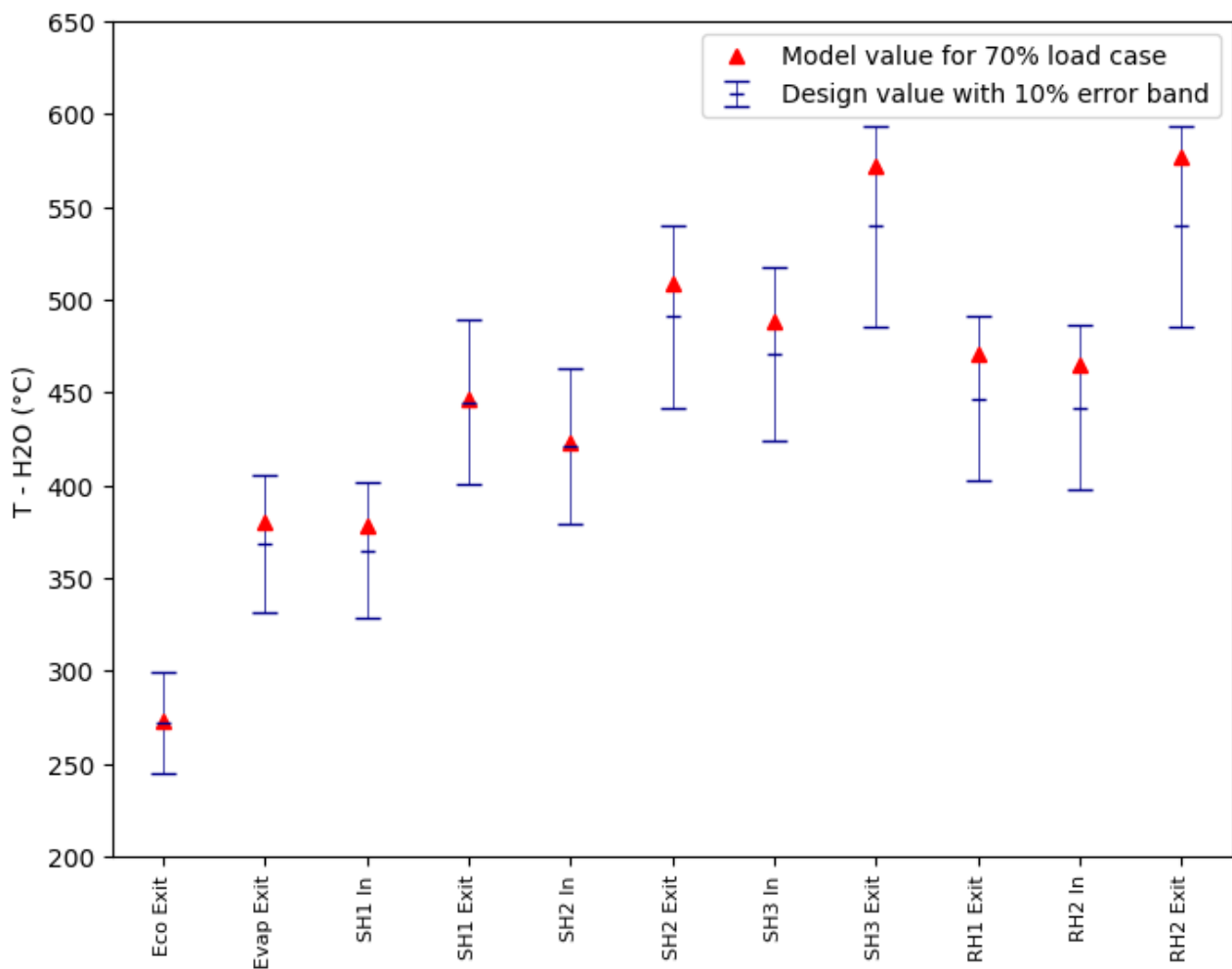


Figure 4.26 Water/steam temperature results at 70% MCR

However, in the case of 70% MCR, the model also overpredicts the MS temperatures at all the SH inlets and exits. This inconsistency in the model performance could suggest that the slagging and fouling resistance presented in Section 3.5.2 is general and not applicable to the case study boiler. Additionally, the heat transfer in the furnace computed by the model is significantly lower when compared to the design specifications, which results in a higher FET of the flue gas.

This difference in heat transfer could suggest that the design analysis might not consider the high-ash loading effects, limiting the amount of heat transfer in the boiler. This difference in flue gas FET affects the heat transfer of the downstream heat exchangers, contributing to the difference in the steam temperatures.

The high FET also cause the model to over predict the flue gas temperatures. However, the over predictions in the flue gas temperatures are all within an acceptable range, as shown in Figure 4.27 and Figure 4.28.

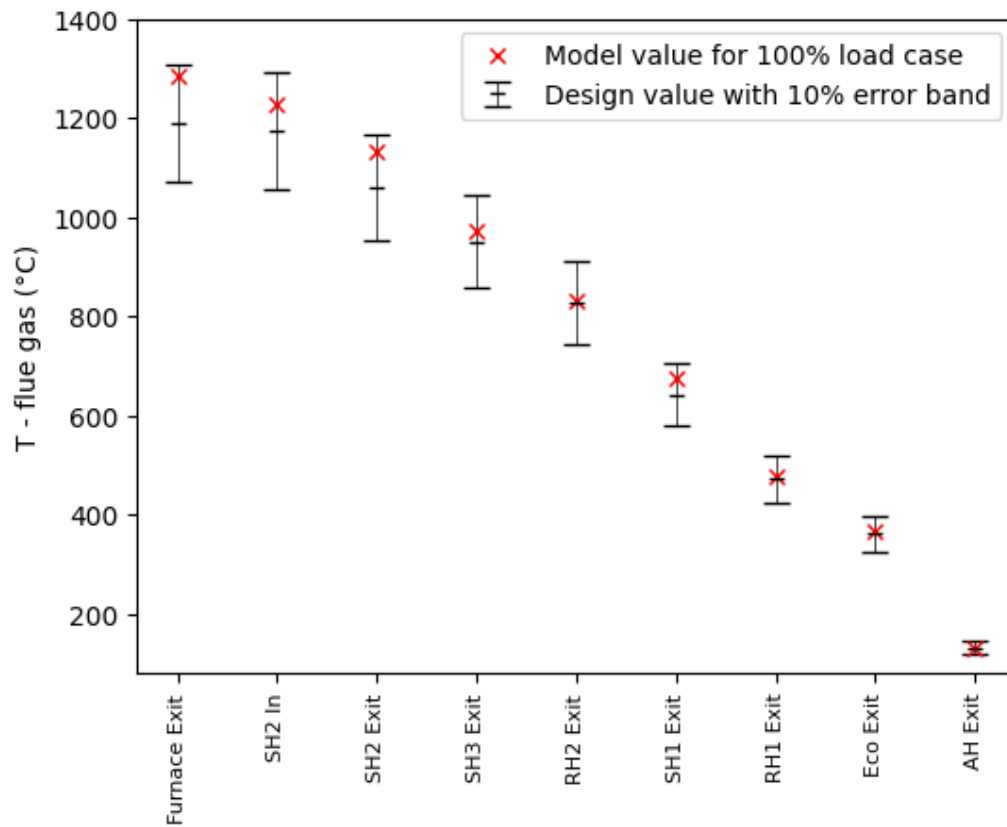


Figure 4.27 Flue gas temperature results at 100% MCR

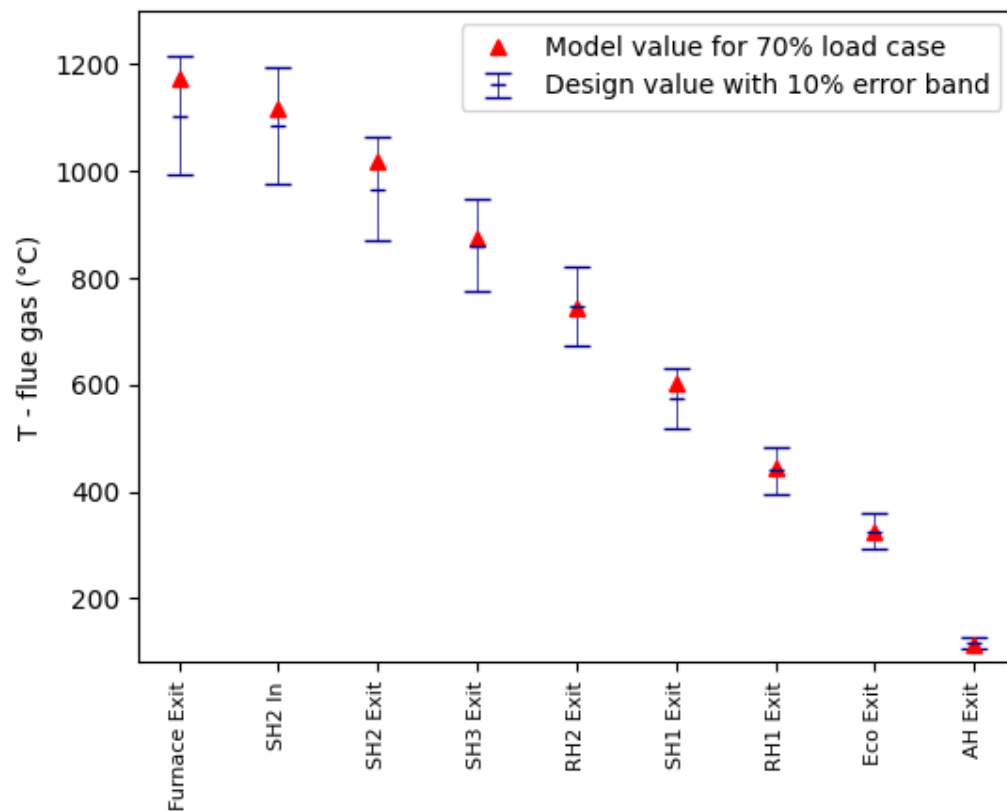


Figure 4.28 Flue gas temperature results at 70% MCR

4.4.3 Validation using designer specification: heat transfer results

The model's temperature over predictions also reflects in the heat transfer results. The total boiler heat load computed by the model is higher than the heat load from the designer specification as shown in Figure 4.29.

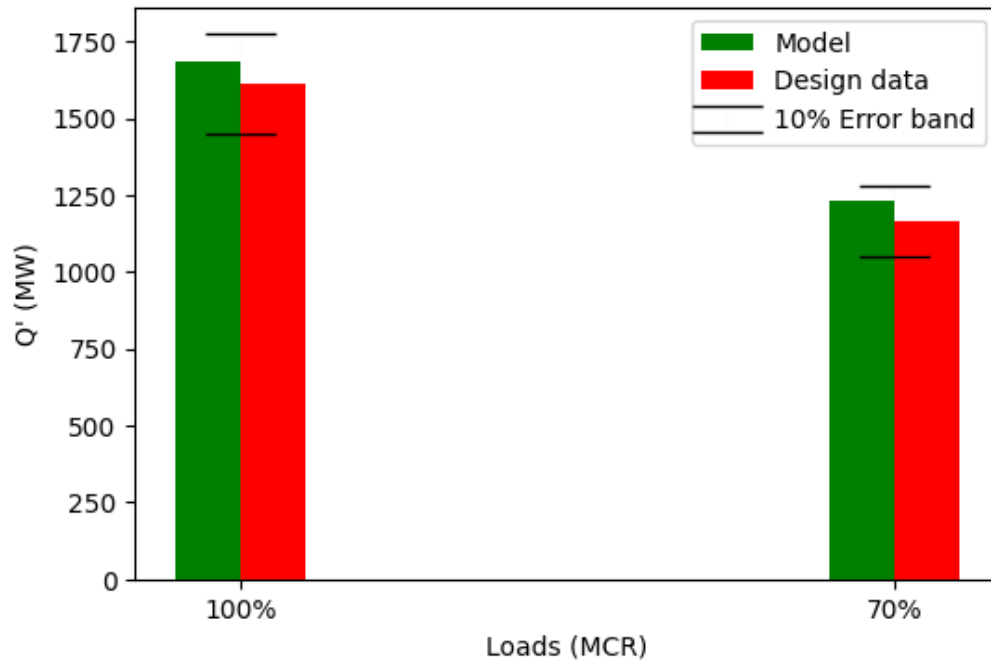


Figure 4.29 Total boiler heat transfer results

In the 100% MCR case, most of the heat load difference is shown to be absorbed by the re-heat steam (RS), as shown in Figure 4.30.

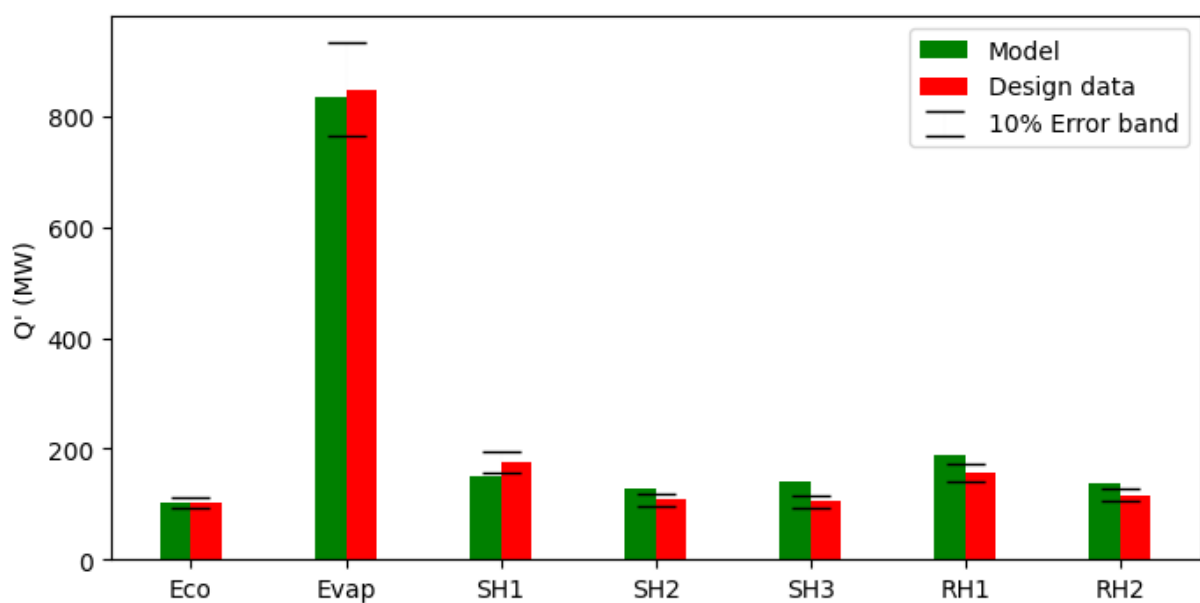


Figure 4.30 Heat exchanger heat transfer results at 100% MCR

As shown in Figure 4.30, the furnace heat transfer from the design specifications is significantly higher than the model result, which results in a lower flue gas FET, as discussed in Section 4.4.2.

In the 70% MCR load case, the difference in heat load is evenly distributed amongst the MS and the RS, as shown in Figure 4.31.

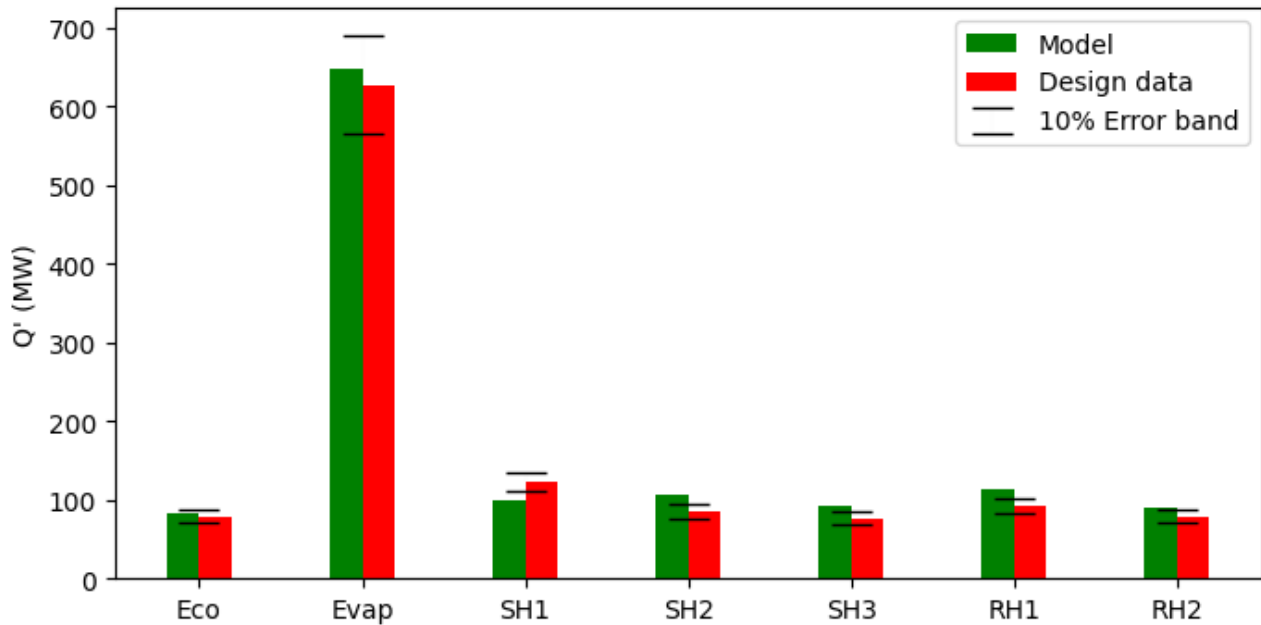


Figure 4.31 Heat exchanger heat transfer results at 70% MCR

Additionally, in both load cases, the heat distribution in the boiler computed by the model does not match the design specifications, since the model under-predicts the heat transfer in SH1.

The validation against the designer specification showed that the model does provide sensible results. However, the model still requires calibration to ensure the model can capture realistic ash deposition and heat distribution of the boiler.

5. Results and Discussion

As discussed in Section 4.4, the ash deposition (fouling or slagging) resistances suggested by the literature are not applicable for the case study boiler, which implies that the boiler model still requires calibration regarding these thermal resistances. The model is calibrated using the plant data at full-load operating conditions, and the details of the validation exercise are discussed in the subsequent sections. The calibrated model is then validated using the plant data at two other part-loads operating conditions. Lastly, the model is applied to simulate plant operations in the wet operating mode.

5.1 Plant measurements and data processing

The load cases as mentioned above refer to the steady-state plant operations at a certain MCR. However, CFPPs rarely operate at a steady state for an extended period since the plant energy output is required to meet the grid's demand. For that reason, if the generator load readings from the plant data have not varied by more than 1% MCR for at least one hour, the plant is considered to be operating at a quasi-steady-state, as shown in Figure 5.1.

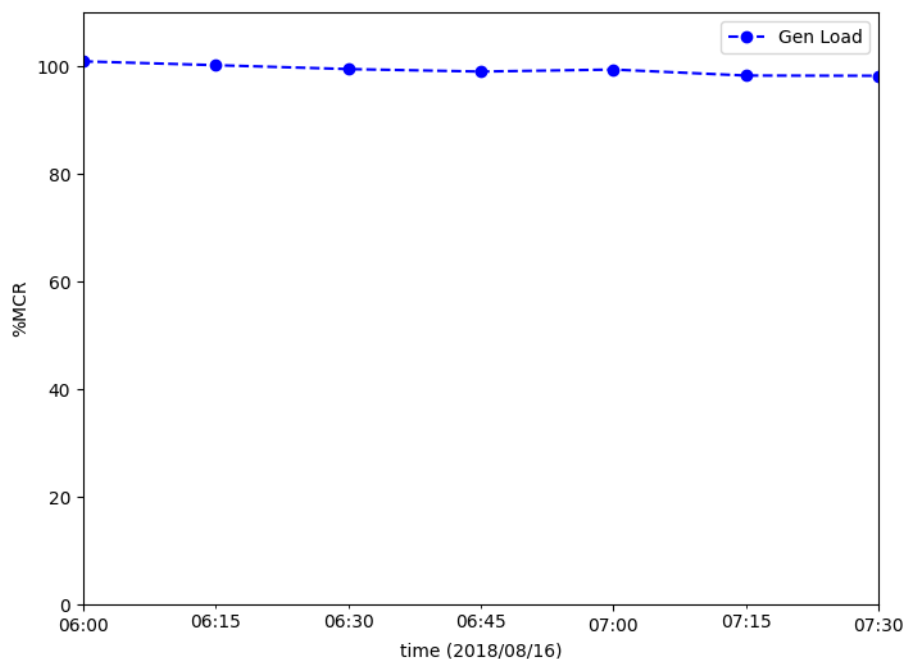


Figure 5.1 Generator load reading for the full load case

These steady-state criteria are applied to identify all the steady-state load cases used for the calibration and validation exercises, which have periods shown in Table 5.1.

Table 5.1 Steady-state load operation periods

Load (% MCR)	Starting time	End time
100	2018/08/16 06:00	2018/08/16 07:30
81	2018/09/25 01:00	2018/09/25 05:15
63	2018/10/18 23:00	2018/10/19 00:15
35	2018/10/10 00:00	2018/10/10 03:00

5.1.1 Measurement data processing

In the case study boiler, there are typically multiple measurements taken at each measuring point; therefore, the mean of these sensor measurements is used for the calibration and validation exercises. In this case, multiple measurements are used to validate each parameter; therefore, the accuracy of the model results are checked using the 95% confidence interval (CI). The 95% confidence level is the commonly used range, which can be computed using the following equation:

$$CI = \bar{x} \pm 1.96 \frac{\sigma}{\sqrt{n}} \quad (5-1)$$

The term *CI* in Equation (5-1) refers to the confidence interval, which is computed using the measurement mean ($\bar{x}$), sample size (n) and standard deviation (σ).

Additionally, the steam temperature at each heat exchanger terminal is measured at each attemperator's inlet and outlet, which branches the steam into four legs. However, the steam flow rate in each leg is not measured; therefore, these steam temperatures are determined using the mass and energy balance of each attemperator leg (see Appendix O).

One of the parameters needed to perform this mass and energy balance analysis is the steam pressure at each heat exchanger terminal, some of which are not measured. For that reason, the steam pressures at these unmeasured points are determined using linear interpolation.

Another parameter that can be used to examine the model's accuracy is the flue gas temperature across each heat exchanger. However, not all flue gas temperatures across each heat exchanger are measured; therefore, the validation is done mainly by comparing the model results to the processed water, steam, or mixture data.

5.1.2 Fuel and air inputs

The coal specifications are a set of parameters that are not measured on-site but determined monthly at sophisticated testing facilities. Ideally, the coal specifications used in the model should be from the same months as the periods in Table 5.1. However, the lab reports obtained from the site engineer only contain coal specifications from March 2020 to January 2021 (see Appendix O). For that reason, the coal specifications of the case study loads are approximated as the mean of all the given specifications and are shown in Table 5.2.

Table 5.2 Coal specifications used in the model (as-received basis)

Symbol	Parameter	Unit	Value
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	19484.5
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4420
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0413
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.1094
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0184
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0108
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.0923
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.2858

Another fuel-related input parameter, which is not directly extracted from the plant data, is the mass flow rate of the infeed coal since the measurements of this parameter are unclear in the plant data. Instead, the infeed coal mass flow rate is determined using the mass and energy balance analysis of the entire boiler [34] together with the boiler heat load obtained from the plant data.

Additionally, the infeed air mass flow rate is an input parameter that is not directly measured at the case study boiler. For that reason, the infeed air mass flow rate for each load case is determined using the combustion analysis presented in Section 3.2.1 and the flue gas oxygen content measured at the boiler exit, which can be directly extracted from the plant data.

The methods mentioned above for determining the infeed coal and air mass flow rates are proven to be valid since the total boiler heat load reported from the model, after calibration at 100% MCR, show sufficient accuracy when compared to the boiler heat load at all the different loads obtained from the plant data. The results are summarised in Figure 5.2.

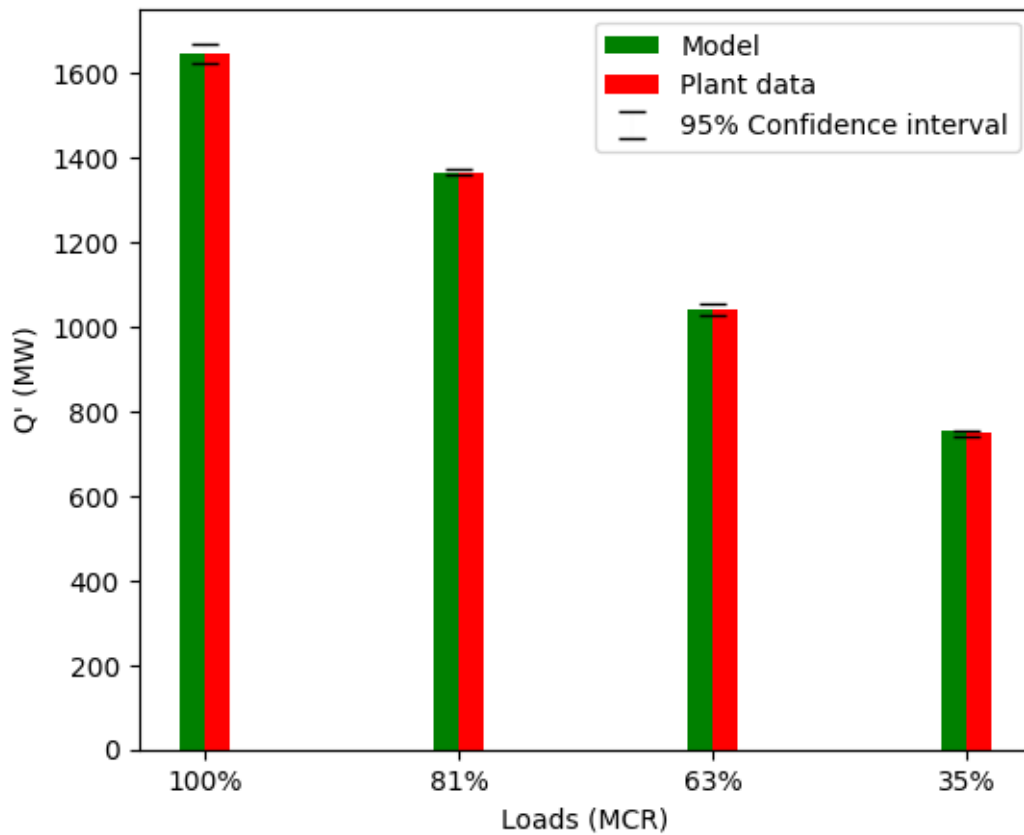


Figure 5.2 Total boiler heat load

5.2 Boiler simulation at full load conditions

As mentioned before, the processed plant data at 100% MCR is used to calibrate the boiler model (see Appendix Q for model inputs and boundary conditions). The main parameters for the calibration exercise are the water/steam pressure loss coefficients and the ash deposition resistances. The water side pressure loss coefficients are calibrated since the pressure measurements differ from the designer specifications, and the ash deposition resistances are calibrated due to the reasons discussed in Section 4.4. The calibration method and the model results are presented in the subsequent sections.

5.2.1 Model calibration for ash deposition resistances

The ash deposition resistances of the model are calibrated by increasing the ash deposition thicknesses to match the respective heat exchanger loads obtained from the plant data. The calibration process starts at the platen superheater (SH2, directly downstream of the furnace) and moves up towards the heat exchanger at the boiler exit (Eco).

After the first round of adjustments, the inlet steam temperatures of some heat exchangers will change due to the change in the heat transfer of the upstream heat exchanger. This change in inlet temperature will influence the heat transfer in the current heat exchanger, and therefore the ash deposition thickness needs to be re-adjusted to match the heat load in the current heat exchanger. An iterative process is therefore required.

Additionally, the ash deposition resistances of the duct wall and sling tubes in each heat exchanger are also modelled. Furthermore, the values of these ash deposition resistances are the same as their respective heat exchanger tubes.

As shown in Table 5.3, the final calibrated values differ from the values suggested in the literature.

Table 5.3 Fouling/slugging resistance of the calibrated model

Fouling/slugging resistance ($\frac{m^2 K}{W}$)	SH1	SH2	SH3	RH1	RH2	Eco
Model	0.0008	0.0113	0.0070	0.0086	0.0067	0.0195
Literature [33], [43]	0.0043	0.0091	0.0043	0.0043	0.0043	0.0089

The ash deposition resistance of RH1 is significantly higher than SH1, which is a surprising finding since RH1 is downstream of SH1 in the gas flow direction. Upon enquiry the site engineer suggested that this high ash deposition at RH1 is mainly due to the fact that the soot blowers for RH1 are not always available, leading to the heat exchangers having unclean surfaces. Additionally, soot blowing may be biased to allow some fouling or slagging to build up to control temperatures and reduce the spray water requirements.

Another possibility suggested by the site engineer is that the model captured the realistic ash build-up phenomena in RH1. RH1 has fewer soot blowers than originally designed, with a row omitted in the construction phase. For that reason, the soot blowers at RH1 do not remove all the accumulated ash from the previous low load operations. Practical experience has shown that if the remaining ash

gets disturbed, a large amount of ash will fall out of the heat exchanger, causing chain reactions leading to flameout and large pressure disturbance in the gas flow.

5.2.2 Water/steam temperature and pressure results

The calibrated model's water/steam pressure results in the MS stream show sufficient accuracy for the full-load operation simulation, as shown in Figure 5.3. The pressure results are within the confidence intervals, with the maximum error at Eco exit being 0.25%.

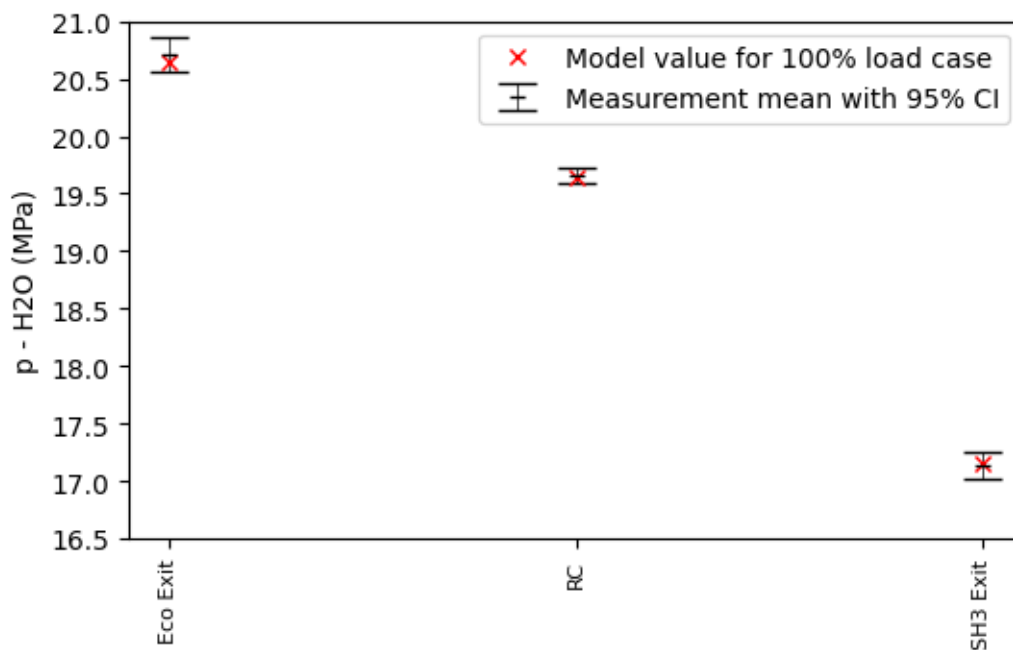


Figure 5.3 MS stream water/steam pressure results at 100% MCR

The steam pressure at the RH2 exit also show sufficient accuracy, as shown in Table 5.4. The model's water/steam temperature results show sufficient accuracy since most temperatures are within the 95% confidence intervals, as shown in Figure 5.4.

Table 5.4 RH2 exit steam pressure result at 100% MCR

Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
3.727	3.748	0.021	0.57%	±0.04

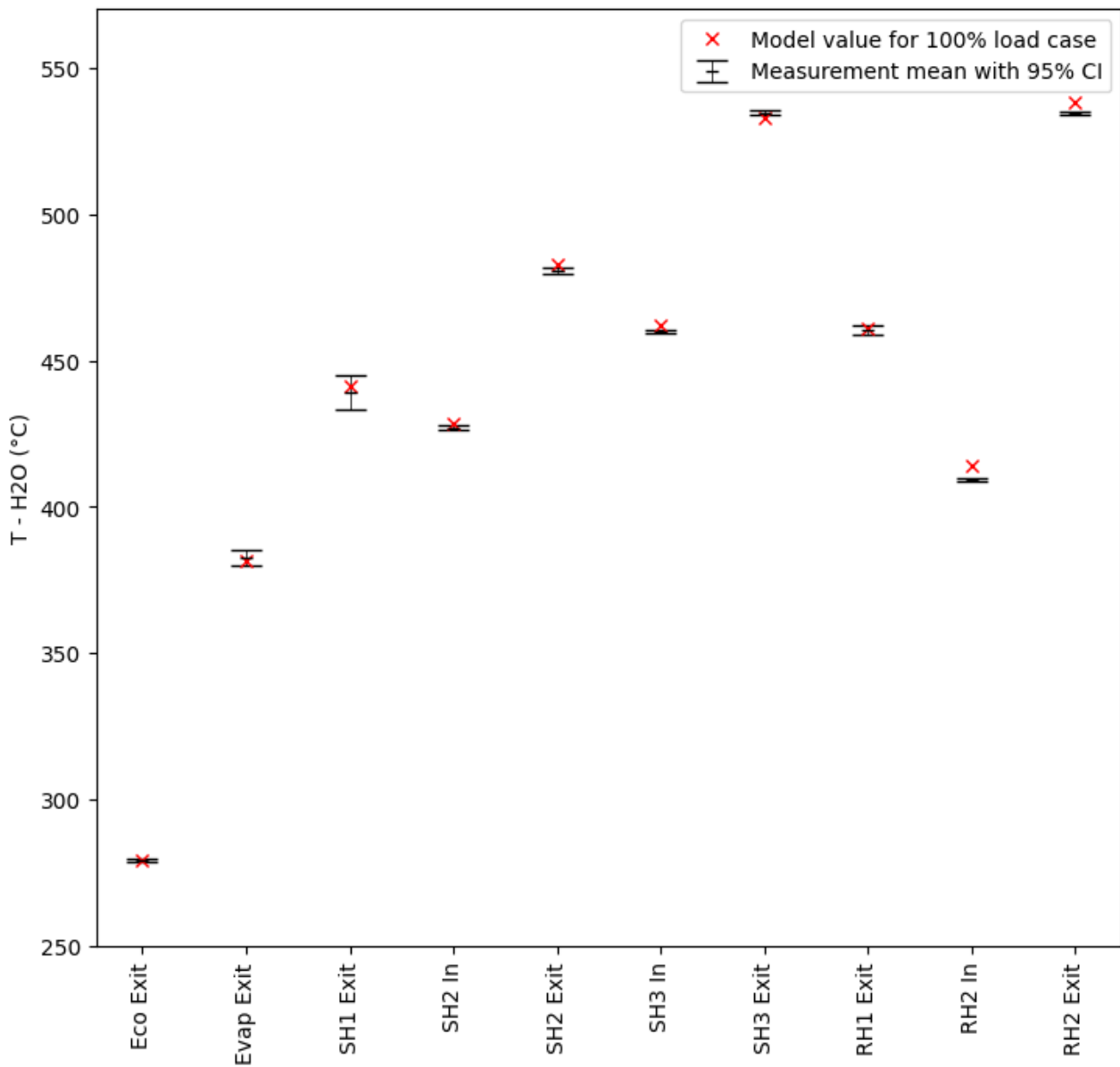


Figure 5.4 Water/steam temperature results at 100% MCR

As shown in Figure 5.4, the steam temperature results at RH2 inlet and exit are slightly out of the confidence intervals. However, the magnitude of these errors presented in Table 5.5 suggests that these errors are relatively insignificant.

Table 5.5 RH2 inlet and exit steam temperature results at 100% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{RH2\ in}$	413.90	409.25	-4.65	-1.14%	±0.48
$T_{RH2\ exit}$	538.24	534.38	-3.86	-0.72%	±0.62

5.2.3 Heat transfer results

The heat transfer results are also shown to be sufficiently accurate since most of the results are within the confidence intervals, as shown in Figure 5.5. The results of SH1 and RH2 are slightly out of the confidence intervals. However, Table 5.6 shows that these results' relative errors are all less than 5.2%.

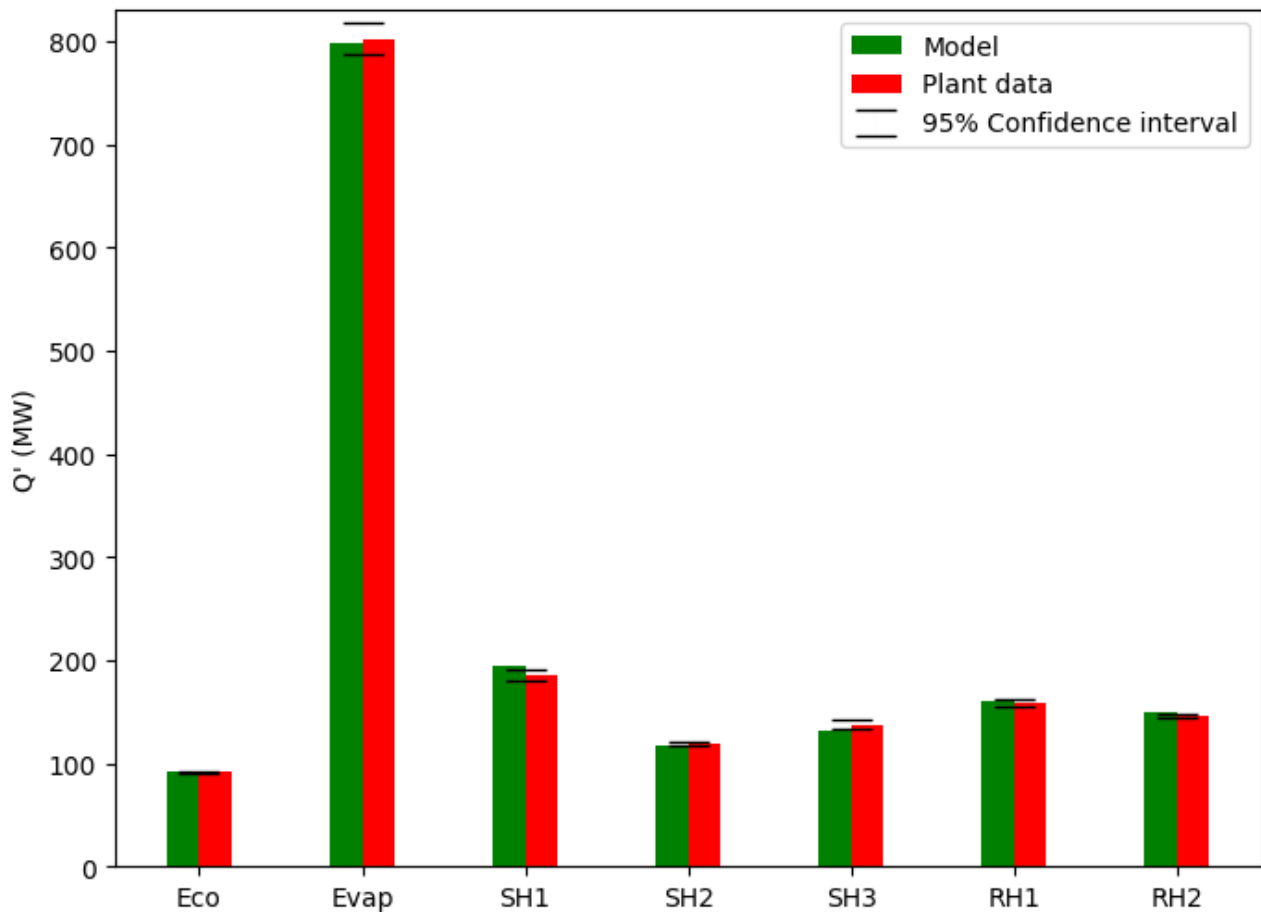


Figure 5.5 Total heat transfer in each heat exchangers at 100% MCR

Table 5.6 Heat transfer results at 100% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	91.98	91.88	-0.09	-0.10%	± 0.79
$\dot{Q}_{Evap}$	798.09	802.51	4.42	0.55%	± 15.06
$\dot{Q}_{SH1}$	195.33	185.70	-9.63	-5.19%	± 6.08
$\dot{Q}_{SH2}$	116.97	119.16	2.19	1.84%	± 1.21
$\dot{Q}_{SH3}$	131.47	137.63	6.16	4.48%	± 4.04
$\dot{Q}_{RH1}$	159.54	158.84	-0.70	-0.44%	± 3.50
$\dot{Q}_{RH2}$	149.85	147.53	-2.31	-1.57%	± 2.15
$\dot{Q}_{Total}$	1643.26	1643.25	-0.01	0.00%	± 23.08

Additionally, the errors in the heat transfer results of these heat exchangers do not have a significant impact on the steam temperatures since the temperature results showed sufficient accuracy and the average relative error of the results is 2.0%.

The Q'T diagram of the model results summarises the boiler's temperature and heat distribution, as shown in Figure 5.6.

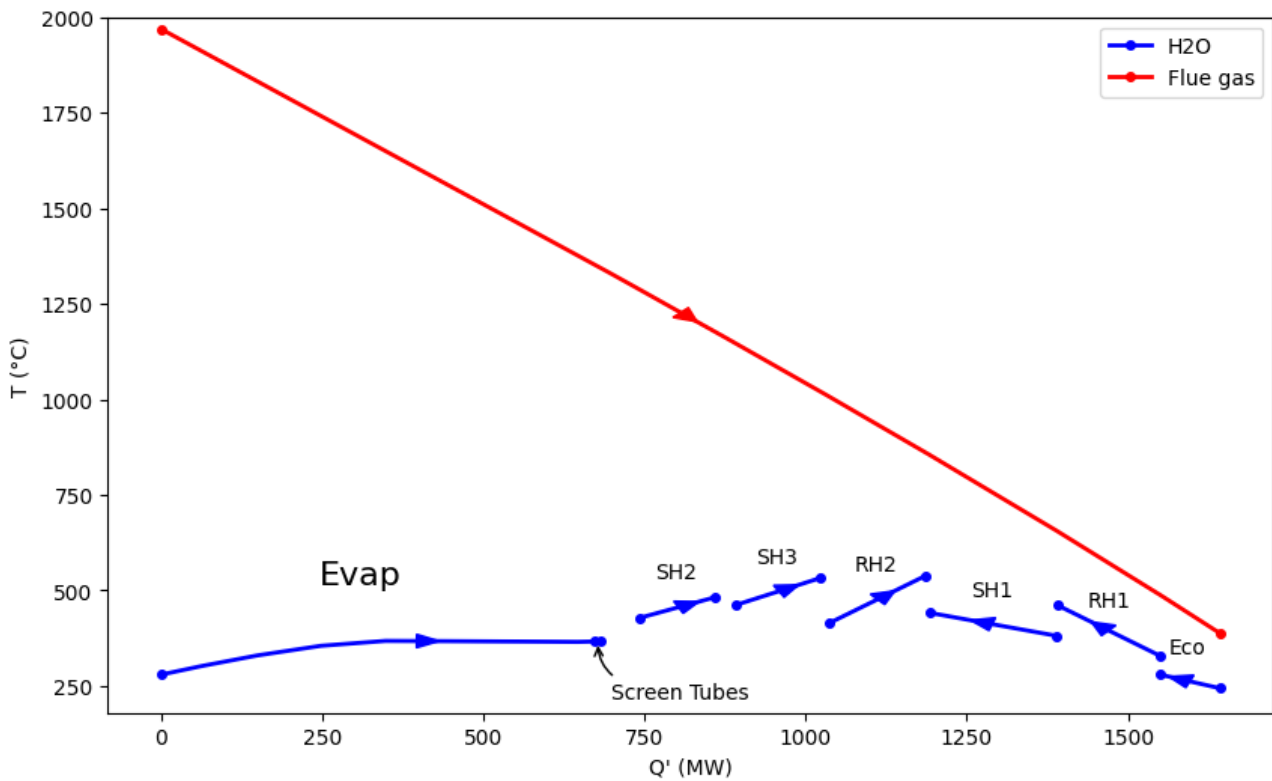


Figure 5.6 Q'T diagram of model results at 100% MCR

The arrows in Figure 5.6 indicate flow directions, which show that the evaporator, SH2, SH3 and RH2 are heat exchangers with parallel flow configurations. On the other hand, SH1, RH1 and Eco are heat exchangers with counterflow configurations. Additionally, the plateau in the evaporator temperature indicates the evaporation phenomenon, which suggests that the boiler is operating under the critical pressure of water.

5.2.4 Metal temperature results

The calibrated model's metal temperature results are also acceptable, as shown in Figure 5.7. Additionally, the model computes the maximum metal temperature of each heat exchange tube pass, which can help identify the tubes at risk during operation. In this case, the results in Table 5.7 suggest that heat exchanger tubes are sufficiently cooled during full-load operation since the maximum metal temperatures are all below the designed maximum operating metal temperatures.

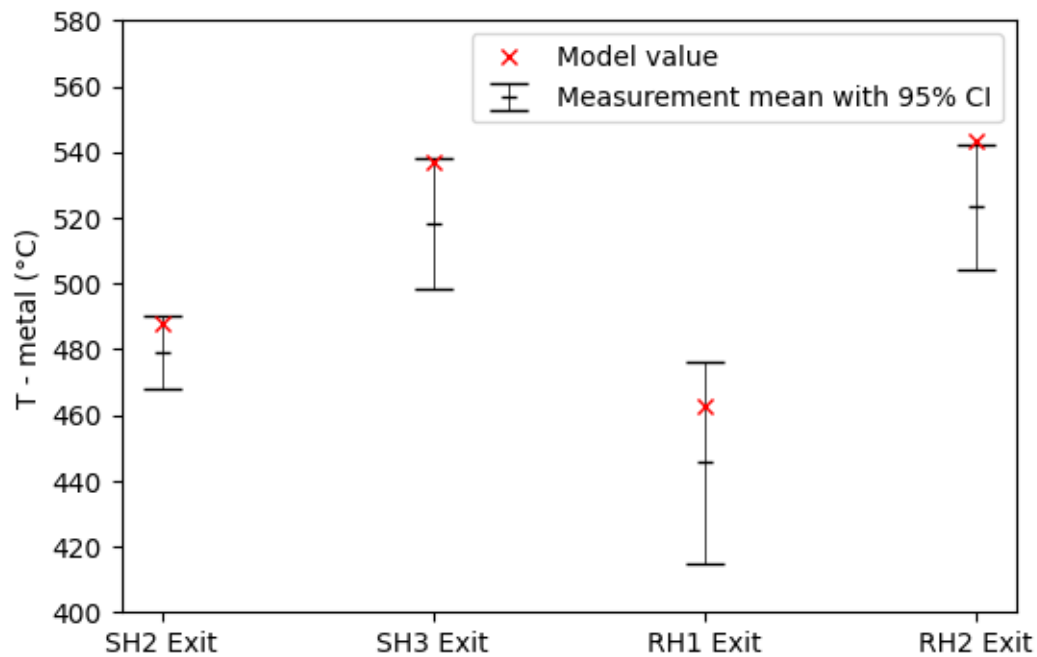


Figure 5.7 Metal temperature results at 100% MCR

Table 5.7 Maximum metal temperatures of each heat exchanger at 100% MCR

Heat exchanger	Model (°C)	Design (°C)	Pass
Eco	281.55	327.00	4
SH1	446.79	474.00	6
SH2	489.81	494.00	2
SH3	538.35	558.00	2
RH1	462.88	512.00	4
RH2	544.01	566.00	2

Additionally, the temperature results of the furnace membrane wall models in Figure 5.8 show that the furnace water wall tubes are operating in the safe range since the maximum metal temperature is well below the designed maximum.

The figure also shows the temperature profile of the furnace wall, which follows the distribution discussed in Section 3.3.4 and the maximum wall temperature is found at the effective mean flame height.

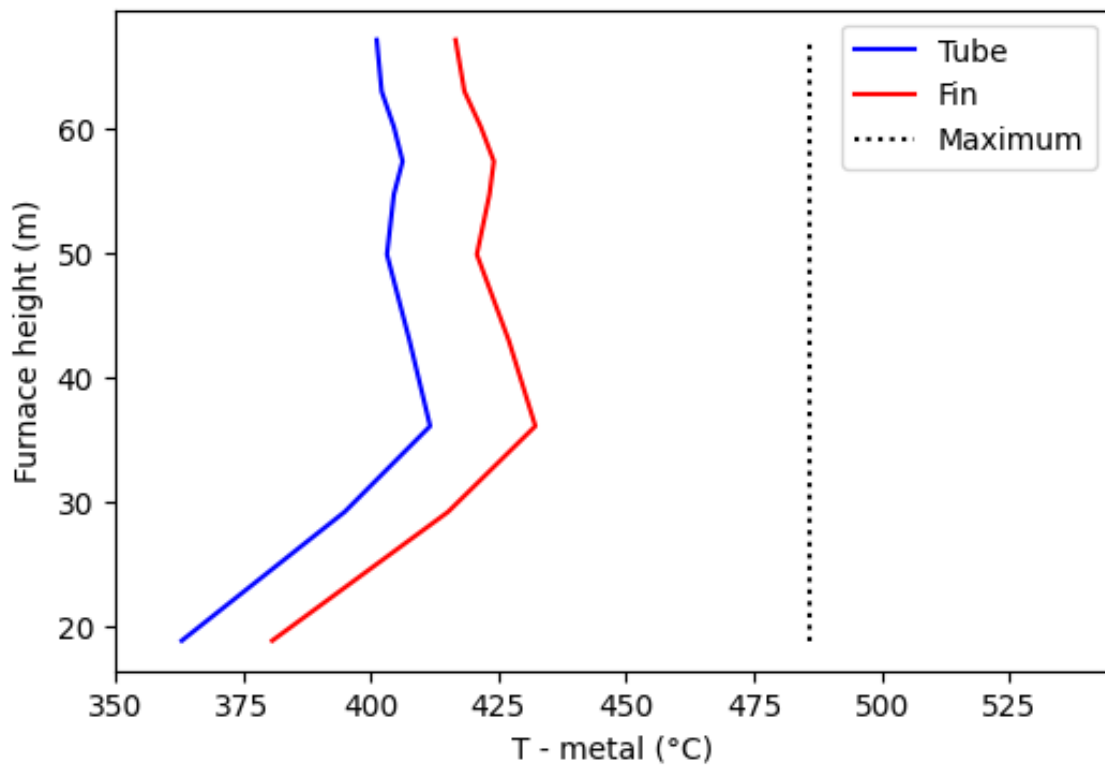


Figure 5.8 Membrane wall tube and fin metal temperature results at 100% MCR

5.3 Simulation of part-load conditions during dry mode operation

After calibration, the boiler model is validated using plant data at two part-loads namely 81% and 63% MCR (see Appendix Q for model inputs and boundary conditions). The boiler operates in dry mode for both load cases, and the results of the validation exercise are presented in the subsequent sections.

5.3.1 Water/steam temperature and pressure results

At 81% MCR load case, the model's MS stream pressure results at ring collector (RC) and SH3 exit are shown to be slightly out of the confidence intervals in Figure 5.9. However, the results in Table 5.8 suggests that these errors are still in the acceptable range.

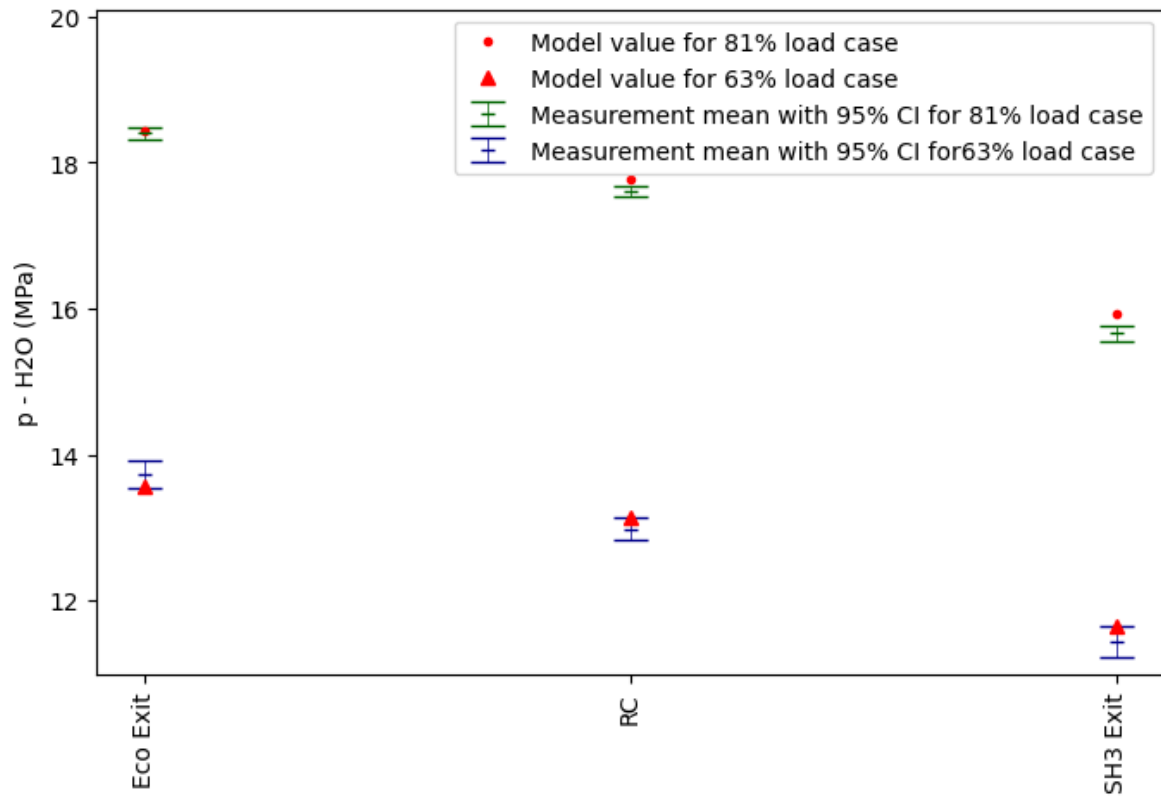


Figure 5.9 MS stream water/steam pressure results at 81% and 63% MCR

Table 5.8 RC and SH3 steam pressure at 81% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
p_{RC}	17.78	17.61	-0.17	-0.95%	± 0.06
$p_{SH3\ exit}$	15.93	15.66	-0.27	-1.71%	± 0.11

The steam pressure at the RH2 exit for the 81% MCR load is also slightly out of the confidence interval, as shown in Table 5.9. However, the relative error is less than 1%.

Table 5.9 RH2 exit steam pressure result at 81% and 63% MCR

Load case	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
81% MCR	3.00	2.98	-0.02	-0.59%	± 0.01
63% MCR	2.32	2.35	0.03	1.13%	± 0.05

The model's water/steam temperature results show sufficient accuracy for the 81% MCR load case, as shown in Figure 5.10. The steam temperature results at SH2 and RH2 exits are slightly out of the confidence intervals. Table 5.10 shows that the relative errors of both temperatures are less than 1%.

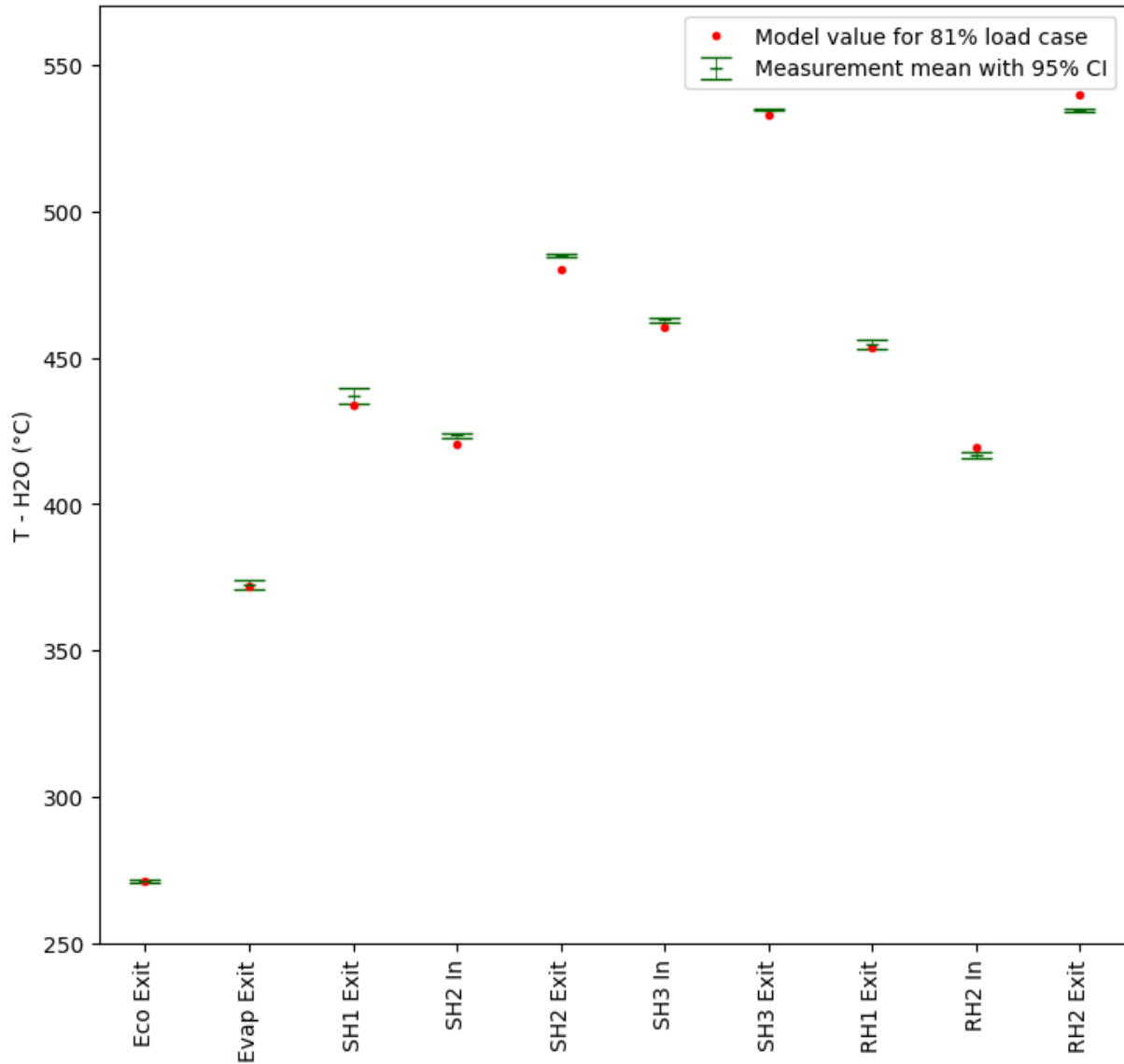


Figure 5.10 Water/steam temperature results at 81% MCR

Table 5.10 SH2 and RH2 exit steam temperature results at 81% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{SH2\ exit}$	480.17	484.97	4.80	0.99%	±0.50
$T_{RH2\ exit}$	539.67	534.37	-5.30	-0.99%	±0.51

The temperature results also showed sufficient accuracy for the 63% MCR load case, as shown in Figure 5.11. In this case, the steam temperature results at SH2 exit and RH2 inlet are out of the confidence intervals. However, Table 5.11 shows that the errors of these temperature results are insignificant since the relative errors are all less than 2%.

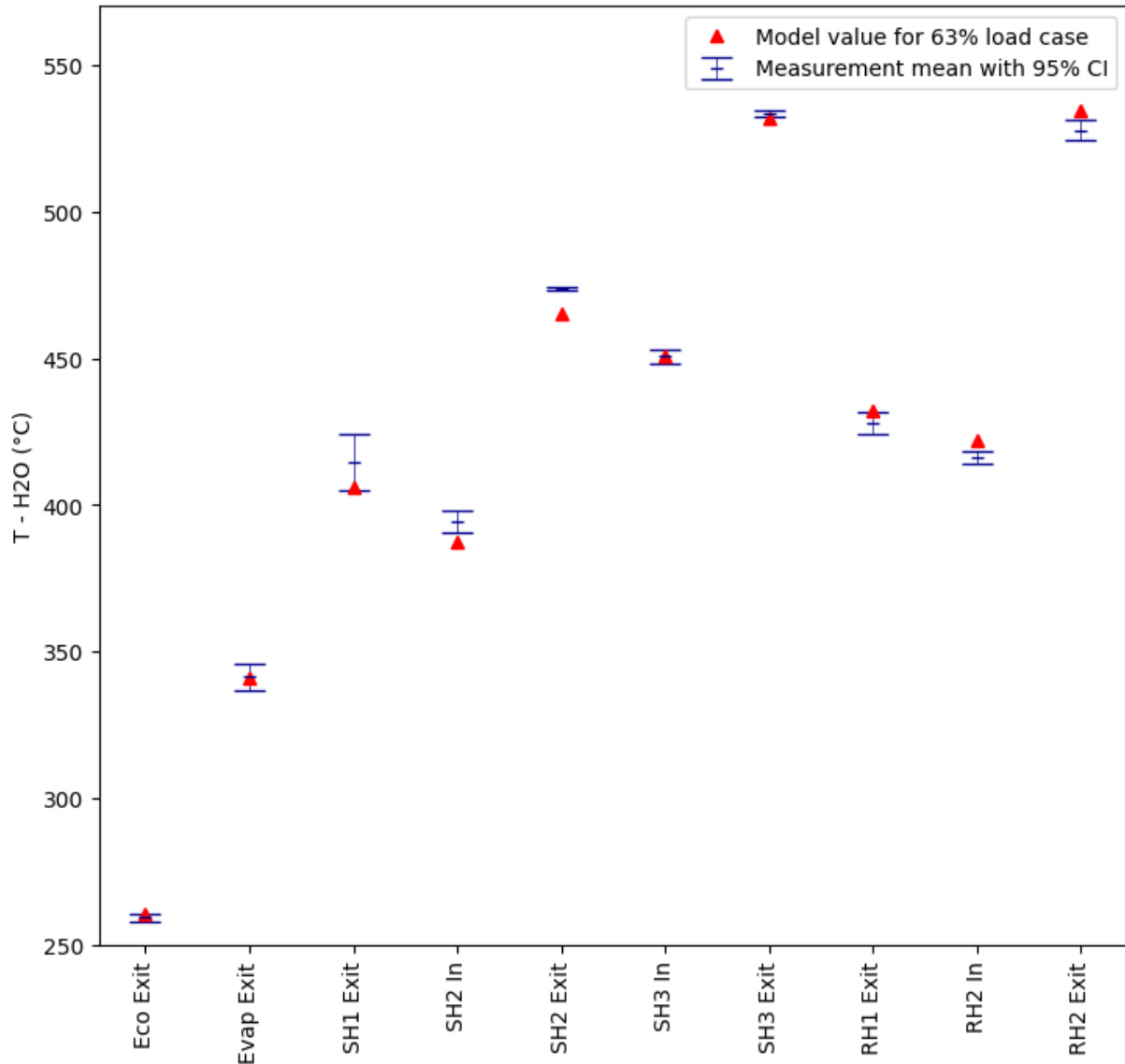


Figure 5.11 Water/steam temperature results at 63% MCR

Table 5.11 SH2 exit and RH2 inlet steam temperature results at 63% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{SH2\ exit}$	465.43	473.83	8.39	1.77%	±0.46
$T_{RH2\ in}$	422.14	416.36	-5.78	-1.39%	±2.18

5.3.2 Heat transfer results

As observed from the temperature results in Table 5.9, the model slightly overpredicts the steam temperature at RH2 exit at 81% MCR load. The result suggests that the model overpredicts the performance of RH2, which is evident from the heat transfer results shown in Figure 5.12. The model overpredicts the performance of SH3. This overprediction is expected since the model underpredicts the steam temperature at SH2 exit. However, Table 5.12 shows that errors in these heat transfers are still less than 5%.

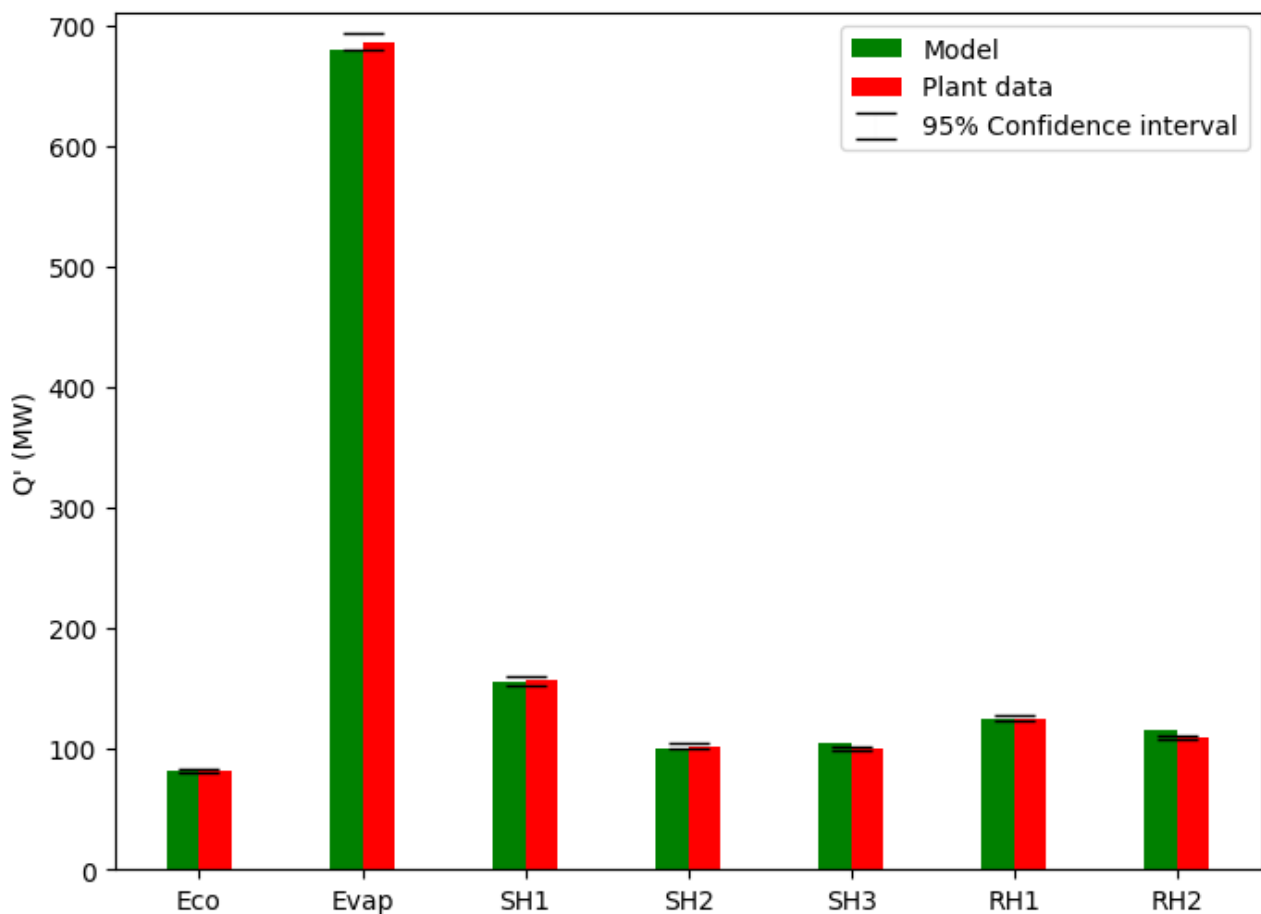


Figure 5.12 Total heat transfer in each heat exchangers at 81% MCR

Table 5.12 Heat transfer results at 81% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	82.12	81.46	-0.66	-0.81%	± 1.21
$\dot{Q}_{Evap}$	679.79	686.57	6.78	0.99%	± 6.40
$\dot{Q}_{SH1}$	155.22	156.54	1.32	0.84%	± 3.24
$\dot{Q}_{SH2}$	100.90	102.60	1.70	1.66%	± 1.60
$\dot{Q}_{SH3}$	105.03	100.11	-4.93	-4.92%	± 1.40
$\dot{Q}_{RH1}$	124.31	125.62	1.30	1.04%	± 1.60
$\dot{Q}_{RH2}$	116.13	111.78	-4.34	-3.88%	± 1.34
$\dot{Q}_{Total}$	1363.44	1363.91	0.47	0.03%	± 6.77

In this case, the results still show sufficient accuracy since the average relative error is only 2.0%.

The overprediction in SH3 performance is more significant in the 63% MCR load, as shown in Figure 5.13. In this case, the average relative error of the heat transfer results is 3.6% and Table 5.13 shows that the relative error of SH3 result is close to 10%.

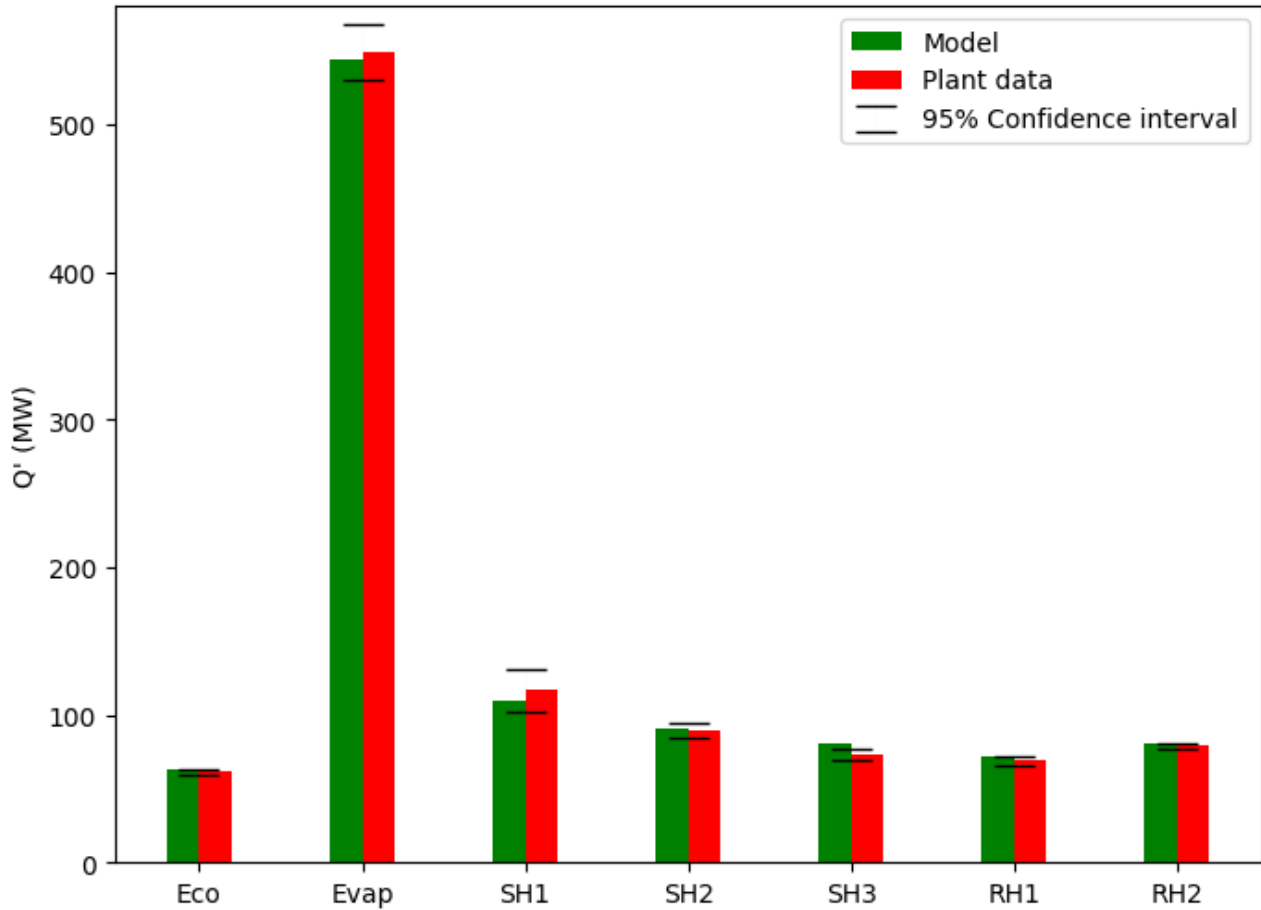


Figure 5.13 Total heat transfer in each heat exchanger at 63% MCR

Table 5.13 Heat transfer results at 63% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	63.24	61.51	-1.73	-2.81%	± 2.19
$\dot{Q}_{Evap}$	543.82	549.00	5.18	0.94%	± 18.58
$\dot{Q}_{SH1}$	109.61	116.76	7.15	6.12%	± 14.69
$\dot{Q}_{SH2}$	90.28	89.31	-0.97	-1.09%	± 5.21
$\dot{Q}_{SH3}$	80.55	73.45	-7.10	-9.66%	± 3.78
$\dot{Q}_{RH1}$	72.22	69.19	-3.03	-4.39%	± 3.00
$\dot{Q}_{RH2}$	80.72	81.13	0.40	0.50%	± 2.16
$\dot{Q}_{Total}$	1040.34	1040.35	0.01	0.00%	± 15.46

The results in Figure 5.12 and Figure 5.13 suggest that the calibrated model overpredicts the performance of SH3 at the part-load simulations. This finding suggests that there could be an additional ash deposition build-up in SH3 for the part-load operations. Since at part-load, the products/flue gas flow is slower due to decreased infeed air and fuel mass flow rate. The slower flue gas flow allows more time for ash to accumulate on the heat exchanger surfaces. Additionally, the ash could accumulate more at SH3 since the pitch of SH3 is a lot tighter than its downstream heat exchanger in the flue gas flow direction (see Appendix S for SH3 geometry). However, the errors in the heat transfer results of SH3 for the part-load simulations are still within an acceptable range since the temperature results are shown to be sufficiently accurate.

Another observation from the temperature and heat transfer results is that the available heat for each heat exchanger decreases with decreasing loads, as shown in Figure 5.14 and Figure 5.15. As shown in Figure 5.14 and Figure 5.15, the temperature differences between the flue gas and water/steam decrease as the load decreases. This decrease in temperature differences implies less thermal potential to drive heat transfer in each heat exchanger.

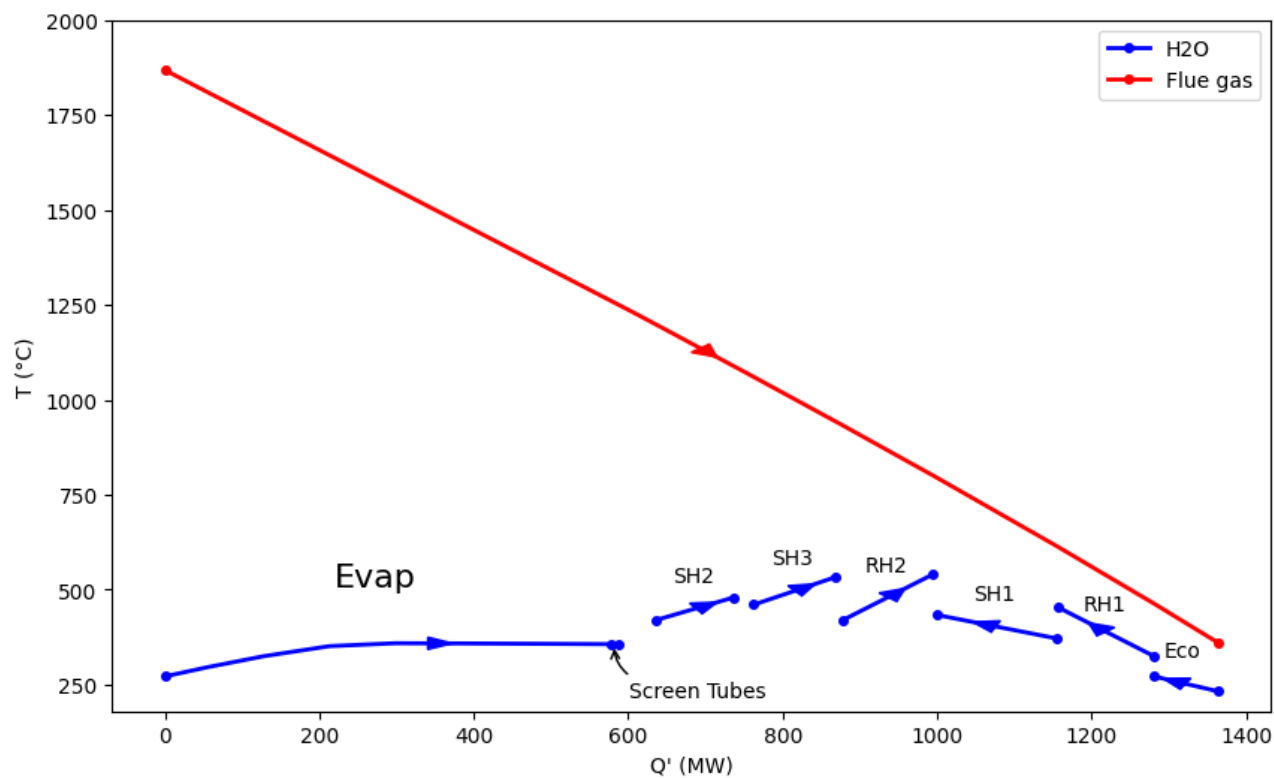


Figure 5.14 Q'T diagram of model results at 81% MCR

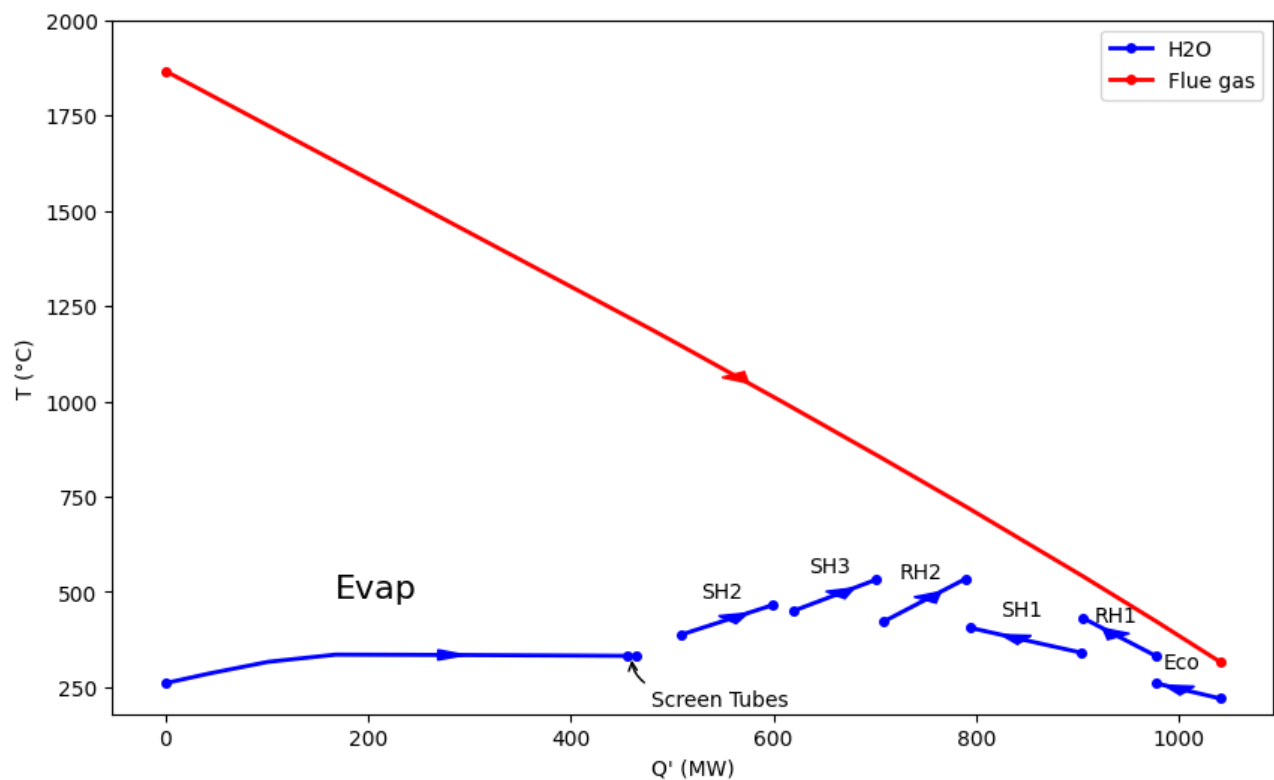


Figure 5.15 Q'T diagram of model results at 63% MCR

5.3.3 Metal temperature results

For simulation of both load cases, the metal temperatures in Figure 5.16 and Figure 5.17 indicate that the model results are sufficiently accurate. The maximum metal temperature results show that the heat exchanger tubes are sufficiently cooled during both part-load operations, as shown in Table 5.14.

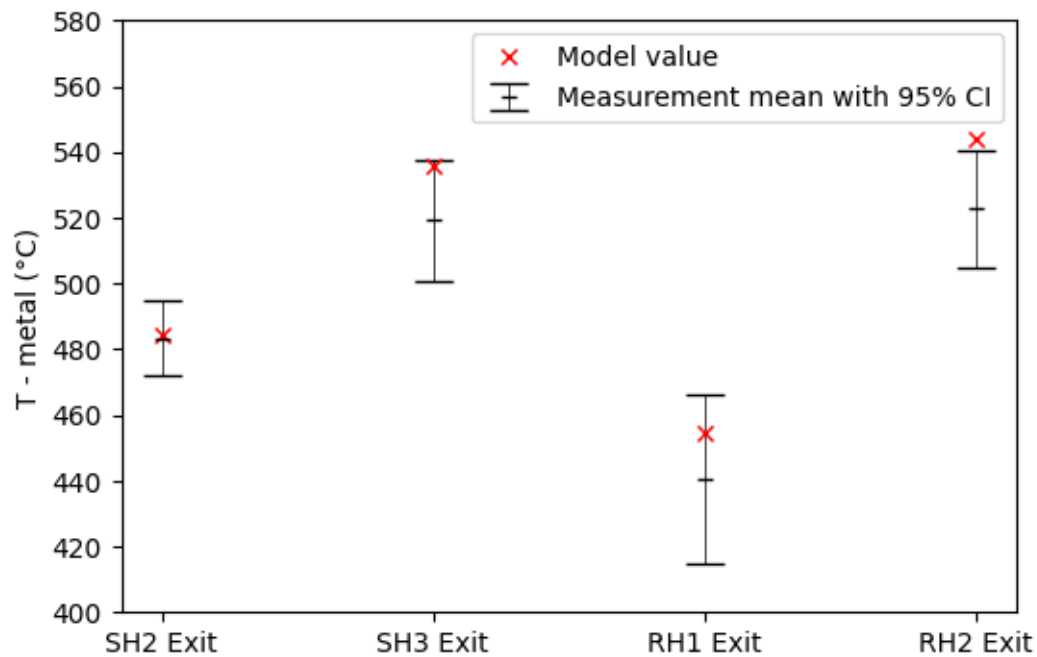


Figure 5.16 Metal temperature results at 81% MCR

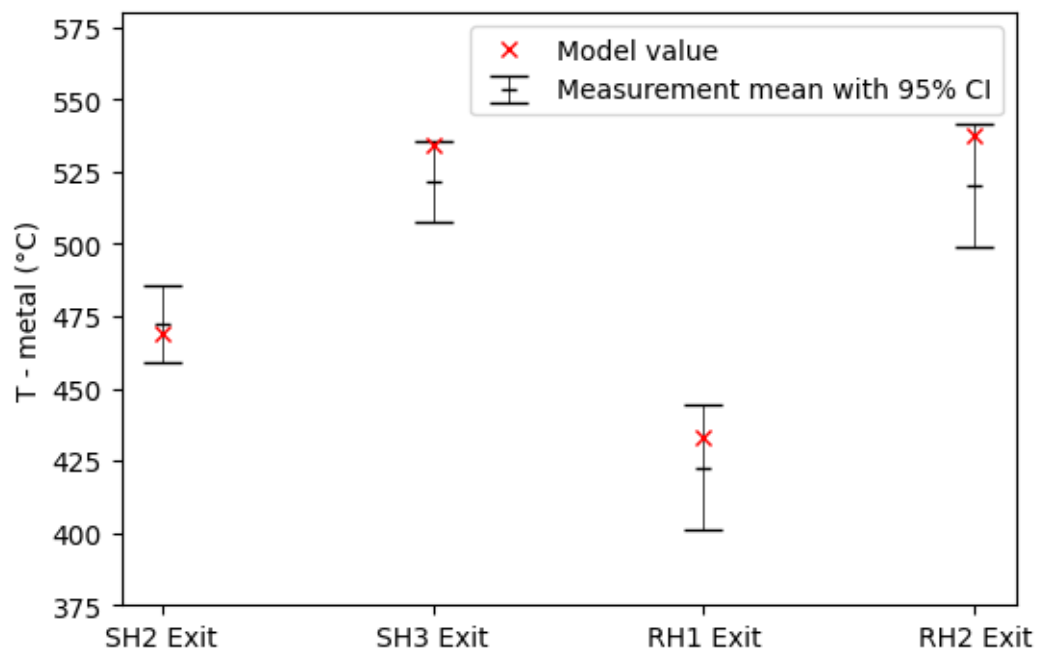


Figure 5.17 Metal temperature results at 63% MCR

Table 5.14 Maximum metal temperature of each heat exchanger at 81% and 63% MCR

Heat exchanger	81% MCR			63% MCR		
	Model (°C)	Design (°C)	Pass	Model (°C)	Design (°C)	Pass
Eco	273.48	327.00	4	262.06	327.00	4
SH1	438.10	474.00	6	409.54	474.00	6
SH2	486.11	494.00	2	470.54	494.00	2
SH3	536.91	558.00	2	534.91	558.00	2
RH1	454.96	512.00	4	433.28	512.00	4
RH2	544.33	566.00	2	537.68	566.00	2

In this case, the temperature results of the furnace membrane wall models in Figures 5.11 and figure 5.12 show that the metal temperatures of membranes decrease as the load drops. This temperature decrease is because the operating pressure decreases with load, which results in a lower saturation temperature. For that reason, the membrane wall is cooled by a mixture or steam with a lower temperature, thus resulting in cooler metal temperatures.

Additionally, the figures also show that the maximum wall temperature for each part-load operation is at different heights different due to the flame biases (see Appendix Q for flame biases).

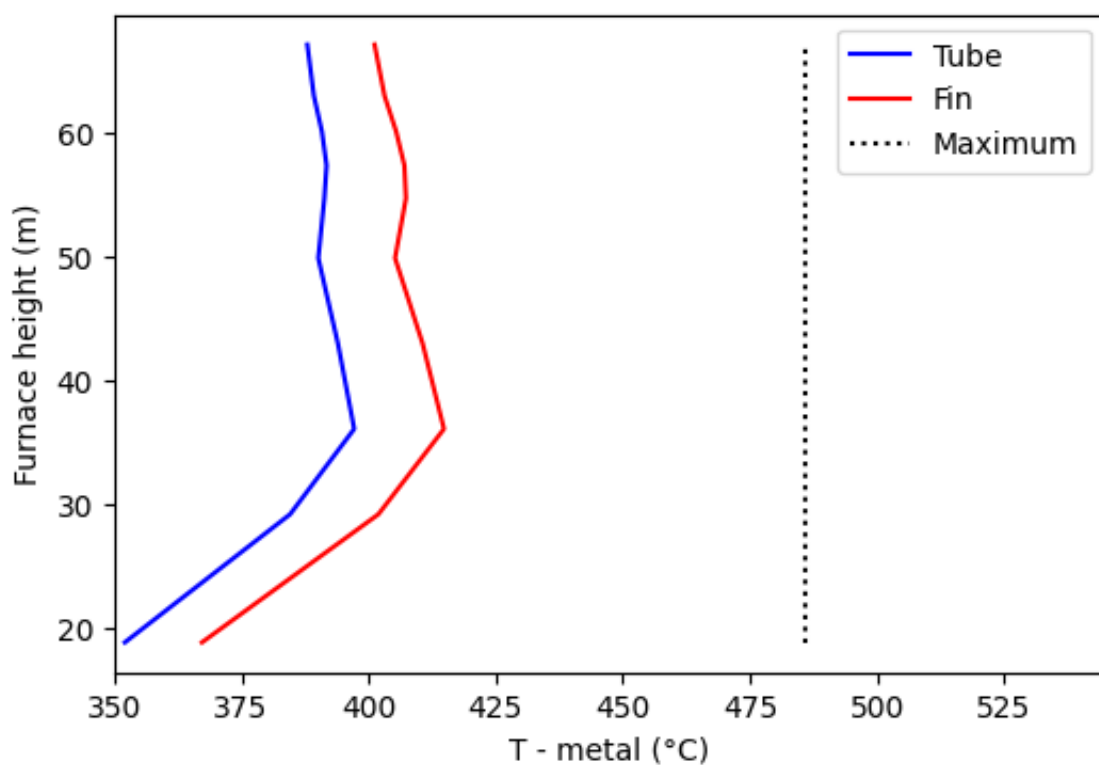


Figure 5.18 Membrane wall tube and fin metal temperature results at 81% MCR

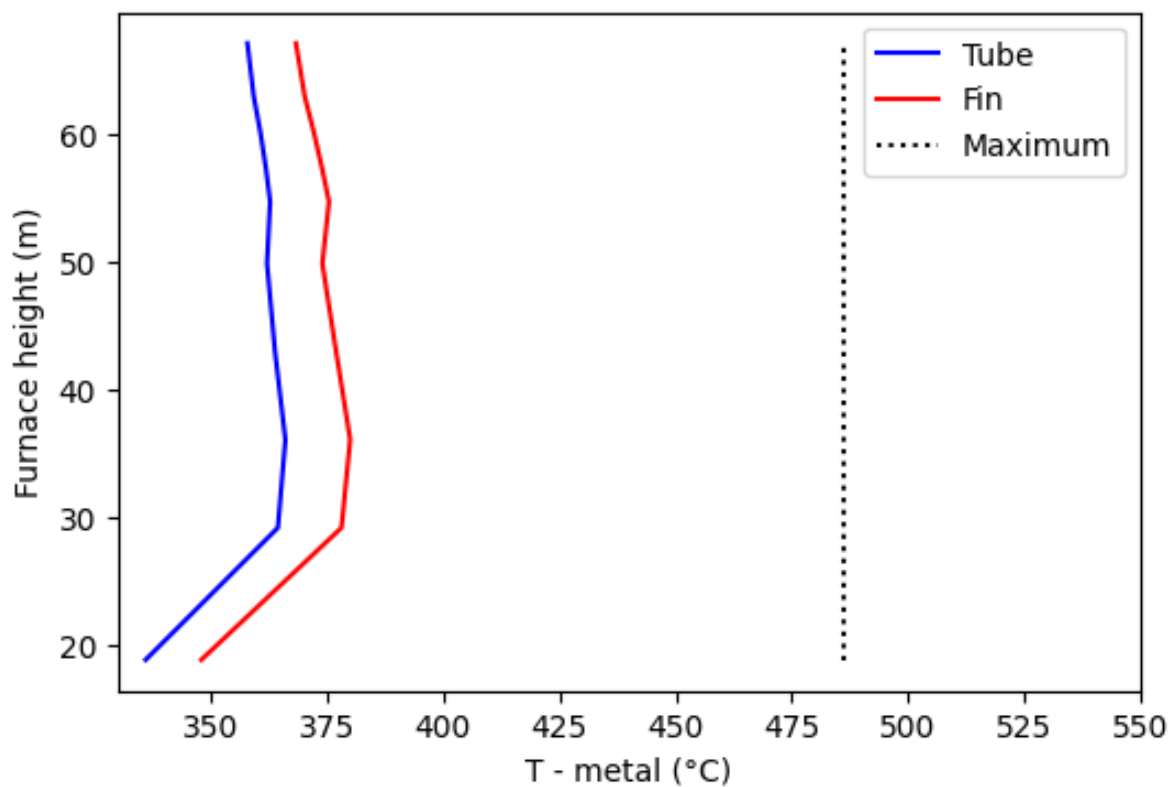


Figure 5.19 Membrane wall tube and fin metal temperature results at 63% MCR

5.4 Simulation of low load operation during wet mode operation

After validating the calibrated model using the data of two dry-mode operating loads, the model is applied to simulate boiler operations under wet mode at 35% MCR. As mentioned in Chapter 3, a once-through boiler typically operates in the wet mode for loads below 40% MCR (see Appendix Q for model inputs and boundary conditions). The results of the wet-mode operation simulation are presented in the subsequent sections.

5.4.1 Water/steam temperature and pressure results

For the wet-mode simulation, the model's water/steam pressure results in the MS stream are shown in Figure 5.20. Here, all the simulated pressures are outside of the confidence intervals. However, Table 5.15 suggests that these results are still within an acceptable range since the relative errors are all less than 3%. The steam pressure at the RH2 exit is also out of the confidence interval. However, the relative error of this pressure result is less than 5%, as shown in Table 5.16.

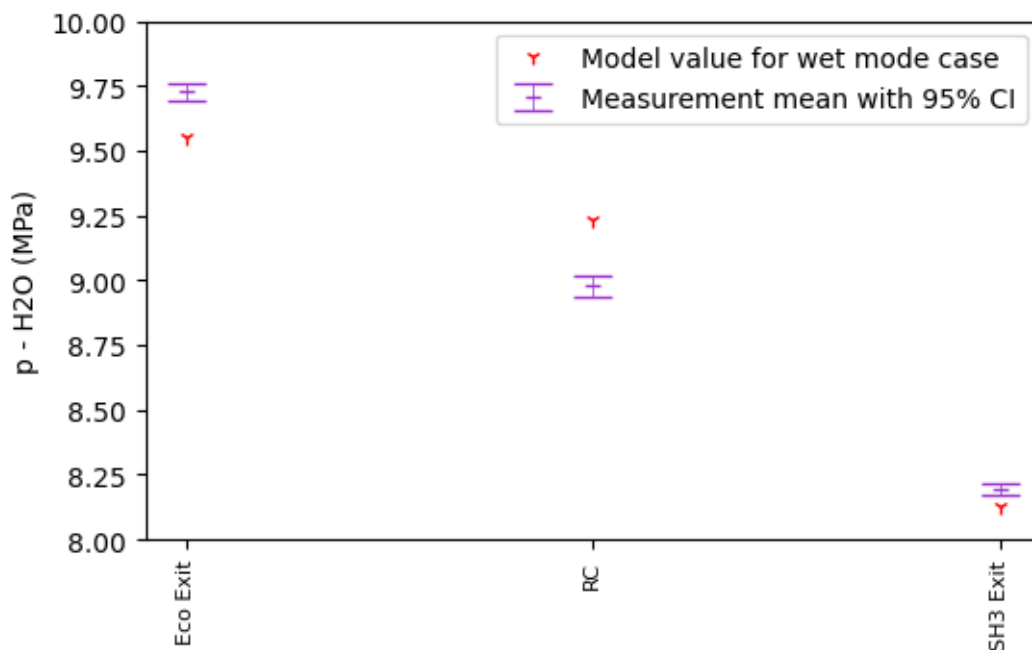


Figure 5.20 MS stream water/steam pressure results at 35% MCR

Table 5.15 RC and SH3 steam pressure at 35% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
$p_{Eco\ exit}$	9.55	9.73	0.18	1.83%	± 0.03
p_{RC}	9.23	8.98	-0.25	-2.82%	± 0.04
$p_{SH3\ exit}$	8.13	8.20	0.06	0.75%	± 0.02

Table 5.16 RH2 exit steam pressure result at 35% MCR

Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
1.29	1.36	0.06	4.57%	± 0.01

The water/steam temperature results in Figure 5.21 suggest that the model underpredicts the performance of SH1 for the wet-mode simulation. The steam temperature at SH1 exit and SH2 inlet are the two results with the most significant errors, as shown in Table 5.17.

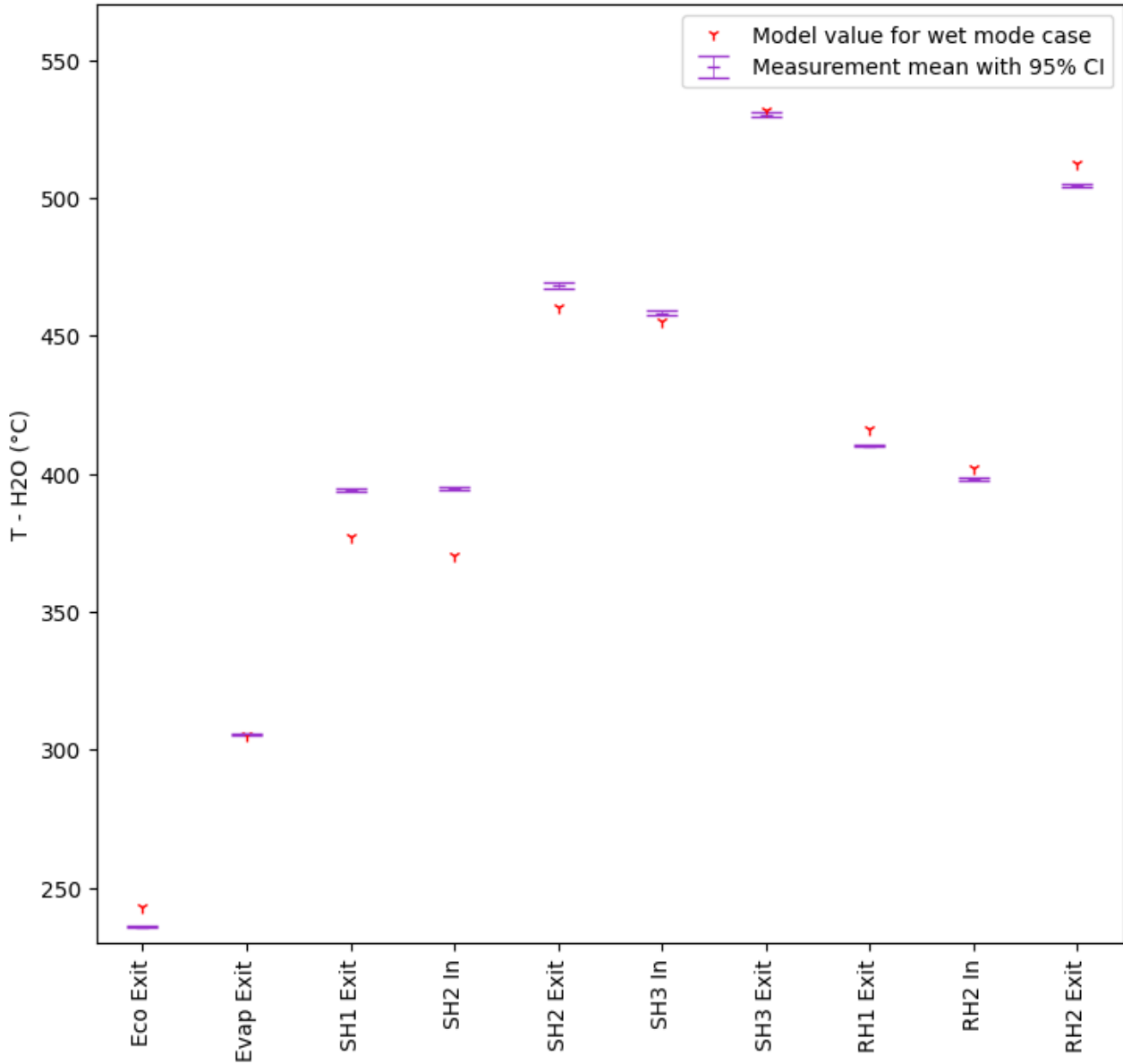


Figure 5.21 Water/steam temperature results at 35% MCR

Table 5.17 SH1 exit and SH2 inlet steam temperature results at 35% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{SH1\ exit}$	377.29	394.19	16.90	4.29%	±0.60
$T_{SH2\ in}$	370.14	394.60	24.46	6.20%	±0.60

As shown in Table 5.17, the plant data is almost identical for SH1 exit and SH2 inlet. This result does not seem logical since the plant data suggests that the steam leaving SH1 is first attemperated before entering SH2 (see Appendix Q for the spray water flow rate of attemperator 2). The rest of the steam temperature results show good accuracy, suggesting that the SH1 exit and SH2 inlet measurements might not truly reflect the actual steam temperatures during wet-mode operation.

5.4.2 Heat transfer results

Other evidence can be found by analysing the heat transfer results, as shown in Figure 5.22. The errors in the heat transfer results of SH1, SH2, and SH3 are significant. However, the heat transfer values obtained from the plant data are based on the temperature and pressure measurements; therefore, the temperature measurement errors at SH1 exit and SH2 inlet will have a knock-on effect on the heat transfer values of the downstream heat exchangers. In this case, the average relative error in the heat transfer result is 10.4% and Table 5.18 shows that the maximum relative error is 18.1%.

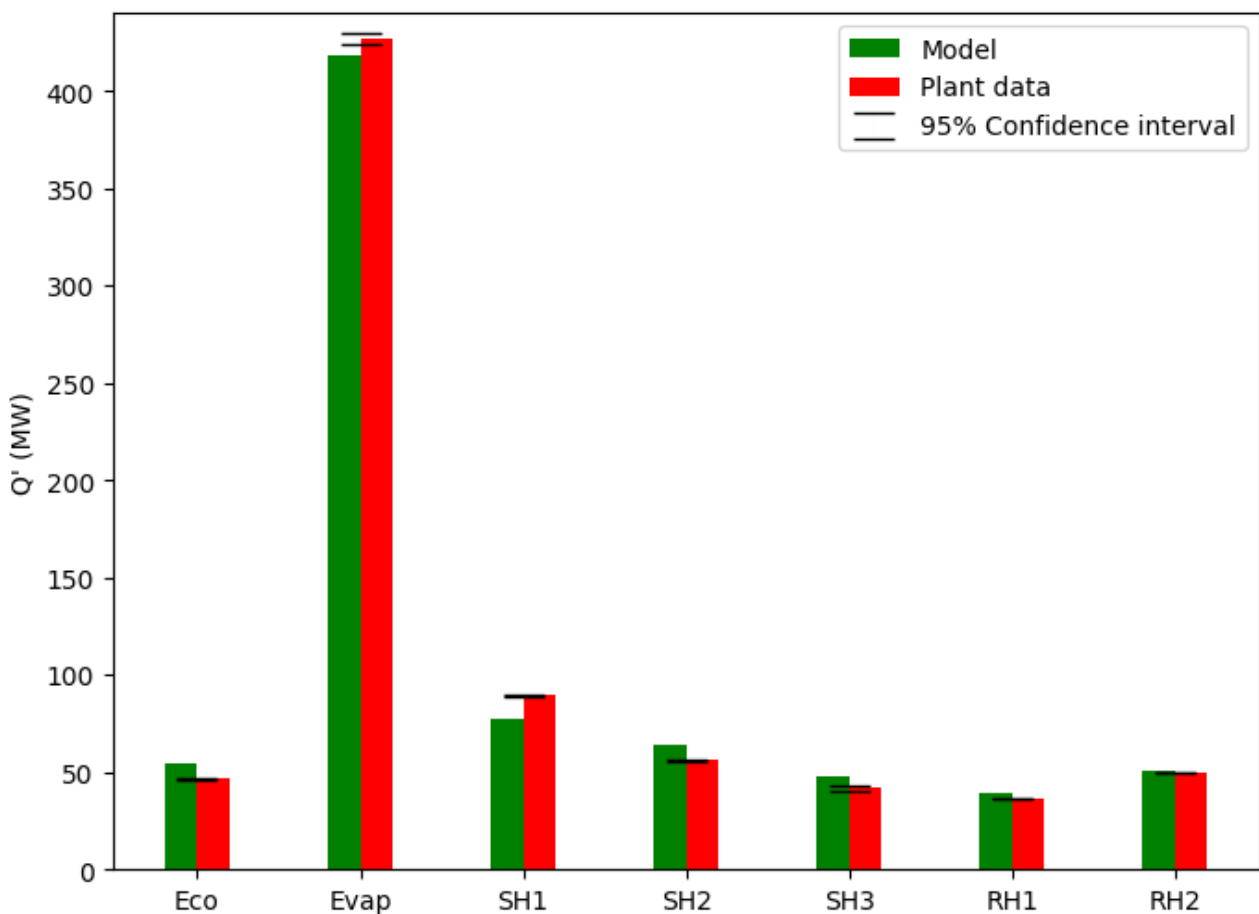


Figure 5.22 Total heat transfer in each heat exchangers at 35% MCR

Table 5.18 Heat transfer results at 35% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	55.01	46.60	-8.41	-18.06%	± 0.69
$\dot{Q}_{Evap}$	418.39	426.65	8.26	1.94%	± 2.88
$\dot{Q}_{SH1}$	77.32	89.37	12.05	13.48%	± 0.65
$\dot{Q}_{SH2}$	64.33	56.29	-8.04	-14.28%	± 0.49
$\dot{Q}_{SH3}$	48.21	41.81	-6.41	-15.32%	± 1.58
$\dot{Q}_{RH1}$	39.37	36.53	-2.84	-7.78%	± 0.26
$\dot{Q}_{RH2}$	50.46	49.56	-0.89	-1.80%	± 0.24
$\dot{Q}_{Total}$	752.98	746.80	-6.18	-0.83%	± 4.75

Additionally, Figure 5.23 shows that the model's cumulative heat transfer results of MS and RS for the wet-mode operation correspond closely to the measured data, thus implying that the model can capture the overall heat transfer effects in both MS and RS. In this case, Table 5.18 shows that the relative errors of the cumulative heat transfer results are all under 5%.

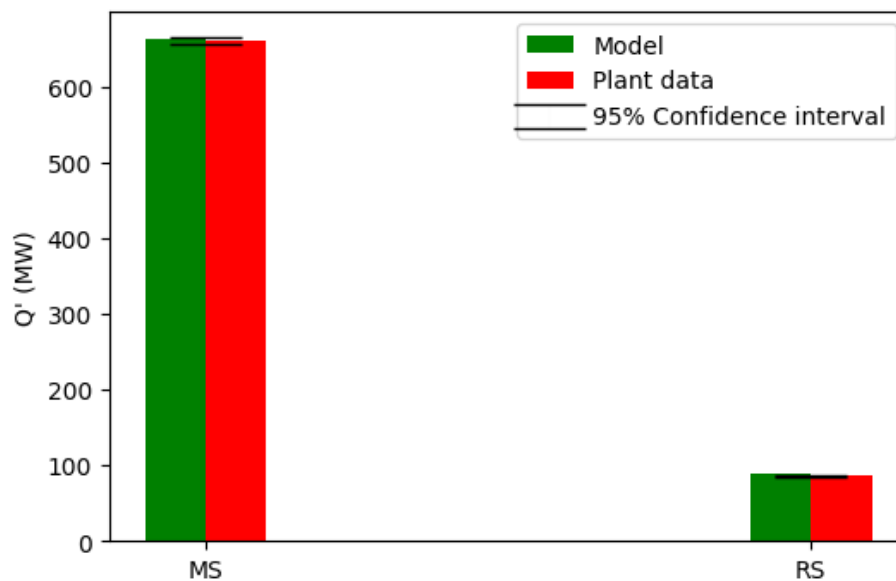


Figure 5.23 Cumulative heat transfer in MS and RS at 35% MCR

Table 5.19 cumulative heat transfer results of MS and RS at 35% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{MS}$	663.25	660.71	-2.54	-0.38%	± 4.40
$\dot{Q}_{RS}$	89.83	86.09	-3.74	-4.34%	± 0.46
$\dot{Q}_{Total}$	752.98	746.80	-6.18	-0.83%	± 4.75

If the measured steam temperatures at SH1 exit and SH2 inlet for the 35% MCR load are indeed correct, the heat transfer results imply that the ash deposition of the heat exchangers upstream of SH1 in the flue gas flow direction is significantly higher during wet mode operation. Additionally, the heat transfer in the boiler can also be influenced by the dead spots in the heat exchangers. These dead spots are caused by the low up-flow velocities and non-uniform gas flow profiles, resulting from the staggered burner firing arrangements during low-load operating conditions. However, the plant data suggests that the case study boiler did not shut down for an extended period due to maintenance between the case study periods. Moreover, Table 5.1 shows that the periods between the case study operations are relatively close. Therefore, a drastic change in ash deposition does not seem feasible. Some doubt, therefore, still remains about the accuracy of the measured temperatures in the wet mode operation case. Nonetheless, the results demonstrate the ability of the model to capture unusual events and study the ash deposition phenomenon at different operating conditions (including low load operation).

As mentioned in Section 5.3.2, the available heat for each heat exchanger decreases with decreasing load due to the decrease in temperature differences between the flue gas and water/steam. For that reason, the temperature of the infeed water to the boiler is dropped to maintain a certain level of thermal potential, as shown in Figure 5.24.

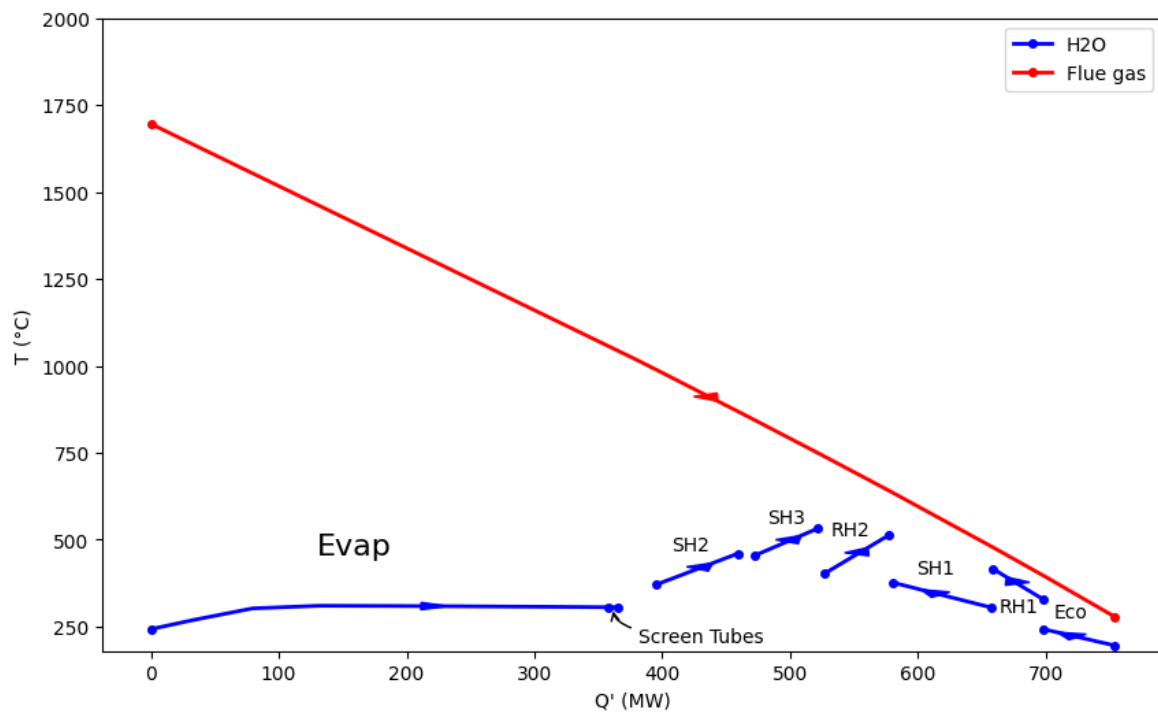


Figure 5.24 Q'T diagram of model results at 35% MCR

5.4.3 Metal temperature results

The metal temperature shown in Figure 5.25 is another parameter that can be used to validate the model performance for the 35% MCR simulation, which the results show sufficient accuracy. In the case study wet-mode operation, the maximum metal temperature results shown in Table 5.20 suggests that the heat exchanger tubes are sufficiently cooled during operation.

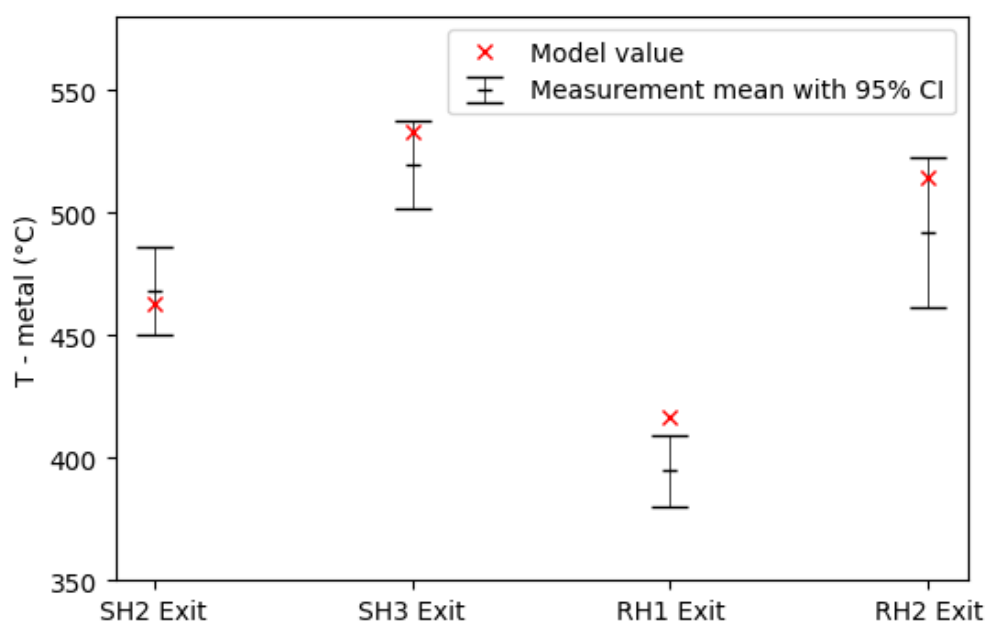


Figure 5.25 Metal temperature results at 35% MCR

Table 5.20 Maximum metal temperatures of each heat exchanger at 35% MCR

Heat exchanger	Model (°C)	Design (°C)	Pass
Eco	244.20	327.00	4
SH1	379.60	474.00	6
SH2	463.81	494.00	2
SH3	533.50	558.00	2
RH1	416.95	512.00	4
RH2	514.79	566.00	2

As discussed in Section 3.1, the once-through boilers have a minimum water flow rate in the furnace water walls to prevent overheating during wet-mode operation. In this case, the temperature results in Figure 5.26 show that the furnace water wall tubes are operating in the safe range even during wet-mode operation.

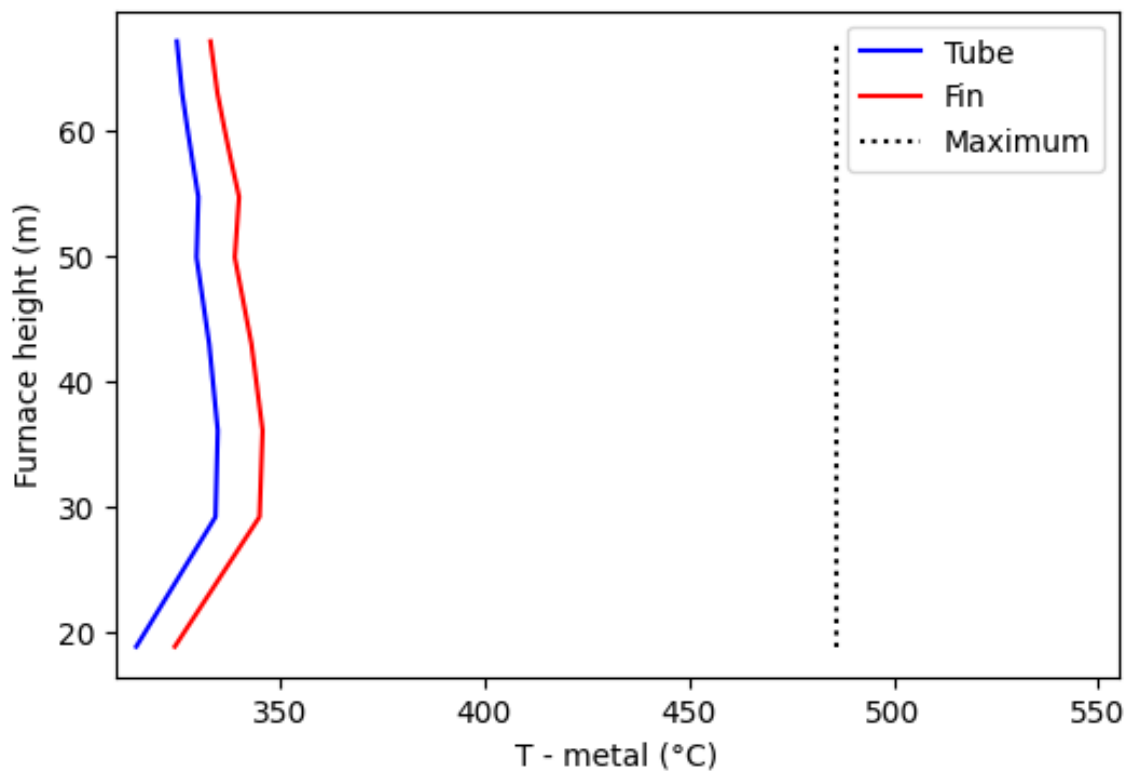


Figure 5.26 Membrane wall tube and fin metal temperature results at 35% MCR

5.5 Gas side pressure drop calculations at full load conditions

As shown in Section 4.4.1, the model demonstrates the ability to compute the flue gas pressure drop across the heat exchangers for the designed full load operation. This feature is also validated using the plant data of full load operation, for which the results are shown in Figure 5.27. The model's pressure results are not as accurate as those in Figure 4.23. However, Table 5.21 shows that the pressure results that are outside of the confidence intervals have relative errors less than 5%.

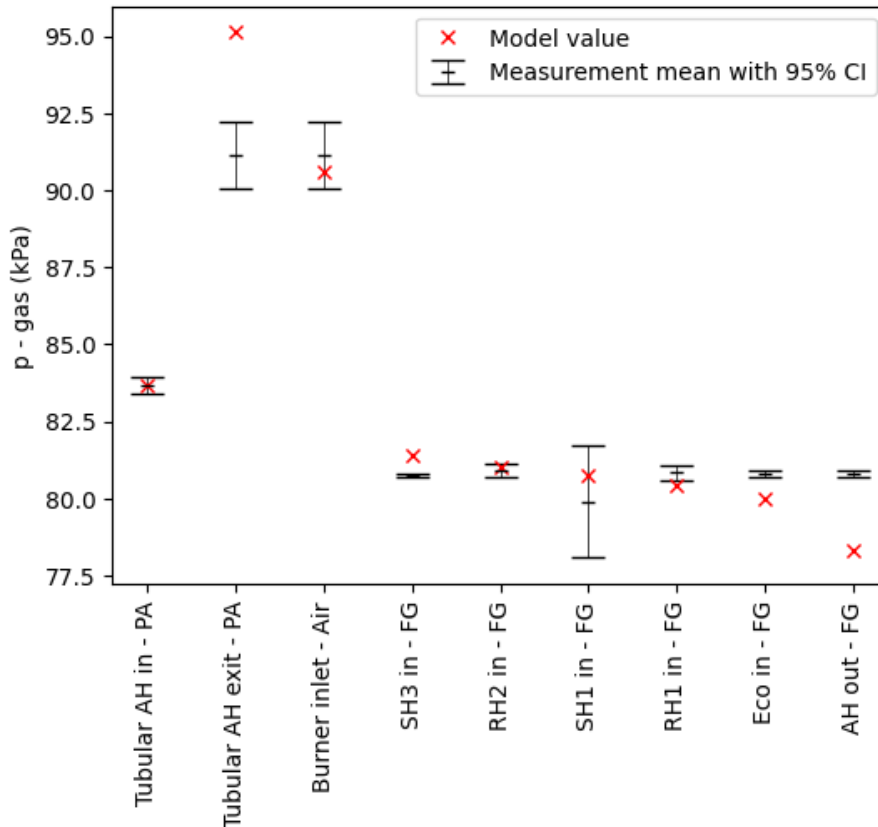


Figure 5.27 Air and flue gas pressure results at 100% MCR

Table 5.21 Out of confidence air and flue gas pressure results at 100% MCR

	Model (kPa)	Plant data (kPa)	Absolute error (kPa)	Relative error	95% confidence interval (kPa)
$p_{PA\ Tub\ AH\ exit}$	95.11	91.14	3.98	4.37%	± 1.08
$p_{FG\ SH3\ in}$	81.41	80.77	0.64	0.79%	± 0.06
$p_{FG\ Eco\ in}$	80.00	80.80	-0.80	-0.99%	± 0.11
$p_{FG\ AH\ exit}$	78.32	80.80	-2.48	-3.08%	± 0.11

The model's pressure results can be used to determine the total flue gas pressure drop across the boiler as shown in Table 5.22. These results can be used to assess whether the forced-draught fan is out of capacity or not.

Table 5.22 Model results for flue gas pressure drop across the boiler at 100% MCR

	Pressure drop across the furnace and heat exchangers	Pressure drop across Air heaters	Total pressure drop across the entire boiler
Symbol	$\Delta p_{FG\ HX}$	$\Delta p_{FG\ AH}$	$\Delta p_{FG\ total}$
Unit	<i>kPa</i>	<i>kPa</i>	<i>kPa</i>
Value	-2.92	-1.49	-4.40

6. Conclusion and recommendations

This aim of the project was to develop a quasi-steady-state integrated thermofluid network model for a complete tower-type once-through boiler. The model must provide sufficiently accurate results at a low computational cost and requisite component resolution for the boiler operator to gain insight into the boiler operation under different operating conditions, including low-load operation.

A challenge in the development process is that the existing literature has limited information on fouling and slagging for high ash coal used in South African CFPPs. Despite this, the model needs to account for the effects of fouling/slagging since it has significant impact on the heat transfer in the boiler. To address this, a model calibration methodology was developed that uses real plant data in an iterative process to determine suitable fouling/slagging thermal resistance values.

Based on the results obtained, the following conclusions may be highlighted:

- The integrated thermofluid network model developed in this project consists of multiple components, with each heat exchanger component discretised on a pass-by-pass basis. The evaporator model includes a thermal network model that can capture temperatures at different regions of the membrane water wall.
- The model validation using designer specifications shows that the model computes sensible results. However, the heat transfer results show that the typical ash deposition resistances suggested in literature are not directly applicable for the case study boiler. The model requires additional calibration to realistically account for the ash deposition effects.
- A model calibration methodology was developed and applied using plant data of boiler operation at 100% MCR. The results show sufficient accuracy compared to plant data, with the average error of the predicted heat exchanger heat loads being 2.0% and the maximum error being 5.2%.
- The calibrated model is first applied to simulate boiler operations under dry mode at 81% and 63% MCR loads. The 81% load case's average error in the heat exchanger duty is 2.0% and the maximum is 4.9%. For the 63% load case, the average error is 3.6% and the maximum is 9.7%. However, the heat transfer results at 63% MCR show that the model significantly overpredicts the performance of SH3. This finding suggests that the ash deposition on SH3 could be higher during this particular operating condition than at full-load. This conclusion is supported by the plant operating experience as reported by the site engineer.
- For the wet-mode (low-load) operation simulation at 35% MCR, the results are shown to be less accurate, especially the steam temperatures at SH1 exit and SH2 inlet. In this case, the average error in the heat exchanger duty is 10.4% and the maximum is 18.1%. However, the

plant measurements in this case are suspect since it does not reflect the attemperation taking place after leaving SH1 before entering SH2. If these measurements are indeed correct, the results imply that the ash deposition on the heat exchangers is significantly higher during wet mode operation than at full load operation. Additionally, the heat transfer in the boiler can also be influenced by the dead spots in the heat exchangers. These dead spots are caused by the low up-flow velocities and non-uniform gas flow profiles, resulting from the staggered burner firing arrangements during low-load operating conditions. Still, the cumulative heat transfer results correspond closely to the measured data, thus implying that the model can capture the integrated overall heat transfer effects.

- Lastly, the model's metal temperature results are shown to be sufficiently accurate and can be used to identify the heat exchanger tube passes that are most at risk of overheating.
- The model demonstrated the ability to compute sufficiently accurate heat transfer results if the ash deposition is modelled correctly. This feature can be used in reverse to study the ash deposition phenomenon during operation.
- The heat distribution used in the furnace model is based on the experimental data from the literature. The experiments were conducted before 1987 on a different type of CFPP boiler. For that reason, this empirical correlation might not accurately reflect the heat distribution in the case study furnace. The recommendation is to use a CFD model of the case study furnace to investigate the heat distribution at different operating loads, which can help find a more realistic correlation.
- The model showed sufficient accuracy in simulating quasi-steady-state boiler operation. However, the model does not integrate any features associated with the transient operation. For that reason, control feature integration is a way to improve the model for dynamic simulations.

7. List of References

- [1] D. R. Walwyn and A. C. Brent, "Renewable energy gathers steam in South Africa," *Renew. Sustain. Energy Rev.*, vol. 41, pp. 390–401, 2015, doi: 10.1016/j.rser.2014.08.049.
- [2] Agora Energiewende, "Flexibility in thermal power plants-with a focus on existing coal-fired power plants," 2017. [Online]. Available: https://www.agora-energiewende.de/fileadmin/Projekte/2017/Flexibility_in_thermal_plants/115_flexibility-report-WEB.pdf.
- [3] D. Taler, M. Trojan, P. Dzierwa, K. Kaczmarek, and J. Taler, "Numerical simulation of convective superheaters in steam boilers," *Int. J. Therm. Sci.*, vol. 129, no. April, pp. 320–333, 2018, doi: 10.1016/j.ijthermalsci.2018.03.005.
- [4] R. Laubscher and P. Rousseau, "CFD study of pulverized coal-fired boiler evaporator and radiant superheaters at varying loads," *Appl. Therm. Eng.*, vol. 160, no. June, p. 114057, 2019, doi: 10.1016/j.applthermaleng.2019.114057.
- [5] P. Rosslee, "Integrated process and control modelling of water recirculation in once-through boilers during low load and transient operation," University of Cape Town, 2020.
- [6] D. O. Glushkov, A. I. Matiushenko, A. E. Nurpeiis, and A. V. Zhuikov, "An experimental investigation into the fuel oil-free start-up of a coal-fired boiler by the main solid fossil fuel with additives of brown coal, biomass and charcoal for ignition enhancement," *Fuel Process. Technol.*, vol. 223, no. August, p. 106986, 2021, doi: 10.1016/j.fuproc.2021.106986.
- [7] S. Li, Z. Chen, E. He, B. Jiang, Z. Li, and Q. Wang, "Combustion characteristics and NO_x formation of a retrofitted low-volatile coal-fired 330 MW utility boiler under various loads with deep-air-staging," *Appl. Therm. Eng.*, vol. 110, no. x, pp. 223–233, 2017, doi: 10.1016/j.applthermaleng.2016.08.159.
- [8] X. Zhang, Z. Chen, M. Zhang, L. Zeng, and Z. Li, "Combustion stability, burnout and NO_x emissions of the 300-MW down-fired boiler with bituminous coal: Load variation and low-load comparison with anthracite," *Fuel*, vol. 295, no. x, p. 120641, 2021, doi: 10.1016/j.fuel.2021.120641.
- [9] S. Wang, D. Yang, Y. Zhao, and M. Qu, "Heat transfer characteristics of spiral water wall tube in a 1000 MW ultra-supercritical boiler with wide operating load mode," *Appl. Therm. Eng.*, vol. 130, pp. 501–514, 2018, doi: 10.1016/j.applthermaleng.2017.10.114.
- [10] S. Echi, A. Bouabidi, Z. Driss, and M. S. Abid, "CFD simulation and optimization of industrial boiler," *Energy*, vol. 169, pp. 105–114, 2019, doi: 10.1016/j.energy.2018.12.006.
- [11] F. Alobaid, N. Mertens, R. Starkloff, T. Lanz, C. Heinze, and B. Eppele, "Progress in dynamic simulation of thermal power plants," *Prog. Energy Combust. Sci.*, vol. 59, pp. 79–162, 2017, doi: 10.1016/j.pecs.2016.11.001.
- [12] N. Modliński, K. Szczepanek, D. Nabagło, P. Madejski, and Z. Modliński, "Mathematical procedure for predicting tube metal temperature in the second stage reheater of the operating flexibly steam boiler," *Appl. Therm. Eng.*, vol. 146, no. September 2018, pp. 854–865, 2019, doi: 10.1016/j.applthermaleng.2018.10.063.

- [13] N. Modliński, P. Madejski, T. Janda, K. Szczepanek, and W. Kordylewski, "A validation of computational fluid dynamics temperature distribution prediction in a pulverized coal boiler with acoustic temperature measurement," *Energy*, vol. 92, no. x, pp. 77–86, 2015, doi: 10.1016/j.energy.2015.05.124.
- [14] T. Chen, Y. jun Zhang, M. ran Liao, and W. zong Wang, "Coupled modeling of combustion and hydrodynamics for a coal-fired supercritical boiler," *Fuel*, vol. 240, no. March 2018, pp. 49–56, 2019, doi: 10.1016/j.fuel.2018.11.008.
- [15] R. Laubscher and P. Rousseau, "An enhanced model of particle radiation properties in high ash gas-particle dispersion flow through industrial gas-to-steam heat exchangers," *Fuel*, vol. 285, no. July 2020, p. 119153, 2021, doi: 10.1016/j.fuel.2020.119153.
- [16] H.-G. Brummel, "K4 Thermal Radiation of Gas–Solids–Dispersions," *VDI Heat Atlas*, pp. 989–999, 2010, doi: 10.1007/978-3-540-77877-6.
- [17] R. I. Backreedy, L. M. Fletcher, L. Ma, M. Pourkashanian, and A. Williams, "Modelling pulverised coal combustion using a detailed coal combustion model," *Combust. Sci. Technol.*, vol. 178, no. 4, pp. 763–787, 2006, doi: 10.1080/00102200500248532.
- [18] C. Yin, "On gas and particle radiation in pulverized fuel combustion furnaces," *Appl. Energy*, vol. 157, pp. 554–561, 2015, doi: 10.1016/j.apenergy.2015.01.142.
- [19] K. Deng, C. Yang, H. Chen, N. Zhou, and S. Huang, "Start-Up and dynamic processes simulation of supercritical once-through boiler," *Appl. Therm. Eng.*, vol. 115, pp. 937–946, 2017, doi: 10.1016/j.applthermaleng.2017.01.016.
- [20] R. Starkloff, F. Alobaid, K. Karner, B. Eppele, M. Schmitz, and F. Boehm, "Development and validation of a dynamic simulation model for a large coal-fired power plant," *Appl. Therm. Eng.*, vol. 91, pp. 496–506, 2015, doi: 10.1016/j.applthermaleng.2015.08.015.
- [21] M. Richter, G. Oeljeklaus, and K. Görner, "Improving the load flexibility of coal-fired power plants by the integration of a thermal energy storage," *Appl. Energy*, vol. 236, no. October 2018, pp. 607–621, 2019, doi: 10.1016/j.apenergy.2018.11.099.
- [22] J. Taler *et al.*, "Mathematical model of a supercritical power boiler for simulating rapid changes in boiler thermal loading," *Energy*, vol. 175, pp. 580–592, 2019, doi: 10.1016/j.energy.2019.03.085.
- [23] Y. Niu, M. Du, W. Ge, H. Luo, and G. Zhou, "A dynamic nonlinear model for a once-through boiler-turbine unit in low load," *Appl. Therm. Eng.*, vol. 161, no. 2017, p. 113880, 2019, doi: 10.1016/j.applthermaleng.2019.113880.
- [24] D. Wang, Y. Zhou, and X. Li, "A dynamic model used for controller design for fast cut back of coal-fired boiler-turbine plant," *Energy*, vol. 144, pp. 526–534, 2018, doi: 10.1016/j.energy.2017.12.053.
- [25] G. Sankar, D. S. Kumar, and K. R. Balasubramanian, "Computational modeling of pulverized coal fired boilers – A review on the current position," *Fuel*, vol. 236, no. August 2018, pp. 643–665, 2019, doi: 10.1016/j.fuel.2018.08.154.
- [26] M. Trojan, "Modeling of a steam boiler operation using the boiler nonlinear mathematical model," *Energy*, vol. 175, pp. 1194–1208, 2019, doi: 10.1016/j.energy.2019.03.160.
- [27] B. Jin, H. Zhao, and C. Zheng, "Dynamic simulation for mode switching strategy in a conceptual

- 600 MWe oxy-combustion pulverized-coal-fired boiler,” *Fuel*, vol. 137, pp. 135–144, 2014, doi: 10.1016/j.fuel.2014.07.104.
- [28] H. Hajebzadeh, A. N. M. Ansari, and S. Niazi, “Mathematical modeling and validation of a 320 MW tangentially fired boiler: A case study,” *Appl. Therm. Eng.*, vol. 146, no. September 2018, pp. 232–242, 2019, doi: 10.1016/j.applthermaleng.2018.09.102.
- [29] S. Gordon and B. J. McBride, “Computer program for calculation of complex chemical equilibrium compositions rocket performance incident and reflected shocks, and Chapman-Jouguet detonations,” *Comput. Progr. Calc. complex Chem. Equilib. Compos. Rocket Perform. Incid. reflected Shock. ChapmanJouguet detonations*, p. 250, 1971.
- [30] B. Prabir, K. Cen, and J. Louis, *Boilers and Burners: Design and Theory*. New York: Springer, 2012.
- [31] S. Kakaç, *Boilers, evaporators, and condensers*. New York: John Wiley and Sons, 1991.
- [32] A. G. Blokh and R. Viskanta, *Heat Transfer In Steam Boiler Furnaces*. Taylor & Francis, 1988.
- [33] Y. Zhang, Q. Li, and H. Zhou, *Theory and calculation of heat transfer in furnaces*. London: Elsevier, 2016.
- [34] R. Laubscher and P. Rousseau, *Power Plant Boilers Thermofluid Processes and Controls - MEC4117Z Course*, Rev A. University of Cape Town, 2021.
- [35] V. Ganapathy, *Steam Generators and Waste Heat Boilers For Process and Plant Engineers*. 2014.
- [36] X. Zhang, J. Yuan, Z. Chen, Z. Tian, and J. Wang, “A dynamic heat transfer model to estimate the flue gas temperature in the horizontal flue of the coal-fired utility boiler,” *Appl. Therm. Eng.*, vol. 135, no. February, pp. 368–378, 2018, doi: 10.1016/j.applthermaleng.2018.02.067.
- [37] P. Rousseau and R. Laubscher, “A thermofluid network-based model for heat transfer in membrane walls of pulverized coal boiler furnaces,” *Therm. Sci. Eng. Prog.*, vol. 18, no. April, p. 100547, 2020, doi: 10.1016/j.tsep.2020.100547.
- [38] Y. A. Cengel, *Heat Transfer A Practical Approach*, 2nd ed., vol. 4, no. 9. MacGraw-Hill, 2004.
- [39] N. V. Kuznetsov, V. V. Mitor, I. E. Dubovskii, and E. S. Karasina, “Thermal calculation of boiler units (normative method),” *Energiya, Moscow*, 1973.
- [40] N. Modliński, K. Szczepanek, D. Nabagło, P. Madejski, and Z. Modliński, “Mathematical procedure for predicting tube metal temperature in the second stage reheater of the operating flexibly steam boiler,” *Appl. Therm. Eng.*, vol. 146, pp. 854–865, Jan. 2019, doi: 10.1016/j.applthermaleng.2018.10.063.
- [41] V. Gnielinski, “G7 Heat Transfer in Cross-flow Around Single Rows of Tubes and Through Tube Bundles,” *VDI Heat Atlas*, pp. 725–729, 2010, doi: 10.1007/978-3-540-77877-6.
- [42] M. Trojan and D. Taler, “Thermal simulation of superheaters taking into account the processes occurring on the side of the steam and flue gas,” *Fuel*, vol. 150, pp. 75–87, 2015, doi: 10.1016/j.fuel.2015.01.095.
- [43] H. Muller-Steinhagen, “C4 Fouling of Heat Exchanger Surfaces,” *VDI Heat Atlas*, pp. 79–104, 2010, doi: 10.1007/978-3-540-77877-6.

- [44] H. Y. Park *et al.*, “Thermal resistance by slagging and its relationship with ash properties for six coal blends in a commercial coal-fired boiler,” *Fuel*, vol. 235, no. March 2018, pp. 1377–1386, 2019, doi: 10.1016/j.fuel.2018.08.134.
- [45] J.B. Kitto and S.C. Stultz, *Steam: Its generation and use*, vol. 51, no. 1. 2005.
- [46] Y. A. Cengel and M. A. Boles, *Thermodynamics An Engineering Approach*, 7th ed. MacGraw-Hill, 2010.
- [47] M-Tech Industrial, *Flownex Simulation Environment General User Manual*. M-Tech Industrial, 2020.
- [48] M-Tech Industrial, *Flownex Simulation Environment Theory Manual*. M-Tech Industrial, 2020.
- [49] E. S. Gaddis, “L1.4 Pressure Drop of Tube Bundles in Cross Flow,” *VDI Heat Atlas*, pp. 1076–1091, 2010, doi: 10.1007/978-3-540-77877-6_70.
- [50] W. Richter and K. Gorner, “K5 Heat Radiation in Furnaces,” *VDI Heat Atlas*, 2010, doi: 10.1007/978-3-540-77877-6.
- [51] K. G. Schmidt, “M1 Heat Transfer to Finned Tubes,” *VDI Heat Atlas*, pp. 739–740, 2010.
- [52] S. V. Vassilev and C. G. Vassileva, “A new approach for the combined chemical and mineral classification of the inorganic matter in coal. 1. Chemical and mineral classification systems,” *Fuel*, vol. 88, no. 2, pp. 235–245, 2009, doi: 10.1016/j.fuel.2008.09.006.
- [53] “onshape.” 2014, [Online]. Available: <https://www.onshape.com/en/>.

Appendix A. Oxidations of combustion species

The stoichiometric combustion analysis considers the oxidations of carbon, sulphur and hydrogen in the fuel.

The chemical equation for carbon C oxidation is calculated as follows:



where the chemical equation for sulphur S oxidation is calculated as follows:



Furthermore, the chemical equation for hydrogen H oxidation is calculated as follows:



Appendix B. Correlations for Gurvich method

The flame centre modification factor M can be determined using empirical correlation presented by Kakaç [31] and Prabir et al. [30]:

$$M = A' - B'(X_r + \Delta X) \quad (\text{B-1})$$

where X_r is the relative height of the highest temperature zone in the furnace, and it is defined as follows:

$$X_r = \frac{H_{r,eff}}{H_{furnace}} \quad (\text{B-2})$$

where $H_{r,eff}$ [m] is the effective flame height and $H_{furnace}$ [m] is the furnace height. These heights are measured from the hopper mid-plane, as shown in Figure B.1.

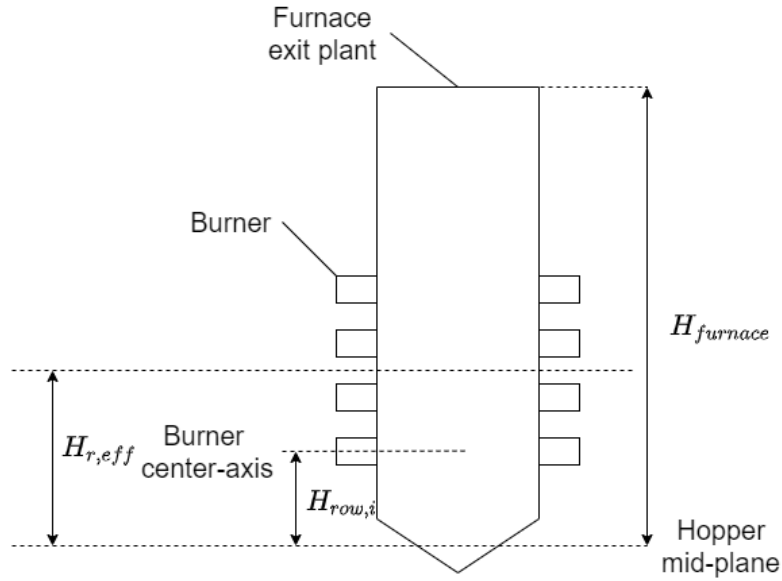


Figure B.1 Heights for flame modification factor calculation

The effective flame height is taken as the weighted average of each burner height $H_{row,i}$, expressed as follows:

$$H_{r,eff} = \frac{\sum n_i \dot{m}_{fuel \text{ per row}} H_{row,i}}{\sum n_i \dot{m}_{fuel \text{ per row}}} \quad (\text{B-3})$$

where n_i is the number of burners in the burner row, $\dot{m}_{fuel \text{ per row}} \left[\frac{kg}{s} \right]$ is the fuel consumption per burner in a row.

The ΔX is the correction factor for the actual flame core position. For horizontal burners $\Delta X = 0$. For tilting horizontal burners at $\mp 20^\circ C$, $\Delta X = \mp 0.1$. For burners arranged on front walls or opposing walls, $\Delta X = 0.05 \sim 0.1$.

The coefficients A' and B' depends on the type of fuel and the furnace construction and can be found in Table B.1.

Table B.1 Flame modification factors [30], [31]

Fuel	A'	B'
Gas, fuel oil	0.54	0.2
High volatile or reactive coal	0.59	0.5
Low reactive fuels (anthracite or high ash coal)	0.56	0.5

The furnace wall efficiency factor (ψ) is determined using two empirical correlations, namely the angular coefficient (x_{ang}), and the furnace fouling factor (ξ):

$$\psi = x_{ang} \xi \quad (B-4)$$

The angular coefficient is a geometric factor, which accounts for the radiation that bypasses a pitched tubular water wall. The radiation that bypasses the tubes is absorbed by a continuous plane with an emissivity and temperature close to the tubes.

For exposed radiant superheater (exit plane), tangential furnace tubes and slag screens, the angular coefficient equals 1. In comparison, the angular coefficient for furnace water walls can be found using empirical correlations presented by Kakaç [31] and Prabir et al. [30], shown in Figure B.2.

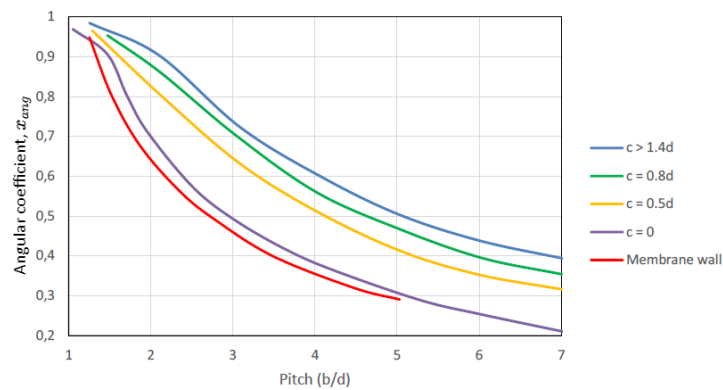


Figure B.2 Angular coefficients as a function of the pitch [30], [31]

In Figure B.2, the water wall tube pitch (b [m]) is divided by the tube outside diameter (d [m]). The term c [m] in Figure B.2 refers to the distance from the centre of the tube to the continuous plane behind the tube row.

For membrane walls, the function for the angular coefficient is obtained from fitting the curve in Figure B.2 [34], and it is expressed as follows:

$$x_{ang} = 1.15484 \left(\frac{b}{d} \right)^{-0.850475} \quad (\text{B-5})$$

The thermal properties of the ash deposits on the furnace walls inhibit the heat transfer from the gases to the working fluid inside the tubes. The high thermal resistance of the ash causes the surface temperature exposed to the hot gases to increase substantially, reducing the radiation heat transfer due to the lower temperature gradient between the gas and the walls. Furthermore, the emissivity of the ash deposits is also lower than that of the tube material. For that reason, these effects are accounted for by applying a furnace fouling factor. For the furnace exit plane, furnace fouling factor equals 1 [30], [31]. Whereas for the furnace water walls, the fouling factors are dependant on their configurations and fuel type, which their values are presented by Kakaç [31] and Prabir et al. [30], shown in Table B.2.

Table B.2 Fouling factory for various water wall configurations and fuels [30], [31]

Water wall configuration	Fuel type	ξ
Water wall or studded water walls	Gas, heavy oil and gas mixture	0.65
	Heavy oil	0.55
	Coal	0.35~0.45
	All grate-fired fuels	0.6
	Other fuels	0.2
Water wall with refractory backing	All fuels	0.2

The radiation loss fraction ($f_{rad\ loss}$) accounts for the radiation loss through the boiler skin casings, calculated from the American Boiler Manufacturer Association (ABMA) charts as an empirical relation. Furthermore, the radiation loss fraction is used in ASME Power Test Code PTC 4.1-464 [34].

The standard calculation for the radiation loss fraction can be expressed as follows:

$$f_{rad\ loss} = F_1 F_2 F_3 \exp \left(a' + b' \times \ln(\dot{Q}_{steam}) + c' \times [(\dot{Q}_{steam})]^2 + d' \times [\ln(\dot{Q}_{steam})]^3 \right) \quad (B-6)$$

The boiler unit is assumed to have 28K temperature difference between the skin casing and the ambient air. Additionally, the wind speed is assumed to be $0.51 \frac{m}{s}$ and external surface emissivity is assumed to be 0.95.

The steam heat load ($\dot{Q}_{steam}$ [MW]) is the net heat transfer into the water/steam flows inside the boiler.

The exponential coefficients in Equation (B-6) are defined as:

$$\begin{aligned} a' &= 1.5551 \\ b' &= -0.72889 \\ c' &= 4.4410 \times 10^{-2} \\ d' &= -1.4924 \times 10^{-4} \end{aligned} \quad (B-7)$$

The correction factor for the temperature difference and wind speed (F_1) can be calculated as follows:

$$F_1 = \Delta T_{casing} (a + b \Delta T_{casing} + c v_{wind} + d v_{wind} \Delta T_{casing}) \quad (B-8)$$

where ΔT_{casing} is the temperature difference between the skin casing and the atmosphere [K] and the wind speed (v_{wind}) is given in $\frac{m}{s}$. The boiler skin casing temperature is typically assumed to be between 35 and 45°C.

The coefficients in Equation (B-8) are defined as:

$$\begin{aligned} a &= 2.8766 \times 10^{-2} \\ b &= 6.7708 \times 10^{-5} \\ c &= 9.7636 \times 10^{-3} \\ d &= 5.0479 \times 10^{-6} \end{aligned} \quad (\text{B-9})$$

The correction factor for the number of furnace water-cooled walls (F_2) is defined as follows:

$$F_2 = 1 - 0.0063N_{walls} \quad (\text{B-10})$$

where the number of water-cooled walls (N_{walls}) in most boiler furnaces is 4.

The correction factor (F_3) is used for the boiler calculations are at part loads, and it is defined as follows:

$$F_3 = \frac{100}{MCR_{load}} \quad (\text{B-11})$$

where MCR_{load} [%] is the load percentage of the boiler.

Appendix C. Furnace gas and particle emissivity with high ash loading

The method of calculating the effective gas and particle emissivity (ϵ_{G+P}) is presented by Brummel [16].

The effective gas and particle emissivity is calculated as follows:

$$\epsilon_{G+P} = (1 - \beta_{G+P}) \left(\frac{1 - \exp(-\phi_{G+P})}{1 + \beta_{G+P} \exp(-\phi_{G+P})} \right) \quad (C-1)$$

where the calculation parameter (β_{G+P}) is defined as follows:

$$\beta_{G+P} = \frac{\gamma - 1}{\gamma + 1} \quad (C-2)$$

where the radiation scattering factor (γ) is defined as follows:

$$\gamma = \left(1 + \frac{2Q_{scat}}{Q_{abs}} \right)^{\frac{1}{2}} \quad (C-3)$$

In Equation (C-3), the back-scattering efficiency (Q_{scat}) is an empirical function of the ash particle diameter (d_{fa} [μm]), which is presented by Laubscher et al. [34] and Richter et al. [50]:

$$Q_{scat} = \frac{6.2188 \times 10^{-3} - 1.0492 \times 10^{-2} \cdot d_{fa} + 7.287 \times 10^{-3} \cdot d_{fa}^2 - 2.1925 \times 10^{-5} \cdot d_{fa}^3}{1.851 \times 10^{-1} - 2.0405 \times 10^{-3} \cdot d_{fa}^2 + 6.254 \times 10^{-4} \cdot d_{fa}^3} \quad (C-4)$$

The particle absorptivity (Q_{abs}) is also an empirical function of the ash particle diameter, which is presented by Laubscher et al. [34] and Richter et al. [50]:

$$Q_{abs} = 0.275 \cdot d_{fa}^{0.298} - 0.305 \quad (C-5)$$

The optical thickness of the gas-particulate dispersion is expressed as follows:

$$\begin{aligned} \phi_{G+P} &= (Q_{abs} A_P L_P + K_{emi,G} + K_{coke}) S \gamma \text{ for furnace} \\ \phi_{G+P} &= (Q_{abs} A_P L_P + K_{emi,G}) S \gamma \text{ for others} \end{aligned} \quad (C-6)$$

where particle area A_p is calculated as follows:

$$A_p = \left(\frac{3}{2\rho_{fa}d_{fa}} \right) \quad (C-7)$$

The particle density of ash (ρ_{fa}) is typically assumed to be $2200 \frac{kg}{m^3}$ and the ash particle diameter in Equation (C-7) is given in meters.

The ash loading effect L_p is defined as follows:

$$L_p = \frac{\dot{m}_{ash}}{\dot{V}_{fg}} \quad (C-8)$$

where $\dot{V}_{fg}$ is the flue gas volume flow rate and $\dot{m}_{ash}$ is the mass flow rate of ash in the flow.

The gas absorption coefficient ($K_{emi,G}$) is calculated using the weighted-sum-of-grey-gas (WSGGM) model, which can be expressed as follows:

$$K_{emi,G} = \frac{-\ln(1 - \epsilon_g)}{S}$$

$$\epsilon_g = \sum_{i=1}^3 a_i (1 - \exp(-k_{G,i} p_i S)) \quad (C-9)$$

$$a_i = b_{0,i} + b_{1,i} T_{G+P}$$

where T_{G+P} [K] is the gas and particle flow temperature. The mean beam length of the gas and particle radiations (S) can be expressed as follows [30], [31]:

$$S = 3.6 \frac{V}{A} \quad (C-10)$$

where V refers to the volume of the flow regio and A refers to the area of the surfaces in contact with the gas and particle mixture.

For the furnace, the mean beam length ($S_{furnace}$) can be expressed as follows:

$$S_{furnace} = 3.6 \frac{V_{furnace}}{A_{furnace}} \quad (C-11)$$

where $V_{furnace}$ is the volume of the furnace and $A_{furnace}$ is the planar furnace area, which consists of the furnace internal surface area and the exit plane.

For tube bundles, the mean beam length ($S_{tube,bundle}$) can be expressed as follows:

$$S_{tube,bundle} = 3.6 \frac{\left(s_t s_l - \left(\frac{\pi d_o^2}{4} \right) \right)}{\pi d_o} \text{ for in-line}$$

$$S_{tube,bundle} = 3.6 \frac{\left(\frac{s_t s_l}{2} - \left(\frac{\pi d_o^2}{8} \right) \right)}{\pi d_o} \text{ for staggered} \quad (C-12)$$

where d_o is the outer tube diameter, s_t is the transversal pitch and s_l is the longitudinal pitch of the tube bank.

The term p_i [bar] in Equation (C-9) refers to tri-atomic gas (CO_2 , H_2O and SO_2) partial pressure in the gas and particle flow, which is determined using Dalton's law of partial pressures:

$$p_i = X_i p_{G+P} \quad (C-13)$$

where p_{G+P} is the fluid pressure of the gas and particle flow.

The mole fraction of each tri-atomic gas in the mixture (X_i) is defined as follows:

$$X_i = \frac{n_i}{n_{G+P}} \quad (C-14)$$

where the number of moles of each tri-atomic gas (n_i) is defined as follows:

$$n_i = \dot{m}_i M_i \quad (C-15)$$

where $\dot{m}_i$ is the mass flow rate of each tri-atomic gas and M_i is the gas' molar mass.

The number of moles of the gas and particle flow (n_{G+P}) is determined similarly to the number of moles of tri-atomic gases, which can be expressed as follows:

$$n_{G+P} = \dot{m}_{G+P} M_{G+P} \quad (C-16)$$

where $\dot{m}_{G+P}$ is the gas and particle mass flow rate and M_{G+P} is the molar mass of the gas and particle mixture.

The empirical coefficients used in WSGGM are presented by Richter et al. [50], and they are shown in Table C.1.

Table C.1 WSGGM model coefficients [50]

i	$b_{0,i} [-]$	$b_{1,i} \left[\frac{1}{K} \right]$	$k_{G,i} \left[\frac{1}{m \cdot bar} \right]$
1	0.13	0.000265	0.0
2	0.595	-0.000115	0.824
3	0.275	-0.000115	25.907

The coke absorption coefficient K_{coke} is found using an empirical correlation [34]:

$$K_{coke} = \left[\frac{1}{10.2} \cdot \frac{1}{m \cdot MPa} \right] \cdot 0.5 \cdot 0.1 \cdot p_{G+P} \quad (C-17)$$

where p_{G+P} is used in Equation (C-17) is in MPa .

Appendix D. Furnace heat flux nonuniformity factor

The heat flux nonuniformity factor η_k account for the heat flux distribution along with the height of the boiler and assumes a homogeneous heat flux distribution in the cross-sectional directions of the furnace.

The distribution of η_k generally has a shape shown in Figure D.1.

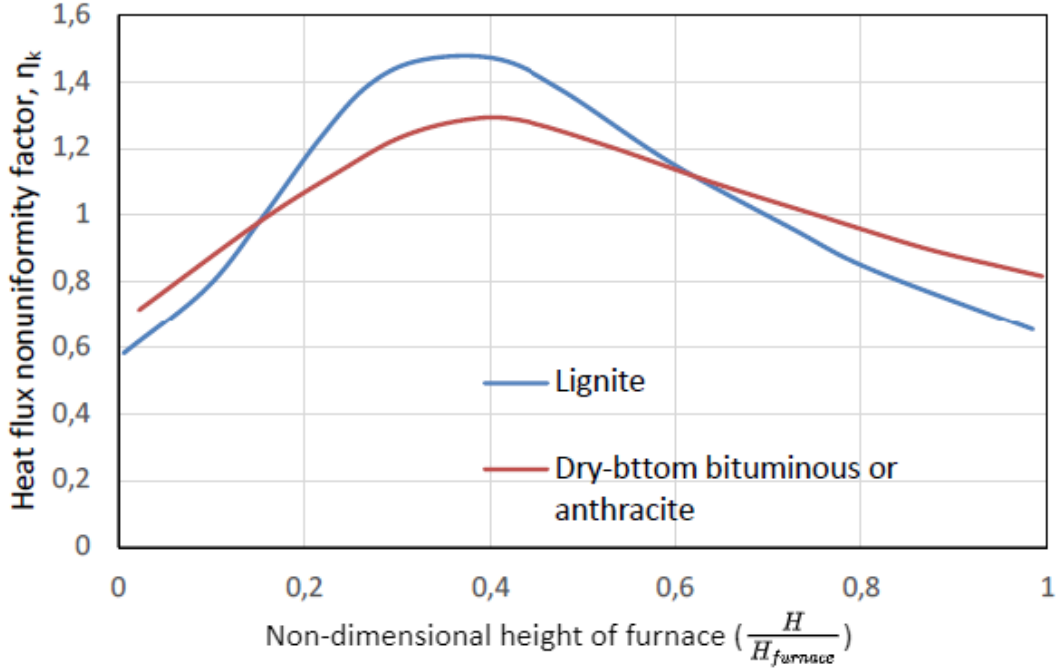


Figure D.1 Nonuniformity of heat flux distribution for coal and lignite fuels, with a centre burners axis located at $\frac{H}{H_{furnace}} = 0.185$ and 0.27 [30], [31]

Integration of the curves in Figure D.1 between the nondimensional heights of $0 \sim 1$ should yield a value of 1.0. These curves are a modified probability density function because their integral should result in 1 to ensure the integrated heat flux equals the total heat transfer. In the case where the numerical integration of the curves is not equal to one, the η_k should be adjusted to ensure the integral equals 1.

These curves presented by Kakaç [31] and Prabir et al. [30] were digitised by Laubscher et al. [34]. The functions were fitted to these exponential Gaussian distribution curves using the least square regression [34], and it can be expressed as follows:

$$\eta_k = c_1 \frac{\lambda}{2} \exp \left(\frac{\lambda}{2} \left(2\mu^{c_2} + \lambda\sigma^2 - \frac{2H}{H_{furnace}} \right) \operatorname{erfc} \left(\frac{\mu^{c_2} + \lambda\sigma^2 - \frac{H}{H_{furnace}}}{\sqrt{2}\sigma} \right) \right) \quad (D-1)$$

where the empirical coefficients (c_1, c_2, μ, λ and σ) in Equation (D-1) are shown in Table D.1.

Table D.1 Fitted parameters for heat flux nonuniformity curves [34]

Fuel	c_1	c_2	λ	μ	σ
Lignite	1.4728	1.3324	1.7131	0.2275	0.1907
Bituminous coal	1.9745	1.808	0.9821	0.2275	0.2371
Oil/gas	1.5214	2.9035	1.575	0.2675	0.1960

The furnace exit plane of once-through type boiler sits at nondimensional heights of 1, and therefore the associated heat flux nonuniformity factor ($\eta_{k,furnace,exit}$) can be calculated as follows:

$$\eta_{k,furnace,exit} = c_1 \frac{\lambda}{2} \exp \left(\frac{\lambda}{2} (2\mu^{c_2} + \lambda\sigma^2 - 2) \operatorname{erfc} \left(\frac{\mu^{c_2} + \lambda\sigma^2 - 1}{\sqrt{2}\sigma} \right) \right) \quad (\text{D-2})$$

Appendix E. Furnace-superheater interaction factor

The superheater-furnace interaction factor (β) is an empirical constant shown in Figure E.1.

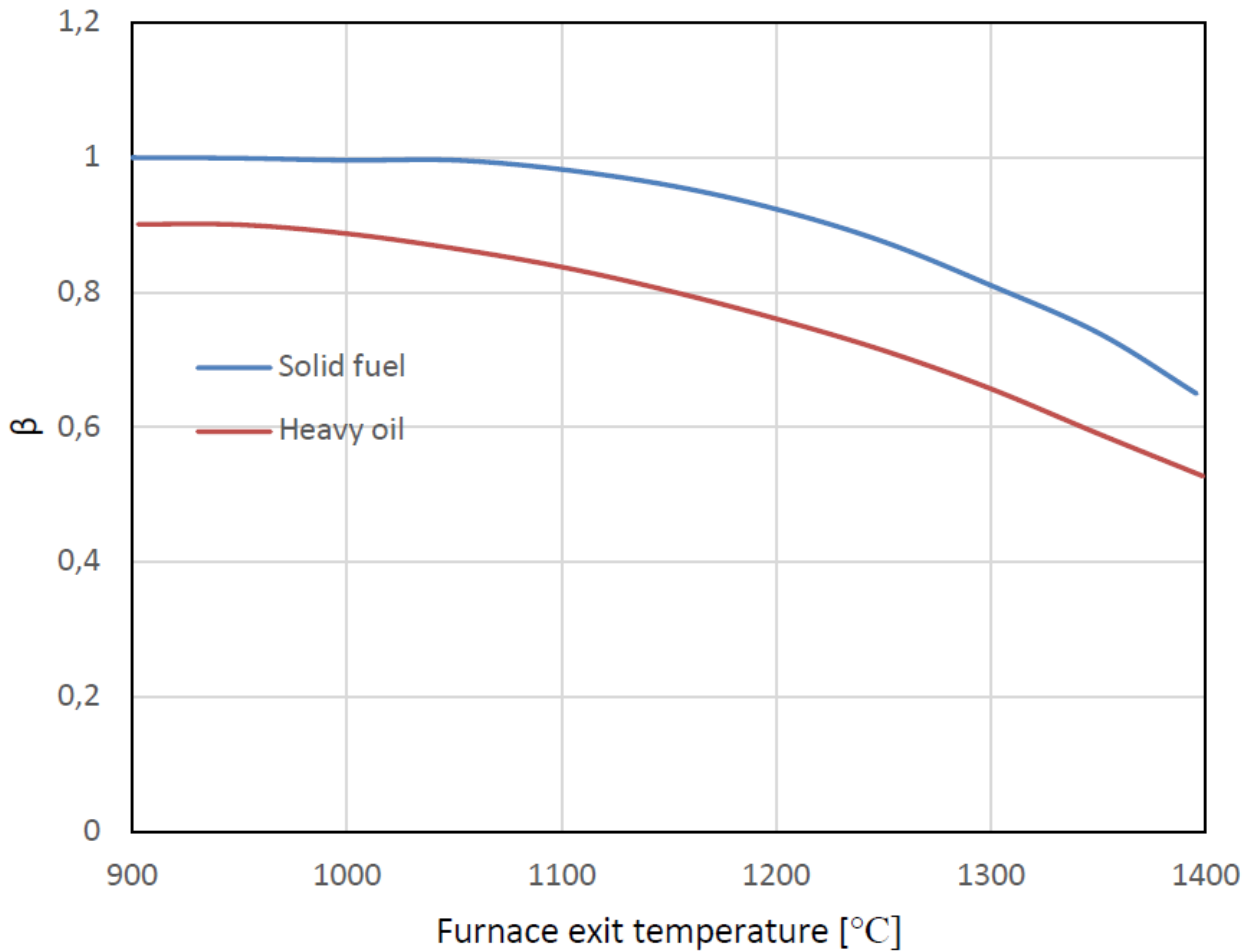


Figure E.1 Superheater-furnace interaction factor [30], [31]

The curves in Figure E.1 were fitted by Laubscher et al. [34] using the polynomial assumption, and they are expressed as follows:

$$\beta = -2.18793 \times 10^{-6} \cdot T^2 + 4.36338 \times 10^{-3} \cdot T - 1.16738 \text{ for solid fuel} \quad (\text{E-1})$$

$$\beta = -1.43393 \times 10^{-6} \cdot T^2 + 2.53702 \times 10^{-3} \cdot T - 0.217727 \text{ for heavy oil}$$

where T in Equation (E-1) is the furnace exit temperature of the products.

Appendix F. Tube bank heat exchanger geometry

The heat exchangers in the case study boiler, such as superheater, reheater and economiser, are ducted tube banks with the arrangement shown in Figure F.1.

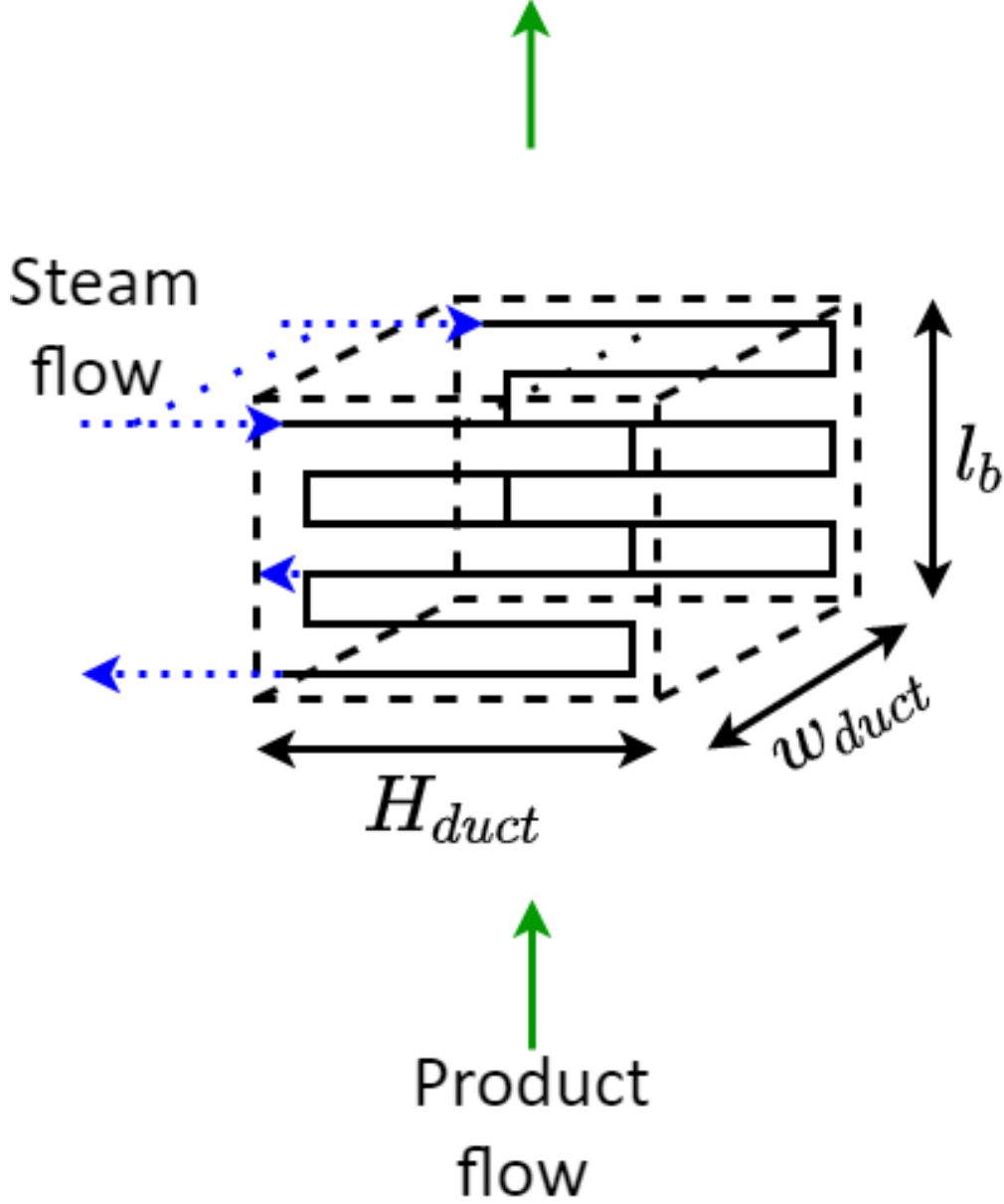


Figure F.1 Tube bank heat exchanger layout

As shown in Figure F.1, the products stream flow through the heat exchanger with an inlet flow area equal to the open duct area ($A_{duct,open}$), which can be calculated using the depth (H_{duct}) and the width of the duct (w_{duct}):

$$A_{duct,open} = H_{duct}w_{duct} \quad (F-1)$$

Additionally, the duct walls contain a vertical tube bank (vertical water wall), which also receives heat from the products stream. The heat transfer area of this tube bank ($A_{vertical\ WW}$) can be assumed as the same as the internal surface area of the duct wall, which can be calculated using the length (l_b), depth and the width of the duct:

$$A_{vertical\ WW} = 2l_b(H_{duct} + w_{duct}) \quad (F-2)$$

For the products flowing in the heat exchanger, its corresponding one-dimensional control volume can be represented using the inlet flow area, control volume length ($l_{products}$) and the wetted perimeter (P_{wet}).

The length of the control volume can be determined using the inlet flow area and the volume of the products in the heat exchanger ($V_{products}$):

$$l_{products} = \frac{V_{products}}{A_{duct,open}} \quad (F-3)$$

where the volume of the products in the heat exchanger is the difference between the internal duct volume (V_{duct}) and the tube bank volume (V_{tubes}):

$$V_{products} = V_{duct} - V_{tubes} \quad (F-4)$$

The internal duct volume can be calculated using the open duct area and the duct length:

$$V_{duct} = A_{duct,open}l_b \quad (F-5)$$

Whereas the tube bank volume refers to the physical volume of the tubes, which can be calculated using the number of tubes in one pass (n_{fwt}) together with the outer diameter (d_o) and mean path length of each tube (l_{path}):

$$V_{tubes} = n_{fwt}l_{path}\left(\frac{\pi}{4}d_o^2\right) \quad (F-6)$$

The wetted perimeter (P_{wet}) of the control volume can be determined by assuming the outer tube diameter as the hydraulic diameter of the control volume. Based on this assumption, the wetted perimeter of the control volume can be calculated as follows:

$$P_{wet} = \frac{4A_{duct,open}}{d_o} \quad (F-7)$$

Each tube pass generally has multiple tubes in both transverse and longitudinal directions, with the arrangement shown in Figure F.2.

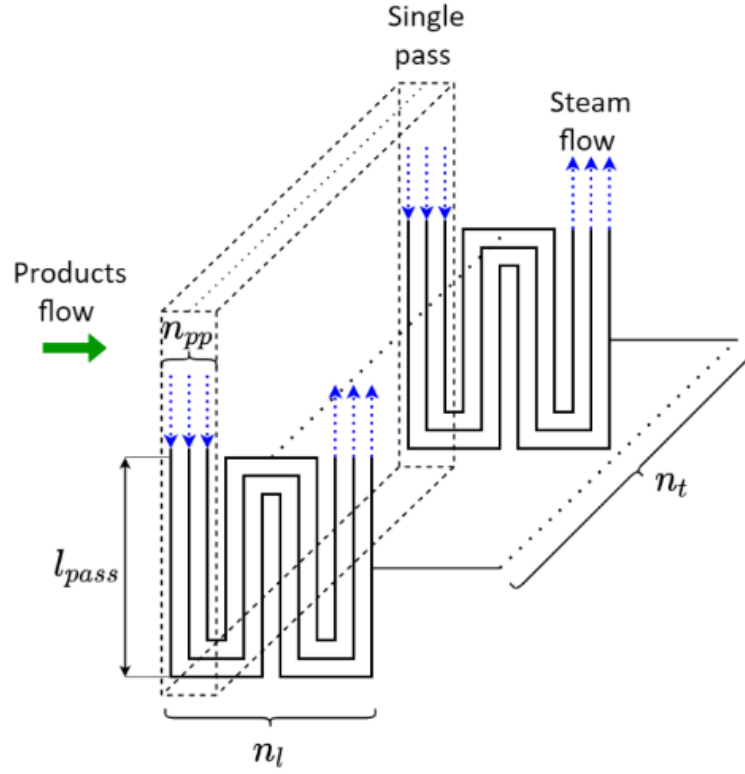


Figure F.2 Tube bank layout

In Figure F.2, n_{pp} is the number of tubes flowing in the parallel path, which can be determined using the number of tubes in the longitudinal direction (n_l) and the number of passes of each tube (n_{pass}):

$$n_{pp} = \frac{n_l}{n_{pass}} \quad (\text{F-8})$$

The overall mean path length of each tube can be calculated by multiplying the mean tube length of each pass (l_{pass}) by the number of passes of each tube:

$$l_{path} = n_{pass} l_{pass} \quad (\text{F-9})$$

The total number of tubes in one tube pass can be calculated using the number of passes of each tube and the number of tubes in the transverse direction:

$$n_{fwt} = \frac{n_{pp}}{2} (n_{t,odd} + n_{t,even}) \quad (\text{F-10})$$

where $n_{t,odd}$ is the number of tubes in the transverse direction for the odd rows and $n_{t,even}$ is the number of tubes in the transverse direction for the even rows.

In the case of in-line tube banks, the number of tubes in the transverse direction for both even and odd rows is the same. Whereas for the staggered tube banks, the numbers might be different between even and odd rows.

The duct length shown in Figure F.1 can also be determined from the tube bank geometry, which is shown in Figure F.3.

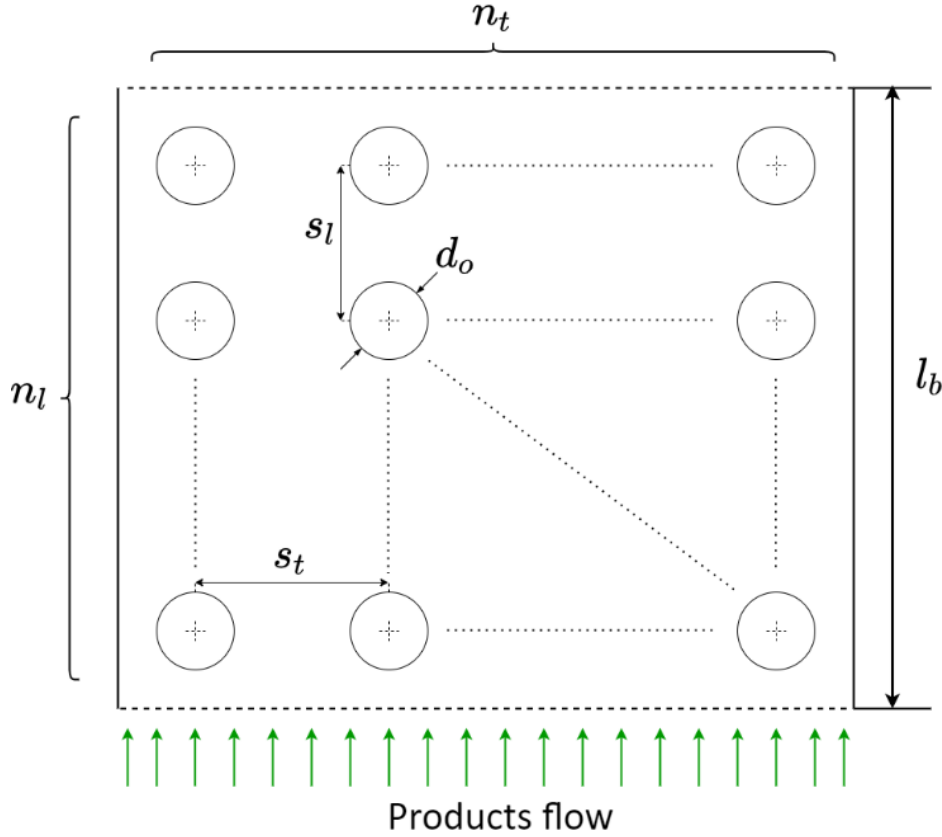


Figure F.3 Tube bank layout side view

As shown in Figure F.3, the duct length is related to the number of tubes in the longitudinal directions and the tube bank's longitudinal pitch (s_l):

$$l_b = s_l(n_l + 1) \quad (\text{F-11})$$

With the above-mentioned geometrical parameters of tube banks established, the overall external surface area of the tube banks (A_{ext}) can be calculated by multiplying the external surface area of each tube by the total number of tubes in one tube pass:

$$A_{ext} = n_{fwt} l_{path} (\pi d_o) \quad (\text{F-12})$$

where d_o is the outer diameter of the tube.

In the case of the platen superheater, the tubes are in close proximity to each other in the longitudinal direction; therefore, the overall external surface area of the tube banks can be approximated using the projected surface area shown in Figure F.4.

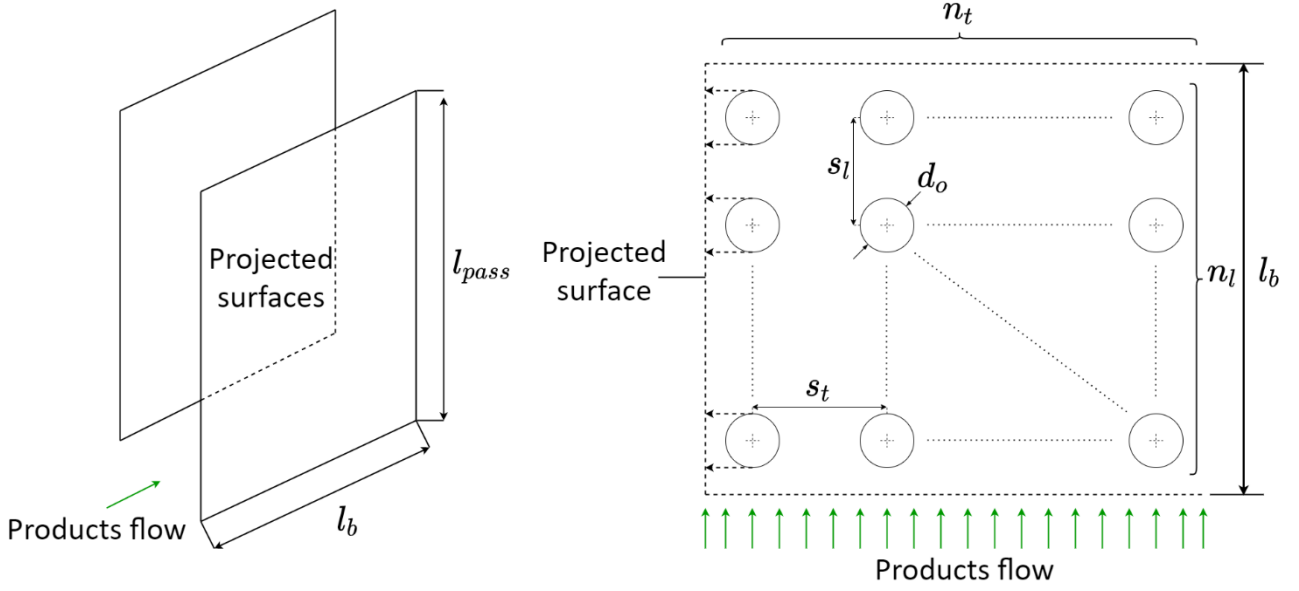


Figure F.4 Platen superheater projected surface

As shown in Figure F.4, the projected surfaces are imagined as sheets of heat transfer areas, in which each column of tubes in the transverse direction has two projected surfaces due to symmetry. For that reason, the projected surface area can be calculated by multiplying the area of each sheet by the total number of sheets:

$$A_{ext} = 2n_t l_{pass} l_b \quad (\text{F-13})$$

where n_t refers to the number of tubes in the transverse direction.

The equation for calculating the overall internal surface area of the tube banks (A_{int}) is similar to Equation (F-12), except the tube inner diameter (d_i) is used instead of the outer diameter:

$$A_{int} = n_{fwt} l_{path} (\pi d_i) \quad (\text{F-14})$$

where the inner diameter of the tube can be calculated by subtracting the tube wall thickness (t_{wall}) from the outer tube diameter:

$$d_i = d_o - 2t_{wall} \quad (\text{F-15})$$

Appendix G. Sling tube geometry

In the case study boiler, the steam or mixture leaving the furnace branches to the vertical water wall and sling tubes, as shown in Figure G.1.

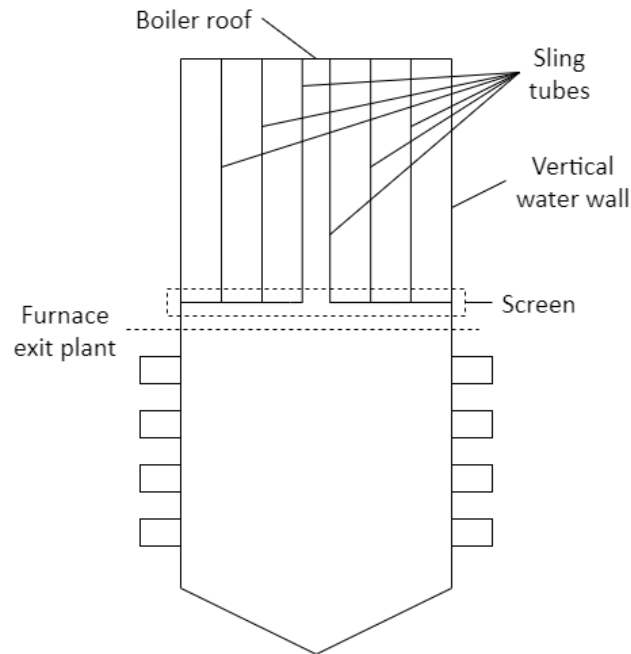


Figure G.1 Sling tube layout

As shown in Figure G.1, the sling tubes form a screen at the furnace exit with a layout shown in Figure G.2.

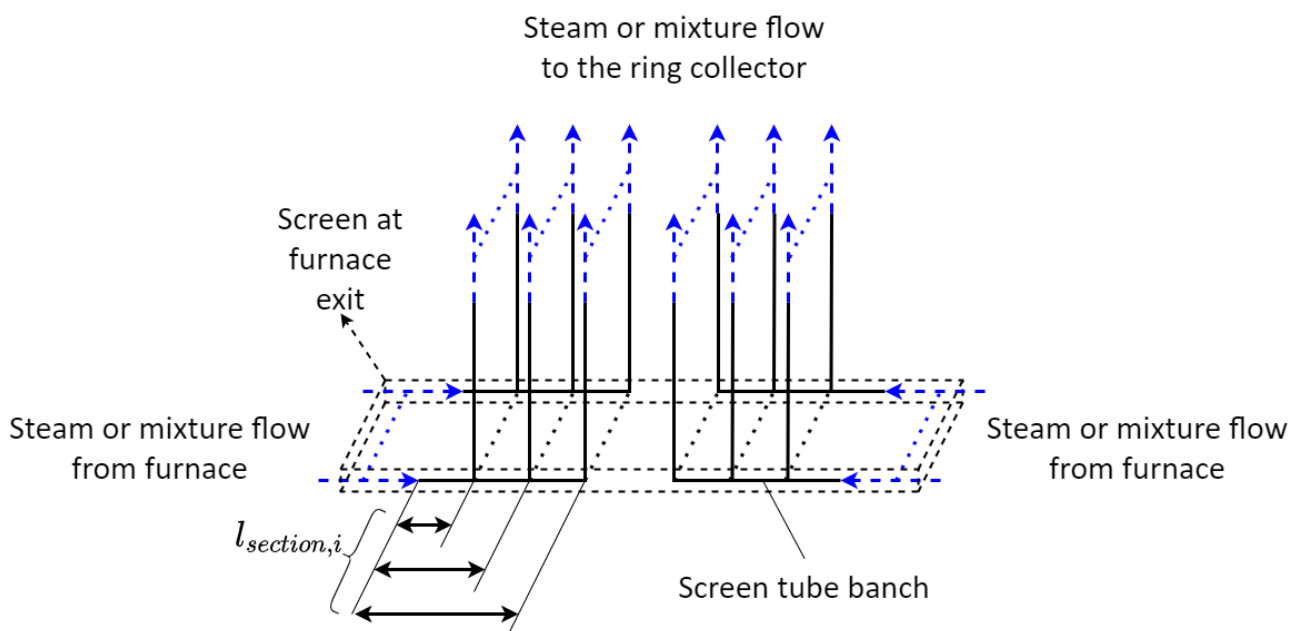


Figure G.2 Sling tube screen layout

As shown in Figure G.2, each sling tube branch in the screen contains multiple sections, and the length of each section is denoted as $l_{section,i}$.

Although the steam or mixture in each section of the branch flows up towards the boiler roof at a different point, the control volume of each branch can still be modelled as a single tube, as shown in Figure G.3.

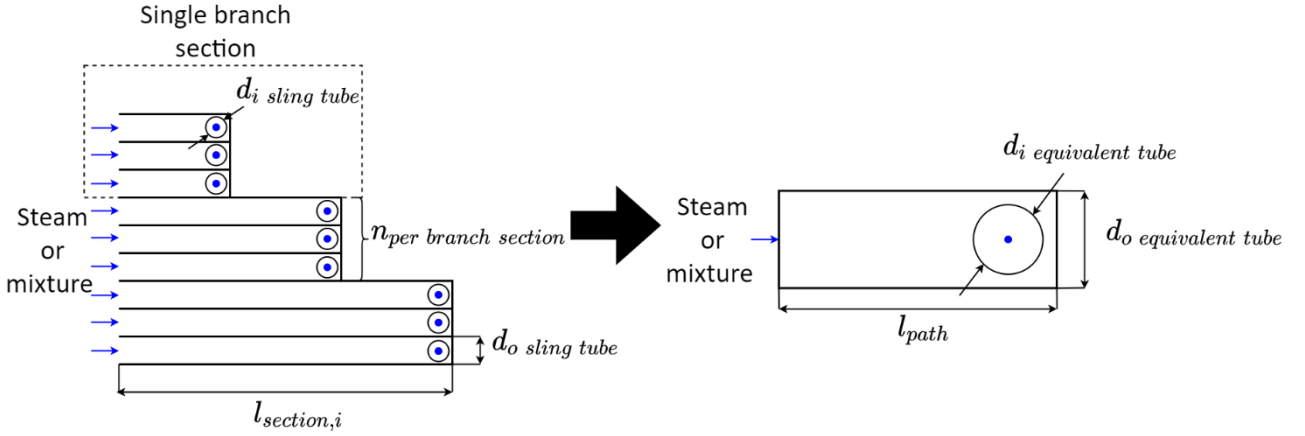


Figure G.3 Screen branch control volume conversion

In Figure G.3, the converted equivalent tube control volume (on the right-hand side) has a length (l_{path}) equal to the longest section of the branch. The volume of the equivalent tube is the same as the volume of each branch (on the left-hand side), which is expressed by the following equation:

$$\frac{\pi d_{o \text{ equivalent tube}}^2 l_{path}}{4} = \frac{\pi d_{o \text{ sling tube}}^2 l_{total}}{4} \quad (G-1)$$

where the term $d_{o \text{ sling tube}}$ refers to the outer diameter of the sling tube and $d_{o \text{ equivalent tube}}$ refers to the outer diameter of the equivalent tube.

The total branch tube length (l_{total}) can be calculated by summing the length of all the tubes in the branch, which is given by the following equation:

$$l_{total} = \sum_{i=1}^{n_{sections}} l_{section,i} n_{per \text{ branch section}} \quad (G-2)$$

where $n_{sections}$ refers to the number of sections in the branch and $n_{per \text{ branch section}}$ refers to the number of tubes in each section.

By rearranging Equation (G-1), the relationship between the outer diameter of the equivalent tube and the sling tube can be expressed using the following equation:

$$d_{o \text{ equivalent tube}} = d_{o \text{ sling tube}} \sqrt{\frac{l_{total}}{l_{path}}} \quad (G-3)$$

The inner diameter of the equivalent tube ($d_{i \text{ equivalent tube}}$) and the sling tube ($d_{i \text{ sling tube}}$) can also be linked using the same relationship, which can be expressed using the following equation:

$$d_{i \text{ equivalent tube}} = d_{i \text{ sling tube}} \sqrt{\frac{l_{total}}{l_{path}}} \quad (G-4)$$

where the inner diameter of the sling tube can be calculated using the outer diameter and the thickness ($t_{sling tube}$) of the sling tube:

$$d_{i \text{ sling tube}} = d_{o \text{ equivalent tube}} - 2t_{sling tube} \quad (G-5)$$

With both the inner and outer diameter of the equivalent tube known, the thickness of the equivalent tube ($t_{equivalent tube}$) can then be calculated using the following equation:

$$t_{equivalent tube} = \frac{d_{o \text{ equivalent tube}} - d_{i \text{ equivalent tube}}}{2} \quad (G-6)$$

By applying the conversion mentioned above, the sling tube screen can be treated as a tube bank heat exchanger with a single row of tubes in the longitudinal (flow) direction, as shown in Figure G.4.

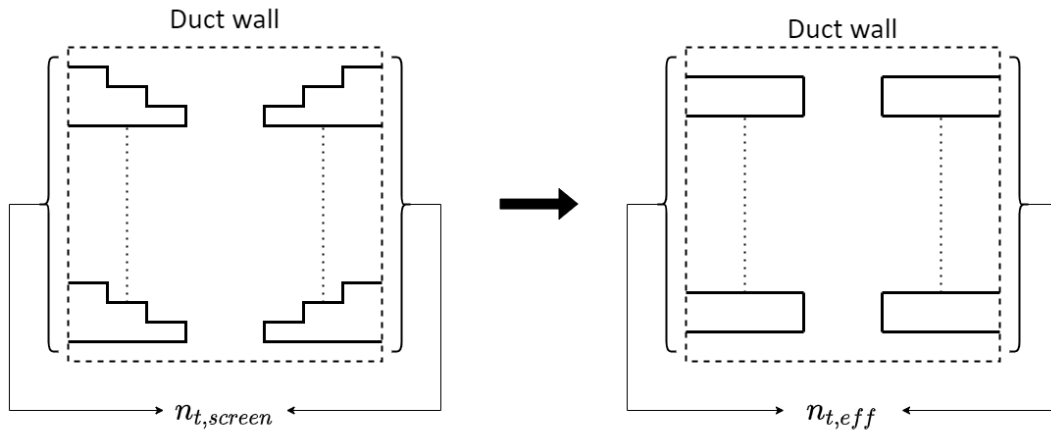


Figure G.4 Sling tube screen conversion

As shown in Figure G.4, the number of equivalent tubes in the transverse direction ($n_{t,eff}$) is equal to the number of screen branches, which can be determined using the total number of sling tubes in the transverse direction ($n_{t,screen}$), the number of sections in the branch and number of tubes in each section:

$$n_{t,eff} = \frac{n_{t,screen}}{n_{sections} n_{per\ branch\ section}} \quad (G-7)$$

By using the geometric parameters mentioned before, the external surface area of the screen (A_{ext}) can be calculated using the following equation:

$$A_{ext} = \pi d_{o\ sling\ tube} l_{total} n_{t,eff} \quad (G-8)$$

The equation for calculating the internal surface area of the screen (A_{int}) is similar to Equation (G-8), except the inner tube diameter is used instead of the outer diameter:

$$A_{int} = \pi d_{i\ sling\ tube} l_{total} n_{t,eff} \quad (G-9)$$

As shown in Figure G.1, the vertical sections of the sling tubes are situated in the heat exchanger ducts with a structure shown in Figure G.5.

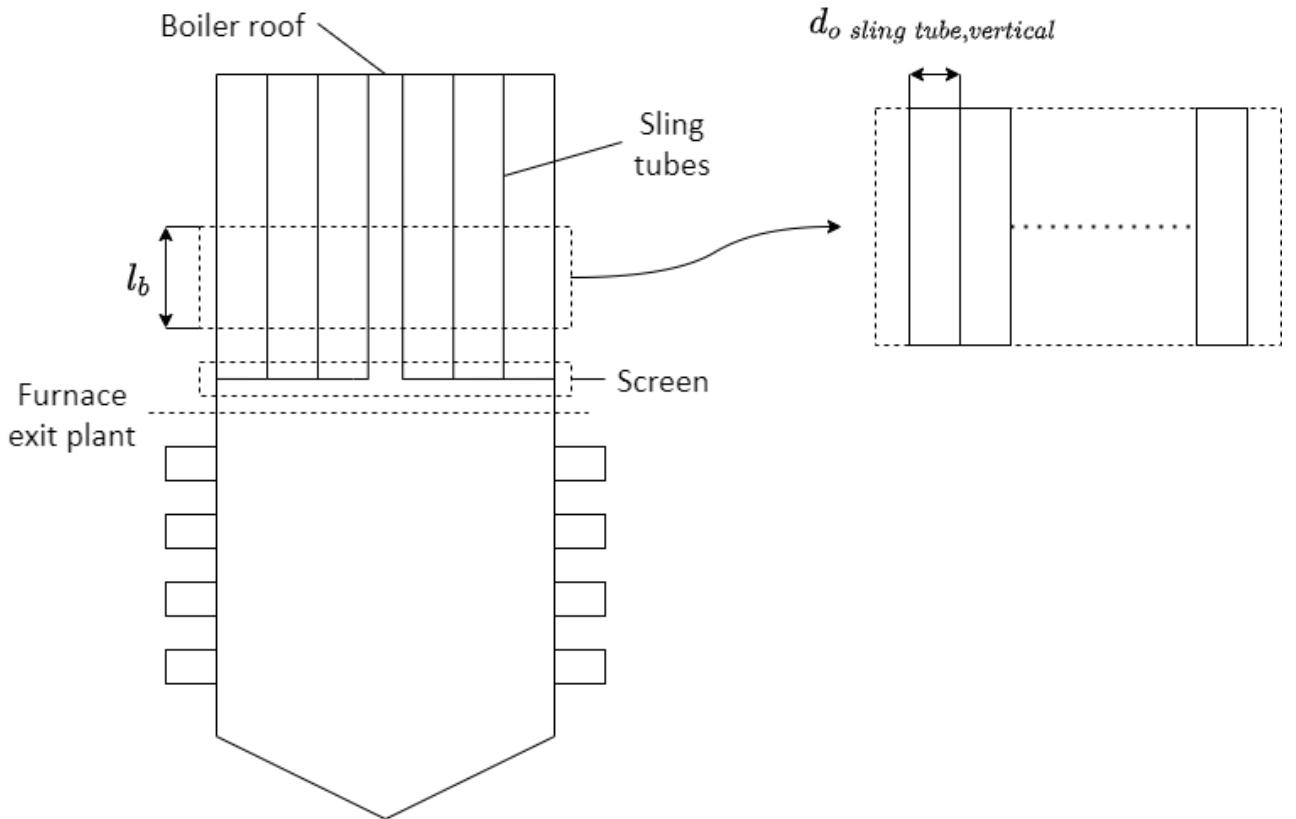


Figure G.5 vertical sling tube layout

As shown in Figure G.5, the external surface area of the vertical sling tubes ($A_{sling\ tubes}$) can be calculated using the diameters of the outer diameter of the vertical sling tube ($d_{o\ sling\ tube,vertical}$), the heat exchanger duct length (l_b) and the total number of sling tubes:

$$A_{sling\ tubes} = \pi d_{o\ sling\ tube,vertical} l_b n_{t,screen} \quad (G-10)$$

Appendix H. Fin efficiency

In the case study boiler, finned tubes are used in the economiser with a layout shown in Figure H.1.

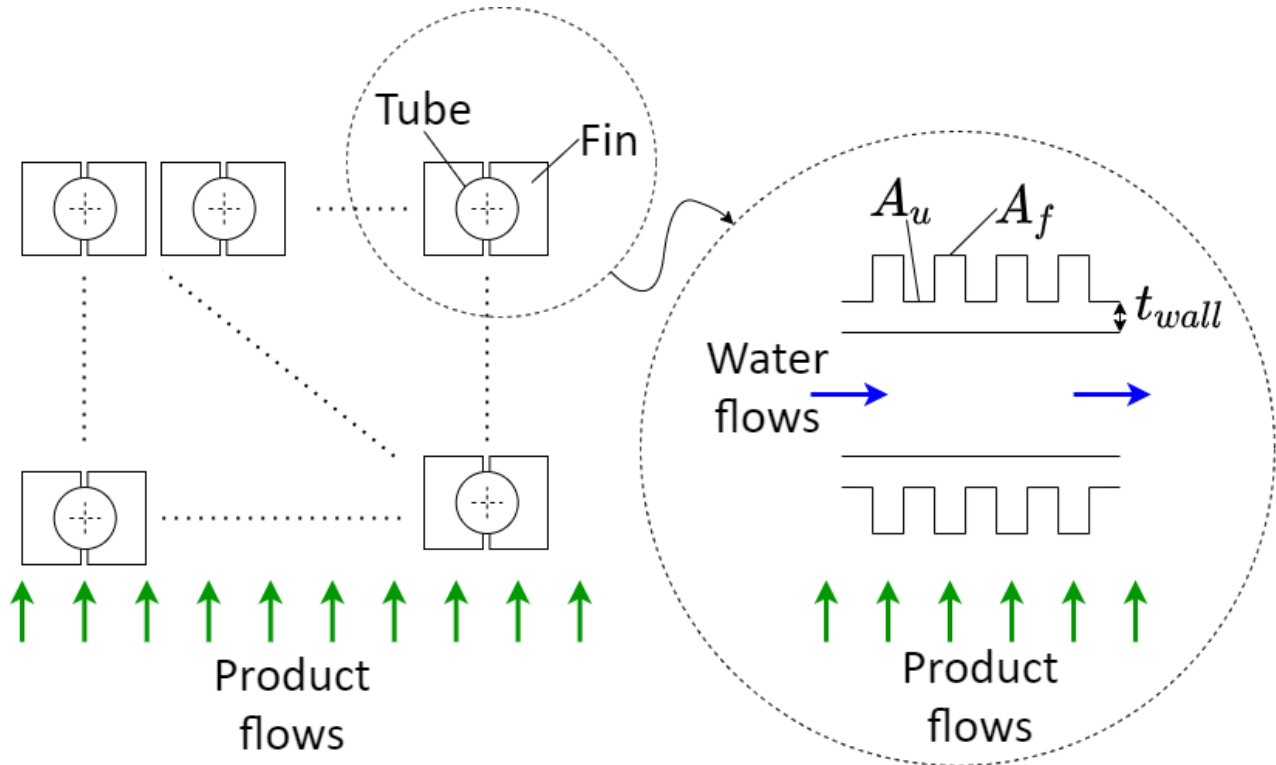


Figure H.1 Finned tube layout

In Figure H.1, the term A_f refers to the total fin area and A_u refers to the total unfinned tube area.

The total fin area can be calculated by analysing the geometry of the fins, as shown in Figure H.2.

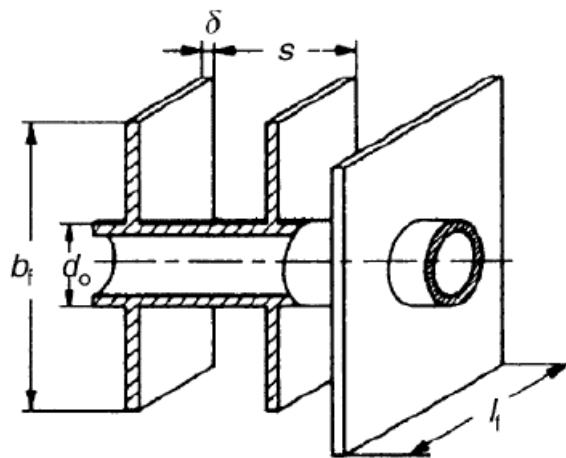


Figure H.2 Rectangular fin schematics from Schmidt [51]

As shown in Figure H.2, the total fin area can be calculated by summing the surface area of the outer edges ($A_{f,edge}$) and the faces ($A_{f,face}$) of all the fins:

$$A_f = A_{f,edge} + A_{f,face} \quad (H-1)$$

The surface area of the outer edges can be calculated using the length (l_f), width (b_f) and thickness (δ) of each fin together with the total number of fins (n_{fin}):

$$A_{f,rim} = 2n_{fin}\delta(l_f + b_f) \quad (H-2)$$

Whereas the surface area of the faces can be calculated by subtracting the tubes' cross-sectional area from the area of the fins' rectangular faces:

$$A_{f,face} = 2n_{fin}(2l_fb_f - \left(\frac{\pi d_o^2}{4}\right)) \quad (H-3)$$

The total number of fins in the tube bank can be calculated using the number of fins ($n_{fin\ per\ pass}$) and the number of tubes in one tube pass (n_{fwt} , see Appendix F for details) together with the number of tube passes (n_{pass} , see Appendix F for details):

$$n_{fin} = n_{pass}n_{fin\ per\ pass}n_{fwt} \quad (H-4)$$

where the number of fins in each tube pass can be calculated using the mean tube length of each pass (l_{pass} , see Appendix F for details) and the pitch of the fins (s):

$$n_{fin\ per\ pass} = round\left(\frac{l_{pass}}{s}\right) + 1 \quad (H-5)$$

The unfinned tube area shown in Figure H.1 can be calculated by subtracting the area blocked by the fins from the overall external surface area of the tube banks (A_{ext} , see Appendix F for details):

$$A_u = A_{ext} - \pi d_o \delta n_{fin} \quad (H-6)$$

where d_o refers to the outer diameter of the tube.

With the total fin area and unfinned tube area known, the overall external surface area of the finned tube banks ($A_{ext,f}$) can be calculated by summing the two areas:

$$A_{ext,f} = A_u + A_f \quad (H-7)$$

For the finned tube heat exchangers, the increase in heat transfer due to the fin surfaces is described using the fin efficiency (η_f), discussed in the work of Schmidt [51]. The correlation used to determine the fin efficiency is dependent on the fin geometry, as shown in the following equation:

$$\eta_f = \frac{\tan (X)}{X} \quad (\text{H-8})$$

In Equation (H-8), the term X is a variable related to the thermal properties and the geometry of the fin, which can be expressed using the following equation:

$$X = \phi \frac{d_o}{2} \sqrt{\frac{2h_{ext}}{k_f \delta}} \quad (\text{H-9})$$

where h_{ext} refers to the external heat transfer coefficient of the tube bank and k_f refers to the thermal conductivity of the fin material.

The term ϕ in Equation (H-9) is a fin geometry related variable, and for rectangular fins, the variable can be described using the following expression:

$$\phi = (\phi' - 1)[1 + 0.35 \ln (\phi')] \quad (\text{H-10})$$

where the term ϕ' can be calculated using the following equation:

$$\phi' = 1.28 \frac{b_f}{d_o} \sqrt{\left(\frac{l_f}{b_f} - 0.2\right)} \quad (\text{H-11})$$

Appendix I. Fluid property polynomials

For the boiler model developed in the project, the ideal gas enthalpies of each product constituent can be found in the NIST database. The enthalpies extracted from the database can be used to fit polynomials, as shown in the following equation:

$$h_{gas} = a_0 + a_1 T_{gas} + a_2 T_{gas}^2 + a_3 T_{gas}^3 + a_4 T_{gas}^4 \quad (I-1)$$

where h_{gas} refers to the ideal gas enthalpy of the gas species with the unit $\frac{kJ}{kg}$ and T_{gas} is the temperature of the gas with the unit K .

For the constituents of the products referenced from 25 degree Celsius, the polynomial coefficients (a_i) are presented in Table I.1.

Table I.1 Ideal gas enthalpy polynomial coefficients

Gas specie	a_0	a_1	a_2	a_3	a_4
Ash	-2.1765×10^2	7.3000×10^{-1}	0	0	0
CO ₂	-2.3232×10^2	6.5424×10^{-1}	4.5762×10^{-4}	-1.3346×10^{-7}	1.5198×10^{-11}
H ₂ O	1.8612×10^3	1.8895×10^0	1.5342×10^{-4}	1.6077×10^{-7}	-5.0689×10^{-11}
N ₂	-2.8297×10^2	8.9719×10^{-1}	1.8479×10^{-4}	-3.6656×10^{-8}	2.7798×10^{-12}
O ₂	-2.5727×10^2	8.0307×10^{-1}	2.1810×10^{-4}	-6.0835×10^{-8}	7.3371×10^{-12}
SO ₂	-1.7790×10^2	5.4154×10^{-1}	1.8500×10^{-4}	1.0116×10^{-19}	1.0588×10^{-22}
UC	-2.1169×10^2	7.1000×10^{-1}	0	0	0

For sulphur dioxide, the ideal gas fluid properties can be found in the NIST database, and the data can be used to fit polynomials, as shown in the following equation:

$$\phi_{SO_2} = b_0 + b_1 T_{SO_2} + b_2 T_{SO_2}^2 \quad (I-2)$$

where ϕ_{SO_2} refers to the fluid property and T_{SO_2} refers to the temperature of sulphur dioxide.

For sulphur dioxide referenced from 25 degrees Celsius, the polynomial coefficients (b_i) of each fluid property are presented in Table I.2.

Table I.2 Ideal gas fluid property polynomial coefficients of sulphur dioxide

Symbol	Fluid property	b_0	b_1	b_2
μ_{SO_2}	Dynamics viscosity	-1.3572×10^{-6}	4.8908×10^{-8}	-5.8856×10^{-12}
k_{SO_2}	Thermal conductivity	-3.3710×10^{-3}	4.3803×10^{-5}	3.1228×10^{-9}

Appendix J. Molar mass of ash

The molar mass of ash can be calculated by summing the weighted molar mass of its constituents, as presented in Table J.1.

Table J.1 Ash molar mass [52]

Ash constituent	Weight ratio (%)	Molar mass ($\frac{kg}{kmol}$)	Weighted molar mass ($\frac{kg}{kmol}$)
Aluminium Oxide (Al_2O_3)	20.91	102	21.33
Calcium Oxide (CaO)	12.54	56	7.02
Ferric Oxide (Fe_2O_3)	6.77	160	10.83
Potassium Oxide (K_2O)	1.49	62	0.92
Magnesium Oxide (MgO)	2.35	40	0.94
Sodium Oxide (Na_2O)	0.73	62	0.45
Phosphorus Pentaoxide (P_2O_5)	0.29	142	0.41
Silica (SiO_2)	46.15	60	27.69
Sulphur Trioxide (SO_3)	8.00	80	6.4
Titanium Oxide (TiO_2)	0.77	80	0.62
Total	100	N/A	76.62

Appendix K. Furnace membrane wall geometry

For the case study boiler, the furnace wall consists of membrane tubes with helix angles (α_{WW}) as shown in Figure K.1.

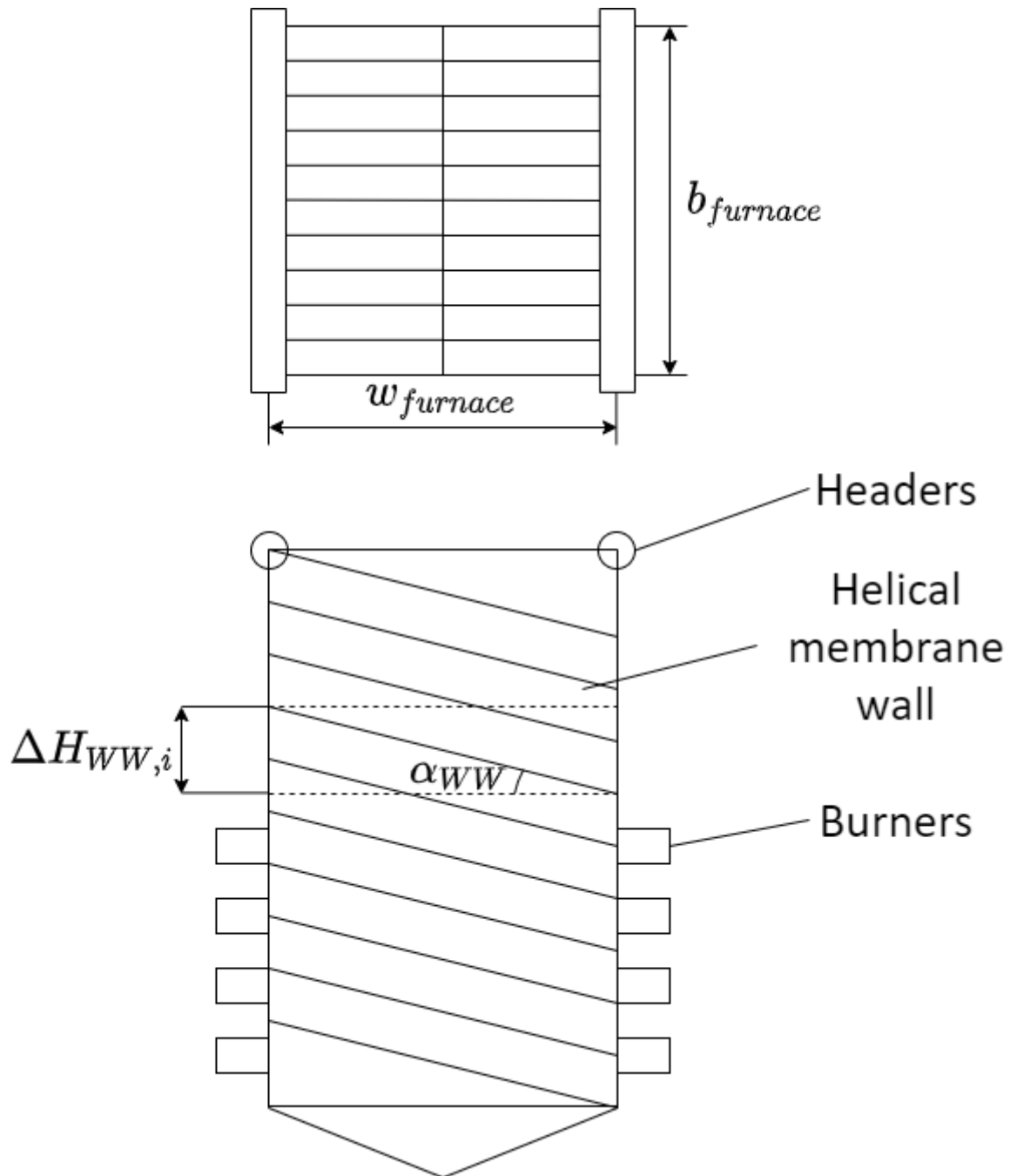


Figure K.1 Furnace membrane wall layout

In Figure K.1, the term $w_{furnace}$ refers to the width and $b_{furnace}$ refers to the depth of the furnace. The term $\Delta H_{WW,i}$ refers to the height of an arbitrary section of the furnace wall, which can be used to determine the geometric parameters of the membrane wall, as shown in Figure K.2.

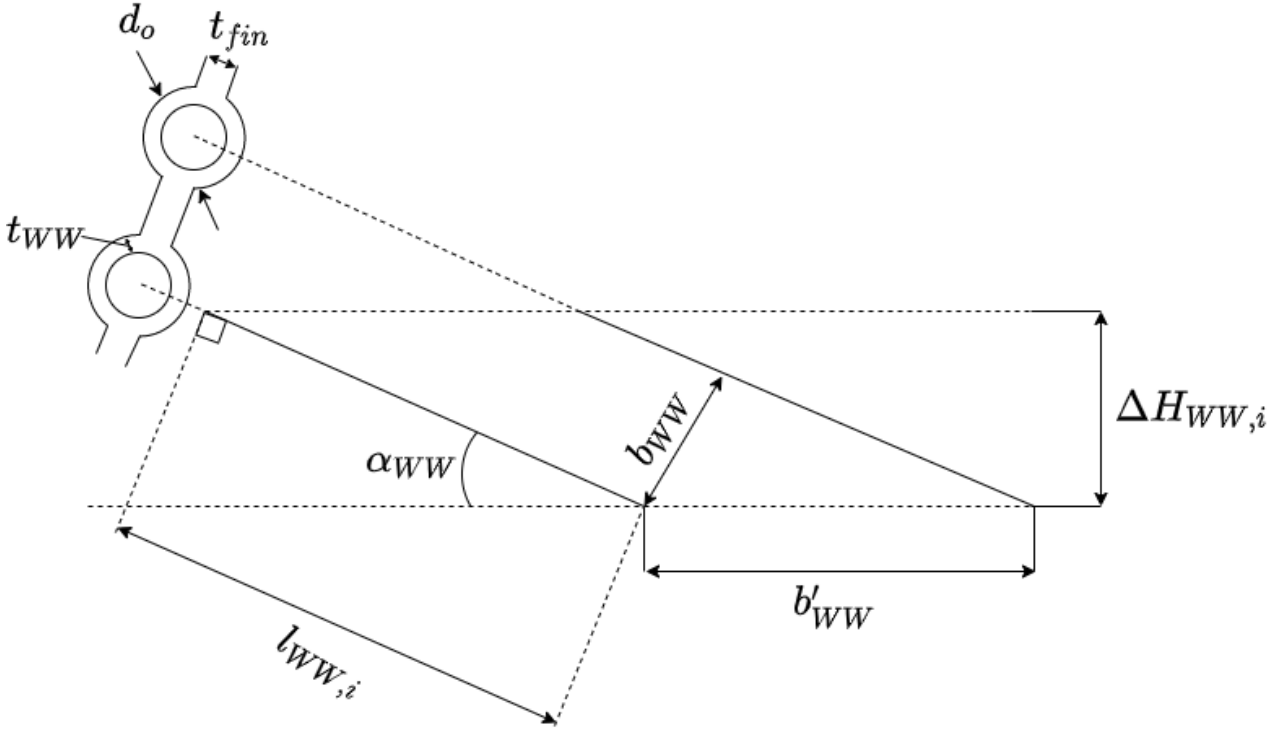


Figure K.2 Membrane wall geometry

In Figure K.2, the term b'_{WW} refers to the axial pitch of the tubes, which can be determined using the circumference of the furnace and the total number of tubes in the wall (n_{WW}):

$$b'_{WW} = \frac{2(w_{furnace} + b_{furnace})}{(n_{WW} - 1)} \quad (K-1)$$

With the axial pitch of the tubes known, the normal pitch of the tubes (b_{WW}) can be calculated using trigonometry:

$$b_{WW} = b'_{WW} \sin(\alpha_{WW}) \quad (K-2)$$

The length of the tubes in the section ($l_{WW,i}$) can also be calculated using trigonometry together with the height of the section:

$$l_{WW,i} = \frac{\Delta H_{WW,i}}{\sin(\alpha_{WW})} \quad (K-3)$$

Additionally, the outer diameter (d_o), tube thickness (t_{WW}) and the fin/membrane thickness (t_{fin}) are also shown in Figure K.2, which these parameters can be found in the design specifications.

Appendix L. Component model verification

Each Flownex component model developed in the project was verified against an equivalent process model using the programming language Python. The Python models use fluid properties from the open-source database CoolProp for both H_2O and flue gas/products streams.

L.1 Adiabatic flame

For the adiabatic flame model developed in Flownex, the combustion analysis is solved using a script component; therefore, the script's logic needs to be verified to ensure the combustion analysis calculation is executed correctly.

The inputs used in the verification exercise come from Laubscher and Rousseau's boiler specifications [34], which has a coal specification presented in Table L.1.

Table L.1 Coal specifications used for the adiabatic flame model verification

Symbol	Parameter	Unit	Case 1	Case 2
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	28777	16400
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.7080	0.4394
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0450	0.0250
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.0713	0.0961
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0072	0.0111
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0221	0.0100
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.0297	0.1051
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.1167	0.3133

The mass flow rate of the fuel and air is calculated using the mass and energy balance of the boiler's heat load, which is presented together with other inputs in Table L.2.

Table L.2 Inputs used for the adiabatic flame model verification

Symbol	Parameter	Unit	Case 1	Case 2
$\dot{Q}_{steam}$	Heat load	<i>MW</i>	277.6	1446
$C_{unaccount}$	Unaccounted heat loss	%	0.5	0.5
α_{air}	Excess air ratio	<i>N/A</i>	1.20	1.155
Y_{UC}	Carbon loss	$\frac{kg}{kg\ fuel}$	0.01	0.0035
p_{atm}	Ambient air pressure onsite	<i>kPa</i>	101.325	87.1
T_{fuel}	Fuel temperature at burner inlet	°C	90	90
$T_{air\ BNR\ in}$	Air temperature at burner inlet	°C	395.85	287.85
Y_{air,N_2}	Air N_2 content	$\frac{kg}{kg\ air}$	0.7543	0.7668
Y_{air,O_2}	Air O_2 content	$\frac{kg}{kg\ air}$	0.2279	0.2316
Y_{air,H_2O}	Air H_2O content	$\frac{kg}{kg\ air}$	0.0178	0.0016
$\dot{m}_{fuel}$	Fuel mass flow rate	$\frac{kg}{s}$	10.92	100.15
$\dot{m}_{air\ BNR\ in}$	Air mass flow rate	$\frac{kg}{s}$	124.64	627.36

The Flownex model results showed sufficient accuracy compared to the Python model results, as presented in Table L.3.

Table L.3 Adiabatic flame model verification results

Symbol	Parameter	Unit	Case 1		Case 2	
			Flownex	Python	Flownex	Python
T_{flame}	Flame temperature	°C	2336.25	2330.22	2186.10	2191.60
h_{flame}	Flame enthalpy	$\frac{kJ}{kg}$	2710.42	2718.10	2451.92	2475.61
Y_{CO_2}	Combustion products CO_2 content	$\frac{kg}{kg \text{ products}}$	0.2076	0.2076	0.2166	0.2166
Y_{H_2O}	Combustion products H_2O content	$\frac{kg}{kg \text{ products}}$	0.0514	0.0514	0.0466	0.0466
Y_{N_2}	Combustion products N_2 content	$\frac{kg}{kg \text{ products}}$	0.6937	0.6937	0.6628	0.6628
Y_{SO_2}	Combustion products SO_2 content	$\frac{kg}{kg \text{ products}}$	0.0036	0.0036	0.0027	0.0027
Y_{O_2}	Combustion products O_2 content	$\frac{kg}{kg \text{ products}}$	0.0334	0.0334	0.0268	0.0268
w_{ash}	Combustion products ash content	$\frac{kg}{kg \text{ products}}$	0.0095	0.0095	0.0431	0.0431
w_{UC}	Combustion products UC content	$\frac{kg}{kg \text{ products}}$	0.0008	0.0008	0.0014	0.0014

As shown in Table L.3, the enthalpy and flame temperature and enthalpy results differed slightly between the Flownex and Python models. This difference could be since the Python model uses a different fluid property database than the Flownex model. However, the results between the models are still close since the relative differences are all less than 1%.

L.2 Furnace

Similarly, the furnace heat transfer script is the crucial component verified to ensure the Gurvich method and the high ash loading models are executed correctly.

The furnace heat transfer script verification uses the results from the adiabatic flame model shown in Table L.3 and the furnace specifications shown in Table L.4.

Table L.4 Verification furnace geometry

Symbol	Parameter	Unit	Case 1	Case 2
$H_{furnace}$	Furnace height	m	26.283	50.669
$H_{r,eff}$	Flame height	m	7.11	14.79
$V_{furnace}$	Furnace volume	m^3	2554.5	15794.0
$A_{furnace}$	Planar furnace area	m^2	1187.3	4281.0
$A_{furnace,exit\ plane}$	Furnace exit plane area	m^2	106.09	600.40
$d_{o\ ww}$	Water wall tube outer diameter	mm	0.043	0.0762
b_{ww}	Water wall tube pitch	mm	0.1143	0.2222
N/A	Fuel type	N/A	Lignite	Lignite
N/A	Flame modification mode	N/A	High ash coal	High ash coal
N/A	Water all type	N/A	Panel	Membrane
N/A	High ash loading	N/A	No	Yes

The Flownex model results showed sufficient accuracy compared to the Python model results, as presented in Table L.5.

Table L.5 Furnace heat transfer script verification results

Symbol	Parameter	Unit	Case 1		Case 2	
			Flownex	Python	Flownex	Python
$T_{products,exit}$	Products exit temperature	°C	1152.48	1168.28	1410.17	1426.10
$\dot{Q}_{rad\ total}$	Total radiation	MW	166.46	162.42	487.48	480.99
$\dot{Q}_{rad\ to\ WW}$	Radiation to the water wall	MW	158.05	154.60	448.81	444.65
$\dot{Q}_{direct\ rad\ furnace\ exit}$	Radiation through exit plane	MW	8.41	7.82	38.67	36.34
$\dot{q}_{furnace}$	Radiation heat flux	$\frac{kW}{m^2}$	140.20	136.80	113.86	112.35

Again, Table L.5 show that the results differed slightly between the Flownex and Python models. Besides the fluid property database difference, the errors could also propagate through the results of the adiabatic flame model. However, the differences in results between the models are still in the acceptable range.

L.3 Gas-to-steam tube bank heat exchanger

For the tube bank heat exchanger model, the Flownex model is verified using a parametric study since this verification is done on the entire model rather than just the script component.

The parameters chosen for the parametric study are the temperatures and mass flow rates of the fluids at the inlet of the heat exchangers. These parameters are sampled using Latin hypercube sampling with the distribution shown in Table L.6 to Table L.9.

Table L.6 Heat exchanger model verification case 1 input parameter distributions

	$\dot{m}_{H_2O,in} (\frac{kg}{s})$	$\dot{m}_{products,in} (\frac{kg}{s})$	$T_{H_2O,in} (K)$	$T_{products,in} (K)$
Mean	116	135	668	1409
Standard deviation	20	20	20	30

Table L.7 Heat exchanger model verification case 2 input parameter distributions

	$\dot{m}_{H_2O,in} (\frac{kg}{s})$	$\dot{m}_{products,in} (\frac{kg}{s})$	$T_{H_2O,in} (K)$	$T_{products,in} (K)$
Mean	510	695	730	1370
Standard deviation	20	20	25	35

Table L.8 Heat exchanger model verification case 3 input parameter distributions

	$\dot{m}_{H_2O,in} (\frac{kg}{s})$	$\dot{m}_{products,in} (\frac{kg}{s})$	$T_{H_2O,in} (K)$	$T_{products,in} (K)$
Mean	482	694	552	717
Standard deviation	20	20	10	30

Table L.9 Heat exchanger model verification case 4 input parameter distributions

	$\dot{m}_{H_2O,in} (\frac{kg}{s})$	$\dot{m}_{products,in} (\frac{kg}{s})$	$T_{H_2O,in} (K)$	$T_{products,in} (K)$
Mean	482	694	552	717
Standard deviation	20	20	10	25

For each verification case, one hundred points are sampled, and the specifications of heat exchangers in each verification case are shown in Table L.10 to Table L.12 .

Table L.10 Heat exchanger specifications for the model verification

Symbol	Parameter	Unit	Case 1	Case 2	Case 3	Case 4
$Config$	Tube bundle configuration	N/A	Inline	Inline	Inline	Staggered
$H_{duct,in}$	Depth of duct inlet	m	8.408	16.608	15.083	15.083
$H_{duct,exit}$	Depth of duct exit	m	8.408	12.814	15.083	15.083
w_{duct}	Width of duct	m	9.841	24	24.001	24.001
l_R	Length of pre-cavity	m	0	3.61	0.917	0.917
l_b	Length of duct	m	2.343	4	1.834	1.834
d_o	Tube outer diameter	mm	42	47.75	38	38
t_{wall}	Tube wall thickness	mm	5	9.7	3.8	3.8
n_l	Number of longitudinal tubes	N/A	40	68	28	28
$n_{t,odd}$	Number of transverse tubes in odd row	N/A	14	28	140	139
$n_{t,even}$	Number of transverse tubes in even row	N/A	14	28	140	140
n_{pass}	Number of passes	N/A	10	2	14	14
s_l	Tube bundle longitudinal pitch	mm	5.9	59	66.5	66.5
s_t	Tube bundle transverse pitch	mm	65.91	80	77.8	77.8
l_{pass}	Mean tube length of each pass	m	8.408	14.874	105.582	105.582
$R_{slagging,ext}$	External slagging resistance	$\frac{m^2 K}{W}$	0.0097	N/A	N/A	N/A
$R_{fouling,ext}$	External fouling resistance	$\frac{m^2 K}{W}$	N/A	0.0043	0.0043	0.0043

Table L.11 Products specifications for the heat exchanger model verification

Symbol	Parameter	Unit	Case 1	Case 2	Case 3	Case 4
Y_{CO_2}	Combustion products CO_2 content	$\frac{kg}{kg \text{ products}}$	0.2061	0.2178	0.2178	0.2178
Y_{H_2O}	Combustion products H_2O content	$\frac{kg}{kg \text{ products}}$	0.0515	0.0461	0.0461	0.0461
Y_{N_2}	Combustion products N_2 content	$\frac{kg}{kg \text{ products}}$	0.6948	0.6660	0.6660	0.6660
Y_{SO_2}	Combustion products SO_2 content	$\frac{kg}{kg \text{ products}}$	0.0036	0.0027	0.0027	0.0027
Y_{O_2}	Combustion products O_2 content	$\frac{kg}{kg \text{ products}}$	0.0347	0.0269	0.0269	0.0269
w_{ash}	Combustion products ash content	$\frac{kg}{kg \text{ products}}$	0.0085	0.0401	0.0401	0.0401
w_{UC}	Combustion products UC content	$\frac{kg}{kg \text{ products}}$	0.0008	0.0005	0.0005	0.0005

Table L.12 Pressure specifications for the heat exchanger model verification

Symbol	Parameter	Unit	Case 1	Case 2	Case 3	Case 4
$p_{H_2O,in}$	Inlet water or steam pressure	MPa	14.1	17.88	19.01	19.01
$p_{products,in}$	Inlet products pressure	kPa	95	87.1	87.1	87.1

For the first case study, the temperature, pressure and heat transfer results in Figure L.1 and Figure L.2 indicate that the two models' results agree on the systematic level.

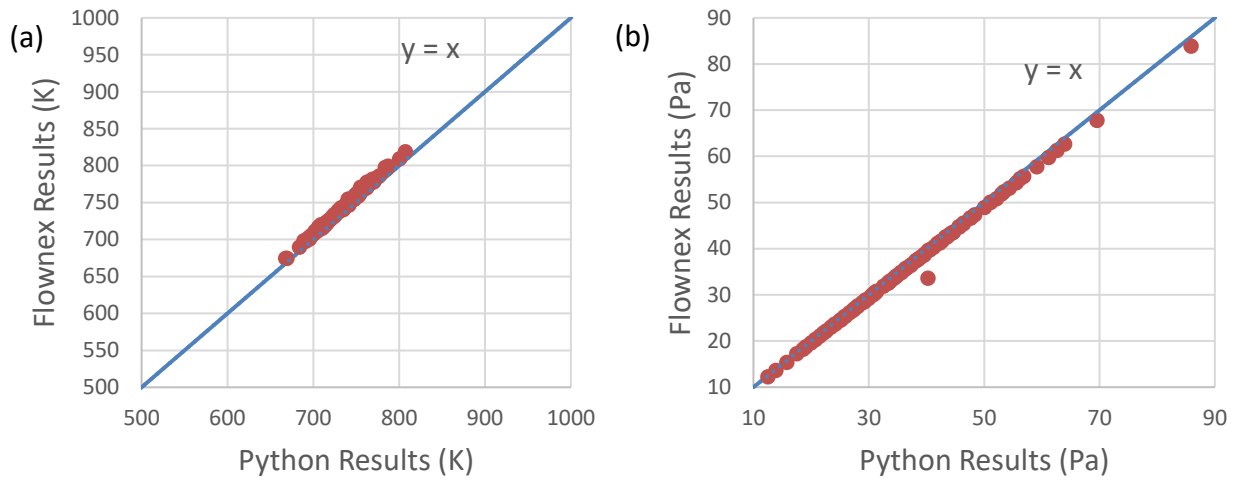


Figure L.1 Heat exchanger model verification case 1 (a) steam temperature (b) products pressure loss results

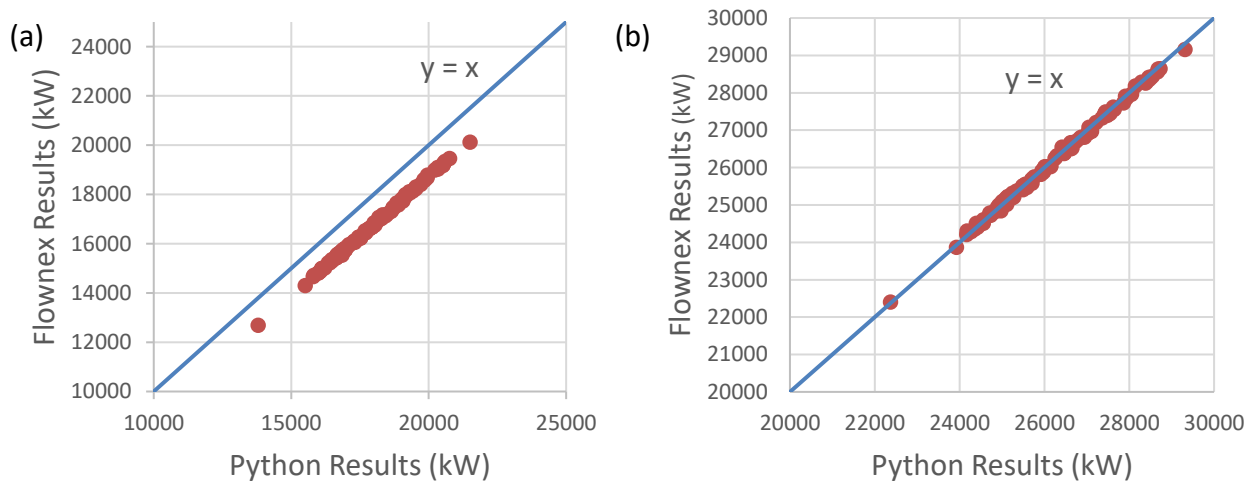


Figure L.2 Heat exchanger model verification case 1 (a) products (b) steam heat transfer results

The indication for the systematic agreement between the two models is that the gradient of the graphs in Figure L.1 and Figure L.2 follows the benchmark trend ($y = x$), indicating the conherency of the models.

Additionally, the error found in Figure L.1 (a) could be due to the same fluid property data difference.

The systematic agreement between the models can be seen for all four verification cases, and the error results in Table show that the Flownex results are sufficiently accurate.

Table L.13 Heat exchanger model verification relative error L2 norms

	$T_{H_2O,exit}$	$T_{products,exit}$	$\Delta p_{products}$	$\dot{Q}_{products}$	$\dot{Q}_{H_2O}$
Case 1	6.81%	0.27%	1.40%	1.24%	2.67%
Case 2	2.73%	2.49%	0.46%	0.37%	0.13%
Case 3	2.43%	2.07%	0.30%	0.08%	0.53%
Case 4	2.43%	2.07%	0.30%	0.08%	0.53%

Since the tubular air heater and the sling tube screen are also modelled using the same modelling methodology; therefore, additional verification of those component models are not verified.

L.4 Rotary air heater

Lastly, the rotary air heater model script component is verified using the rotary air heat sample calculation presented by Laubscher and Rousseau [34]. Furthermore, the verification case study rotary air heater has specifications shown in Table L.14 to Table L.16.

Table L.14 Rotary air heater verification inputs

Symbol	Parameter	Unit	Value
$\dot{m}_{air,in}$	Inlet air mass flow rate	$\frac{kg}{s}$	626.766
$\dot{m}_{products,in}$	Inlet products mass flow rate	$\frac{kg}{s}$	705.049
$T_{air,in}$	Inlet air temperature	°C	38.35
$T_{products,in}$	Inlet products temperature	°C	335.50
$p_{air,in}$	Inlet air pressure	kPa	104.325
$p_{products,in}$	Inlet products pressure	kPa	87.1

Table L.15 Rotary air heater verification gas specifications

Symbol	Parameter	Unit	Air	Products
Y_{CO_2}	CO_2 content	$\frac{kg}{kg \text{ total gas}}$	0	0.2154
Y_{H_2O}	H_2O content	$\frac{kg}{kg \text{ total gas}}$	0.0016	0.0452
Y_{N_2}	N_2 content	$\frac{kg}{kg \text{ total gas}}$	0.7664	0.6684
Y_{SO_2}	SO_2 content	$\frac{kg}{kg \text{ total gas}}$	0	0.0027
Y_{O_2}	O_2 content	$\frac{kg}{kg \text{ total gas}}$	0.2320	0.0298
w_{ash}	Ash content	$\frac{kg}{kg \text{ total gas}}$	0	0.0380
w_{UC}	UC content	$\frac{kg}{kg \text{ total gas}}$	0	0.0005

Table L.16 Verification rotary air heater geometry

Symbol	Parameter	Unit	Value
l_{rot}	Rotary air heater length	m	5
$A_{flow,air}$	Air flow area	m^2	47.79
$A_{flow,products}$	Products flow area	m^2	52.14
A_{HT}	Total heat transfer area	m^2	32945
$D_{h,rot}$	Effective hydraulic diameter	mm	15.166
n_{comp}	Number of compartments in each radial plane	N/A	1000
$D_{duct,0}$	Rotary air heater inner radius	m	0.62
$D_{duct,N}$	Rotary air heater outer radius	m	6.2
$A_{flow,gaps}$	Flow area of the gaps	m^2	0.1
$L_{c,0}$	Width of the compartment closest to the centre	mm	9.73
$L_{c,N}$	Width of the compartment furthest to the centre	mm	9.73

The Flownex model results show sufficient accuracy, as shown in Table.

Table L.17 Rotary air heater verification results

Symbol	Parameter	Unit	Flownex	Sample calc
$\dot{m}_{air,exit}$	Inlet air mass flow rate	$\frac{kg}{s}$	607.837	608.513
$\dot{m}_{products,exit}$	Inlet products mass flow rate	$\frac{kg}{s}$	723.978	723.302
$T_{air,exit}$	Inlet air temperature	$^{\circ}C$	141.52	141.85
$T_{products,exit}$	Inlet products temperature	$^{\circ}C$	251.42	246.59
$\dot{Q}_{air}$	Air side heat transfer	MW	62.79	66.44

Appendix M. Designer specification boundary conditions and inputs

M.1 Fuel specification

Table M.1 Coal specification from designer specifications

Symbol	Parameter	Unit	Value
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	20000
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4842
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0288
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.0762
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0128
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0075
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.1011
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.2894

M.2 Boiler model inputs and boundary conditions

Table M.2 Mass inputs and boundary conditions from designer specifications

Symbol	Parameter	Unit	100% MCR	70% MCR
$\dot{m}_{fuel}$	Infeed fuel mass flow rate	$\frac{kg}{s}$	95.23	69.01
$\dot{m}_{SA}$	Secondary air mass flow rate	$\frac{kg}{s}$	618.46	471.45
$\dot{m}_{PA}$	Primary air mass flow rate	$\frac{kg}{s}$	149.03	109.25
$\dot{m}_{SH3\ exit}$	Superheater 3 exit steam mass flow rate	$\frac{kg}{s}$	577.00	403.90
$\dot{m}_{RH2\ exit}$	Reheater 2 exit steam mass flow rate	$\frac{kg}{s}$	519.07	360.04
$\dot{m}_{attemp\ 1}$	Attemperator 1 spray water mass flow rate	$\frac{kg}{s}$	6.06	4.24
$\dot{m}_{attemp\ 2}$	Attemperator 2 spray water mass flow rate	$\frac{kg}{s}$	20.20	14.14
$\dot{m}_{attemp\ 3}$	Attemperator 3 spray water mass flow rate	$\frac{kg}{s}$	14.40	9.90
$\dot{m}_{RH\ attemp}$	Reheat attemperator spray water mass flow rate	$\frac{kg}{s}$	7.10	1.64

Table M.3 Pressure inputs and boundary conditions from designer specifications

Symbol	Parameter	Unit	100% MCR	70% MCR
p_{FW}	Final feedwater heater pressure	MPa	21.60	16.18
$p_{RH1\ in}$	Reheater 1 Inlet pressure	MPa	4.15	2.38
$p_{FG\ AH\ exit}$	Flue gas pressure at air heater exit	kPa	79.52	80.06
$p_{PA\ AH\ exit}$	Primary air pressure at air heater exit	kPa	90.13	89.43

Table M.4 Temperature inputs and boundary conditions from designer specifications

Symbol	Parameter	Unit	100% MCR	70% MCR
T_{FW}	Final feedwater heater temperature	°C	248.00	227.00
$T_{RH1\ in}$	Reheater 1 Inlet temperature	°C	332.00	333.00
$T_{RH1\ attemp}$	Reheat attemperator spray water temperature	°C	180.00	180.00
$T_{SA\ in}$	Secondary air temperature at air heater inlet	°C	43.00	42.00
$T_{PA\ in}$	Primary air temperature at air heater inlet	°C	48.00	47.00
$T_{PA\ BNR\ in}$	Primary air temperature at burner inlet	°C	90.00	90.00

Table M.5 Other inputs from designer specifications

Symbol	Parameter	Unit	100% MCR	70% MCR
$\%O_2$	Oxygen content in flue gas at boiler exit	%	3.50	3.91
$f_{Rad\ loss}$	Radiation loss	%	0.20	0.20
w_{AF}	Air to fuel ratio	$\frac{kg\ air}{kg\ fuel}$	8.059	8.415

M.3 Burner biases

Table M.6 Flame biases from designer specifications

	Row 1	Row 2	Row 3	Row 4	Row 5
100% MCR	On	On	On	On	Off
70% MCR	On	On	On	On	Off

M.4 Heat exchanger tube pressure loss parameters

Table M.7 Heat exchanger tube pressure loss parameters for the validation using designer specifications

Heat exchanger	Internal surface roughness (μm)	$K_{forward}$
Eco	200	9
Furnace	200	1e-6
Vertical water wall	200	N/A
Sling tubes	200	N/A
Sling tube screen	200	1e-6
SH 1	200	0.1
SH 2	200	1.4
SH3	200	6.8
RH1	200	1e-6
RH2	200	0.8

Appendix N. Designed case simulation results

N.1 Full load case (100% MCR)

Table N.1 Water/steam pressure results at designed 100% MCR

	Model (MPa)	Design data (MPa)	Absolute error (MPa)	Relative error	10% error band (MPa)
$p_{Eco\ exit}$	21.44	21.44	0.00	-0.01%	± 2.14
$p_{Furnace\ WW\ exit}$	19.36	19.18	-0.18	-0.94%	± 1.92
p_{RC}	18.70	18.77	0.07	0.37%	± 1.88
$p_{SH1\ exit}$	18.34	18.42	0.08	0.45%	± 1.84
$p_{SH2\ exit}$	17.61	17.71	0.10	0.59%	± 1.77
$p_{SH3\ exit}$	17.10	17.22	0.12	0.67%	± 1.72
$p_{RH1\ exit}$	4.10	4.10	0.00	-0.07%	± 0.41
$p_{RH2\ exit}$	3.94	3.94	0.00	0.02%	± 0.39

Table N.2 Flue gas pressure results at designed 100% MCR

	Model (kPa)	Design data (kPa)	Absolute error (kPa)	Relative error	10% error band (kPa)
$p_{FG\ SH2\ in}$	82.44	81.00	-1.44	-1.78%	± 8.10
$p_{FG\ SH3\ in}$	82.27	80.99	-1.28	-1.58%	± 8.10
$p_{FG\ RH2\ in}$	81.90	80.99	-0.91	-1.12%	± 8.10
$p_{FG\ SH1\ in}$	81.58	80.97	-0.60	-0.74%	± 8.10
$p_{FG\ RH1\ in}$	81.25	80.90	-0.35	-0.43%	± 8.09
$p_{FG\ Eco\ in}$	80.83	80.74	-0.09	-0.11%	± 8.07
$p_{FG\ Eco\ exit}$	80.63	80.63	0.00	0.00%	± 8.06
$p_{FG\ AH\ exit}$	79.52	79.52	0.00	0.00%	± 7.95

Table N.3 Water/steam temperature results at designed 100% MCR

	Model (°C)	Design data (°C)	Absolute error (°C)	Relative error	10% error band (°C)
$T_{Eco\ exit}$	287.91	288.00	0.09	0.03%	±28.80
$T_{Evap\ exit}$	392.84	395.00	2.16	0.55%	±39.50
$T_{SH1\ in}$	390.31	391.00	0.69	0.18%	±39.10
$T_{SH1\ exit}$	446.33	463.00	16.67	3.60%	±46.30
$T_{SH2\ in}$	427.97	442.00	14.03	3.17%	±44.20
$T_{SH2\ exit}$	488.56	497.00	8.44	1.70%	±49.70
$T_{SH3\ in}$	471.78	480.00	8.22	1.71%	±48.00
$T_{SH3\ exit}$	550.00	540.00	-10.00	-1.85%	±54.00
$T_{RH1\ exit}$	486.26	457.00	-29.26	-6.40%	±45.70
$T_{RH2\ in}$	470.37	442.00	-28.37	-6.42%	±44.20
$T_{RH2\ exit}$	586.68	540.00	-46.68	-8.65%	±54.00

Table N.4 Flue gas temperature results at designed 100% MCR

	Model (°C)	Design data (°C)	Absolute error (°C)	Relative error	10% error band (°C)
$T_{FG \text{ furnace exit}}$	1284.23	1190	-94.23	-7.92%	±119.00
$T_{FG \text{ SH2 in}}$	1227.22	1174	-53.22	-4.53%	±117.40
$T_{FG \text{ SH2 exit}}$	1133.59	1061	-72.59	-6.84%	±106.10
$T_{FG \text{ SH3 exit}}$	973.99	952	-21.99	-2.31%	±95.20
$T_{FG \text{ RH2 exit}}$	830.21	828	-2.21	-0.27%	±82.80
$T_{FG \text{ SH1 exit}}$	677.13	643	-34.13	-5.31%	±64.30
$T_{FG \text{ RH1 exit}}$	479.27	473	-6.27	-1.32%	±47.30
$T_{FG \text{ Eco exit}}$	368.36	362	-6.36	-1.76%	±36.20
$T_{FG \text{ AH exit}}$	133.08	132	-1.08	-0.82%	±13.20

Table N.5 Heat transfer results at designed 100% MCR

	Model (MW)	Design data (MW)	Absolute error (MW)	Relative error	10% error band (MW)
$\dot{Q}_{Eco}$	103.67	102.93	-0.74	-0.72%	±10.29
$\dot{Q}_{Evap}$	832.98	848.43	15.45	1.82%	±84.84
$\dot{Q}_{SH1}$	150.15	175.81	25.66	14.60%	±17.58
$\dot{Q}_{SH2}$	129.33	108.06	-21.27	-19.68%	±10.81
$\dot{Q}_{SH3}$	139.93	104.41	-35.52	-34.02%	±10.44
$\dot{Q}_{RH1}$	189.18	156.17	-33.01	-21.14%	±15.62
$\dot{Q}_{RH2}$	138.91	116.57	-22.34	-19.17%	±11.66

Table N.6 Cumulative Heat transfer results at designed 100% MCR

	Model (MW)	Design data (MW)	Absolute error (MW)	Relative error	10% error band (MW)
$\dot{Q}_{MS\ total}$	1356.06	1339.64	-16.42	-1.23%	± 133.96
$\dot{Q}_{RS\ total}$	328.10	272.74	-55.36	-20.30%	± 27.27
$\dot{Q}_{Total}$	1684.07	1612.38	-71.69	-4.45%	± 161.24

N.2 Part load case (70% MCR)

Table N.7 Water/steam pressure results at designed 70% MCR

	Model (MPa)	Design data (MPa)	Absolute error (MPa)	Relative error	10% error band (MPa)
$p_{Eco\ exit}$	16.10	16.08	-0.02	-0.15%	± 1.61
$p_{Furnace\ WW\ exit}$	14.67	14.65	-0.02	-0.12%	± 1.47
p_{RC}	14.37	14.35	-0.02	-0.11%	± 1.44
$p_{SH1\ exit}$	14.11	14.20	0.08	0.58%	± 1.42
$p_{SH2\ exit}$	13.61	13.76	0.14	1.05%	± 1.38
$p_{SH3\ exit}$	13.27	13.39	0.13	0.93%	± 1.34
$p_{RH1\ exit}$	2.88	2.87	0.00	-0.07%	± 0.29
$p_{RH2\ exit}$	2.77	2.77	0.00	-0.04%	± 0.28

Table N.8 Flue gas pressure results at designed 70% MCR

	Model (kPa)	Design data (kPa)	Absolute error (kPa)	Relative error	10% error band (kPa)
$p_{FG\ SH2\ in}$	81.39	81.00	-0.39	-0.48%	± 8.10
$p_{FG\ SH3\ in}$	81.30	81.00	-0.31	-0.38%	± 8.10
$p_{FG\ RH2\ in}$	81.11	80.99	-0.11	-0.14%	± 8.10
$p_{FG\ SH1\ in}$	80.94	80.99	0.05	0.06%	± 8.10
$p_{FG\ RH1\ in}$	80.76	80.95	0.19	0.23%	± 8.10
$p_{FG\ Eco\ in}$	80.53	80.87	0.34	0.41%	± 8.09
$p_{FG\ Eco\ exit}$	80.42	80.76	0.34	0.42%	± 8.08
$p_{FG\ AH\ exit}$	80.06	80.06	0.00	0.00%	± 8.01

Table N.9 Water/steam temperature results at designed 70% MCR

	Model (°C)	Design data (°C)	Absolute error (°C)	Relative error	10% error band (°C)
$T_{Eco\ exit}$	273.28	272.00	-1.28	-0.5%	±27.20
$T_{Evap\ exit}$	380.50	369.00	-11.50	-3.1%	±36.90
$T_{SH1\ in}$	378.43	365.00	-13.43	-3.7%	±36.50
$T_{SH1\ exit}$	446.69	445.00	-1.69	-0.4%	±44.50
$T_{SH2\ in}$	422.91	421.00	-1.91	-0.5%	±42.10
$T_{SH2\ exit}$	508.86	491.00	-17.86	-3.6%	±49.10
$T_{SH3\ in}$	488.03	471.00	-17.03	-3.6%	±47.10
$T_{SH3\ exit}$	572.11	540.00	-32.11	-5.9%	±54.00
$T_{RH1\ exit}$	470.97	447.00	-23.97	-5.4%	±44.70
$T_{RH2\ in}$	465.54	442.00	-23.54	-5.3%	±44.20
$T_{RH2\ exit}$	576.67	540.00	-36.67	-6.8%	±54.00

Table N.10 Flue gas temperature results at designed 70% MCR

	Model (°C)	Design data (°C)	Absolute error (°C)	Relative error	10% error band (°C)
$T_{FG \text{ furnace exit}}$	1173.78	1104	-69.78	-6.32%	±110.40
$T_{FG \text{ SH2 in}}$	1118.26	1086	-32.26	-2.97%	±108.60
$T_{FG \text{ SH2 exit}}$	1018.07	966	-52.07	-5.39%	±96.60
$T_{FG \text{ SH3 exit}}$	872.04	860	-12.04	-1.40%	±86.00
$T_{FG \text{ RH2 exit}}$	743.90	747	3.10	0.41%	±74.70
$T_{FG \text{ SH1 exit}}$	603.84	574	-29.84	-5.20%	±57.40
$T_{FG \text{ RH1 exit}}$	442.38	440	-2.38	-0.54%	±44.00
$T_{FG \text{ Eco exit}}$	322.67	326	3.33	1.02%	±32.60
$T_{FG \text{ AH exit}}$	116.13	116	-0.13	-0.11%	±11.60

Table N.11 Heat transfer results at designed 70% MCR

	Model (MW)	Design data (MW)	Absolute error (MW)	Relative error	10% error band (MW)
$\dot{Q}_{Eco}$	82.38	79.35	-3.03	-3.82%	±7.94
$\dot{Q}_{Evap}$	646.81	627.69	-19.12	-3.05%	±62.77
$\dot{Q}_{SH1}$	100.51	123.14	22.63	18.38%	±12.31
$\dot{Q}_{SH2}$	105.98	85.51	-20.47	-23.94%	±8.55
$\dot{Q}_{SH3}$	93.19	76.83	-16.36	-21.30%	±7.68
$\dot{Q}_{RH1}$	113.11	93.33	-19.78	-21.19%	±9.33
$\dot{Q}_{RH2}$	90.13	79.40	-10.73	-13.51%	±7.94

Table N.12 Cumulative Heat transfer results at designed 70% MCR

	Model (MW)	Design data (MW)	Absolute error (MW)	Relative error	10% error band (MW)
$\dot{Q}_{MS\ total}$	1028.87	992.52	-36.35	-3.66%	±99.25
$\dot{Q}_{RS\ total}$	203.23	172.73	-30.50	-17.66%	±17.27
$\dot{Q}_{Total}$	1232.02	1165.25	-66.77	-5.73%	±116.53

N.3 Error summary

Table N.13 Designed operation simulations error summary

	Steam or water pressure	Flue gas pressure	Steam or water temperature	Flue gas temperature	Heat transfer
Full load case average relative error	0.4%	0.7%	3.1%	3.5%	15.9%
Part load case average relative error	0.1%	0.3%	3.5%	2.6%	15.0%
Full load case maximum relative error	1.0%	1.8%	8.7%	7.9%	34.0%
Part load case maximum relative error	1.1%	0.5%	6.8%	6.3%	23.9%

Appendix O. Mass and energy balance at attemperator

The attemperator of the case study boiler branches the inlet steam into four legs and cools the steam by spraying water in each leg, as shown in Figure O.1.

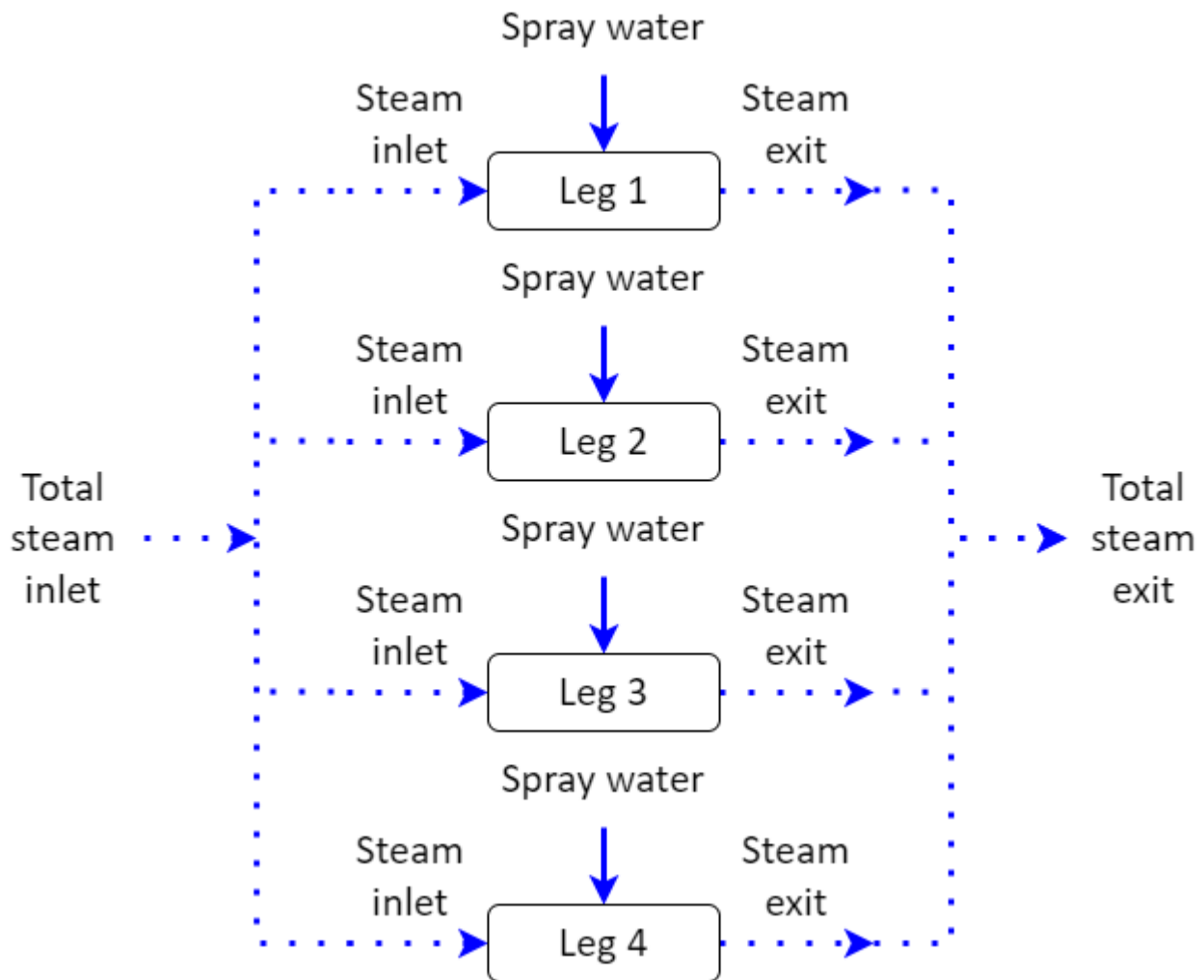


Figure O.1 Case study boiler attemperator schematics

As shown in Figure O.1, steam in each leg joins back into one stream after being attemperated, and the steam temperature of the total steam can be determined using the mass average of each leg.

In the case study boiler, the steam temperature at the inlet ($T_{steam,in\ i}$) and exit ($T_{steam,exit\ i}$) of each attemperator leg are measured, but not the mass flow rates. For that reason, the mass flow rate of steam in each attemperator leg is determined by analysing each leg's mass and energy balance.

Each attemporator leg can be seen as an individual system with a control volume diagram shown in Figure O.2.

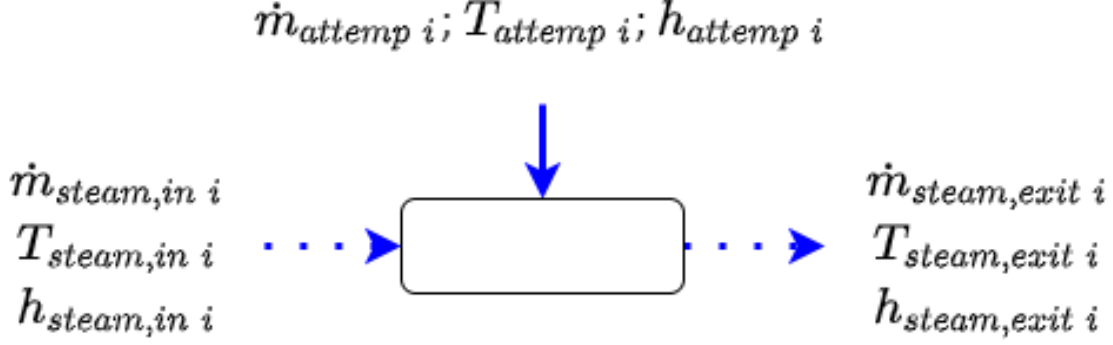


Figure O.2 Individual attemporator leg control volume diagram

Based on Figure O.2, the mass conservation of the system can be expressed using the following equation:

$$\begin{aligned}\sum \dot{m}_{exit} &= \sum \dot{m}_{in} \\ \dot{m}_{steam,exit\ i} &= \dot{m}_{steam,in\ i} + \dot{m}_{attemp\ i}\end{aligned}\quad (O-1)$$

Additionally, by assuming each attemporator leg is well insulated, the energy conservation of the system can be expressed using the following equation:

$$\begin{aligned}\sum \dot{m}_{exit} h_{oe} &= \sum \dot{m}_{in} h_{oi} \\ \dot{m}_{steam,exit\ i} h_{steam,exit\ i} &= \dot{m}_{steam,in\ i} h_{steam,in\ i} + \dot{m}_{attemp\ i} h_{attemp\ i}\end{aligned}\quad (O-2)$$

In Equation (O-2), the enthalpy of the inlet ($h_{steam,in\ i}$) and exit ($h_{steam,exit\ i}$) steam of each leg is obtained using their temperature measurements together with the steam fluid properties. Another parameter required to determine the steam enthalpy is the steam pressure, which is not measured by all the inlet and exit of the heat exchanger. For that reason, the missing steam pressures are linearly interpolated using the known steam pressure measurements.

On the other hand, the spray water temperature ($T_{attemp\ i}$), pressure ($p_{attemp\ i}$) and mass flow rate ($\dot{m}_{attemp\ i}$) are all known properties from the plant data; therefore, the steam mass flow rate at the inlet ($\dot{m}_{steam,in\ i}$) and exit ($\dot{m}_{steam,exit\ i}$) are determined using the spray water mass flow rate ($\dot{m}_{attemp\ i}$) of each leg.

The first step of computing these steam mass flow rates is to substitute Equation (O-1) into Equation (O-2) to obtain the following expression:

$$\begin{aligned} (\dot{m}_{steam,in\ i} + \dot{m}_{attemp\ i})h_{steam,exit\ i} \\ = \dot{m}_{steam,in\ i}h_{steam,in\ i} + \dot{m}_{attemp\ i}h_{attemp\ i} \end{aligned} \quad (O-3)$$

By rearranging Equation (O-3), the equation can be expressed as follow:

$$\dot{m}_{steam,in\ i} = \frac{\dot{m}_{attemp\ i}(h_{attemp\ i} - h_{steam,exit\ i})}{(h_{steam,exit\ i} - h_{steam,in\ i})} \quad (O-4)$$

Since all terms on the right-hand side of Equation (O-4) are known, the steam mass flow rate at the inlet of each leg can be determined.

Once the inlet steam mass flow rate is known, each leg's exit steam mass flow rate can be determined using Equation (O-1).

Once the steam mass flow rate at the inlet and exit of each leg is known, the mass average of the temperature measurements can be determined using the following equations:

$$T_{steam,in} = \sum_{i=1}^4 \frac{\dot{m}_{steam,in\ i} T_{steam,in\ i}}{\sum_{i=1}^4 \dot{m}_{steam,in\ i}} \quad (O-5)$$

$$T_{steam,exit} = \sum_{i=1}^4 \frac{\dot{m}_{steam,exit\ i} T_{steam,exit\ i}}{\sum_{i=1}^4 \dot{m}_{steam,exit\ i}} \quad (O-6)$$

Appendix P. Fuel specification

Table P.1 Coal specification for the first half of 2020

Symbol	Parameter	Unit	2020/03/31	2020/04/30	2020/05/31	2020/06/30	2020/07/31
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	18570	18910	19710	19450	19700
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4170	0.4320	0.4410	0.4450	0.4540
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0397	0.0386	0.0436	0.0413	0.0406
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.1052	0.1022	0.1155	0.1095	0.1076
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0177	0.0172	0.0194	0.0184	0.0181
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0104	0.0101	0.0114	0.0108	0.0106
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.1030	0.1220	0.0910	0.0900	0.0910
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.3070	0.2780	0.2780	0.2850	0.2780

Table P.2 Coal specification for the second half of 2020

Symbol	Parameter	Unit	2020/08/31	2020/09/30	2020/10/31	2020/11/30	2020/12/31
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	19150	19440	20290	19570	19350
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4430	0.4410	0.4610	0.4470	0.4470
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0402	0.0436	0.0427	0.0411	0.0416
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.1064	0.1155	0.1131	0.1089	0.1101
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0179	0.0194	0.0190	0.0183	0.0185
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0105	0.0114	0.0112	0.0107	0.0109
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.0810	0.0720	0.0860	0.0880	0.0810
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.3010	0.2970	0.2670	0.2860	0.2910

Table P.3 Coal specification for January 2021

Symbol	Parameter	Unit	Value
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	20190
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4340
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0413
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.1095
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0184
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0108
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.1100
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.2760

Table P.4 Average values from all the coal specifications

Symbol	Parameter	Unit	Value
HHV_{fuel}	The heating value of fuel	$\frac{kJ}{kg}$	19484.5
Y_C	Fuel C content	$\frac{kg}{kg\ fuel}$	0.4420
Y_H	Fuel H content	$\frac{kg}{kg\ fuel}$	0.0413
Y_O	Fuel O content	$\frac{kg}{kg\ fuel}$	0.1094
Y_N	Fuel N content	$\frac{kg}{kg\ fuel}$	0.0184
Y_S	Fuel S content	$\frac{kg}{kg\ fuel}$	0.0108
Y_{H_2O}	Fuel H_2O content	$\frac{kg}{kg\ fuel}$	0.0923
Y_{ash}	Fuel ash content	$\frac{kg}{kg\ fuel}$	0.2858

Appendix Q. Model boundary conditions and inputs extracted from plant data

Q.1 Full load case (100% MCR)

Table Q.1 Mass inputs and boundary conditions for boiler full load operation simulation

Symbol	Parameter	Unit	100% MCR
$\dot{m}_{fuel}$	Infeed fuel mass flow rate	$\frac{kg}{s}$	97.60
$\dot{m}_{SA}$	Secondary air mass flow rate	$\frac{kg}{s}$	589.00
$\dot{m}_{PA}$	Primary air mass flow rate	$\frac{kg}{s}$	149.00
$\dot{m}_{SH3\ exit}$	Superheater 3 exit steam mass flow rate	$\frac{kg}{s}$	575.12
$\dot{m}_{RH2\ exit}$	Reheater 2 exit steam mass flow rate	$\frac{kg}{s}$	525.09
$\dot{m}_{attemp\ 1}$	Attemperator 1 spray water mass flow rate	$\frac{kg}{s}$	0.00
$\dot{m}_{attemp\ 2}$	Attemperator 2 spray water mass flow rate	$\frac{kg}{s}$	14.31
$\dot{m}_{attemp\ 3}$	Attemperator 3 spray water mass flow rate	$\frac{kg}{s}$	17.06
$\dot{m}_{RH\ attemp}$	Reheat attemperator spray water mass flow rate	$\frac{kg}{s}$	20.66
$\dot{m}_{Circ\ pump}$	Circulation pump water flow rate	$\frac{kg}{s}$	0.00

Table Q.2 Pressure inputs and boundary conditions for boiler full load operation simulation

Symbol	Parameter	Unit	100% MCR
p_{FW}	Final feedwater heater pressure	<i>MPa</i>	21.59
$p_{RH1\ in}$	Reheater 1 Inlet pressure	<i>MPa</i>	3.81
$p_{FG\ AH\ exit}$	Flue gas pressure at air heater exit	<i>kPa</i>	78.32
$p_{PA\ AH\ exit}$	Primary air pressure at air heater exit	<i>kPa</i>	90.57

Table Q.3 Temperature inputs and boundary conditions for boiler full load operation simulation

Symbol	Parameter	Unit	100% MCR
T_{FW}	Final feedwater heater temperature	°C	244.09
$T_{RH1\ in}$	Reheater 1 Inlet temperature	°C	328.25
$T_{RH1\ attemp}$	Reheat attemperator spray water temperature	°C	141.07
$T_{SA\ in}$	Secondary air temperature at air heater inlet	°C	33.65
$T_{PA\ in}$	Primary air temperature at air heater inlet	°C	46.25
$T_{PA\ BNR\ in}$	Primary air temperature at burner inlet	°C	81.63

Table Q.4 Other inputs for boiler full load operation simulation

Symbol	Parameter	Unit	100% MCR
$\%O_2$	Oxygen content in flue gas at boiler exit	%	2.99
$f_{Rad\ loss}$	Radiation loss	%	0.33
w_{AF}	Air to fuel ratio	$\frac{kg\ air}{kg\ fuel}$	6.995

Q.2 Dry mode part load cases (81% and 63% MCR)

Table Q.5 Mass inputs and boundary conditions for boiler part-load operation simulations

Symbol	Parameter	Unit	81% MCR	63% MCR
$\dot{m}_{fuel}$	Infeed fuel mass flow rate	$\frac{kg}{s}$	80.68	61.10
$\dot{m}_{SA}$	Secondary air mass flow rate	$\frac{kg}{s}$	539.27	402.13
$\dot{m}_{PA}$	Primary air mass flow rate	$\frac{kg}{s}$	125.00	100.00
$\dot{m}_{SH3\ exit}$	Superheater 3 exit steam mass flow rate	$\frac{kg}{s}$	470.67	356.77
$\dot{m}_{RH2\ exit}$	Reheater 2 exit steam mass flow rate	$\frac{kg}{s}$	428.56	323.70
$\dot{m}_{attemp\ 1}$	Attemperator 1 spray water mass flow rate	$\frac{kg}{s}$	0.00	0.00
$\dot{m}_{attemp\ 2}$	Attemperator 2 spray water mass flow rate	$\frac{kg}{s}$	11.75	11.11
$\dot{m}_{attemp\ 3}$	Attemperator 3 spray water mass flow rate	$\frac{kg}{s}$	12.59	6.13
$\dot{m}_{RH\ attemp}$	Reheat attemperator spray water mass flow rate	$\frac{kg}{s}$	11.75	2.61
$\dot{m}_{Circ\ pump}$	Circulation pump water flow rate	$\frac{kg}{s}$	0.00	0.00

Table Q.6 Pressure inputs and boundary conditions for boiler part-load operation simulations

Symbol	Parameter	Unit	81% MCR	63% MCR
p_{FW}	Final feedwater heater pressure	<i>MPa</i>	19.06	13.92
$p_{RH1\ in}$	Reheater 1 Inlet pressure	<i>MPa</i>	3.07	2.38
$p_{FG\ AH\ exit}$	Flue gas pressure at air heater exit	<i>kPa</i>	78.76	79.41
$p_{PA\ AH\ exit}$	Primary air pressure at air heater exit	<i>kPa</i>	90.33	90.08

Table Q.7 Temperature inputs and boundary conditions for boiler part-load operation simulations

Symbol	Parameter	Unit	81% MCR	63% MCR
T_{FW}	Final feedwater heater temperature	°C	232.63	220.72
$T_{RH1\ in}$	Reheater 1 Inlet temperature	°C	324.60	332.29
$T_{RH1\ attemp}$	Reheat attemperator spray water temperature	°C	133.99	125.88
$T_{SA\ in}$	Secondary air temperature at air heater inlet	°C	34.68	29.80
$T_{PA\ in}$	Primary air temperature at air heater inlet	°C	48.27	43.24
$T_{PA\ BNR\ in}$	Primary air temperature at burner inlet	°C	80.19	74.64

Table Q.8 Other inputs for boiler part-load operation simulations

Symbol	Parameter	Unit	81% MCR	63% MCR
$\%O_2$	Oxygen content in flue gas at boiler exit	%	4.23	3.88
$f_{Rad\ loss}$	Radiation loss	%	0.35	0.39
w_{AF}	Air to fuel ratio	$\frac{kg\ air}{kg\ fuel}$	7.580	7.425

Q.3 Wet mode part load case (35% MCR)

Table Q.9 Mass inputs and boundary conditions for boiler wet mode operation simulation

Symbol	Parameter	Unit	35% MCR
$\dot{m}_{fuel}$	Infeed fuel mass flow rate	$\frac{kg}{s}$	43.91
$\dot{m}_{SA}$	Secondary air mass flow rate	$\frac{kg}{s}$	358.04
$\dot{m}_{PA}$	Primary air mass flow rate	$\frac{kg}{s}$	56.00
$\dot{m}_{SH3\ exit}$	Superheater 3 exit steam mass flow rate	$\frac{kg}{s}$	248.61
$\dot{m}_{RH2\ exit}$	Reheater 2 exit steam mass flow rate	$\frac{kg}{s}$	210.17
$\dot{m}_{attemp\ 1}$	Attemperator 1 spray water mass flow rate	$\frac{kg}{s}$	0.00
$\dot{m}_{attemp\ 2}$	Attemperator 2 spray water mass flow rate	$\frac{kg}{s}$	2.44
$\dot{m}_{attemp\ 3}$	Attemperator 3 spray water mass flow rate	$\frac{kg}{s}$	1.08
$\dot{m}_{RH\ attemp}$	Reheat attemperator spray water mass flow rate	$\frac{kg}{s}$	2.30
$\dot{m}_{Circ\ pump}$	Circulation pump water flow rate	$\frac{kg}{s}$	14.91

Table Q.10 Pressure inputs and boundary conditions for boiler wet mode operation simulation

Symbol	Parameter	Unit	35% MCR
p_{FW}	Final feedwater heater pressure	<i>MPa</i>	9.75
$p_{RH1\ in}$	Reheater 1 Inlet pressure	<i>MPa</i>	1.34
$p_{FG\ AH\ exit}$	Flue gas pressure at air heater exit	<i>kPa</i>	79.56
$p_{PA\ AH\ exit}$	Primary air pressure at air heater exit	<i>kPa</i>	86.61

Table Q.11 Temperature inputs and boundary conditions for boiler wet mode operation simulation

Symbol	Parameter	Unit	35% MCR
T_{FW}	Final feedwater heater temperature	°C	189.13
$T_{RH1\ in}$	Reheater 1 Inlet temperature	°C	328.59
$T_{RH1\ attemp}$	Reheat attemperator spray water temperature	°C	106.61
$T_{SA\ in}$	Secondary air temperature at air heater inlet	°C	35.37
$T_{PA\ in}$	Primary air temperature at air heater inlet	°C	55.41
$T_{PA\ BNR\ in}$	Primary air temperature at burner inlet	°C	60.51

Table Q.12 Other inputs for boiler wet mode operation simulation

Symbol	Parameter	Unit	35% MCR
$\%O_2$	Oxygen content in flue gas at boiler exit	%	6.01
$f_{Rad\ loss}$	Radiation loss	%	0.43
w_{AF}	Air to fuel ratio	$\frac{kg\ air}{kg\ fuel}$	8.479
L_{SV}	Water level at separation vessel	<i>m</i>	4.30

Q.4 Burner biases

Table Q.13 Burner biases for the boiler operation simulations

	Row 1	Row 2	Row 3	Row 4	Row 5
100% MCR	On	On	On	On	Off
81% MCR	On	On	On	Off	Off
63% MCR	On	Off	On	On	Off
35% MCR	On	On	Off	Off	Off

Q.5 Heat exchanger tube pressure loss parameters

Table Q.14 Heat exchanger tube pressure loss parameters for the boiler operation simulations

Heat exchanger	Internal surface roughness (μm)	$K_{forward}$
Eco	200	235
Furnace	200	1e-6
Vertical water wall	200	N/A
Sling tubes	200	N/A
Sling tube screen	200	1e-6
SH 1	200	23.8
SH 2	200	1e-6
SH3	200	12
RH1	25	1e-6
RH2	1e-6	1e-6

Appendix R. Plant operation simulation results

R.1 Full load case (100% MCR)

Table R.1 Water/steam pressure results at 100% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
$p_{Eco\ exit}$	20.65	20.70	0.05	0.25%	± 0.15
p_{RC}	19.64	19.66	0.02	0.10%	± 0.07
$p_{SH3\ exit}$	17.15	17.13	-0.02	-0.10%	± 0.11
$p_{RH2\ exit}$	3.73	3.75	0.02	0.57%	± 0.04

Table R.2 Gas pressure results at 100% MCR

	Model (kPa)	Plant data (kPa)	Absolute error (kPa)	Relative error	95% confidence interval (kPa)
$p_{PATub\ AH\ in}$	95.11	91.14	3.98	4.37%	± 1.08
$p_{PATub\ AH\ exit}$	90.57	91.14	-0.57	-0.62%	± 1.08
$p_{Air\ BNR\ in}$	83.66	83.66	0.01	0.01%	± 0.26
$p_{FG\ SH3\ in}$	81.41	80.77	0.64	0.79%	± 0.06
$p_{FG\ RH2\ in}$	81.04	80.89	0.15	0.19%	± 0.22
$p_{FG\ SH1\ in}$	80.72	79.90	0.83	1.04%	± 1.82
$p_{FG\ RH1\ in}$	80.41	80.84	-0.43	-0.53%	± 0.26
$p_{FG\ Eco\ in}$	80.00	80.80	-0.80	-0.99%	± 0.11
$p_{FG\ AH\ exit}$	78.32	80.80	-2.48	-3.08%	± 0.11

Table R.3 Water/steam temperature results at 100% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{Eco\ exit}$	279.45	279.41	-0.04	-0.01%	±0.37
$T_{Evap\ exit}$	381.71	382.82	1.11	0.29%	±2.61
$T_{SH1\ exit}$	441.40	439.07	-2.33	-0.53%	±0.78
$T_{SH2\ in}$	428.62	427.06	-1.56	-0.36%	±1.25
$T_{SH2\ exit}$	482.62	480.78	-1.84	-0.38%	±0.61
$T_{SH3\ in}$	462.03	460.01	-2.03	-0.44%	±0.86
$T_{SH3\ exit}$	532.91	534.61	1.70	0.32%	±0.80
$T_{RH1\ exit}$	460.99	460.30	-0.69	-0.15%	±0.47
$T_{RH2\ in}$	413.90	409.25	-4.65	-1.14%	±0.48
$T_{RH2\ exit}$	538.24	534.38	-3.86	-0.72%	±0.62

Table R.4 Heat transfer results at 100% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	91.98	91.88	-0.09	-0.10%	± 0.79
$\dot{Q}_{Evap}$	798.09	802.51	4.42	0.55%	± 15.06
$\dot{Q}_{SH1}$	195.33	185.70	-9.63	-5.19%	± 6.08
$\dot{Q}_{SH2}$	116.97	119.16	2.19	1.84%	± 1.21
$\dot{Q}_{SH3}$	131.47	137.63	6.16	4.48%	± 4.04
$\dot{Q}_{RH1}$	159.54	158.84	-0.70	-0.44%	± 3.50
$\dot{Q}_{RH2}$	149.85	147.53	-2.31	-1.57%	± 2.15

Table R.5 Cumulative Heat transfer results at 100% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{MS\ total}$	1333.84	1336.89	3.05	0.23%	± 18.11
$\dot{Q}_{RS\ total}$	309.38	306.37	-3.02	-0.98%	± 5.50
$\dot{Q}_{Total}$	1643.26	1643.25	-0.01	0.00%	± 23.08

Table R.6 Metal temperature results at 100% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{metal\ SH2\ exit}$	487.64	479.07	8.58	1.79%	±10.83
$T_{metal\ SH3\ exit}$	536.74	518.37	18.37	3.54%	±19.74
$T_{metal\ RH1\ exit}$	462.62	445.46	17.17	3.85%	±30.63
$T_{metal\ RH2\ exit}$	543.29	523.39	19.90	3.80%	±18.99

Table R.7 Furnace wall metal temperature results at 100% MCR

	Tube (°C)	Fin (°C)
Segment 1	363.08	380.78
Segment 2	394.94	415.18
Segment 3	411.67	432.26
Segment 4	407.63	427.05
Segment 5	403.23	420.82
Segment 6	404.58	423.29
Segment 7	406.29	424.14
Segment 8	404.59	421.60
Segment 9	402.19	418.39
Segment 10	401.22	416.65

R.2 Part load case 1 (81% MCR)

Table R.8 Water/steam pressure results at 81% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
$p_{Eco\ exit}$	18.44	18.41	-0.04	-0.20%	± 0.09
p_{RC}	17.78	17.61	-0.17	-0.95%	± 0.06
$p_{SH3\ exit}$	15.93	15.66	-0.27	-1.71%	± 0.11
$p_{RH2\ exit}$	3.00	2.98	-0.02	-0.59%	± 0.01

Table R.9 Gas pressure results at 81% MCR

	Model (kPa)	Plant data (kPa)	Absolute error (kPa)	Relative error	95% confidence interval (kPa)
$p_{PATub\ AH\ in}$	93.61	90.71	2.90	3.20%	± 1.47
$p_{PATub\ AH\ exit}$	90.33	90.71	-0.38	-0.42%	± 0.82
$p_{Air\ BNR\ in}$	83.48	83.42	0.06	0.07%	± 0.26
$p_{FG\ SH3\ in}$	80.96	80.76	0.20	0.24%	± 0.02
$p_{FG\ RH2\ in}$	80.69	80.90	-0.22	-0.27%	± 0.20
$p_{FG\ SH1\ in}$	80.45	79.89	0.56	0.70%	± 1.82
$p_{FG\ RH1\ in}$	80.21	80.89	-0.68	-0.84%	± 0.14
$p_{FG\ Eco\ in}$	79.91	80.83	-0.92	-1.14%	± 0.09
$p_{FG\ AH\ exit}$	78.76	80.82	-2.07	-2.56%	± 0.08

Table R.10 Water/steam temperature results at 81% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{Eco\ exit}$	271.57	271.28	-0.29	-0.11%	±0.62
$T_{Evap\ exit}$	371.86	372.56	0.70	0.19%	±1.59
$T_{SH1\ exit}$	433.86	437.02	3.16	0.72%	±0.89
$T_{SH2\ in}$	420.32	423.50	3.18	0.75%	±0.50
$T_{SH2\ exit}$	480.17	484.97	4.80	0.99%	±0.57
$T_{SH3\ in}$	460.58	462.87	2.29	0.50%	±0.32
$T_{SH3\ exit}$	532.68	534.72	2.03	0.38%	±0.36
$T_{RH1\ exit}$	453.37	454.56	1.19	0.26%	±0.24
$T_{RH2\ in}$	419.59	416.66	-2.94	-0.71%	±0.94
$T_{RH2\ exit}$	539.67	534.37	-5.30	-0.99%	±0.51

Table R.11 Heat transfer results at 81% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	82.12	81.46	-0.66	-0.81%	± 1.21
$\dot{Q}_{Evap}$	679.79	686.57	6.78	0.99%	± 6.40
$\dot{Q}_{SH1}$	155.22	156.54	1.32	0.84%	± 3.24
$\dot{Q}_{SH2}$	100.90	102.60	1.70	1.66%	± 1.60
$\dot{Q}_{SH3}$	105.03	100.11	-4.93	-4.92%	± 1.40
$\dot{Q}_{RH1}$	124.31	125.62	1.30	1.04%	± 1.60
$\dot{Q}_{RH2}$	116.13	111.78	-4.34	-3.88%	± 1.34

Table R.12 Cumulative Heat transfer results at 81% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{MS\ total}$	1123.06	1127.28	4.21	0.37%	± 5.77
$\dot{Q}_{RS\ total}$	240.44	238.44	-2.00	-0.84%	± 2.18
$\dot{Q}_{Total}$	1363.44	1363.91	0.47	0.03%	± 6.77

Table R.13 Metal temperature results at 81% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{metal\ SH2\ exit}$	484.29	483.26	1.04	0.21%	±11.49
$T_{metal\ SH3\ exit}$	535.65	519.37	16.28	3.13%	±18.40
$T_{metal\ RH1\ exit}$	454.76	440.66	14.10	3.20%	±25.79
$T_{metal\ RH2\ exit}$	543.78	522.82	20.96	4.01%	±17.86

Table R.14 Furnace wall metal temperature results at 81% MCR

	Tube (°C)	Fin (°C)
Segment 1	352.08	367.25
Segment 2	384.58	401.95
Segment 3	397.19	414.85
Segment 4	393.99	410.65
Segment 5	390.13	405.23
Segment 6	391.32	407.37
Segment 7	391.76	407.07
Segment 8	390.83	405.43
Segment 9	389.26	403.16
Segment 10	387.97	401.20

R.3 Part load case 2 (63% MCR)

Table R.15 Water/steam pressure results at 63% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
$p_{Eco\ exit}$	13.57	13.73	0.16	1.16%	± 0.19
p_{RC}	13.14	12.99	-0.15	-1.15%	± 0.15
$p_{SH3\ exit}$	11.65	11.44	-0.20	-1.76%	± 0.21
$p_{RH2\ exit}$	2.32	2.35	0.03	1.13%	± 0.05

Table R.16 Gas pressure results at 63% MCR

	Model (kPa)	Plant data (kPa)	Absolute error (kPa)	Relative error	95% confidence interval (kPa)
$p_{PATub\ AH\ in}$	92.11	90.55	1.56	1.72%	± 0.79
$p_{PATub\ AH\ exit}$	90.08	90.55	-0.48	-0.53%	± 0.79
$p_{Air\ BNR\ in}$	83.32	83.29	0.02	0.02%	± 0.09
$p_{FG\ SH3\ in}$	80.38	80.75	-0.38	-0.47%	± 0.02
$p_{FG\ RH2\ in}$	80.23	80.90	-0.67	-0.83%	± 0.21
$p_{FG\ SH1\ in}$	80.11	79.89	0.21	0.27%	± 1.82
$p_{FG\ RH1\ in}$	79.98	80.88	-0.90	-1.11%	± 0.22
$p_{FG\ Eco\ in}$	79.81	80.87	-1.07	-1.32%	± 0.08
$p_{FG\ AH\ exit}$	79.41	80.87	-1.46	-1.80%	± 0.08

Table R.17 Water/steam temperature results at 63% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{Eco\ exit}$	260.55	259.49	-1.06	-0.41%	±1.34
$T_{Evap\ exit}$	341.02	341.44	0.42	0.12%	±4.59
$T_{SH1\ exit}$	406.37	414.87	8.50	2.05%	±3.62
$T_{SH2\ in}$	387.71	394.28	6.57	1.67%	±0.46
$T_{SH2\ exit}$	465.43	473.83	8.39	1.77%	±2.39
$T_{SH3\ in}$	450.72	450.85	0.13	0.03%	±1.09
$T_{SH3\ exit}$	531.74	533.55	1.81	0.34%	±0.27
$T_{RH1\ exit}$	432.22	427.95	-4.26	-1.00%	±0.46
$T_{RH2\ in}$	422.14	416.36	-5.78	-1.39%	±2.18
$T_{RH2\ exit}$	534.33	527.78	-6.55	-1.24%	±3.40

Table R.18 Heat transfer results at 63% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	63.24	61.51	-1.73	-2.81%	± 2.19
$\dot{Q}_{Evap}$	543.82	549.00	5.18	0.94%	± 18.58
$\dot{Q}_{SH1}$	109.61	116.76	7.15	6.12%	± 14.69
$\dot{Q}_{SH2}$	90.28	89.31	-0.97	-1.09%	± 5.21
$\dot{Q}_{SH3}$	80.55	73.45	-7.10	-9.66%	± 3.78
$\dot{Q}_{RH1}$	72.22	69.19	-3.03	-4.39%	± 3.00
$\dot{Q}_{RH2}$	80.72	81.13	0.40	0.50%	± 2.16

Table R.19 Cumulative Heat transfer results at 63% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{MS\ total}$	887.51	890.04	2.53	0.28%	± 11.57
$\dot{Q}_{RS\ total}$	152.95	150.31	-2.63	-1.75%	± 4.03
$\dot{Q}_{Total}$	1040.34	1040.35	0.01	0.00%	± 15.46

Table R.20 Metal temperature results at 63% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{metal\ SH2\ exit}$	469.00	472.16	-3.16	-0.67%	±13.42
$T_{metal\ SH3\ exit}$	533.98	521.56	12.42	2.38%	±14.17
$T_{metal\ RH1\ exit}$	433.17	422.72	10.45	2.47%	±21.57
$T_{metal\ RH2\ exit}$	537.32	520.43	16.89	3.25%	±21.46

Table R.21 Furnace wall metal temperature results at 63% MCR

	Tube (°C)	Fin (°C)
Segment 1	335.93	347.91
Segment 2	364.40	378.10
Segment 3	366.01	379.96
Segment 4	363.78	376.93
Segment 5	362.06	373.97
Segment 6	362.73	375.40
Segment 7	361.81	373.89
Segment 8	360.64	372.15
Segment 9	359.19	370.16
Segment 10	357.83	368.28

R.4 Wet mode part load case (35% MCR)

Table R.22 Water/steam pressure results at 35% MCR

	Model (MPa)	Plant data (MPa)	Absolute error (MPa)	Relative error	95% confidence interval (MPa)
$p_{Eco\ exit}$	9.55	9.73	0.18	1.83%	± 0.03
p_{RC}	9.23	8.98	-0.25	-2.82%	± 0.04
$p_{SH3\ exit}$	8.13	8.20	0.06	0.75%	± 0.02
$p_{RH2\ exit}$	1.29	1.36	0.06	4.57%	± 0.01

Table R.23 Gas pressure results at 35% MCR

	Model (kPa)	Plant data (kPa)	Absolute error (kPa)	Relative error	95% confidence interval (kPa)
$p_{PATub\ AH\ in}$	87.33	86.74	0.60	0.69%	± 0.34
$p_{PATub\ AH\ exit}$	86.61	86.74	-0.13	-0.15%	± 0.34
$p_{Air\ BNR\ in}$	82.49	82.48	0.01	0.01%	± 0.19
$p_{FG\ SH3\ in}$	79.98	80.79	-0.81	-1.01%	± 0.05
$p_{FG\ RH2\ in}$	79.89	80.91	-1.02	-1.26%	± 0.19
$p_{FG\ SH1\ in}$	79.81	79.91	-0.10	-0.13%	± 1.86
$p_{FG\ RH1\ in}$	79.72	80.92	-1.19	-1.47%	± 0.19
$p_{FG\ Eco\ in}$	79.61	80.93	-1.32	-1.63%	± 0.05
$p_{FG\ AH\ exit}$	79.56	80.93	-1.37	-1.69%	± 0.05

Table R.24 Water/steam temperature results at 35% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{Eco\ exit}$	242.84	235.92	-6.92	-2.93%	±0.39
$T_{Evap\ exit}$	305.18	305.78	0.61	0.20%	±0.34
$T_{SH1\ exit}$	377.29	394.19	16.90	4.29%	±0.60
$T_{SH2\ in}$	370.14	394.60	24.46	6.20%	±1.13
$T_{SH2\ exit}$	460.31	468.16	7.84	1.68%	±0.74
$T_{SH3\ in}$	455.38	458.35	2.97	0.65%	±0.94
$T_{SH3\ exit}$	531.57	530.26	-1.32	-0.25%	±0.40
$T_{RH1\ exit}$	416.33	410.13	-6.20	-1.51%	±0.13
$T_{RH2\ in}$	401.87	398.15	-3.71	-0.93%	±0.34
$T_{RH2\ exit}$	512.59	504.70	-7.89	-1.56%	±0.61

Table R.25 Heat transfer results at 35% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{Eco}$	55.01	46.60	-8.41	-18.06%	± 0.69
$\dot{Q}_{Evap}$	418.39	426.65	8.26	1.94%	± 2.88
$\dot{Q}_{SH1}$	77.32	89.37	12.05	13.48%	± 0.65
$\dot{Q}_{SH2}$	64.33	56.29	-8.04	-14.28%	± 0.49
$\dot{Q}_{SH3}$	48.21	41.81	-6.41	-15.32%	± 1.58
$\dot{Q}_{RH1}$	39.37	36.53	-2.84	-7.78%	± 0.26
$\dot{Q}_{RH2}$	50.46	49.56	-0.89	-1.80%	± 0.24

Table R.26 Cumulative Heat transfer results at 35% MCR

	Model (MW)	Plant data (MW)	Absolute error (MW)	Relative error	95% confidence interval (MW)
$\dot{Q}_{MS\ total}$	663.25	660.71	-2.54	-0.38%	± 4.40
$\dot{Q}_{RS\ total}$	89.83	86.09	-3.74	-4.34%	± 0.46
$\dot{Q}_{Total}$	752.98	746.80	-6.18	-0.83%	± 4.75

Table R.27 Metal temperature results at 35% MCR

	Model (°C)	Plant data (°C)	Absolute error (°C)	Relative error	95% confidence interval (°C)
$T_{metal\ SH2\ exit}$	462.78	468.38	-5.60	-1.20%	±17.79
$T_{metal\ SH3\ exit}$	532.96	519.72	13.24	2.55%	±17.92
$T_{metal\ RH1\ exit}$	416.90	394.88	22.02	5.58%	±14.44
$T_{metal\ RH2\ exit}$	514.58	492.04	22.55	4.58%	±30.75

Table R.28 Furnace wall metal temperature results at 35% MCR

	Tube (°C)	Fin (°C)
Segment 1	315.03	324.44
Segment 2	334.37	345.12
Segment 3	334.92	345.87
Segment 4	332.83	343.15
Segment 5	329.74	339.09
Segment 6	330.18	340.13
Segment 7	328.85	338.33
Segment 8	327.53	336.57
Segment 9	326.22	334.84
Segment 10	324.94	333.15

R.5 Error summary

Table R.29 Steam/water pressure results error summary

	100% MCR	81% MCR	63% MCR	35% MCR
Average relative error	0.3%	0.9%	1.3%	2.5%
Maximum relative error	0.6%	1.7%	1.8%	4.6%

Table R.30 Gas pressure results error summary

	100% MCR	81% MCR	63% MCR	35% MCR
Average relative error	1.3%	1.0%	0.9%	0.9%
Maximum relative error	4.4%	3.2%	1.8%	1.7%

Table R.31 Steam/water temperature results error summary

	100% MCR	81% MCR	63% MCR	35% MCR
Average relative error	0.4%	0.6%	1.0%	2.0%
Maximum relative error	1.1%	1.0%	2.0%	6.2%

Table R.32 Heat transfer results error summary

	100% MCR	81% MCR	63% MCR	35% MCR
Average relative error	2.0%	2.0%	3.6%	10.4%
Maximum relative error	5.2%	4.9%	9.7%	18.1%

Table R.33 Metal temperature results error summary

	100% MCR	81% MCR	63% MCR	35% MCR
Average relative error	3.2%	2.6%	2.2%	3.5%
Maximum relative error	3.9%	4.0%	3.2%	5.6%

Appendix S. Case study boiler geometry

S.1 Boiler

Table S.1 Boiler overall geometry

Symbol	Parameter	Unit	Value
b_{boiler}	Boiler depth	m	104
H_{boiler}	Boiler height	m	19.286
w_{boiler}	Boiler width	m	19.286

S.2 Furnace

Table S.2 Furnace geometry

Symbol	Parameter	Unit	Value
$b_{furnace}$	Furnace depth	m	61.7
$H_{furnace}$	Furnace height	m	19.286
$w_{furnace}$	Furnace width	m	19.286
$H_{hopper\ base}$	Hopper base height	m	4.515
$H_{hopper\ mid-plane}$	Hopper mid-plane height	m	11.6575
$H_{hopper\ top}$	Hopper top height	m	18.8
$V_{furnace}$	Furnace volume	m^3	21434.60
$A_{furnace}$	Planar furnace area	m^2	5180
$A_{furnace,exit\ plane}$	Furnace exit plane area	m^2	371.95

S.3 Furnace water wall (helical)

Table S.3 Furnace water wall geometry

Symbol	Parameter	Unit	Value
$d_{o\,ww\,1}$	Water wall tube outer diameter 1	<i>mm</i>	44.5
$d_{o\,ww\,2}$	Water wall tube outer diameter 2	<i>mm</i>	48.3
$t_{ww\,1}$	Water wall tube thickness 1	<i>mm</i>	5.6
$t_{ww\,2}$	Water wall tube thickness 2	<i>mm</i>	6.3
$t_{fin\,ww\,1}$	Water wall tube fin thickness 1	<i>mm</i>	11.2
$t_{fin\,ww\,2}$	Water wall tube fin thickness 2	<i>mm</i>	12.6
$\alpha_{WW\,1}$	Water wall tube helix angle 1	°	14.971
$\alpha_{WW\,2}$	Water wall tube helix angle 2	°	16.549
$n_{WW\,1}$	Number of parallel water wall tubes 1	<i>N/A</i>	320
$n_{WW\,2}$	Number of parallel water wall tubes 2	<i>N/A</i>	320
H_{shift}	Height where tube set 1 shift to set 2	<i>m</i>	53.3

S.4 Burners

Table S.4 Burner heights

	Unit	Row 1	Row 2	Row 3	Row 4	Row 5
Upper	<i>m</i>	22.472	28.184	33.896	39.607	45.319
Lower	<i>m</i>	19.969	25.681	31.393	37.104	42.816

S.5 Superheaters

Table S.5 Superheater geometry

Symbol	Parameter	Unit	SH1	SH2	SH3
$Config$	Tube bundle configuration	N/A	Inline	Inline	Inline
H_{duct}	Depth of duct	m	19.286	19.286	19.286
w_{duct}	Width of duct	m	19.286	19.286	19.286
l_b	Length of duct	m	3.95	3.28	2.7
d_o	Tube outer diameter	mm	44.5	44.5	44.5
t_{wall}	Tube wall thickness	mm	5.1	5.5	5.2
n_l	Number of longitudinal tubes	N/A	24	48	30
$n_{t,odd}$	Number of transverse tubes in odd row	N/A	160	18	40
$n_{t,even}$	Number of transverse tubes in even row	N/A	160	18	40
n_{pass}	Number of passes	N/A	6	2	2
s_l	Tube bundle longitudinal pitch	mm	85	70	70
s_t	Tube bundle transverse pitch	mm	120	960	480
l_{pass}	Mean tube length of each pass	m	13.067	19.4	19.4
A_{ext}	Tube bundle external surface area	m^2	7014.828	2343.286	3291.132
A_{int}	Tube bundle internal surface area	m^2	5406.934	1764.047	2530.515

S.6 Reheaters

Table S.6 Reheater geometry

Symbol	Parameter	Unit	RH1	RH2
$Config$	Tube bundle configuration	N/A	Inline	Inline
H_{duct}	Depth of duct	m	19.286	19.286
w_{duct}	Width of duct	m	19.286	19.286
l_b	Length of duct	m	5.6	2.8
d_o	Tube outer diameter	mm	58.65	57
t_{wall}	Tube wall thickness	mm	3.6	3.8
n_l	Number of longitudinal tubes	N/A	36	20
$n_{t,odd}$	Number of transverse tubes in odd row	N/A	160	80
$n_{t,even}$	Number of transverse tubes in even row	N/A	160	80
n_{pass}	Number of passes	N/A	4	2
s_l	Tube bundle longitudinal pitch	mm	110	110
s_t	Tube bundle transverse pitch	mm	120	240
l_{pass}	Mean tube length of each pass	m	19.65	19.55
A_{ext}	Tube bundle external surface area	m^2	20854.651	5601.334
A_{int}	Tube bundle internal surface area	m^2	18294.489	4854.489

S.7 Economiser

Table S.7 Economiser geometry

Symbol	Parameter	Unit	Eco
$Config$	Tube bundle configuration	N/A	Inline
H_{duct}	Depth of duct	m	19.286
w_{duct}	Width of duct	m	19.286
l_b	Length of duct	m	1.143
d_o	Tube outer diameter	mm	51
t_{wall}	Tube wall thickness	mm	4.5
n_l	Number of longitudinal tubes	N/A	8
$n_{t,odd}$	Number of transverse tubes in odd row	N/A	160
$n_{t,even}$	Number of transverse tubes in even row	N/A	160
n_{pass}	Number of passes	N/A	4
s_l	Tube bundle longitudinal pitch	mm	127
s_t	Tube bundle transverse pitch	mm	120
l_{pass}	Mean tube length of each pass	m	17.4
$A_{ext,f}$	Finned tube bundle external surface area	m^2	31346.457
A_{int}	Tube bundle internal surface area	m^2	2938.721
t_{fin}	Tube fin thickness	mm	3
l_f	Fin length	mm	115
b_f	Fin width	mm	122

S.8 Tubular air heater

Table S.8 Tubular air heater geometry

Symbol	Parameter	Unit	Cold	Hot
<i>Config</i>	Tube bundle configuration	<i>N/A</i>	Staggered	Staggered
H_{duct}	Depth of duct	<i>m</i>	2.6	2.6
w_{duct}	Width of duct	<i>m</i>	25.28	25.28
l_b	Length of duct	<i>m</i>	2.6	2.6
d_o	Tube outer diameter	<i>mm</i>	51	51
t_{wall}	Tube wall thickness	<i>mm</i>	2.9	2.3
n_l	Number of longitudinal tubes	<i>N/A</i>	36	42
$n_{t,odd}$	Number of transverse tubes in odd row	<i>N/A</i>	294	296
$n_{t,even}$	Number of transverse tubes in even row	<i>N/A</i>	294	296
n_{pass}	Number of passes	<i>N/A</i>	1	3
s_l	Tube bundle longitudinal pitch	<i>mm</i>	60	60
s_t	Tube bundle transverse pitch	<i>mm</i>	85	80
l_{pass}	Mean tube length of each pass	<i>m</i>	3.385	3.0517
A_{ext}	Tube bundle external surface area	<i>m</i> ²	5740.220	18235.572
A_{int}	Tube bundle internal surface area	<i>m</i> ²	5087.411	16590.795

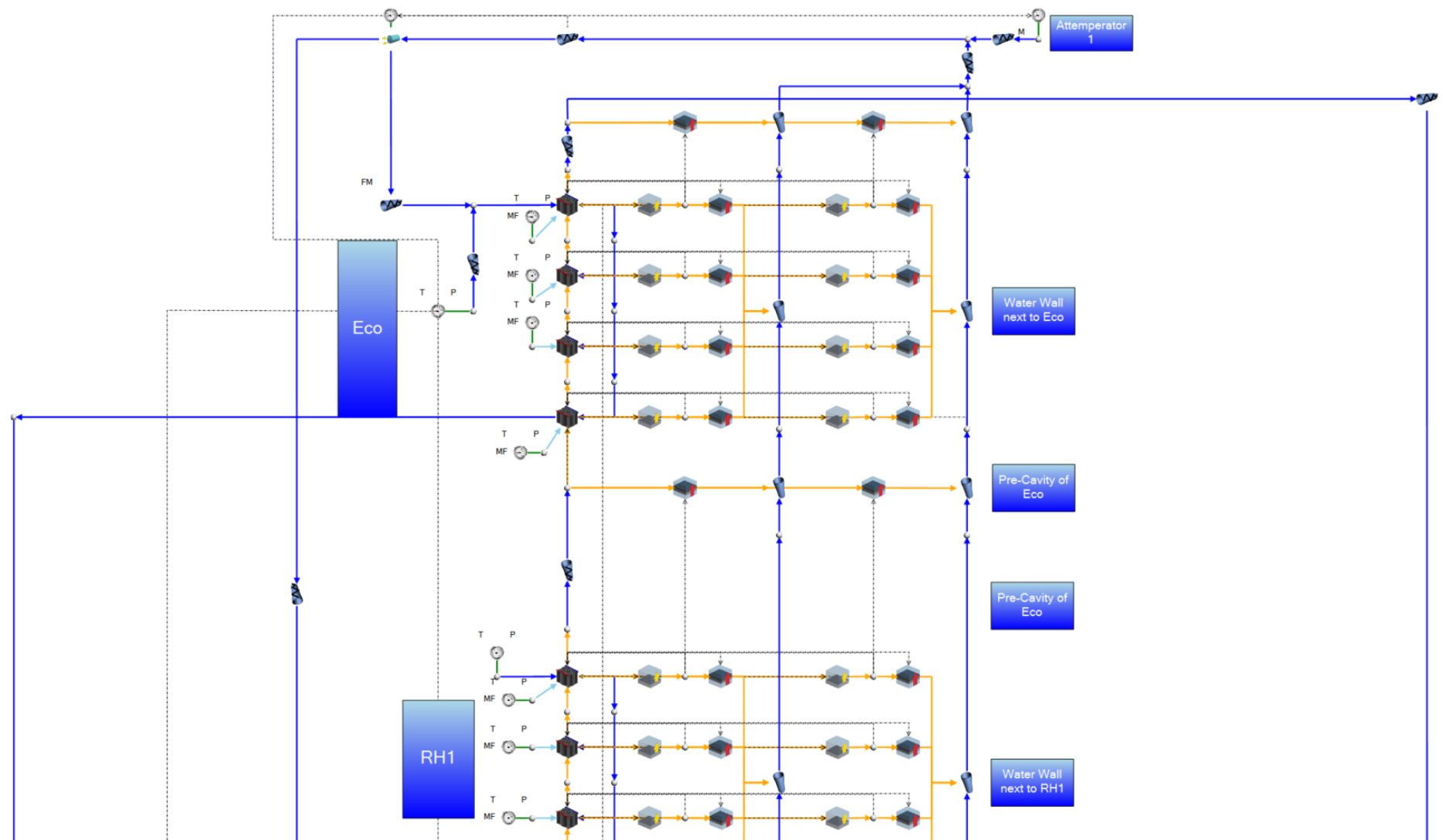
S.9 Rotary air heater

Table S.9 Rotary air heater geometry

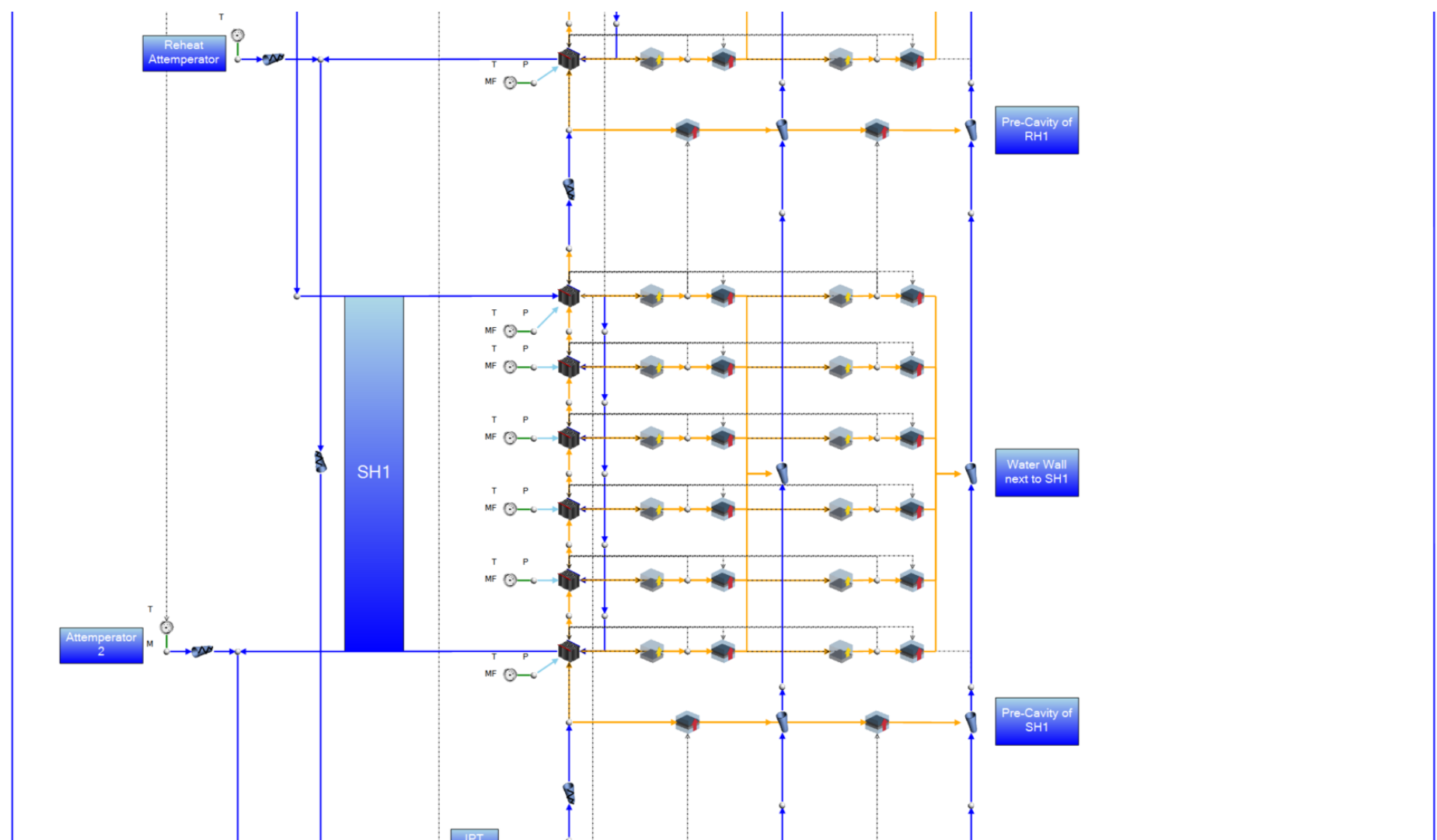
Symbol	Parameter	Unit	Value
l_{rot}	Rotary air heater length	m	11.4
$A_{flow,air}$	Air flow area	m^2	55.6144
$A_{flow,products}$	Products flow area	m^2	70.3086
A_{HT}	Total heat transfer area	m^2	69010
$D_{h,rot}$	Effective hydraulic diameter	mm	2.64
n_{comp}	Number of compartments in each radial plane	N/A	2178
$D_{duct,0}$	Rotary air heater inner radius	m	3
$D_{duct,N}$	Rotary air heater outer radius	m	14.5
$A_{flow,gaps}$	Flow area of the gaps	m^2	0.72
$L_{c,0}$	Width of the compartment closest to the centre	mm	2.64
$L_{c,N}$	Width of the compartment furthest to the centre	mm	2.64

Appendix T. Flownex boiler model layout

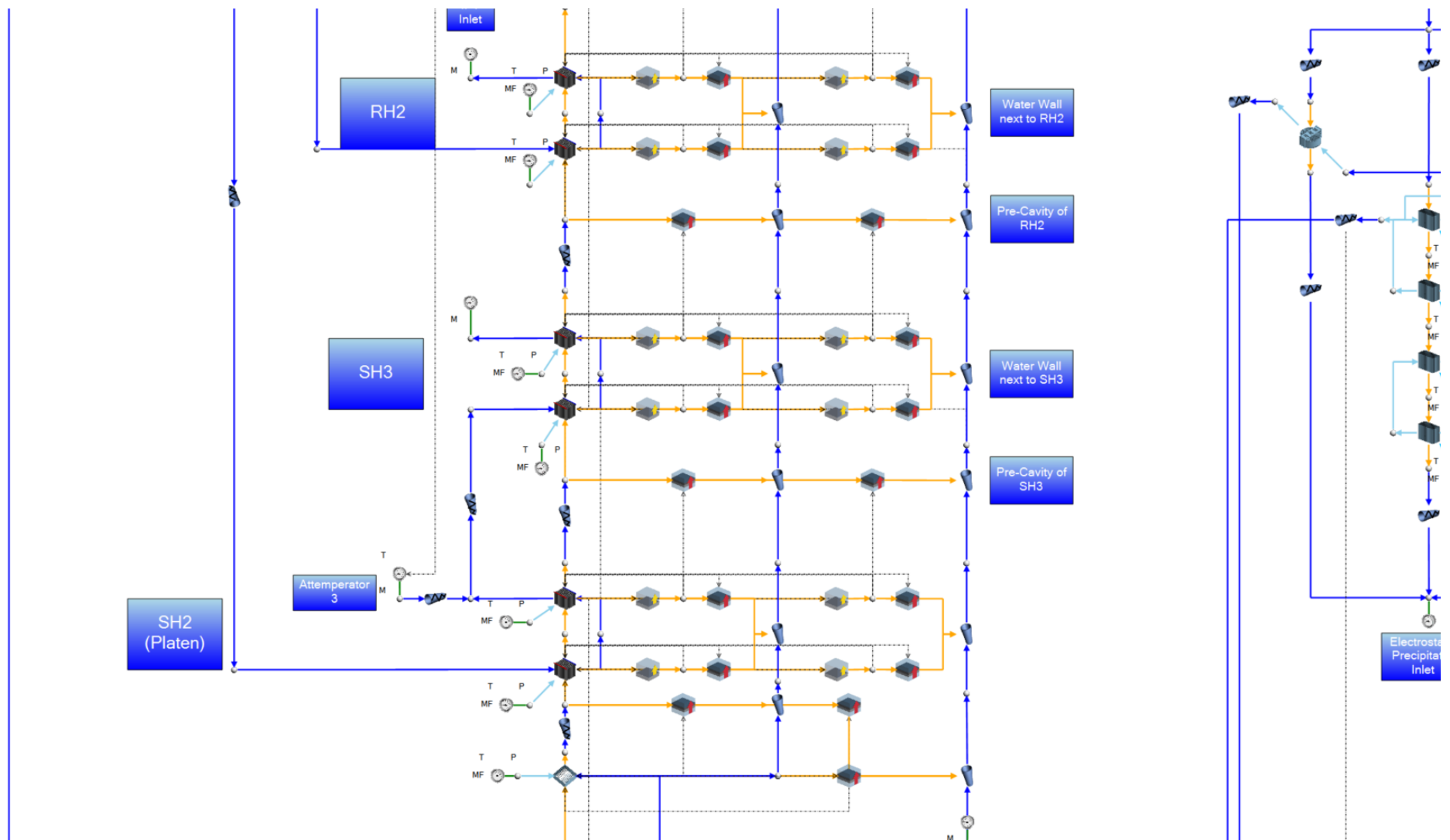
Model top section:



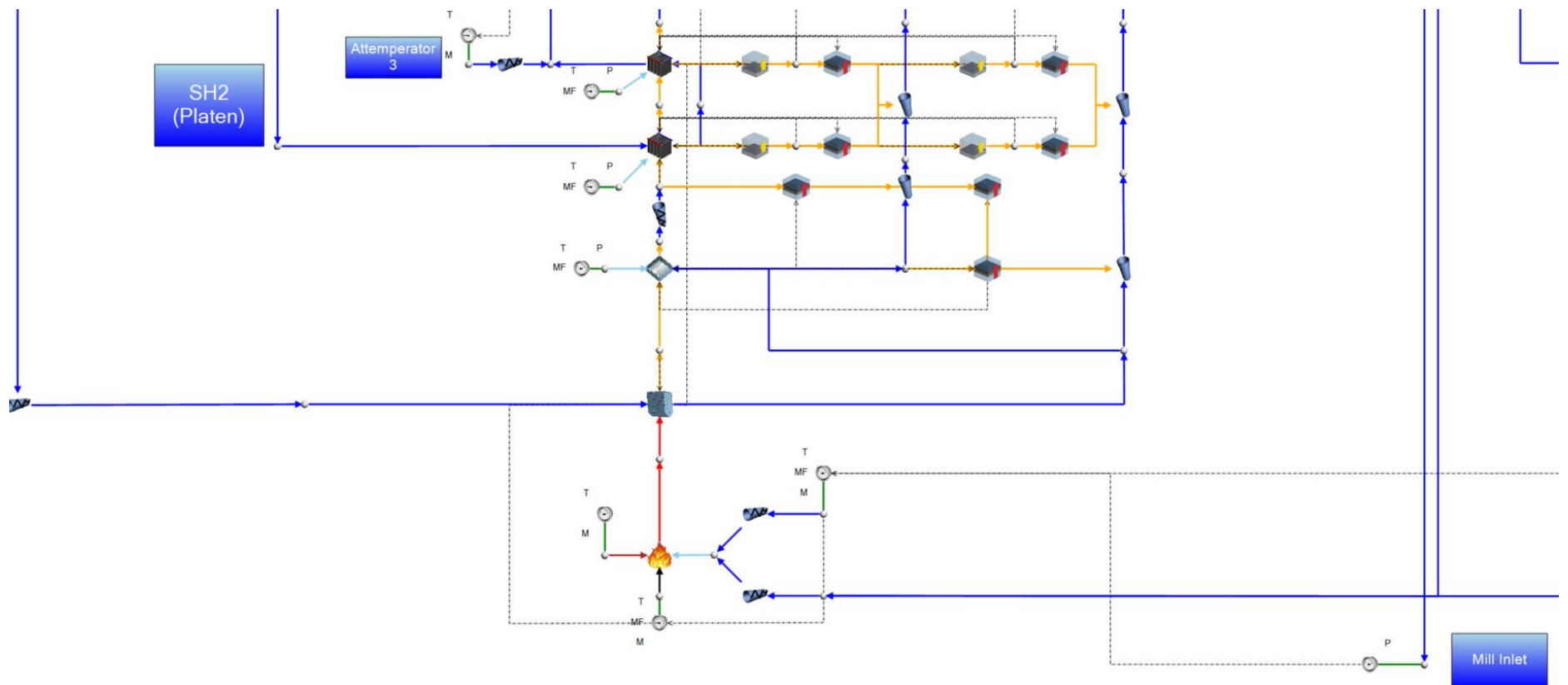
Model mid-high section:



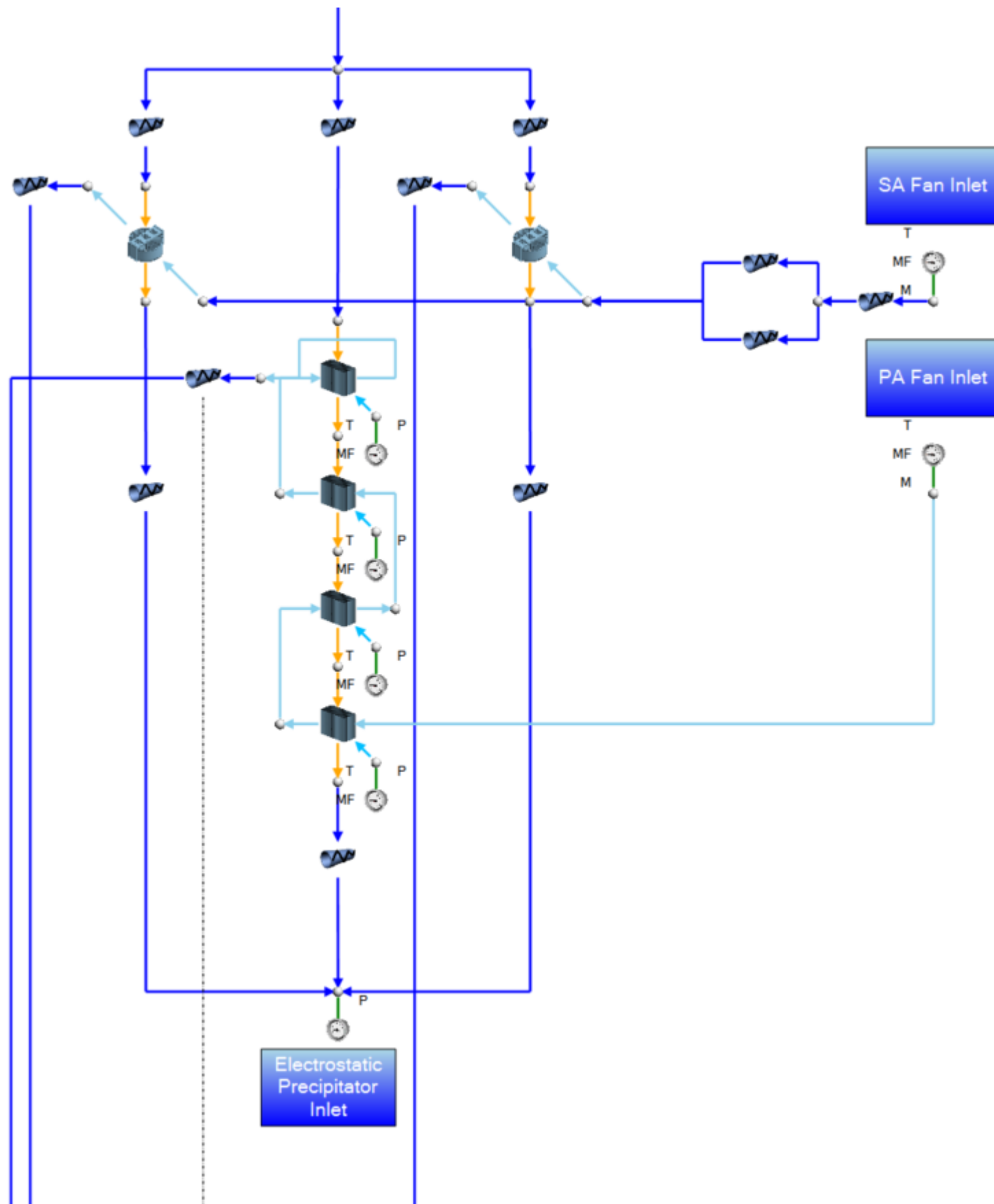
Model mid-low section:



Model low section:



Air handling section:



Appendix U. Flownex component model C# scripts

The C# scripts in the Flownex compound components developed in this project are presented in the following pages:

- Combustion analysis script – page 202 to 217
- Furnace geometry script – page 218 to 236
- Furnace heat transfer script – page 237 to 307
- Heat exchanger script – page 308 to 402
- Sling tube screen script – page 403 to 472
- Tubular air heater script – page 473 to 543
- Rotary air heater script – page 544 to 568

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll
using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.DataReferences.Materials;

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Thermal fluid properties
    IPS.Properties.Double _mfuelmain;
    IPS.Properties.Double _mfueladd1;
    IPS.Properties.Double _mair;
    IPS.Properties.Double _Tfuelmain;
    IPS.Properties.Double _Tfueladd1;
    IPS.Properties.Double _Tair;
    IPS.Properties.Double _Pamb;

    //Mass fraction of fuel air mix
    IPS.Properties.DataReferenceProperty _FuelAirmix;
    IPS.Properties.DataReferenceProperty _Products;
    IPS.Properties.Double _YUC;
    IPS.Properties.List<IPS.Properties.Double> _MassFractionsfuelmain;
    IPS.Properties.List<IPS.Properties.Double> _MassFractionsfueladd1;
    IPS.Properties.List<IPS.Properties.Double> _MassFractionsair;

    //HHV and cp
    IPS.Properties.Double _HHVfuelmain;
    IPS.Properties.Double _HHVfueladd1;
    IPS.Properties.Double _cpfueladd1;

    //Outputs

    //Mass flow rate and fuel constituents of the fuel air mixture
    IPS.Properties.Double _mmix;
    IPS.Properties.List<IPS.Properties.Double> _MassFractionsmix;

    //Outlet temperature and massfractions
    IPS.Properties.Double _Tproductout;

```

```

IPS.Properties.Double _hflame;
IPS.Properties.Double _mproducts;
IPS.Properties.Double _Qproduct;
IPS.Properties.List<IPS.Properties.Double> _MassFractionsout;

//Others
double MUC;
double HHVUC;
double Mreactant;
double Moxygen;
double Mproduct;
double mreactant;
double Sboxygen;
double Sbproduct;
double cpcoal;
double hfuelmainin;
double hfueladd1in;
double hfueladd2in;
double hair;

//do pre simulation initialisation here
public override void Initialise()
{

}

//do post simulation cleanup here
public override void Cleanup()
{
}

//script main execution function - called every cycle
public override void Execute(double Time)
{
    //Clear the mass of the product
    _mproducts.Value = 0;

    //Calculate Air Fuel mixture mass flow rate
    _mmix.Value = _mfueladd1.Value + _mfuelmain.Value + _mair.Value;

    if (_mmix.Value == 0)
    {

        _mmix.Value = 0;
    }
}

```

```

    }
    else
    {

        // Process massfraction list into individual

        List<double> fractionsfuelmain = new List<double>();
        List<double> fractionsfueladd1 = new List<double>();
        List<double> fractionsair = new List<double>();
        List<double> fractionsmix = new List<double>();

        for(int i = 0; i < _MassFractionsfuelmain.Count; i++)
        {
            fractionsfuelmain.Add(_MassFractionsfuelmain[i].Value);
            fractionsfueladd1.Add(_MassFractionsfueladd1[i].Value);
            fractionsair.Add(_MassFractionsair[i].Value);
            fractionsmix.Add(_MassFractionsmix[i].Value);
        }

        double[] MassFractionsfuelmain = fractionsfuelmain.ToArray();
        double[] MassFractionsfueladd1 = fractionsfueladd1.ToArray();
        double[] MassFractionsair = fractionsair.ToArray();
        double[] MassFractionsmix = fractionsmix.ToArray();

        MixedFluids fluidin = (MixedFluids)_FuelAirmix.Value;
        MixedFluids fluidout = (MixedFluids)_Products.Value;

        //Create lists to store the seperated data
        double[] mmixcomposition = new double[fractionsmix.Count];
        double[] mfuelmaincomposition = new
double[fractionsfuelmain.Count];
        double[] mfueladd1composition = new
double[fractionsfueladd1.Count];
        double[] maircomposition = new double[fractionsair.Count];
        double[] mproductsout = new double[MassFractionsout.Count];
        double[] reactantspecie = new double[fractionsfuelmain.Count];
        double[] oxygenspecie = new double[fractionsfueladd1.Count];
        double[] productspecie = new double[fractionsfueladd1.Count];

        //Convert YUC(%) to MUC

        MUC = _YUC.Value / 100;

        //Process input mass fractions
        for(int i = 0; i < fractionsfuelmain.Count; i++)

```

```

{
    //Calculate mass of each constituent in each stream

    mfuelmaincomposition[i] = _mfuelmain.Value *
_MassFractionsfuelmain[i].Value;

    mmixcomposition[i] += mfuelmaincomposition[i];

    mfueladd1composition[i] = _mfueladd1.Value *
_MassFractionsfueladd1[i].Value;

    mmixcomposition[i] += mfueladd1composition[i];

    maircomposition[i] = _mair.Value *
_MassFractionsair[i].Value;

    mmixcomposition[i] += maircomposition[i];

    _MassFractionsmix[i].Value = mmixcomposition[i] /
_mmix.Value;
}

//For UC
mproductsout[6] = _mfuelmain.Value * MUC;

mmixcomposition[1] = mmixcomposition[1] - mproductsout[6];

//Pass the masses of constituents to the product list
for (int i = 0; i < fractionsmix.Count; i += 2)
{
    if(i == 6)
    {
        mproductsout[i / 2] = mmixcomposition[i] +
mmixcomposition[i - 1];
    }

    else if (i == 8)
    {
        mproductsout[i / 2] = mmixcomposition[i] +
mmixcomposition[i - 1];
    }

    else

```

Combustion analysis script

```

        {
            mproductsout[i / 2] = mmixcomposition[i];
        }
    }
    //Seperate Oxygen out for molar mass extraction
    oxygenspecie[8] = 1;

    //Stoichiometric analysis
    for (int i = 1; i < fractionsmix.Count ; i += 2)
    {
        //Oxidation of Hydrogen
        if(i == 3)
        {
            //Seperate the reactant and product out for
            molar mass extraction

            reactantspecie[i] = 1;
            productspecie[i + 1] = 1;

            //Extract molar mass
            Mreactant = 2 *
            fluidin.GetMolarMass(reactantspecie, Parent.Identifier.Value);
            Moxygen =
            fluidin.GetMolarMass(oxygenspecie, Parent.Identifier.Value);
            Mproduct =
            fluidin.GetMolarMass(productspecie, Parent.Identifier.Value);

            //Calculate Sb ratios
            Sboxygen = 0.5 * (Moxygen / Mreactant);
            Sbproduct = Mproduct / Mreactant;

            //Calculate mass of oxygen required for
            combustion and mass of the product

            mproductsout[4] -= Sboxygen *
            mmixcomposition[i];

            mproductsout[(i + 1) / 2] += Sbproduct *
            mmixcomposition[i];

            //Clear the array for later use
            reactantspecie[i] = 0;
            productspecie[i + 1] = 0;
        }

        else if(i == 5)
        {
            continue;
        }
    }

```

```

    }

    else if(i == 7)
    {
        continue;
    }

    //Oxidation of othere species
    else
    {
        //Seperate the reactant and product out for
        molar mass extraction

        reactantspecie[i] = 1;
        productspecie[i + 1] = 1;

        //Extract molar mass
        Mreactant =
fluidin.GetMolarMass(reactantspecie, Parent.Identifier.Value);
        Moxygen =
fluidin.GetMolarMass(oxygenspecie, Parent.Identifier.Value);
        Mproduct =
fluidin.GetMolarMass(productspecie, Parent.Identifier.Value);

        //Calculate Sb ratios
        Sboxygen = Moxygen / Mreactant;
        Sbproduct = Mproduct / Mreactant;

        //Calculate mass of oxygen required for
        combustion and mass of the product

        mproductsout[4] -= Sboxygen *
mmixcomposition[i];

        mproductsout[(i + 1) / 2] += Sbproduct *
mmixcomposition[i];

        //Clear the array for later use
        reactantspecie[i] = 0;
        productspecie[i + 1] = 0;
    }
}

//Convert mass into mass fractions
double[] MassFractionsproduct = new
double[_MassFractionsout.Count];

for (int i = 0; i < mproductsout.Length; i++)

```

```

    {
        _mproducts.Value += mproductsout[i];

        MassFractionsproduct[i] = mproductsout[i] / _mmix.Value;

        _MassFractionsout[i].Value = MassFractionsproduct[i];

    }

    //Calculate the flame enthalpy
    hfuelmainin = cpcoal * (_Tfuelmain.Value - 298.15);
    hfueladd1in = _cpfueladd1.Value * (_Tfueladd1.Value - 298.15);
    hair = fluidin.GetEnthalpy(_Tair.Value, _Pamb.Value,
MassFractionsair, Parent.Identifier.Value);
    _Qproduct.Value = (_mfuelmain.Value * hfuelmainin +
    _mfueladd1.Value * hfueladd1in) + _mair.Value * hair + (_mfuelmain.Value * _HHVfuelmain.Value
    + _mfueladd1.Value * _HHVfueladd1.Value) - mproductsout[6] * HHVUC;
    _hflame.Value = _Qproduct.Value / _mproducts.Value;
    _Tproductout.Value =
fluidout.GetTemperatureFromPH(_Pamb.Value, _hflame.Value, MassFractionsproduct,
Parent.Identifier.Value);

    }

}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{
    Execute(0.0);
}

//any processing you want to do while solving steady state
public override void ExecuteSteadyState()
{
    Execute(0.0);
}

//any processing you want to do after steady state
public override void ExecuteAfterSteadyState()
{
    Execute(0.0);
}

```



```

//constructor initialises parameters
public Script()
{
    //Inputs

    //Thermal fluid properties
    _mfuelmain = new IPS.Properties.Double();
    _mfueladd1 = new IPS.Properties.Double();
    _mair = new IPS.Properties.Double();
    _Tfuelmain = new IPS.Properties.Double();
    _Tfueladd1 = new IPS.Properties.Double();
    _Tair = new IPS.Properties.Double();
    _Pamb = new IPS.Properties.Double();

    //Mass fraction of flue air mix
    _FuelAirmix = new IPS.Properties.DataReferenceProperty(null);
    _FuelAirmix.Types = new Type[]{typeof(MixedFluids)};
    _Products = new IPS.Properties.DataReferenceProperty(null);
    _Products.Types = new Type[]{typeof(MixedFluids)};
    _YUC = new IPS.Properties.Double();
    _MassFractionsfuelmain = new IPS.Properties.List<IPS.Properties.Double>();
    _MassFractionsfueladd1 = new IPS.Properties.List<IPS.Properties.Double>();
    _MassFractionsair = new IPS.Properties.List<IPS.Properties.Double>();

    //HHV
    _HHVfuelmain = new IPS.Properties.Double();
    _HHVfueladd1 = new IPS.Properties.Double();
    _cpfueladd1 = new IPS.Properties.Double();

    //Outputs

    //Mass flow rate and fuel constituents of the fuel air mixture
    _mmix = new IPS.Properties.Double();
    _MassFractionsmix = new IPS.Properties.List<IPS.Properties.Double>();

    //Outlet temperature and massfractions
    _Tproductout = new IPS.Properties.Double();
    _hflame = new IPS.Properties.Double();
    _mproducts = new IPS.Properties.Double();
    _Qproduct = new IPS.Properties.Double();
    _MassFractionsout = new IPS.Properties.List<IPS.Properties.Double>();

    //units

```

```
//Inputs

//Thermal fluid properties
_mfuelmain.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mfueladd1.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mair.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_Tfuelmain.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tfueladd1.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tair.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Pamb.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

//HHV
_HHVfuelmain.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_HHVfueladd1.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_cpfueladd1.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;

//Outputs

//Mass flow rate and fuel constituents of the fuel air mixture
_mmix.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Outlet temperature and massfractions
_Tproductout.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Qproduct.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_hflame.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_mproducts.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Initilaise

//Inputs
//Thermal fluid properties
_mfuelmain.Value = 0;
_mfueladd1.Value = 0;
_mair.Value = 0;
_Tfuelmain.Value = 0;
_Tfueladd1.Value = 0;
_Tair.Value = 0;
_Pamb.Value = 0;

//Mass fraction of flue air mix
_YUC.Value = 0;

//HHV
_HHVfuelmain.Value = 0;
_HHVfueladd1.Value = 0;
```

```

_cpfueladd1.Value = 0;

//Outputs

//Mass flow rate and fuel constituents of the fuel air mixture
_mmix.Value = 0;

//Outlet temperature and massfractions
_Tproductout.Value = 0;
_hflame.Value = 0;
_Qproduct.Value = 0;
_mproducts.Value = 0;

//Others
HHVUC = 33820 * Math.Pow(10, 3);

//Others
Mreactant = 0;
Mxygen = 0;
Mproduct = 0;
mreactant = 0;
Sboxygen = 0;
Sbproduct = 0;
cpcoal = 710;
hfuelmainin = 0;
hfueladd1in = 0;
hair = 0;

}

//property declarations to make
//parameters visible to outside world

//Inputs

//Thermal fluid properties

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Main fuel mass flow rate")]
[GridOrder(11)]
public IPS.Properties.Double mfuelmain
{
    get
    {
        return _mfuelmain;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Additional fuel mass flow rate")]
[GridOrder(12)]
public IPS.Properties.Double mfueladd1
{
    get
    {
        return _mfueladd1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air mass flow rate")]
[GridOrder(13)]
public IPS.Properties.Double mair
{
    get
    {
        return _mair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Main fuel inlet temperature")]
[GridOrder(14)]
public IPS.Properties.Double Tfuelmain
{
    get
    {
        return _Tfuelmain;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Additional fuel inlet temperature")]
[GridOrder(15)]
public IPS.Properties.Double Tfueladd1
{
    get
    {
        return _Tfueladd1;
    }
}

```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air inlet temperature")]
[GridOrder(16)]
public IPS.Properties.Double Tair
{
    get
    {
        return _Tair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Ambient pressure")]
[GridOrder(17)]
public IPS.Properties.Double Pamb
{
    get
    {
        return _Pamb;
    }
}

//Mass fraction of fuel air mix

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Fuel Air mix")]
[GridOrder(18)]
public IPS.Properties.DataReferenceProperty FuelAirmix
{
    get
    {
        return _FuelAirmix;
    }
}

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Products")]
[GridOrder(19)]
public IPS.Properties.DataReferenceProperty Products
{
    get
    {
        return _Products;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Unburnt Carbon %")]
[GridOrder(20)]
public IPS.Properties.Double YUC
{
    get
    {
        return _YUC;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mass fractions of the main fuel")]
[GridOrder(21)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsfuelmain
{
    get
    {
        return _MassFractionsfuelmain;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mass fractions of the additional fuel")]
[GridOrder(22)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsfueladd1
{
    get
    {
        return _MassFractionsfueladd1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mass fractions of the air")]
[GridOrder(23)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsair
{
    get
    {
        return _MassFractionsair;
    }
}

```

```

}

//HHV and cp

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("HHV of the main fuel")]
[GridOrder(24)]
public IPS.Properties.Double HHVfuelmain
{
    get
    {
        return _HHVfuelmain;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("HHV of the additional fuel")]
[GridOrder(25)]
public IPS.Properties.Double HHVfueladd1
{
    get
    {
        return _HHVfueladd1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("cp of the additional fuel")]
[GridOrder(26)]
public IPS.Properties.Double cpfueladd1
{
    get
    {
        return _cpfueladd1;
    }
}

//Output

//Mass flow rate and fuel constituents of the fuel air mixture

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fuel Air mix mass flow rate")]
[GridOrder(27)]
public IPS.Properties.Double mmix

```

```

{
    get
    {
        return _mmix;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Mass fractions of fuel air mix")]
[GridOrder(28)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsmix
{
    get
    {
        return _MassFractionsmix;
    }
}

//Outlet temperature and massfractions

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Product outlet temperature")]
[GridOrder(29)]
public IPS.Properties.Double Tproductout
{
    get
    {
        return _Tproductout;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat release from the combustion")]
[GridOrder(30)]
public IPS.Properties.Double Qproduct
{
    get
    {
        return _Qproduct;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flame enthalpy")]
[GridOrder(31)]

```



```
public IPS.Properties.Double hflame
{
    get
    {
        return _hflame;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Product mass flow rate")]
[GridOrder(32)]
public IPS.Properties.Double mproducts
{
    get
    {
        return _mproducts;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Mass fractions of products")]
[GridOrder(33)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsout
{
    get
    {
        return _MassFractionsout;
    }
}
}
```

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll
using System;
using IPS.Properties;
using IPS.Scripting;

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs
    IPS.Properties.Double _Hfurnace;
    IPS.Properties.Double _Hi;
    IPS.Properties.Double _Hhoppertop;
    IPS.Properties.Double _numberofnodes;
    IPS.Properties.Double _WWthickness1;
    IPS.Properties.Double _WWthickness2;
    IPS.Properties.Double _Fthickness1;
    IPS.Properties.Double _Fthickness2;
    IPS.Properties.Double _piperoughness;
    IPS.Properties.Double _scale;
    IPS.Properties.Double _slag;
    IPS.Properties.Double _width;
    IPS.Properties.Double _depth;
    IPS.Properties.Double _Hanglechange;
    IPS.Properties.Double _Helixangle1;
    IPS.Properties.Double _Helixangle2;
    IPS.Properties.Double _kforward;
    IPS.Properties.Double _numeroftubes1;
    IPS.Properties.Double _numeroftubes2;
    IPS.Properties.Double _WWouterdiameter1;
    IPS.Properties.Double _WWouterdiameter2;

    //Exposed Water Wall Height inputs
    IPS.Properties.Double _WWheight1;
    IPS.Properties.Double _WWheight2;
    IPS.Properties.Double _WWheight3;
    IPS.Properties.Double _WWheight4;
    IPS.Properties.Double _WWheight5;
    IPS.Properties.Double _WWheight6;
    IPS.Properties.Double _WWheight7;
    IPS.Properties.Double _WWheight8;

```

```

IPS.Properties.Double _WWheight9;
IPS.Properties.Double _WWheight10;

//Output
IPS.Properties.List<IPS.Properties.Double> _Fthickness;
IPS.Properties.List<IPS.Properties.Double> _WWouterdiameter;
IPS.Properties.List<IPS.Properties.Double> _WWinnerdiameter;
IPS.Properties.List<IPS.Properties.Double> _pipelength;
IPS.Properties.Double _Lhoppermid;
IPS.Properties.List<IPS.Properties.Double> _pitchX;
IPS.Properties.List<IPS.Properties.Double> _pitchN;
IPS.Properties.List<IPS.Properties.Double> _numeroftubes;
IPS.Properties.List<IPS.Properties.Double> _WWthickness;
IPS.Properties.List<IPS.Properties.Double> _WWheight;
IPS.Properties.Double _pitchNavg;
IPS.Properties.Double _WWouterdiameteravg;

double pitchNsum;
double WWouterdiametersum;

//do pre simulation initialisation here
public override void Initialise()
{
    //Don't start initialise variables at the very first time step
    if(_Hfurnace.Value == 0)
    {
        _Hfurnace.Value = 0;
    }

    else
    {
        //Transfer Water Wall Height inputs to the list
        _WWheight[0].Value = _WWheight1;
        _WWheight[1].Value = _WWheight2;
        _WWheight[2].Value = _WWheight3;
        _WWheight[3].Value = _WWheight4;
        _WWheight[4].Value = _WWheight5;
        _WWheight[5].Value = _WWheight6;
        _WWheight[6].Value = _WWheight7;
        _WWheight[7].Value = _WWheight8;
        _WWheight[8].Value = _WWheight9;
        _WWheight[9].Value = _WWheight10;

        //Clean the sums
    }
}

```

```

pitchNsum = 0;

WWouterdiametersum = 0;

_Lhoppermid.Value = _Hhoppertop.Value - _Hi.Value;

for(int i = 0; i < _numberofnodes.Value; i++)
{
    if(_WWheight[i].Value <= _Hanglechange.Value)
    {
        if(i == 0)
        {
            _pipelength[i].Value =
(_WWheight[i].Value - _Hhoppertop.Value) / Math.Sin(_Helixangle1.Value);
        }

        else
        {
            _pipelength[i].Value =
(_WWheight[i].Value - _WWheight[i - 1].Value) / Math.Sin(_Helixangle1.Value);
        }

        _pitchX[i].Value = (2 * (_width.Value +
_depth.Value)) / (_numeroftubes1.Value - 1);

        _pitchN[i].Value = _pitchX[i].Value *
Math.Sin(_Helixangle1.Value);

        _numeroftubes[i].Value =

        _Fthickness[i].Value = _Fthickness1.Value;

        _WWthickness[i].Value = _WWthickness1.Value;

        _WWouterdiameter[i].Value =

        _WWouterdiameter1.Value;

        _WWinnerdiameter[i].Value =
_WWouterdiameter[i].Value - 2 * _WWthickness1.Value;

        pitchNsum += _pitchN[i].Value;

        WWouterdiametersum +=

        _WWouterdiameter[i].Value;

```

```

    }

    else
    {
        _pipelength[i].Value = (_WWheight[i].Value -
        _WWheight[i - 1].Value) / Math.Sin(_Helixangle2.Value);

        _pitchX[i].Value = (2 * (_width.Value +
        _depth.Value)) / (_numeroftubes2.Value - 1);

        _pitchN[i].Value = _pitchX[i].Value *
        Math.Sin(_Helixangle2.Value);

        _numeroftubes[i].Value =
        _numeroftubes2.Value;

        _Fthickness[i].Value = _Fthickness2.Value;

        _WWthickness[i].Value = _WWthickness2.Value;

        _WWouterdiameter[i].Value =
        _WWouterdiameter2.Value;

        _WWinnerdiameter[i].Value =
        _WWouterdiameter[i].Value - 2 * _WWthickness2.Value;

        pitchNsum += _pitchN[i].Value;

        WWouterdiametersum +=
        _WWouterdiameter[i].Value;

    }

}

_pitchNavg.Value = pitchNsum / _numberofnodes.Value;

_WWouterdiameteravg.Value = WWouterdiametersum /
_numberofnodes.Value;

}

}

//do post simulation cleanup here

```

```

public override void Cleanup()
{
}

//script main execution function - called every cycle
public override void Execute(double Time)
{
    //calculates the base 10 logarithm of the input
    //_Output.Value = Math.Log10(_Input.Value);
}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{
    Execute(0.0);
}

//any processing you want to do while solving steady state
public override void ExecuteSteadyState()
{
    Execute(0.0);
}

//any processing you want to do after steady state
public override void ExecuteAfterSteadyState()
{
    Execute(0.0);
}

//constructor initialises parameters
public Script()
{
    //Inputs
    _Hi = new IPS.Properties.Double();
    _Hfurnace = new IPS.Properties.Double();
    _Hhoppertop = new IPS.Properties.Double();
    _numberofnodes = new IPS.Properties.Double();
    _WWthickness1 = new IPS.Properties.Double();
    _WWthickness2 = new IPS.Properties.Double();
    _Fthickness1 = new IPS.Properties.Double();
    _Fthickness2 = new IPS.Properties.Double();
    _piperoughness = new IPS.Properties.Double();
    _scale = new IPS.Properties.Double();
    _slag = new IPS.Properties.Double();
}

```

```

_width = new IPS.Properties.Double();
_depth = new IPS.Properties.Double();
_Hanglechange = new IPS.Properties.Double();
_Helixangle1 = new IPS.Properties.Double();
_Helixangle2 = new IPS.Properties.Double();
_kforward = new IPS.Properties.Double();
_numberoftubes1 = new IPS.Properties.Double();
_numberoftubes2 = new IPS.Properties.Double();
_WWouterdiameter1 = new IPS.Properties.Double();
_WWouterdiameter2 = new IPS.Properties.Double();

//Exposed Water Wall Height inputs
_WWheight1 = new IPS.Properties.Double();
_WWheight2 = new IPS.Properties.Double();
_WWheight3 = new IPS.Properties.Double();
_WWheight4 = new IPS.Properties.Double();
_WWheight5 = new IPS.Properties.Double();
_WWheight6 = new IPS.Properties.Double();
_WWheight7 = new IPS.Properties.Double();
_WWheight8 = new IPS.Properties.Double();
_WWheight9 = new IPS.Properties.Double();
_WWheight10 = new IPS.Properties.Double();

//Outputs
_Fthickness = new IPS.Properties.List<IPS.Properties.Double>();
_WWouterdiameter = new IPS.Properties.List<IPS.Properties.Double>();
_WWwinnerdiameter = new IPS.Properties.List<IPS.Properties.Double>();
_Lhoppermid = new IPS.Properties.Double();
_pipelength = new IPS.Properties.List<IPS.Properties.Double>();
_pitchX = new IPS.Properties.List<IPS.Properties.Double>();
_pitchN = new IPS.Properties.List<IPS.Properties.Double>();
_numberoftubes = new IPS.Properties.List<IPS.Properties.Double>();
_WWthickness = new IPS.Properties.List<IPS.Properties.Double>();
_WWheight = new IPS.Properties.List<IPS.Properties.Double>();
_pitchNavg = new IPS.Properties.Double();
_WWouterdiameteravg = new IPS.Properties.Double();

//Unit

//Inputs
_Hi.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hhoppertop.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWthickness1.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWthickness2.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

```

```

_Fthickness1.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Fthickness2.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_piperoughness.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_scale.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_slag.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_width.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_depth.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Handlechange.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Helixangle1.DisplayUnitGroup = IPS.Units.UnitGroup.Angle;
_Helixangle2.DisplayUnitGroup = IPS.Units.UnitGroup.Angle;
_WWouterdiameter1.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWouterdiameter2.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Exposed Water Wall Height inputs
_WWheight1.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight2.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight3.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight4.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight5.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight6.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight7.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight8.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight9.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWheight10.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Outputs
_Lhoppermid.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_pitchNavg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_WWouterdiameteravg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Initilise
//Inputs
_Hi.Value = 1e-6;
_Hfurnace.Value = 0;
_Hhoppertop.Value = 1e-6;
_numberofnodes.Value = 0;
_WWthickness1.Value = 1e-6;
_WWthickness2.Value = 1e-6;
_Fthickness1.Value = 1e-7;
_Fthickness2.Value = 1e-7;
_piperoughness.Value = 1e-6;
_scale.Value = 1e-7;
_slag.Value = 1e-7;
_width.Value = 0;
_depth.Value = 0;

```



```

_Hanglechange.Value = 1e-6;
_kforward.Value = 1e-6;
_Helixangle1.Value = 1e-6;
_Helixangle2.Value = 1e-6;
_numberoftubes1.Value = 1e-6;
_numberoftubes2.Value = 1e-6;
_WWouterdiameter1.Value = 1e-6;
_WWouterdiameter2.Value = 1e-6;

//Exposed Water Wall Height inputs
_WWheight1.Value = 1e-6;
_WWheight2.Value = 1e-6;
_WWheight3.Value = 1e-6;
_WWheight4.Value = 1e-6;
_WWheight5.Value = 1e-6;
_WWheight6.Value = 1e-6;
_WWheight7.Value = 1e-6;
_WWheight8.Value = 1e-6;
_WWheight9.Value = 1e-6;
_WWheight10.Value = 1e-6;

//Outputs
_Lhoppermid.Value = 1e-6;
_pitchNavg.Value = 1e-6;
_WWouterdiameteravg.Value = 1e-6;

pitchNsum = 1e-6;

WWouterdiametersum = 1e-6;

}

//property declarations to make
//parameters visible to outside world

//Inputs

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace base height (Hopper Mid-plane)")]
[GridOrder(11)]
public IPS.Properties.Double Hi
{
    get
    {

```

```

        return _Hi;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace height")]
[GridOrder(12)]
public IPS.Properties.Double Hfurnace
{
    get
    {
        return _Hfurnace;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Hopper Top Height")]
[GridOrder(13)]
public IPS.Properties.Double Hhoppertop
{
    get
    {
        return _Hhoppertop;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of water wall nodes")]
[GridOrder(14)]
public IPS.Properties.Double numberofnodes
{
    get
    {
        return _numberofnodes;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall thickness 1")]
[GridOrder(15)]
public IPS.Properties.Double WWthickness1
{
    get
    {
        return _WWthickness1;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall thickness 2")]
[GridOrder(16)]
public IPS.Properties.Double WWthickness2
{
    get
    {
        return _WWthickness2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin thickness 1")]
[GridOrder(17)]
public IPS.Properties.Double Fthickness1
{
    get
    {
        return _Fthickness1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin thickness 2")]
[GridOrder(18)]
public IPS.Properties.Double Fthickness2
{
    get
    {
        return _Fthickness2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Pipe internal roughness")]
[GridOrder(19)]
public IPS.Properties.Double piperoughness
{
    get
    {
        return _piperoughness;
    }
}

```

```
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fouling scale thickness")]
[GridOrder(20)]
public IPS.Properties.Double scale
{
    get
    {
        return _scale;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fouling slag thickness")]
[GridOrder(21)]
public IPS.Properties.Double slag
{
    get
    {
        return _slag;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace width")]
[GridOrder(22)]
public IPS.Properties.Double width
{
    get
    {
        return _width;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace depth")]
[GridOrder(23)]
public IPS.Properties.Double depth
{
    get
    {
        return _depth;
    }
}
```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Height at which helical angle changes")]
[GridOrder(24)]
public IPS.Properties.Double Hanglechange
{
    get
    {
        return _Hanglechange;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall pipe helix angle 1")]
[GridOrder(25)]
public IPS.Properties.Double Helixangle1
{
    get
    {
        return _Helixangle1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall pipe helix angle 2")]
[GridOrder(26)]
public IPS.Properties.Double Helixangle2
{
    get
    {
        return _Helixangle2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("K forward")]
[GridOrder(27)]
public IPS.Properties.Double kforward
{
    get
    {
        return _kforward;
    }
}

```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of tubes in water wall membrane 1")]
[GridOrder(28)]
public IPS.Properties.Double numberoftubes1
{
    get
    {
        return _numberoftubes1;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of tubes in water wall membrane 2")]
[GridOrder(29)]
public IPS.Properties.Double numberoftubes2
{
    get
    {
        return _numberoftubes2;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall pipe outer diameter 1")]
[GridOrder(30)]
public IPS.Properties.Double WWouterdiameter1
{
    get
    {
        return _WWouterdiameter1;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall pipe outer diameter 2")]
[GridOrder(31)]
public IPS.Properties.Double WWouterdiameter2
{
    get
    {
        return _WWouterdiameter2;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 1 elevation")]
[GridOrder(32)]
public IPS.Properties.Double WWheight1
{
    get
    {
        return _WWheight1;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 2 elevation")]
[GridOrder(33)]
public IPS.Properties.Double WWheight2
{
    get
    {
        return _WWheight2;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 3 elevation")]
[GridOrder(34)]
public IPS.Properties.Double WWheight3
{
    get
    {
        return _WWheight3;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 4 elevation")]
[GridOrder(35)]
public IPS.Properties.Double WWheight4
{
    get
    {
        return _WWheight4;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
```

```
[DisplayName("Water wall section 5 elevation")]
[GridOrder(36)]
public IPS.Properties.Double WWheight5
{
    get
    {
        return _WWheight5;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 6 elevation")]
[GridOrder(37)]
public IPS.Properties.Double WWheight6
{
    get
    {
        return _WWheight6;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 7 elevation")]
[GridOrder(38)]
public IPS.Properties.Double WWheight7
{
    get
    {
        return _WWheight7;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 8 elevation")]
[GridOrder(39)]
public IPS.Properties.Double WWheight8
{
    get
    {
        return _WWheight8;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 9 elevation")]
```



```

[GridOrder(40)]
public IPS.Properties.Double WWheight9
{
    get
    {
        return _WWheight9;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall section 10 elevation")]
[GridOrder(41)]
public IPS.Properties.Double WWheight10
{
    get
    {
        return _WWheight10;
    }
}

//Outputs

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Membrane wall fin thickness")]
[GridOrder(42)]
public IPS.Properties.List<IPS.Properties.Double> Fthickness
{
    get
    {
        return _Fthickness;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Water wall outer diameters")]
[GridOrder(43)]
public IPS.Properties.List<IPS.Properties.Double> WWouterdiameter
{
    get
    {
        return _WWouterdiameter;
    }
}

[PropertyUsage(UseProperty.RESULT)]

```

```
[DisplayName("Water wall inner diameters")]
[GridOrder(44)]
public IPS.Properties.List<IPS.Properties.Double> WWinnerdiameter
{
    get
    {
        return _WWinnerdiameter;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Hopper upper half elevations")]
[GridOrder(45)]
public IPS.Properties.Double Lhoppermid
{
    get
    {
        return _Lhoppermid;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Water wall pipe section length")]
[GridOrder(46)]
public IPS.Properties.List<IPS.Properties.Double> pipelength
{
    get
    {
        return _pipelength;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Membrane wall axial pitch")]
[GridOrder(47)]
public IPS.Properties.List<IPS.Properties.Double> pitchX
{
    get
    {
        return _pitchX;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Membrane wall normal pitch")]
```

```

[GridOrder(48)]
public IPS.Properties.List<IPS.Properties.Double> pitchN
{
    get
    {
        return _pitchN;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of tubes in membrane wall")]
[GridOrder(49)]
public IPS.Properties.List<IPS.Properties.Double> numeroftubes
{
    get
    {
        return _numeroftubes;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Water wall pipe thickness")]
[GridOrder(50)]
public IPS.Properties.List<IPS.Properties.Double> WWthickness
{
    get
    {
        return _WWthickness;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Water wall height")]
[GridOrder(51)]
public IPS.Properties.List<IPS.Properties.Double> WWheight
{
    get
    {
        return _WWheight ;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Average water wall normal pitch")]
[GridOrder(52)]

```

```
public IPS.Properties.Double pitchNavg
{
    get
    {
        return _pitchNavg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Average water wall outer diameter")]
[GridOrder(53)]
public IPS.Properties.Double WWouterdiameteravg
{
    get
    {
        return _WWouterdiameteravg ;
    }
}

}
```

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll
using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.DataReferences.Materials;

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Thermal fluid properties
    IPS.Properties.Double _mfuel;
    IPS.Properties.Double _mproducts;
    IPS.Properties.Double _Tflame;
    IPS.Properties.Double _Tfollowsurface;
    IPS.Properties.Double _Pamb;

    //Geometrical parameters
    IPS.Properties.Double _Vfurnace;
    IPS.Properties.Double _Afurnace;
    IPS.Properties.Double _Afurnacewall;
    IPS.Properties.Double _Aexitplane;
    IPS.Properties.Double _Arefrac;
    IPS.Properties.Double _Hfurnace;
    IPS.Properties.Double _dfurnace;
    IPS.Properties.Double _bfurnace;
    IPS.Properties.Double _Hburner;
    IPS.Properties.Double _Hhoppermid;
    IPS.Properties.Double _Hhoppertop;
    IPS.Properties.Double _Ahopperupperhalf;
    IPS.Properties.List<IPS.Properties.Double> _WWheight;
    IPS.Properties.List<IPS.Properties.Double> _HwwHfurnace;
    IPS.Properties.List<IPS.Properties.Double> _HwwHfurnacemean;

    //Burner height calculation

    //Height of each burner row
    IPS.Properties.Double _Hrow1;
    IPS.Properties.Double _Hrow2;
    IPS.Properties.Double _Hrow3;

```

```
IPS.Properties.Double _Hrow4;
IPS.Properties.Double _Hrow5;
IPS.Properties.Double _Hrow6;
IPS.Properties.Double _Hrow7;
IPS.Properties.Double _Hrow8;
IPS.Properties.Double _Hrow9;
IPS.Properties.Double _Hrow10;

//Mass flow distribution per row
IPS.Properties.Double _mrow1;
IPS.Properties.Double _mrow2;
IPS.Properties.Double _mrow3;
IPS.Properties.Double _mrow4;
IPS.Properties.Double _mrow5;
IPS.Properties.Double _mrow6;
IPS.Properties.Double _mrow7;
IPS.Properties.Double _mrow8;
IPS.Properties.Double _mrow9;
IPS.Properties.Double _mrow10;

//Flyash ratios
IPS.Properties.Double _Yflyash;

//Other parameters
IPS.Properties.Double _fradloss;

//Mass fraction of flue gas
IPS.Properties.DataReferenceProperty _FluidDataReference;
IPS.Properties.List<IPS.Properties.Double> _MassFractions;

//Furnace Specs
IPS.Properties.Double _delx;
IPS.Properties.Double _Walltype;
IPS.Properties.Double _Fueltypeforflame;
IPS.Properties.Double _Fluetypeforashsize;
IPS.Properties.Double _Fueltypeforeta;
IPS.Properties.Double _Firingconfig;
IPS.Properties.Double _Comboforxi;

//High ash loading effect inputs and constants
IPS.Properties.Double _HighAshLoading;

//Outputs

//Effective burner height
```

```
IPS.Properties.Double _HBurnerEff;

//Hopper Height Ratio
IPS.Properties.Double _HhopperHfurnace;
IPS.Properties.Double _HhopperHfurnacemean;

//Total mass flow going into the burners
IPS.Properties.Double _mtotal;

//Ash flow rates
IPS.Properties.Double _mFA;
IPS.Properties.Double _mBA;

//Mass fraction of flue gas and mass of the product out
IPS.Properties.List<IPS.Properties.Double> _MassFractionsout;
IPS.Properties.Double _mproductsout;

//Outlet temperature and enthalpy
IPS.Properties.Double _Tfgin;
IPS.Properties.Double _Tfgout;
IPS.Properties.Double _hfgout;

//Intermediate step coefficients
IPS.Properties.Double _kcoke;
IPS.Properties.Double _x1;
IPS.Properties.Double _x2;
IPS.Properties.Double _xi;
IPS.Properties.Double _xirefrac;
IPS.Properties.Double _A;
IPS.Properties.Double _B;
IPS.Properties.Double _dfa;

//Intermediate step coefficients
IPS.Properties.Double _sfurnace;
IPS.Properties.Double _xb;
IPS.Properties.Double _Mflame;
IPS.Properties.Double _xang;
IPS.Properties.Double _beta;
IPS.Properties.Double _ppn;

//Non-uniformaty factors
IPS.Properties.Double _etafurnace;
IPS.Properties.List<IPS.Properties.Double> _etaww;
IPS.Properties.Double _etahoppermid;
IPS.Properties.Double _etaintegral;
```

```
IPS.Properties.Double _etaintegralnohopper;
IPS.Properties.Double _etadiff;
IPS.Properties.List<IPS.Properties.Double> _etawwadj;
IPS.Properties.Double _etaadjintegral;
IPS.Properties.Double _etaadjintegralnohopper;
IPS.Properties.Double _etahoppermidadj;

//Heat transfer
IPS.Properties.Double _Qfg;
IPS.Properties.Double _qfg;
IPS.Properties.Double _Qradout;
IPS.Properties.Double _Qwalls;
IPS.Properties.Double _Qhopperupperhalf;
IPS.Properties.Double _Qremain;
IPS.Properties.Double _qww;
IPS.Properties.List<IPS.Properties.Double> _qwwnode;
IPS.Properties.List<IPS.Properties.Double> _Qww;
IPS.Properties.Double _Qwwsum;

//Coefficient check
IPS.Properties.Double _VC;
IPS.Properties.Double _kg;
IPS.Properties.Double _kfa;
IPS.Properties.Double _kradflame;
IPS.Properties.Double _epiflame;
IPS.Properties.Double _epifurnace;
IPS.Properties.Double _psiavg;

//High ash loading volume flow rate
IPS.Properties.Double _rhofginfurnace;
IPS.Properties.Double _mfg;
IPS.Properties.Double _Vfg;

//High ash loading effect coefficient check
IPS.Properties.Double _coeffA;
IPS.Properties.Double _Qfracabs;
IPS.Properties.Double _Qfracscatter;
IPS.Properties.Double _coeffgamma;
IPS.Properties.Double _coeffbeta;
IPS.Properties.Double _Lp;
IPS.Properties.Double _epiG;
IPS.Properties.List<IPS.Properties.Double> _coeffa;
IPS.Properties.List<IPS.Properties.Double> _coeffb0;
IPS.Properties.List<IPS.Properties.Double> _coeffb1;
IPS.Properties.List<IPS.Properties.Double> _coeffkG;
```



```
IPS.Properties.Double _KeG;
IPS.Properties.Double _coeffKcoke;
IPS.Properties.Double _phiGandP;
IPS.Properties.Double _epiGandP;
IPS.Properties.Double _Tfgavg;

//Check enthalpies
IPS.Properties.Double _TBA;
IPS.Properties.Double _hBA;
IPS.Properties.Double _hflame;
IPS.Properties.Double _hproductsin;

//Heights Relative to Hopper Tops
IPS.Properties.List<IPS.Properties.Double> _HwwHhoppertop;

//Check Areas
IPS.Properties.List<IPS.Properties.Double> _Afurnacewallsectionalcheck;
IPS.Properties.Double _Afurnacewallcheck;
IPS.Properties.Double _Afurnacetotalcheck;

//Effective burner height fraction terms
double nume;
double denome;

//Constants
double sigma;
double dfa;
double pfurnace;

//Non-uniformaty factor coefficients
double c1;
double c2;
double lambda;
double mu;
double sigmahf;
double inerfchopper;
double erfchopper;

//Others
double mproductsout;
double MBA;
double Mproductsout;
double MMfg;
double gammaH2O;
```

```

double gamman;
double ppH2O;
double ppn;
double psiavg;
double h_flame;
double h_product;
double VC;
double kg;
double rhofg;
double kfa;
double kflame;
double epiflame;
double epifurnace;
double HburnertoHfurnace;

//erfc function constants and variables
double inerfcfurnace;
double t ;
double erfcfurnace;
double a1;
double a2;
double a3;
double a4;
double a5;
double p;
int sign;

//High Ash Loading constants
double rhoash;

//do pre simulation initialisation here
public override void Initialise()
{
    //Pre-simulation cleaning

    _xirefrac.Value = 0.1;

    _Hburner.Value = 1e-6;

    _HBurnerEff.Value = 1e-6;

    _xb.Value = 1e-6;

    _Mflame.Value = 1e-6;

```

```
_sfurnace.Value = 10;

_xang.Value = 1e-6;

_etafurnace.Value = 1e-6;

_etaintegral.Value = 1e-6;

_etaadjintegral.Value = 1e-6;

_etahoppermidadj.Value = 1e-6;

_etaintegralnohopper.Value = 1e-6;

_etahoppermid.Value = 1e-6;

_etaadjintegralnohopper.Value = 1e-6;

_VC.Value = 15000;

_kg.Value = 2.8;

_kfa.Value = 80;

_kradflame.Value = 1.8;

_epiflame.Value = 0.5;

_epifurnace.Value = 0.5;

_psiavg.Value = 1;

for(int i = 0; i < _etawwadj.Count; i++)
{
    _etawwadj[i].Value = 0;
}

if(_mfuel.Value <= 1e-6)
{
    _mfuel.Value = _mfuel.Value;
}

else
{
```

coal

```
//Process furnace specs to coefficients

if(_Fueltypeforflame.Value == 0) //Default: High volatile or reactive
{
    _A.Value = 0.59;
    _B.Value = 0.5;
    _x1.Value = 0.5;
}

else if(_Fueltypeforflame.Value == 1) //Gas or fuel oil
{
    _A.Value = 0.54;
    _B.Value = 0.2;
    _x1.Value = 1; // Assume low reactivity
}

else if(_Fueltypeforflame.Value == 2) //Low reactive fuels
{
    _A.Value = 0.56;
    _B.Value = 0.5;
    _x1.Value = 1; // Anthracite
}

else //Use default case
{
    _A.Value = 0.59;
    _B.Value = 0.5;
    _x1.Value = 0.5;
}

if(_Fluetyperashsize.Value == 0) //Default case
{
    _dfa.Value = 16e-6;
}

else if(_Fluetyperashsize.Value == 1)
{
    _dfa.Value = 13e-6;
}

else if(_Fluetyperashsize.Value == 2)
{
    _dfa.Value = 20e-6;
}
```

```

else //Use default value
{
    _dfa.Value = 16e-6;
}

if(_Fueltypeforeta.Value == 0) //Default: Bituminous Coal
{
    c1 = 1.9745;
    c2 = 1.8080;
    lambda = 0.9821;
    mu = 0.2275;
    sigmahf = 0.2371;
}

else if(_Fueltypeforeta.Value == 1) //Lignite
{
    c1 = 1.4728;
    c2 = 1.3324;
    lambda = 1.7131;
    mu = 0.2275;
    sigmahf = 0.1907;
}

else if(_Fueltypeforeta.Value == 2) //Oil/ gas
{
    c1 = 1.5214;
    c2 = 2.9035;
    lambda = 1.575;
    mu = 0.2675;
    sigmahf = 0.1968;
}

else //Use default value
{
    c1 = 1.9745;
    c2 = 1.8080;
    lambda = 0.9821;
    mu = 0.2275;
    sigmahf = 0.2371;
}

if(_Firingconfig.Value == 0) //Default: suspension firing
{
    _x2.Value = 0.1;
}

```

```

}

else if(_Firingconfig.Value == 1) //Default: Grate firing
{
    _x2.Value = 0.03;
}

else //Use default value
{
    _x2.Value = 0.1;
}

if(_Comboforxi.Value == 0) //Default: Water wall with Coal
{
    _xi.Value = 0.45;
}

else if(_Comboforxi.Value == 1) //Water Wall with Gas or Gas-Heavy
{
    _xi.Value = 0.65;
}

else if(_Comboforxi.Value == 2) //Water wall with heavy oil
{
    _xi.Value = 0.55;
}

else if(_Comboforxi.Value == 3) //Water wall with all grate-fired
{
    _xi.Value = 0.6;
}

else if(_Comboforxi.Value == 4) //Water wall with other fuels
{
    _xi.Value = 0.2;
}

else if(_Comboforxi.Value == 5) //Water wall with refractory backing
{
    _xi.Value = 0.2;
}

else //Use default value

```

oil mix

fuels

Furnace heat transfer script

```

{
    _xi.Value = 0.45;
}

_kcoke.Value = 10.2;

//Clean out the values

_mttotal.Value = 0;

_Qwwsum.Value = 0;

nume = 0;

denome = 0;

_mfg.Value = 0;

_epiG.Value = 0;

_Afurnacewallcheck.Value = 0;

_Afurnacetotalcheck.Value = 0;

//dfa

dfa = _dfa.Value;

//Calculate the effective burner height

nume += _mrow1.Value * _mfuel.Value * (_Hrow1.Value -
_Hoppermid.Value);

denome += _mrow1.Value * _mfuel.Value;

_mttotal.Value += _mrow1.Value;

nume += _mrow2.Value * _mfuel.Value * (_Hrow2.Value -
_Hoppermid.Value);

denome += _mrow2.Value * _mfuel.Value;

_mttotal.Value += _mrow2.Value;

nume += _mrow3.Value * _mfuel.Value * (_Hrow3.Value -

```

```

_Hoppermid.Value);

denome += _mrow3.Value * _mfuel.Value;

_mttotal.Value += _mrow3.Value;

nume += _mrow4.Value * _mfuel.Value * (_Hrow4.Value -
_Hoppermid.Value);

denome += _mrow4.Value * _mfuel.Value;

_mttotal.Value += _mrow4.Value;

nume += _mrow5.Value * _mfuel.Value * (_Hrow5.Value -
_Hoppermid.Value);

denome += _mrow5.Value * _mfuel.Value;

_mttotal.Value += _mrow5.Value;

nume += _mrow6.Value * _mfuel.Value * (_Hrow6.Value -
_Hoppermid.Value);

denome += _mrow6.Value * _mfuel.Value;

_mttotal.Value += _mrow6.Value;

nume += _mrow7.Value * _mfuel.Value * (_Hrow7.Value -
_Hoppermid.Value);

denome += _mrow7.Value * _mfuel.Value;

_mttotal.Value += _mrow7.Value;

nume += _mrow8.Value * _mfuel.Value * (_Hrow8.Value -
_Hoppermid.Value);

denome += _mrow8.Value * _mfuel.Value;

_mttotal.Value += _mrow8.Value;

nume += _mrow9.Value * _mfuel.Value * (_Hrow9.Value -
_Hoppermid.Value);

denome += _mrow9.Value * _mfuel.Value;

```



```

        _mtotal.Value += _mrow9.Value;

        nume += _mrow10.Value * _mfuel.Value * (_Hrow10.Value -
_Hoppermid.Value);

        denome += _mrow10.Value * _mfuel.Value;

        _mtotal.Value += _mrow10.Value;

        _HBurnerEff.Value = (nume / denome) + _Hoppermid.Value;

        _Hburner.Value = _HBurnerEff.Value;

//Pre-simulation data process

//Mean Beam Length

        _sfurnace.Value = 3.6 * (_Vfurnace.Value / _Afurnace.Value);

//Coefficients

        _xb.Value = ((_Hburner.Value - _Hoppermid.Value) /
(_Hfurnace.Value - _Hoppermid.Value)) + _delx.Value;

        _Mflame.Value = _A.Value - (_B.Value * _xb.Value);

        if(_Walltype.Value == 0) //Default to be panel wall
        {
            _xang.Value = 1;
        }

        else if(_Walltype.Value == 1) // Membrane wall
        {
            _xang.Value = 1.15484 * Math.Pow((_bfurnace.Value /
_dofurnace.Value), -0.850475);
        }

        else // Set to default
        {
            _xang.Value = 1;
        }

        psiavg = (_xang.Value * _xi.Value * (_Afurnace.Value - _Arefrac.Value
- _Aexitplane.Value) + _xirefrac.Value * _Arefrac.Value + _xi.Value * _Aexitplane.Value) /

```

```

(_Afurnace.Value);

    _psiavg.Value = psiavg;

    // Height to height ratio of the burner

    HburnertoHfurnace = ((_Hburner.Value - _Hhoppermid.Value) /
(_Hfurnace.Value - _Hhoppermid.Value));

    //Non-uniformity factor of furnace exit plane

    //Term inside the furnace erfc function

    inerfcfurnace = (Math.Pow(mu, c2) + lambda * Math.Pow(sigmahf,
2) - 1) / (Math.Pow(2, 0.5) * sigmahf);

    //Erf function

    // Save the sign of x

    if (inerfcfurnace < 0)
    {
        sign = -1;
    }

    inerfcfurnace = Math.Abs(inerfcfurnace);

    // A&S formula 7.1.26
    t = 1.0 / (1.0 + p * inerfcfurnace);
    erfcfurnace = 1.0 - (((((a5 * t + a4) * t) + a3) * t + a2) * t + a1) * t *
Math.Exp(-inerfcfurnace * inerfcfurnace);
    erfcfurnace = sign * erfcfurnace;
    erfcfurnace = 1.0 - erfcfurnace;

    _etafurnace.Value = c1 * (lambda / 2) * Math.Exp((lambda / 2) * (2 *
Math.Pow(mu, c2) + lambda * Math.Pow(sigmahf, 2) - 2)) * erfcfurnace;

    //Find the non-uniformity factor for hopper upper half

    _HhopperHfurnace.Value = (_Hhoppertop.Value -
_Hhoppermid.Value) / (_Hfurnace.Value - _Hhoppermid.Value);

    _HhopperHfurnacemean.Value = _HhopperHfurnace.Value / 2;

```

```

    inerfchopper = (Math.Pow(mu, c2) + lambda * Math.Pow(sigmahf,
2) - _HhopperHfurnacemean.Value) / (Math.Pow(2, 0.5) * sigmahf);

    sign = 1;

    if (inerfchopper < 0)
    {
        sign = -1;
    }

    inerfchopper = Math.Abs(inerfchopper);
    t = 1.0 / (1.0 + p * inerfchopper);
    erfchopper = 1.0 - (((((a5 * t + a4) * t) + a3) * t + a2) * t + a1) * t *
Math.Exp(-inerfchopper * inerfchopper);
    erfchopper = sign * erfchopper;
    erfchopper = 1.0 - erfchopper;

    _etahoppermid.Value = c1 * (lambda / 2) * Math.Exp((lambda / 2) *
(2 * Math.Pow(mu, c2) + lambda * Math.Pow(sigmahf, 2) - 2 * _HhopperHfurnacemean.Value)) *
erfchopper;

//Water wall non-uniformaty factor calculation

double[] inerfcww = new double[_WWheight.Count];

double[] erfcww =new double[_WWheight.Count];

for(int i = 0; i < _WWheight.Count; i++)
{
    _HwwHfurnace[i].Value = (_WWheight[i].Value -
_Hhoppermid.Value) / (_Hfurnace.Value - _Hhoppermid.Value);

    _HwwHhoppertop[i].Value = (_WWheight[i].Value -
_Hhoppertop.Value) / (_Hfurnace.Value - _Hhoppertop.Value);

    if(i == 0)
    {
        _HwwHfurnacemean[i].Value =
(( _HwwHfurnace[i].Value - _HhopperHfurnace.Value) / 2) + _HhopperHfurnace.Value;

        _Afurnacewallsectionalcheck[i].Value =
_Afurnacewall.Value * _HwwHhoppertop[i].Value;

        _Afurnacewallcheck.Value +=

```

```

_Afurnacewallsectionalcheck[i].Value;
    }

    else
    {
        _HwwHfurnacemean[i].Value =
        ((_HwwHfurnace[i].Value - _HwwHfurnace[i - 1].Value) / 2) + _HwwHfurnace[i - 1].Value;

        _Afurnacewallsectionalcheck[i].Value =
        _Afurnacewall.Value * (_HwwHhoppertop[i].Value - _HwwHhoppertop[i - 1].Value);

        _Afurnacewallcheck.Value +=
        _Afurnacewallsectionalcheck[i].Value;
    }

    inerfcww[i] = (Math.Pow(mu, c2) + lambda *
    Math.Pow(sigmahf, 2) - _HwwHfurnacemean[i].Value) / (Math.Pow(2, 0.5) * sigmahf);

    sign = 1;
    if (inerfcww[i] < 0)
    {
        sign = -1;
    }

    inerfcww[i] = Math.Abs(inerfcww[i]);
    t = 1.0 / (1.0 + p * inerfcww[i]);
    erfcww[i] = 1.0 - (((((a5 * t + a4) * t) + a3) * t + a2) * t + a1)
    * t * Math.Exp(-inerfcww[i] * inerfcww[i]);
    erfcww[i] = sign * erfcww[i];
    erfcww[i] = 1.0 - erfcww[i];

    _etaww[i].Value = c1 * (lambda / 2) * Math.Exp((lambda /
    2) * (2 * Math.Pow(mu, c2) + lambda * Math.Pow(sigmahf, 2) - 2 * _HwwHfurnacemean[i].Value))
    * erfcww[i];
}

_Afurnacetotalcheck.Value = _Afurnacewallcheck.Value +
_Ahopperupperhalf.Value;

for(int i = 0; i < _WWheight.Count; i++)
{
    //Numerically integrate non-uniformaty factor

    if(i == 0)

```

```

{
    _etaintegral.Value += _etahoppermid.Value *
(_Ahopperupperhalf.Value / _Afurnacetotalcheck.Value);

    _etaintegral.Value += _etaww[i].Value *
(_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

    _etaintegralnohopper.Value += _etaww[i].Value
* (_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

}

else
{

    _etaintegral.Value += _etaww[i].Value *
(_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

    _etaintegralnohopper.Value += _etaww[i].Value
* (_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

}

}

_etadiff.Value = Math.Abs(1 - _etaintegral.Value) * 100;

_etafurnace.Value = etafurnace.Value / _etaintegral.Value;

_etahoppermidadj.Value = _etahoppermid.Value /
_etaintegral.Value;

for(int i = 0; i < _HwwHfurnacemean.Count; i++)
{
    _etawwadj[i].Value = _etaww[i].Value / _etaintegral.Value;

    //Integrate the adjusted non-uniformity factor

    if(i == 0)
    {
        _etaadjintegral.Value +=
_etahoppermidadj.Value * (_Ahopperupperhalf.Value / _Afurnacetotalcheck.Value);

        _etaadjintegral.Value += _etawwadj[i].Value *

```

```

(_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

        _etaadjintegralnohopper.Value +=
        _etawwadj[i].Value * (_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);
    }

    else
    {
        _etaadjintegral.Value += _etawwadj[i].Value *
        (_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);

        _etaadjintegralnohopper.Value +=
        _etawwadj[i].Value * (_Afurnacewallsectionalcheck[i].Value / _Afurnacetotalcheck.Value);
    }

}

if(_HighAshLoading == 1)
{
    //Initialise constants

    //b0 Coefficients
    _coeffb0[0].Value = 0.130;
    _coeffb0[1].Value = 0.595;
    _coeffb0[2].Value = 0.275;

    //b1 Coefficients
    _coeffb1[0].Value = 0.000265;
    _coeffb1[1].Value = -0.000150;
    _coeffb1[2].Value = -0.000115;

    //KG Coefficients
    _coeffkG[0].Value = 0.0;
    _coeffkG[1].Value = 0.824;
    _coeffkG[2].Value = 25.907;

    //Calculate Coefficient A
    _coeffA.Value = 3 / (2 * rhoash * _dfa.Value);

    //Calculate Qabs using emperical relation
    _Qfracabs.Value = 0.275 * Math.Pow((_dfa.Value / 1e-6),
0.298) - 0.305;

    //Calculate Qscattering using emperical realtion
    _Qfracscatter.Value = (6.2188e-3 - 1.0492e-2 * (_dfa.Value

```

```

/ 1e-6) + 7.287e-3 * Math.Pow((_dfa.Value / 1e-6), 2) - 2.1925e-5 * Math.Pow((_dfa.Value / 1e-6),
3)) / (1.851e-1 - 2.0405e-3 * Math.Pow((_dfa.Value / 1e-6), 2) + 6.254e-4 * Math.Pow((_dfa.Value
/ 1e-6), 3));

```

```

//Calculate Coefficient gamma
_coeffgamma.Value = Math.Pow( 1 + ((2 *
_Qfracscatter.Value) / _Qfracabs.Value), 0.5);

```

```

//Calculate Coefficient Beta
_coeffbeta.Value = (_coeffgamma.Value -1) /
(_coeffgamma.Value + 1);

```

```

    }
    else
    {
        denome = denome;
    }
}

```

```

//do post simulation cleanup here
public override void Cleanup()
{
}

```

```

//script main execution function - called every cycle
public override void Execute(double Time)
{

```

```

    // Set a random flame temperature in case the flame temperature data was
not transfered

```

```

    if(_Tflame.Value <= 300)
    {
        _Tflame.Value = 300; //Set a random flame temperature to avoid
NaN(in case flame temperature data was not transfered)
    }
    else
    {
        _Tflame.Value = _Tflame.Value;
    }

```

```

//Don't start initialise variables at the very first time step
if(_mproducts.Value == 0)
{
    _mproducts.Value = 0;
}

else if(_Tfgout.Value < 300)
{
    _Tfgout.Value = 1500;
}

else if(_Tfollowsurface.Value < 300)
{
    _Tfollowsurface.Value = 300;
}

else if(_Tfgavg.Value < 300)
{
    _Tfgavg.Value = 1500;
}

else
{
    _beta.Value = -2.18793 * Math.Pow(10, -6) *
Math.Pow((_Tfollowsurface.Value - 273.15), 2) + 4.36338 * Math.Pow(10, -3) *
(_Tfollowsurface.Value - 273.15) - 1.16738;

    //Assume furnace is at ambient pressure
    pfurnace = _Pamb.Value;

    List<double> fractions = new List<double>();
    foreach(IPS.Properties.Double val in _MassFractions)
    {
        fractions.Add(val.Value);
    }
    double[] MassFractions = fractions.ToArray();

    MixedFluids fluid = (MixedFluids)_FluidDataReference.Value;

    //Create a new lists for the products leaving the furnace

    _mFA.Value = _Yflyash * _mproducts.Value * MassFractions[0];

```



```

_mBA.Value = (1 - _Yflyash) * _mproducts.Value * MassFractions[0];

mproductsout = _mproducts.Value - _mBA.Value;

_mproductsout.Value = mproductsout;

MBA = _mBA.Value / _mproducts.Value;

Mproductsout = mproductsout / _mproducts.Value;

double[] Mproductout = new double[fractions.Count];

for(int i =0; i < fractions.Count ; i++)
{
    if(i == 0)
    {
        Mproductout[i] = (_Yflyash.Value *
_mproducts.Value * MassFractions[i]) / mproductsout;
    }

    else
    {
        Mproductout[i] = (_mproducts.Value *
MassFractions[i]) / mproductsout;
    }

    _MassFractionsout[i].Value = Mproductout[i];
}

// Find out the cp comtribute from BA

double[] PureAsh = new double[fractions.Count];

PureAsh[0] = 1;

_hBA.Value = fluid.GetEnthalpy(_TBA.Value, pfurnace, PureAsh,
Parent.Identifier.Value);

//Create an array for flue gas mass fraction

double[] Mfg = new double[fractions.Count];

double[] fgMolarMass = new double[fractions.Count - 2];

```

```

double[] Pureconstituent = new double[fractions.Count];

//Pre-loop clean up

_mfg.Value = 0;

for(int i = 1; i < (fractions.Count - 1); i++)
{
    Mfg[i] = Mproductout[i] / (1 - (Mproductout[0] +
Mproductout[6]));

    _mfg.Value += Mproductout[i] * _mproducts.Value;

    Pureconstituent[i] = 1;

    fgMolarMass[i - 1] = fluid.GetMolarMass(Pureconstituent,
Parent.Identifier.Value);

    Pureconstituent[i] = 0;
}

//Calculate tri-atomic gases volume and partial fraction

//Initialise

double[] gammafg = new double[fractions.Count - 2];

double[] ppfg = new double[fractions.Count - 2];

MMfg = fluid.GetMolarMass(Mfg, Parent.Identifier.Value);

for(int i = 0; i < (fractions.Count - 2); i++)
{
    gammafg[i] = (Mfg[i + 1] * MMfg) / fgMolarMass[i];

    ppfg[i] = _Pamb.Value * gammafg[i];
}

gammaH2O = gammafg[1];

gamman = gammafg[0] + gammafg[1] + gammafg[4];

ppH2O = ppfg[1];

```

```

ppn = ppfg[0] + ppfg[1] + ppfg[4];

_ppn.Value = ppn;

//Other coefficients

h_flame = fluid.GetEnthalpy(_Tflame.Value, pfurnace,
MassFractions, Parent.Identifier.Value);

_hflame.Value = h_flame;

_hproductsin.Value = (_mproducts.Value * _hflame.Value -
_mBA.Value * _hBA.Value) / _mproducts.Value;

_Tfgin.Value = fluid.GetTemperatureFromPH(pfurnace,
_hproductsin.Value, Mproductout, Parent.Identifier.Value);

h_product = fluid.GetEnthalpy(_Tfgout.Value, pfurnace,
Mproductout, Parent.Identifier.Value);

VC = ((_hproductsin.Value - h_product) / (_Tfgin.Value -
_Tfgout.Value)) * (mproductsout / _mfuel.Value);

_VC.Value = VC;

//Calculate furnace emissivity

if(_HighAshLoading.Value == 1)
{
    //Get Flue gas density

    _rhofginfurnace.Value = fluid.GetDensity(_Tfgavg.Value,
_Pamb.Value, Mfg, Parent.Identifier.Value);

    //Calculate Flue gas volume flow rate

    _Vfg.Value = _mfg.Value / _rhofginfurnace.Value;

    //Calculate Lp
    _Lp.Value = _mFA.Value / _Vfg.Value;

    //Calcaulate Coefficient ai and epsilon G

    //Pre-loop clean up

```

```

        _epiG.Value = 0;

        for(int i = 0; i < _coeffa.Count; i++)
        {
            _coeffa[i].Value = _coeffb0[i].Value +
            _coeffb1[i].Value * _Tfgavg.Value;

            _epiG.Value += _coeffa[i].Value * (1 -
            Math.Exp(-1 * _coeffkG[i].Value * ppn * 1e-5 * _sfurnace.Value));
        }

        //Calculate K emissivity of Gas
        _KeG.Value = -1 * (Math.Log(1 - _epiG.Value) /
        _sfurnace.Value);

        //Calculate coke particulated absorption coefficient
        _coeffKcoke.Value = _kcoke.Value * _x1.Value * _x2.Value
        * _Pamb.Value * 1e-6;

        //Calculate the optical thickness of gas particulate
        dispersion
        _phiGandP.Value = (_Qfracabs.Value * _coeffA.Value *
        _Lp.Value + _KeG.Value + _coeffKcoke.Value) * _sfurnace.Value * _coeffgamma.Value;

        //Calculate effective gas-particulate emissivity
        _epiGandP.Value = (1 - _coeffbeta.Value) * ((1 -
        Math.Exp(-1 * _phiGandP.Value)) / (1 + _coeffbeta.Value * Math.Exp(-1 * _phiGandP.Value)));

        //Update the effective flame emissivity
        epiflame = _epiGandP.Value;
        _epiflame.Value = epiflame;

        //Calculate furnace emmisivity
        epifurnace = _epiGandP.Value / ((psiavg * (1 -
        _epiGandP.Value)) + _epiGandP.Value);

        _epifurnace.Value = epifurnace;
    }

    else
    {

```

[illegible]

```

        _qfg.Value = _Qfg.Value / (_Afurnacewall.Value +
_Ahopperupperhalf.Value + _Aexitplane.Value); //Effective areas exposed to radiation

        _Qradout.Value = _qfg.Value * _Aexitplane.Value * etafurnace *
beta;

        _Qwalls.Value = _Qfg.Value - _Qradout.Value;

        _qww.Value = _Qwalls.Value / (_Afurnacewall.Value +
_Ahopperupperhalf.Value); //Effective heat flux to all the walls

        _Qhopperupperhalf.Value = _etahoppermidadj.Value * _qww.Value
* _Ahopperupperhalf.Value;

        _Qremain.Value = _Qwalls.Value - _Qhopperupperhalf.Value;

        //Clear up before the loop

        _Qwwsum.Value = 0.0;

        for(int i = 0; i < _HwwHfurnace.Count ; i++)
        {

            if(i == 0)
            {
                _qwwnode[i].Value = _etawwadj[i].Value *
_qww.Value;

                _Qww[i].Value = (_qwwnode[i].Value *
(_Afurnacewall.Value * _HwwHhoppertop[i].Value)) / 1000;
            }

            else
            {
                _qwwnode[i].Value = _etawwadj[i].Value *
_qww.Value;

                _Qww[i].Value = (_qwwnode[i].Value *
(_Afurnacewall.Value * (_HwwHhoppertop[i].Value - _HwwHhoppertop[i - 1].Value))) / 1000;
            }

            _Qwwsum.Value += _Qww[i].Value * 1000;

        }

```

```

    }

}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{
    Execute(0.0);
}

//any processing you want to do while solving steady state
public override void ExecuteSteadyState()
{
    Execute(0.0);
}

//any processing you want to do after steady state
public override void ExecuteAfterSteadyState()
{
    Execute(0.0);
}

//constructor initialises parameters
public Script()
{

    //inputs

    //Thermal fluid properties
    _mfuel = new IPS.Properties.Double();
    _mproducts = new IPS.Properties.Double();
    _Tflame = new IPS.Properties.Double();
    _Tfollowsurface = new IPS.Properties.Double();
    _Pamb = new IPS.Properties.Double();

    //Geometrical parameters
    _Vfurnace = new IPS.Properties.Double();
    _Afurnace = new IPS.Properties.Double();
    _Afurnacewall = new IPS.Properties.Double();
    _Aexitplane = new IPS.Properties.Double();
    _Arefrac = new IPS.Properties.Double();
    _Hfurnace = new IPS.Properties.Double();
    _Hhoppermid = new IPS.Properties.Double();

```

```

_dfurnace = new IPS.Properties.Double();
_bfurnace = new IPS.Properties.Double();
_Hburner = new IPS.Properties.Double();
_Hhoppertop = new IPS.Properties.Double();
_Ahopperupperhalf = new IPS.Properties.Double();
_WWheight = new IPS.Properties.List<IPS.Properties.Double>();
_HwwHfurnace = new IPS.Properties.List<IPS.Properties.Double>();
_HwwHfurnacemean = new IPS.Properties.List<IPS.Properties.Double>();

//Burner height

//Height of each burner row
_Hrow1 = new IPS.Properties.Double();
_Hrow2 = new IPS.Properties.Double();
_Hrow3 = new IPS.Properties.Double();
_Hrow4 = new IPS.Properties.Double();
_Hrow5 = new IPS.Properties.Double();
_Hrow6 = new IPS.Properties.Double();
_Hrow7 = new IPS.Properties.Double();
_Hrow8 = new IPS.Properties.Double();
_Hrow9 = new IPS.Properties.Double();
_Hrow10 = new IPS.Properties.Double();

//Mass flow distribution per row
_mrow1 = new IPS.Properties.Double();
_mrow2 = new IPS.Properties.Double();
_mrow3 = new IPS.Properties.Double();
_mrow4 = new IPS.Properties.Double();
_mrow5 = new IPS.Properties.Double();
_mrow6 = new IPS.Properties.Double();
_mrow7 = new IPS.Properties.Double();
_mrow8 = new IPS.Properties.Double();
_mrow9 = new IPS.Properties.Double();
_mrow10 = new IPS.Properties.Double();

//Flyash ratio
_Yflyash = new IPS.Properties.Double();

//Other parameters
_fradloss = new IPS.Properties.Double();

//Mass fraction of flue gas
_MassFractions = new IPS.Properties.List<IPS.Properties.Double>();
_FluidDataReference = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReference.Types = new Type[] {typeof(MixedFluids)};

```



```

//Furnace Specs
_delx = new IPS.Properties.Double();
_Walltype = new IPS.Properties.Double();
_Fueltypeforflame = new IPS.Properties.Double();
_Fluetypeforashsize = new IPS.Properties.Double();
_Fueltypeforeta = new IPS.Properties.Double();
_Firingconfig = new IPS.Properties.Double();
_Comboforxi = new IPS.Properties.Double();

//High ash loading effect inputs and constants
_HighAshLoading = new IPS.Properties.Double();

//Outputs

//Output
_HBurnerEff = new IPS.Properties.Double();

//Hopper Height Ratio
_HhopperHfurnace = new IPS.Properties.Double();
_HhopperHfurnacemean = new IPS.Properties.Double();

//Total mass flow going into the burners
_mttotal = new IPS.Properties.Double();

//Ash flow rates
_mFA = new IPS.Properties.Double();
_mBA = new IPS.Properties.Double();

//Mass fraction of flue gas and mass of the product out
_MassFractionsout = new IPS.Properties.List<IPS.Properties.Double>();
_mproductsout = new IPS.Properties.Double();

//Outlet temerature and enthalpy
_Tfgin = new IPS.Properties.Double();
_Tfgout = new IPS.Properties.Double();
_hfgout = new IPS.Properties.Double();

//Intermediate step coefficients
_kcoke = new IPS.Properties.Double();
_x1 = new IPS.Properties.Double();
_x2 = new IPS.Properties.Double();
_xi = new IPS.Properties.Double();
_xirefrac = new IPS.Properties.Double();
_A = new IPS.Properties.Double();

```

```

_B = new IPS.Properties.Double();
_dfa = new IPS.Properties.Double();

//Intermediate step coefficients
_sfurnace = new IPS.Properties.Double();
_xb = new IPS.Properties.Double();
_Mflame = new IPS.Properties.Double();
_xang = new IPS.Properties.Double();
_beta = new IPS.Properties.Double();
_ppn = new IPS.Properties.Double();

//Non-uniformaty factors
_etafurnace = new IPS.Properties.Double();
_etaww = new IPS.Properties.List<IPS.Properties.Double>();
_etahoppermid = new IPS.Properties.Double();
_etaintegral = new IPS.Properties.Double();
_etaintegralnohopper = new IPS.Properties.Double();
_etadiff = new IPS.Properties.Double();
_etawwadj = new IPS.Properties.List<IPS.Properties.Double>();
_etaadjintegral = new IPS.Properties.Double();
_etaadjintegralnohopper = new IPS.Properties.Double();
_etahoppermidadj = new IPS.Properties.Double();

//Heat transfer
_Qfg = new IPS.Properties.Double();
_qfg = new IPS.Properties.Double();
_Qradout = new IPS.Properties.Double();
_Qwalls = new IPS.Properties.Double();
_Qhopperupperhalf = new IPS.Properties.Double();
_Qremain = new IPS.Properties.Double();
_qww = new IPS.Properties.Double();
_Qww = new IPS.Properties.List<IPS.Properties.Double>();
_qwwnode = new IPS.Properties.List<IPS.Properties.Double>();
_Qwwsum = new IPS.Properties.Double();

//Coefficient check
_VC = new IPS.Properties.Double();
_kg = new IPS.Properties.Double();
_kfa = new IPS.Properties.Double();
_kradflame = new IPS.Properties.Double();
_epiflame = new IPS.Properties.Double();
_epifurnace = new IPS.Properties.Double();
_psiavg = new IPS.Properties.Double();

//High ash loading volume flow rate

```

```

_rhofginfurnace = new IPS.Properties.Double();
_mfg = new IPS.Properties.Double();
_Vfg = new IPS.Properties.Double();

//High ash loading effect coefficient check
_coeffA = new IPS.Properties.Double();
_Qfracabs = new IPS.Properties.Double();
_Qfracscatter = new IPS.Properties.Double();
_coeffgamma = new IPS.Properties.Double();
_coeffbeta = new IPS.Properties.Double();
_Lp = new IPS.Properties.Double();
_epiG = new IPS.Properties.Double();
_coeffa = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb0 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb1 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffkG = new IPS.Properties.List<IPS.Properties.Double>();
_KeG = new IPS.Properties.Double();
_coeffKcoke = new IPS.Properties.Double();
_phiGandP = new IPS.Properties.Double();
_epiGandP = new IPS.Properties.Double();
_Tfgavg = new IPS.Properties.Double();

//Check enthalpies
_TBA = new IPS.Properties.Double();
_hBA = new IPS.Properties.Double();
_hflame = new IPS.Properties.Double();
_hproductsin = new IPS.Properties.Double();

//Check Areas
_Afurnacewallcheck = new IPS.Properties.Double();
_Afurnacetotalcheck = new IPS.Properties.Double();
_Afurnacewallsectionalcheck = new
IPS.Properties.List<IPS.Properties.Double>();
_HwwHhoppertop = new IPS.Properties.List<IPS.Properties.Double>();

//units

//Thermal fluid properties
_mfuel.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mproducts.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_Tflame.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tfollowsurface.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Pamb.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

```

```

//Geometrical parameters
_Vfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Volume;
_Afurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Afurnacewall.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aexitplane.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Arefrac.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Hburner.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hhoppermid.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_dfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_bfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hhoppertop.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Ahopperupperhalf.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

//Burner height

//Height of each burner row
_Hrow1.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow2.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow3.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow4.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow5.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow6.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow7.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow8.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow9.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hrow10.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Mass flow distribution per row
_mrow1.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow2.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow3.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow4.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow5.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow6.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow7.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow8.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow9.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;
_mrow10.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;

//Flyash ratio
_Yflyash.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;

//Outputs

```

```

//Burner height
_HBurnerEff.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Total mass flow going into the burners
_mttotal.DisplayUnitGroup = IPS.Units.UnitGroup.Mass_Per_Mass;

//Ash flow rates
_mFA.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mBA.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Intermediate step coefficients
_dfa.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Mass of the product out
_mproductsout.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Outlet temperature and enthalpy
_Tfgin.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tfgout.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_hfgout.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;

//Intermediate step coefficients
_sfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_ppn.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

//Heat transfer
_Qfg.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_qfg.DisplayUnitGroup = IPS.Units.UnitGroup.Power_Flux;
_qww.DisplayUnitGroup = IPS.Units.UnitGroup.Power_Flux;
_Qradout.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qwalls.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qhopperupperhalf.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qremain.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qwwsum.DisplayUnitGroup = IPS.Units.UnitGroup.Power;

//High ash loading volume flow rate
_rhofginfurnace.DisplayUnitGroup = IPS.Units.UnitGroup.Density ;
_mfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_Vfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Volume;
_Tfgavg.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;

//Check enthalpies
_TBA.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_hBA.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_hflame.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;

```

```
_hproductsin.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
```

```
//Check Areas
```

```
_Afurnacewallcheck.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
```

```
_Afurnacetotalcheck.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
```

```
//Initilise all variables
```

```
//Thermal fluid properties
```

```
_mproducts.Value = 1e-6;
```

```
_mproductsout.Value = 1e-6;
```

```
_Tflame.Value = 300;
```

```
_Tfollowsurface.Value = 1e-6;
```

```
_Pamb.Value = 1e-6;
```

```
//Geometrical parameters
```

```
_Vfurnace.Value = 1e-6;
```

```
_Afurnace.Value = 1e-6;
```

```
_Aexitplane.Value = 1e-6;
```

```
_Afurnacewall.Value = 1e-6;
```

```
_Arefrac.Value = 1e-6;
```

```
_Hburner.Value = 1e-6;
```

```
_Hhoppermid.Value = 1e-6;
```

```
_dfurnace.Value = 1e-6;
```

```
_bfurnace.Value = 1e-6;
```

```
_Hfurnace.Value = 1e-6;
```

```
_Hhoppertop.Value = 1e-6;
```

```
_Ahopperupperhalf.Value = 1e-6;
```

```
//Burner height
```

```
//Height of each burner row
```

```
_Hrow1.Value = 1e-6;
```

```
_Hrow2.Value = 1e-6;
```

```
_Hrow3.Value = 1e-6;
```

```
_Hrow4.Value = 1e-6;
```

```
_Hrow5.Value = 1e-6;
```

```
_Hrow6.Value = 1e-6;
```

```
_Hrow7.Value = 1e-6;
```

```
_Hrow8.Value = 1e-6;
```

```
_Hrow9.Value = 1e-6;
```

```
_Hrow10.Value = 1e-6;
```

```
//Mass flow distribution per row
```

```
_mrow1.Value = 1e-6;
_mrow2.Value = 1e-6;
_mrow3.Value = 1e-6;
_mrow4.Value = 1e-6;
_mrow5.Value = 1e-6;
_mrow6.Value = 1e-6;
_mrow7.Value = 1e-6;
_mrow8.Value = 1e-6;
_mrow9.Value = 1e-6;
_mrow10.Value = 1e-6;

//Flyash ratio
_Yflyash.Value = 1e-6;

//Other parameters
_fradloss.Value = 1e-6;

//Burner height
_HBurnerEff.Value = 1e-6;

//Hopper Height Ratios
_HhopperHfurnace.Value = 1e-6;
_HhopperHfurnacemean.Value = 1e-6;

//Total mass flow going into the burners
_mttotal.Value = 1e-6;

//Burner height fraction terms
nume = 1e-6;
denome = 1e-6;

//Furnace Specs
_delx.Value = 0;
_Walltype.Value = 0;
_Fueltypeforflame.Value = 0;
_Fluetypeforashsize.Value = 0;
_Fueltypeforeta.Value = 0;
_Firingconfig.Value = 0;
_Comboforxi.Value = 0;

//High ash loading effect inputs and constants
_HighAshLoading.Value = 0;

//Ash flow rates
_mFA.Value = 1e-6;
```

```

_mBA.Value = 1e-6;

//Mass fraction of flue gas and mass of the product out
_mproductsout.Value = 1e-6;

//Outlet temperature and enthalpy
_Tfgin.Value = 300; //Set a random flame temperature to avoid NaN(in case
flame temperature data was not transfered)
_Tfgout.Value = 1e-6;
_hfgout.Value = 1000;

//Intermediate step coefficients
_kcoke.Value = 10.2;
_x1.Value = 1e-6;
_x2.Value = 1e-6;
_xi.Value = 1e-6;
_xirefrac.Value = 1e-6;
_A.Value = 1e-6;
_B.Value = 1e-6;
_dfa.Value = 1e-6;

//High ash loading volume flow rate
_rhofginfurnace.Value = 1e-6;
_mfg.Value = 1e-6;
_Vfg.Value = 1e-6;

//High ash loading effect coefficient check
_coeffA.Value = 0;
_Qfracabs.Value = 0;
_Qfracscatter.Value = 0;
_coeffgamma.Value = 0;
_coeffbeta.Value = 0;
_Lp.Value = 0;
_epiG.Value = 0;
_KeG.Value = 0;
_coeffKcoke.Value = 0;
_phiGandP.Value = 0;
_epiGandP.Value = 0;
_Tfgavg.Value = 1e-6;

//Constants
sigma = 5.67 * Math.Pow(10, -8);

//Non-uniformaty factor coefficients: Default value
c1 = 1.9745;

```



```
c2 = 1.8080;
lambda = 0.9821;
mu = 0.2275;
sigmahf = 0.2371;
inerfchopper = 1e-6;
erfchopper = 1e-6;

//Others
mproductsout = 1e-6;
MBA = 1e-6;
Mproductsout = 1e-6;
MMfg = 1e-6;
gammaH2O = 1e-6;
gamman = 1e-6;
ppH2O = 1e-6;
ppn = 1e-6;
psiavg = 1e-6;
h_flame = 1e-6;
h_product = 1e-6;
VC = 1e-6;
kg = 1e-6;
rho fg = 1e-6;
kfa = 1e-6;
kflame = 1e-6;
epiflame = 1e-6;
epifurnace = 1e-6;
HburnertoHfurnace = 1e-6;
_etaintegral.Value = 1e-6;
_etadiff.Value = 1e-6;
_etaadjintegral.Value = 1e-6;
_etaintegralnohopper.Value = 1e-6;
_etaadjintegralnohopper.Value = 1e-6;

//erfc function constants and variables
inerfcfurnace = 1e-6;
t = 1e-6;
erfcfurnace = 1e-6;
a1 = 0.254829592;
a2 = -0.284496736;
a3 = 1.421413741;
a4 = -1.453152027;
a5 = 1.061405429;
p = 0.3275911;
sign = 1;
```

```

//High Ash Loading constants
rhoash = 2200;

//Check enthalpies
_TBA.Value = 1073.15;
_hBA.Value = 1000;
_hflame.Value = 1000;
_hproductsin.Value = 1000;

//Check Areas
_Afurnacewallcheck.Value = 1e-6;
_Afurnacetotalcheck.Value = 1e-6;

}

//property declarations to make
//parameters visible to outside world
//Inputs

//Thermal fluid properite

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass flow rate")]
[GridOrder(11)]
public IPS.Properties.Double mfuel
{
    get
    {
        return _mfuel;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Product Inlet mass flow rate")]
[GridOrder(12)]
public IPS.Properties.Double mproducts
{
    get
    {
        return _mproducts;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]

```

```
[DisplayName("Adiabatic flame temperature")]
[GridOrder(13)]
public IPS.Properties.Double Tflame
{
    get
    {
        return _Tflame;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Platen super heater surface temerature")]
[GridOrder(14)]
public IPS.Properties.Double Tfollowsurface
{
    get
    {
        return _Tfollowsurface;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Ambient pressure")]
[GridOrder(15)]
public IPS.Properties.Double Pamb
{
    get
    {
        return _Pamb;
    }
}

//Geometrical parameters

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace volume")]
[GridOrder(16)]
public IPS.Properties.Double Vfurnace
{
    get
    {
        return _Vfurnace;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace total surface area (include exit plane)")]
[GridOrder(17)]
public IPS.Properties.Double Afurnace
{
    get
    {
        return _Afurnace;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace wall surface area")]
[GridOrder(18)]
public IPS.Properties.Double Afurnacewall
{
    get
    {
        return _Afurnacewall;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace exit plane surface area")]
[GridOrder(19)]
public IPS.Properties.Double Aexitplane
{
    get
    {
        return _Aexitplane;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Refractory surface area")]
[GridOrder(20)]
public IPS.Properties.Double Arefrac
{
    get
    {
        return _Arefrac;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
```

```

[DisplayName("Furnace height")]
[GridOrder(21)]
public IPS.Properties.Double Hfurnace
{
    get
    {
        return _Hfurnace;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Hopper mid-plane height")]
[GridOrder(22)]
public IPS.Properties.Double Hhoppermid
{
    get
    {
        return _Hhoppermid;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Hopper top height")]
[GridOrder(23)]
public IPS.Properties.Double Hhoppertop
{
    get
    {
        return _Hhoppertop;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Hopper Upper Half Area")]
[GridOrder(24)]
public IPS.Properties.Double Ahopperupperhalf
{
    get
    {
        return _Ahipperupperhalf;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace tube outer diameter")]

```

```

[GridOrder(25)]
public IPS.Properties.Double dfurnace
{
    get
    {
        return _dfurnace;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Furnace tube pitch")]
[GridOrder(26)]
public IPS.Properties.Double bfurnace
{
    get
    {
        return _bfurnace;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner height")]
[GridOrder(27)]
public IPS.Properties.Double Hburner
{
    get
    {
        return _Hburner;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall height")]
[GridOrder(28)]
public IPS.Properties.List<IPS.Properties.Double> WWheight
{
    get
    {
        return _WWheight ;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Water wall height to frunace height ratio")]
[GridOrder(29)]

```

```

public IPS.Properties.List<IPS.Properties.Double> HwwHfurnace
{
    get
    {
        return _HwwHfurnace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Water wall mean height to frunace height ratio")]
[GridOrder(30)]
public IPS.Properties.List<IPS.Properties.Double> HwwHfurnacemean
{
    get
    {
        return _HwwHfurnacemean;
    }
}

//Burner height

//Height of each burner row
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 1 height")]
[GridOrder(31)]
public IPS.Properties.Double Hrow1
{
    get
    {
        return _Hrow1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 2 height")]
[GridOrder(32)]
public IPS.Properties.Double Hrow2
{
    get
    {
        return _Hrow2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]

```

```
[DisplayName("Burner row 3 height")]
[GridOrder(33)]
public IPS.Properties.Double Hrow3
{
    get
    {
        return _Hrow3;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 4 height")]
[GridOrder(34)]
public IPS.Properties.Double Hrow4
{
    get
    {
        return _Hrow4;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 5 height")]
[GridOrder(35)]
public IPS.Properties.Double Hrow5
{
    get
    {
        return _Hrow5;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 6 height")]
[GridOrder(36)]
public IPS.Properties.Double Hrow6
{
    get
    {
        return _Hrow6;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 7 height")]
```



```

[GridOrder(37)]
public IPS.Properties.Double Hrow7
{
    get
    {
        return _Hrow7;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 8 height")]
[GridOrder(38)]
public IPS.Properties.Double Hrow8
{
    get
    {
        return _Hrow8;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 9 height")]
[GridOrder(39)]
public IPS.Properties.Double Hrow9
{
    get
    {
        return _Hrow9;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Burner row 10 height")]
[GridOrder(40)]
public IPS.Properties.Double Hrow10
{
    get
    {
        return _Hrow10;
    }
}

//Mass flow distribution per row
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 1")]

```

```
[GridOrder(41)]
public IPS.Properties.Double mrow1
{
    get
    {
        return _mrow1;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 2")]
[GridOrder(42)]
public IPS.Properties.Double mrow2
{
    get
    {
        return _mrow2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 3")]
[GridOrder(43)]
public IPS.Properties.Double mrow3
{
    get
    {
        return _mrow3;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 4")]
[GridOrder(44)]
public IPS.Properties.Double mrow4
{
    get
    {
        return _mrow4;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 5")]
[GridOrder(45)]
```

```
public IPS.Properties.Double mrow5
{
    get
    {
        return _mrow5;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 6")]
[GridOrder(46)]
public IPS.Properties.Double mrow6
{
    get
    {
        return _mrow6;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 7")]
[GridOrder(47)]
public IPS.Properties.Double mrow7
{
    get
    {
        return _mrow7;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 8")]
[GridOrder(48)]
public IPS.Properties.Double mrow8
{
    get
    {
        return _mrow8;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 9")]
[GridOrder(49)]
public IPS.Properties.Double mrow9
```

```

{
    get
    {
        return _mrow9;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel mass distribution at row 10")]
[GridOrder(50)]
public IPS.Properties.Double mrow10
{
    get
    {
        return _mrow10;
    }
}

//Flyash ratio

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flyash mass ratio")]
[GridOrder(51)]
public IPS.Properties.Double Yflyash
{
    get
    {
        return _Yflyash;
    }
}

//Other parameters

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Loss fraction of radiation")]
[GridOrder(52)]
public IPS.Properties.Double fradloss
{
    get
    {
        return _fradloss;
    }
}

//Mass fraction of Flue gas

```

```

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Fluid")]
[GridOrder(53)]
public IPS.Properties.DataReferenceProperty FluidDataReference
{
    get
    {
        return _FluidDataReference;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mass fractions")]
[GridOrder(54)]
public IPS.Properties.List<IPS.Properties.Double> MassFractions
{
    get
    {
        return _MassFractions;
    }
}

//Furnace Specs

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Delta X")]
[GridOrder(55)]
public IPS.Properties.Double delx
{
    get
    {
        return _delx;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Type of furnace wall (0: Panel wall/default; 1: Membrane wall)")]
[GridOrder(56)]
public IPS.Properties.Double Walltype
{
    get
    {
        return _Walltype;
    }
}

```

```

    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Fuel Type (0: High volatile or Reactive coal/ default; 1: Gas or Fuel oil; 2:
Low reactive fuel/ Anthracite/ High ash coal)")]
    [GridOrder(57)]
    public IPS.Properties.Double Fueltypeforflame
    {
        get
        {
            return _Fueltypeforflame;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Furnace combination to determine fly ash size (0: 16E-6; 1: 13E-6; 2:
20E-6)")]
    [GridOrder(58)]
    public IPS.Properties.Double Fluetypeforashsize
    {
        get
        {
            return _Fluetypeforashsize;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Fuel Type to determine Non-uniformity factor (0: Bituminous Coal; 1:
Lignite; 2: Oil/Gas)")]
    [GridOrder(59)]
    public IPS.Properties.Double Fueltypeforeta
    {
        get
        {
            return _Fueltypeforeta;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Firing Configuration (0: Bituminous Coal; 1: Lignite; 2: Oil/Gas)")]
    [GridOrder(60)]
    public IPS.Properties.Double Firingconfig
    {
        get
        {

```

```

        return _Firingconfig;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fouling factor indicator (0: WW + Coal/ default, 1: WW + Gas or Gas and
oil mix, 2: WW + Heavy Oil, 3: WW + All grate-fired fuel, 4: WW + other fuel, 5: WW with
refractory backing)")]
[GridOrder(61)]
public IPS.Properties.Double Comboforxi
{
    get
    {
        return _Comboforxi;
    }
}

//High ash loading effect inputs and constants

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("High Ash Loading Effect(0: Do Not Considers, 1: Considers)")]
[GridOrder(62)]
public IPS.Properties.Double HighAshLoading
{
    get
    {
        return _HighAshLoading;
    }
}

//Outputs

//Burner height

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective burner height")]
[GridOrder(63)]
public IPS.Properties.Double HBurnerEff
{
    get
    {
        return _HBurnerEff;
    }
}

```

```
//Hopper Height Ratio
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Hopper height to furnace height ratio")]
[GridOrder(64)]
public IPS.Properties.Double HhopperHfurnace
{
    get
    {
        return _HhopperHfurnace;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Hopper mean height to furnace height ratio")]
[GridOrder(65)]
public IPS.Properties.Double HhopperHfurnacemean
{
    get
    {
        return _HhopperHfurnacemean;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Mass fraction sum into the burners")]
[GridOrder(66)]
public IPS.Properties.Double mtotal
{
    get
    {
        return _mtotal;
    }
}
```

```
//Ash flow rates
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fly ash mass flow rate")]
[GridOrder(67)]
public IPS.Properties.Double mFA
{
    get
    {
```



```

        return _mFA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Bottom ash deposition rate")]
[GridOrder(68)]
public IPS.Properties.Double mBA
{
    get
    {
        return _mBA;
    }
}

//Mass fraction of flue gas

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Mass fractions of the product going out")]
[GridOrder(69)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsout
{
    get
    {
        return _MassFractionsout;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Product out mass flow rate")]
[GridOrder(70)]
public IPS.Properties.Double mproductsoutz
{
    get
    {
        return _mproductsout;
    }
}

//Outlet temperature and enthalpy

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Products inlet temperature")]
[GridOrder(71)]
public IPS.Properties.Double Tfgin

```

```

{
    get
    {
        return _Tfgin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas outlet temperature")]
[GridOrder(72)]
public IPS.Properties.Double Tfgout
{
    get
    {
        return _Tfgout;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas outlet enthalpy")]
[GridOrder(73)]
public IPS.Properties.Double hfgout
{
    get
    {
        return _hfgout;
    }
}

//Intermediate step coefficients

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Coke fractor")]
[GridOrder(74)]
public IPS.Properties.Double kcoke
{
    get
    {
        return _kcoke;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("x1")]

```

```
[GridOrder(75)]
public IPS.Properties.Double x1
{
    get
    {
        return _x1;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("x2")]
[GridOrder(76)]
public IPS.Properties.Double x2
{
    get
    {
        return _x2;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("xi")]
[GridOrder(77)]
public IPS.Properties.Double xi
{
    get
    {
        return _xi;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("xi_refrac")]
[GridOrder(78)]
public IPS.Properties.Double xirefrac
{
    get
    {
        return _xirefrac;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flame modification fractor coefficient A")]
[GridOrder(79)]
```

```

public IPS.Properties.Double A
{
    get
    {
        return _A;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flame modification fractor coefficient B")]
[GridOrder(80)]
public IPS.Properties.Double B
{
    get
    {
        return _B;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Mean ash diameter")]
[GridOrder(81)]
public IPS.Properties.Double Dfa
{
    get
    {
        return _dfa;
    }
}

//Intermediate step coefficients

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Furnace average beam length")]
[GridOrder(82)]
public IPS.Properties.Double sfurnace
{
    get
    {
        return _sfurnace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("xb")]

```

```

[GridOrder(83)]
public IPS.Properties.Double xb
{
    get
    {
        return _xb;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flame modification factor")]
[GridOrder(84)]
public IPS.Properties.Double Mflame
{
    get
    {
        return _Mflame;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Angular coefficient")]
[GridOrder(85)]
public IPS.Properties.Double xang
{
    get
    {
        return _xang;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Reradiation factor of platen superheater")]
[GridOrder(86)]
public IPS.Properties.Double beta
{
    get
    {
        return _beta;
    }
}

//Non-uniformaty factors

[PropertyUsage(UseProperty.RESULT)]

```

```

[DisplayName("Thermal load distribution coefficient")]
[GridOrder(87)]
public IPS.Properties.Double etafurnace
{
    get
    {
        return _etafurnace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Non-uniformaty factors of water wall")]
[GridOrder(88)]
public IPS.Properties.List<IPS.Properties.Double> etaww
{
    get
    {
        return _etaww;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Non-uniformaty factors of projected hopper outer surface")]
[GridOrder(89)]
public IPS.Properties.Double etahoppermid
{
    get
    {
        return _etahoppermid;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Non-uniformaty factor integral")]
[GridOrder(90)]
public IPS.Properties.Double etaintegral
{
    get
    {
        return _etaintegral;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Non-uniformaty factor integral without considering the hopper")]

```

```

[GridOrder(91)]
public IPS.Properties.Double etaintegralnohopper
{
    get
    {
        return _etaintegralnohopper;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Percentage difference of non-uniformaty factor integral")]
[GridOrder(92)]
public IPS.Properties.Double etadiff
{
    get
    {
        return _etadiff;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjust non-uniformaty factors of water wall")]
[GridOrder(93)]
public IPS.Properties.List<IPS.Properties.Double> etawwadj
{
    get
    {
        return _etawwadj;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted non-uniformaty factor integral")]
[GridOrder(94)]
public IPS.Properties.Double etaadjintegral
{
    get
    {
        return _etaadjintegral;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted non-uniformaty factor integral without considering hopper")]
[GridOrder(95)]

```

```

public IPS.Properties.Double etaadjintegralnohopper
{
    get
    {
        return _etaadjintegralnohopper;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted non-uniformity factor of projected hopper outer surface")]
[GridOrder(96)]
public IPS.Properties.Double etahoppermidadj
{
    get
    {
        return _etahoppermidadj;
    }
}

//Heat tranfer

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Total flue gas heat transfer")]
[GridOrder(97)]
public IPS.Properties.Double Qfg
{
    get
    {
        return _Qfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Total heat flux")]
[GridOrder(98)]
public IPS.Properties.Double qfg
{
    get
    {
        return _qfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Radiation bypassed")]

```



```
[GridOrder(99)]
public IPS.Properties.Double Qradout
{
    get
    {
        return _Qradout;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Radiation to all the effective projected wall surfaces")]
[GridOrder(100)]
public IPS.Properties.Double Qwalls
{
    get
    {
        return _Qwalls;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Radiation to the projected hopper outer surface")]
[GridOrder(101)]
public IPS.Properties.Double Qhopperupperhalf
{
    get
    {
        return _Qhopperupperhalf;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Remaining heat going into water walls")]
[GridOrder(102)]
public IPS.Properties.Double Qremain
{
    get
    {
        return _Qremain;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat flux to the furnace wall")]
```

```

[GridOrder(103)]
public IPS.Properties.Double qww
{
    get
    {
        return _qww;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat transfer at each node along water wall height")]
[GridOrder(104)]
public IPS.Properties.List<IPS.Properties.Double> Qww
{
    get
    {
        return _Qww;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat flux at each node along water wall height")]
[GridOrder(105)]
public IPS.Properties.List<IPS.Properties.Double> qwwnode
{
    get
    {
        return _qwwnode;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Net heat transfer to the water wall")]
[GridOrder(106)]
public IPS.Properties.Double Qwwsum
{
    get
    {
        return _Qwwsum;
    }
}

//Coefficient check

[PropertyUsage(UseProperty.RESULT)]

```

```
[DisplayName("VC")]
[GridOrder(107)]
public IPS.Properties.Double vc
{
    get
    {
        return _VC;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kg")]
[GridOrder(108)]
public IPS.Properties.Double KG
{
    get
    {
        return _kg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kfa")]
[GridOrder(109)]
public IPS.Properties.Double KFA
{
    get
    {
        return _kfa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kradflame")]
[GridOrder(110)]
public IPS.Properties.Double KRADFLAME
{
    get
    {
        return _kradflame;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("epiflame")]
```

```

[GridOrder(111)]
public IPS.Properties.Double EPIFLAME
{
    get
    {
        return _epiflame;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("epifurnace")]
[GridOrder(112)]
public IPS.Properties.Double EPIFURNACE
{
    get
    {
        return _epifurnace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("psiavg")]
[GridOrder(113)]
public IPS.Properties.Double PSIAVG
{
    get
    {
        return _psiavg;
    }
}

//High ash loading volume flow rate

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas Density in The Furnace")]
[GridOrder(114)]
public IPS.Properties.Double rhofginfurnace
{
    get
    {
        return _rhofginfurnace;
    }
}

[PropertyUsage(UseProperty.RESULT)]

```

```

[DisplayName("Flue Gas Mass Flow Rate")]
[GridOrder(115)]
public IPS.Properties.Double mfg
{
    get
    {
        return _mfg;
    }
}

[PropertyUsage(PropertyType.RESULT)]
[DisplayName("Flue Gas Volume Flow Rate")]
[GridOrder(116)]
public IPS.Properties.Double Vfg
{
    get
    {
        return _Vfg;
    }
}

//High ash loading effect coefficient check

[PropertyUsage(PropertyType.RESULT)]
[DisplayName("A in High Ash Loading Calculation")]
[GridOrder(117)]
public IPS.Properties.Double coeffA
{
    get
    {
        return _coeffA;
    }
}

[PropertyUsage(PropertyType.RESULT)]
[DisplayName("Q abs in High Ash Loading Calculation")]
[GridOrder(118)]
public IPS.Properties.Double Qfracabs
{
    get
    {
        return _Qfracabs;
    }
}

```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q scatter in High Ash Loading Calculation")]
[GridOrder(119)]
public IPS.Properties.Double Qfracscatter
{
    get
    {
        return _Qfracscatter;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Gamma in High Ash Loading Calculation")]
[GridOrder(120)]
public IPS.Properties.Double coeffgamma
{
    get
    {
        return _coeffgamma;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Beta in High Ash Loading Calculation")]
[GridOrder(121)]
public IPS.Properties.Double coeffbeta
{
    get
    {
        return _coeffbeta;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Lp in High Ash Loading Calculation")]
[GridOrder(122)]
public IPS.Properties.Double Lp
{
    get
    {
        return _Lp;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```

[DisplayName("Epsilon G in High Ash Loading Calculation")]
[GridOrder(123)]
public IPS.Properties.Double epiG
{
    get
    {
        return _epiG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("a in High Ash Loading Calculation")]
[GridOrder(124)]
public IPS.Properties.List<IPS.Properties.Double> coeffa
{
    get
    {
        return _coeffa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b0 in High Ash Loading Calculation")]
[GridOrder(125)]
public IPS.Properties.List<IPS.Properties.Double> coeffb0
{
    get
    {
        return _coeffb0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b1 in High Ash Loading Calculation")]
[GridOrder(126)]
public IPS.Properties.List<IPS.Properties.Double> coeffb1
{
    get
    {
        return _coeffb1;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KG in High Ash Loading Calculation")]

```

```
[GridOrder(127)]
public IPS.Properties.List<IPS.Properties.Double> coeffkG
{
    get
    {
        return _coeffkG;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KeG in High Ash Loading Calculation")]
[GridOrder(128)]
public IPS.Properties.Double KeG
{
    get
    {
        return _KeG;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Kcoke in High Ash Loading Calculation")]
[GridOrder(129)]
public IPS.Properties.Double coeffKcoke
{
    get
    {
        return _coeffKcoke;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi G+P in High Ash Loading Calculation")]
[GridOrder(130)]
public IPS.Properties.Double phiGandP
{
    get
    {
        return _phiGandP;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Epsilon G+P in High Ash Loading Calculation")]
[GridOrder(131)]
```



```

public IPS.Properties.Double epiGandP
{
    get
    {
        return _epiGandP;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Average Flue Gas Temperature")]
[GridOrder(132)]
public IPS.Properties.Double Tfgavg
{
    get
    {
        return _Tfgavg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("ppn")]
[GridOrder(133)]
public IPS.Properties.Double PPn
{
    get
    {
        return _ppn;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Bottom ash temperature")]
[GridOrder(134)]
public IPS.Properties.Double TBA
{
    get
    {
        return _TBA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Bottom ash enthalpy")]
[GridOrder(135)]
public IPS.Properties.Double hBA

```

```

{
    get
    {
        return _hBA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flame enthalpy")]
[GridOrder(136)]
public IPS.Properties.Double hflame
{
    get
    {
        return _hflame;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Product inlet enthalpy")]
[GridOrder(137)]
public IPS.Properties.Double hproductsin
{
    get
    {
        return _hproductsin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Projected Furnace Wall Area Sum Check")]
[GridOrder(138)]
public IPS.Properties.Double Afurnacewallcheck
{
    get
    {
        return _Afurnacewallcheck;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Total Projected Furnace Wall Area")]
[GridOrder(139)]
public IPS.Properties.Double Afurnacetotalcheck
{

```

```

        get
        {
            return _Afurnacetotalcheck;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Furnace wall segment surface area")]
    [GridOrder(140)]
    public IPS.Properties.List<IPS.Properties.Double> Afurnacewallsectionalcheck
    {
        get
        {
            return _Afurnacewallsectionalcheck;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Water wall height to frunace height ratio relative to hopper top")]
    [GridOrder(141)]
    public IPS.Properties.List<IPS.Properties.Double> HwwHhoppertop
    {
        get
        {
            return _HwwHhoppertop;
        }
    }
}

```

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll;
//css_ref Flownex.dll;

using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.Components;
using Flownex.DataReferences.Materials;

//Reference
//Heat Atlas VDI G7 [1]
//Heat Atlas VDI L1 [2]
//MEC4117Z Textbook [3]

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Geometrical

    //Pre-cavity
    IPS.Properties.Double _Lpc;

    //Flue gas Cavity
    IPS.Properties.Double _Hci;
    IPS.Properties.Double _wci;
    IPS.Properties.Double _Hce;
    IPS.Properties.Double _wce;

    //Heat exchanger tubes
    IPS.Properties.Double _Lpath;
    IPS.Properties.Double _Do;
    IPS.Properties.Double _twall;
    IPS.Properties.Double _tfouling;
    IPS.Properties.Double _Ntodd;
    IPS.Properties.Double _Nteven;
    IPS.Properties.Double _Nltotal;
    IPS.Properties.Double _st;

```

```
IPS.Properties.Double _sl;  
IPS.Properties.Double _Npp;  
IPS.Properties.Double _thetaAOA;  
IPS.Properties.Double _Nwall;  
  
//Slag Screen Geometry  
IPS.Properties.Double _Dslag;  
IPS.Properties.Double _sslslag;  
IPS.Properties.Double _nslagdeep;  
  
//Fins  
IPS.Properties.Double _Df;  
IPS.Properties.Double _Lf;  
IPS.Properties.Double _bf;  
IPS.Properties.Double _finthickness;  
IPS.Properties.Double _sfin;  
IPS.Properties.Double _lambdafin;  
  
//Others  
IPS.Properties.Double _beta;  
IPS.Properties.Double _tubebankarrangement;  
IPS.Properties.Double _slagscreenconsideration;  
IPS.Properties.Double _Fueltype;  
IPS.Properties.Double _HEtype;  
IPS.Properties.Double _Pass;  
IPS.Properties.Double _HighAshLoading;  
IPS.Properties.Double _HEeffectiveness;  
IPS.Properties.Double _InternalConvectionCorrelation;  
IPS.Properties.Double _ConvectionCorrelationtosurroundingwalls;  
IPS.Properties.Double _NuCorrelationTypesurroundingwalls;  
IPS.Properties.Double _Foulingoption;  
IPS.Properties.Double _Sootbloweroption;  
IPS.Properties.Double _Slagtype;  
IPS.Properties.Double _HXdiscretization;  
  
//Thermal or fluid properties  
  
//Flue gas side  
IPS.Properties.Double _Qraddirect;  
IPS.Properties.Double _Qbackrad;  
IPS.Properties.DataReferenceProperty _FluidDataReference;  
IPS.Properties.List<IPS.Properties.Double> _MassFractions;  
IPS.Properties.Double _Tfg;
```

```
IPS.Properties.Double _Tpc;  
IPS.Properties.Double _pamb;  
IPS.Properties.Double _ufg;  
IPS.Properties.Double _Twalloutertube;  
IPS.Properties.Double _Twallmean;  
IPS.Properties.Double _Tductwalloutertube;  
IPS.Properties.Double _Tslingtubewalloutertube;  
IPS.Properties.Double _dfa;  
IPS.Properties.Double _n;  
  
//High ash loading Inputs  
IPS.Properties.Double _mproducts;  
  
//Outputs  
  
//Thermal or fluid properties  
  
//Flue gas side  
IPS.Properties.Double _Refg;  
IPS.Properties.Double _Refgmax;  
IPS.Properties.Double _Reheat;  
IPS.Properties.Double _Rewall;  
IPS.Properties.Double _Prfg;  
IPS.Properties.Double _rho fg;  
IPS.Properties.Double _lambda fg;  
IPS.Properties.Double _mu;  
IPS.Properties.Double _muw;  
  
//Geometry(pre-simulation data process)  
  
//Flue Gas Cavity CV  
IPS.Properties.Double _Aci;  
IPS.Properties.Double _Ace;  
IPS.Properties.Double _Aopen;  
IPS.Properties.Double _Vc;  
IPS.Properties.Double _Lfg;  
IPS.Properties.Double _Pfg;  
IPS.Properties.Double _Lc;  
IPS.Properties.Double _LpctoLc;  
IPS.Properties.Double _Aductwall;  
IPS.Properties.Double _Aroof;  
IPS.Properties.Double _Afloor;  
  
//Heat exchanger tubes  
IPS.Properties.Double _Atubeouter;
```

```

IPS.Properties.Double _HTareacorrection;
IPS.Properties.Double _AHTouter;
IPS.Properties.Double _Atubeinner;
IPS.Properties.Double _Di;
IPS.Properties.Double _Dhwall;
IPS.Properties.Double _Nr;
IPS.Properties.Double _Nfwt;
IPS.Properties.Double _sd;
IPS.Properties.Double _Rt;
IPS.Properties.Double _Rl;
IPS.Properties.Double _Rd;
IPS.Properties.Double _ufraction;
IPS.Properties.List<IPS.Properties.Double> _Lwall;
IPS.Properties.List<IPS.Properties.Double> _Awall;
IPS.Properties.Double _psi;
IPS.Properties.Double _Nfinperpass;
IPS.Properties.Double _Nfin;
IPS.Properties.Double _Afin;
IPS.Properties.Double _etaf;
IPS.Properties.Double _etao;
IPS.Properties.Double _Xfin;
IPS.Properties.Double _phifin;
IPS.Properties.Double _phifinprime;
IPS.Properties.Double _sHT;
IPS.Properties.Double _L;
IPS.Properties.Double _episurface;

```

```
//Thermal and fluid properties
```

```
//Radiation Adjustment due to Slag Screen
```

```
IPS.Properties.Double _xangslag;
IPS.Properties.Double _xsslslag;
```

```
//Radiations
```

```
IPS.Properties.Double _Qradabs;
IPS.Properties.Double _Qradpass;
IPS.Properties.Double _Qradback;
IPS.Properties.Double _Qradbacknet;
```

```
//Tube bank pressure loss
```

```
IPS.Properties.Double _Deltap;
IPS.Properties.Double _zeta;
IPS.Properties.Double _zetal;
```

```

IPS.Properties.Double _fal;
IPS.Properties.Double _fat;
IPS.Properties.Double _checkpoint1;
IPS.Properties.Double _zetat;
IPS.Properties.Double _fznl;
IPS.Properties.Double _fzt;
IPS.Properties.Double _fnt;
IPS.Properties.Double _zetao;
IPS.Properties.Double _F;
IPS.Properties.Double _kforward;

//Heat transfer coefficients

//Gas Radiation
IPS.Properties.Double _hgasrad;
IPS.Properties.Double _A;
IPS.Properties.Double _hgasradadj;

//Tube bank extenal surface convective heat transfer unfing Gnlinskis method see [1]
IPS.Properties.Double _fa;
IPS.Properties.Double _Nu0lam;
IPS.Properties.Double _Nu0turb;
IPS.Properties.Double _Nu0row;
IPS.Properties.Double _Nu0;
IPS.Properties.Double _K;
IPS.Properties.Double _Nu;
IPS.Properties.Double _hconv;
IPS.Properties.Double _hcombine;

//Gas Radiation to the duct wall
IPS.Properties.Double _hgasradtowall;
IPS.Properties.Double _hgasradadjtowall;

//Convective heat transfer to the surrounding duct walls
IPS.Properties.Double _fwall;
IPS.Properties.Double _Nuwall;
IPS.Properties.Double _hconvwall;
IPS.Properties.Double _hcombinewall;

//Gas Radiation to the sling/hanger tubes
IPS.Properties.Double _hgasradslingtube;
IPS.Properties.Double _hgasradadjslingtube;

```



```

//Others

//Effectiveness
IPS.Properties.Double _HEeffectivenessext;
IPS.Properties.Double _HEeffectivenessint;

//Constants
double sigmao;

//Intermediate steps

//Tri-atomic gases
IPS.Properties.Double _kg;
IPS.Properties.Double _kfa;
IPS.Properties.Double _krad;
IPS.Properties.Double _epsilonfg;
IPS.Properties.Double _gammaH2O;
IPS.Properties.Double _gamman;
IPS.Properties.Double _ppH2O;
IPS.Properties.Double _ppn;
double MMfg;
double TA_rhoHfg;
double H_diff;
double L_floor;
double thetafloor;
IPS.Properties.Double _cp;

//High Ash Loading constants
double rhoash;

//Platen superheater direct radiation adjustment
IPS.Properties.Double _phiradloss;

//High Ash Loading Outputs
IPS.Properties.Double _rhoHfginHT;
IPS.Properties.Double _mfg;
IPS.Properties.Double _Vfg;

//High Ash Loading Coefficient Check
IPS.Properties.Double _coeffA;
IPS.Properties.Double _Qfracabs;
IPS.Properties.Double _Qfracscatter;
IPS.Properties.Double _coeffgamma;
IPS.Properties.Double _coeffbeta;

```

```

IPS.Properties.Double _Lp;
IPS.Properties.Double _epiG;
IPS.Properties.List<IPS.Properties.Double> _coeffa;
IPS.Properties.List<IPS.Properties.Double> _coeffb0;
IPS.Properties.List<IPS.Properties.Double> _coeffb1;
IPS.Properties.List<IPS.Properties.Double> _coeffkG;
IPS.Properties.Double _KeG;
IPS.Properties.Double _phiGandP;

//Fouling layer thickness passed to the component
IPS.Properties.Double _tfoulingfinal;

//Additional parameters to check with C-schedule
IPS.Properties.Double _ufgmax;
IPS.Properties.Double _ufgheat;

double DuctWHratio;

//do pre simulation initialisation here
public override void Initialise()
{
    //Pre-simulation cleaning to prevent Nan and infinities from previous
simulation interfering current simulation

    //Outputs

    //Thermal and fluid properties

    //Radiations
    _Qradabs.Value = 1e-6;
    _Qradpass.Value = 1e-6;
    _Qradback.Value = 1e-6;
    _Qradbacknet.Value = 1e-6;

    //Tube bank pressure loss
    _Deltap.Value = 1e-6;
    _zeta.Value = 1e-6;
    _zetat.Value = 1e-6;
    _fal.Value = 1e-6;
    _fat.Value = 1e-6;
    _checkpoint1.Value = 1e-6;
    _zetat.Value = 1e-6;
    _fznl.Value = 1e-6;
    _fzt.Value = 1e-6;

```

```

_fnt.Value = 1e-6;
_zetao.Value = 1e-6;
_F.Value = 1e-6;
_kforward.Value = 1e-6;

```

```
//Heat transfer coefficients
```

```
//Gas Radiation
_hgasrad.Value = 1;
_A.Value = 1e-6;
_hgasradadj.Value = 1;

```

```
//Tube bank external surface convective heat transfer unfing Gnlnskis method
```

see [1]

```

_fa.Value = 1e-6;
_Nu0lam.Value = 1e-6;
_Nu0turb.Value = 1e-6;
_Nu0row.Value = 1e-6;
_Nu0.Value = 1e-6;
_K.Value = 1e-6;
_Nu.Value = 1e-6;
_hconv.Value = 1;
_hcombine.Value = 1;

```

```
//Gas Radiation to the duct wall
```

```

_hgasradtowall.Value = 1;
_hgasradadjtowall.Value = 1;

```

```
//Convective heat transfer to the surrounding duct walls
```

```

_fwall.Value = 1e-6;
_Nuwall.Value = 1;
_hconvwall.Value = 1;
_hcombinewall.Value = 1;

```

```
//Gas Radiation to the sling/hanger tubes
```

```

_hgasradslingtube.Value = 1;
_hgasradadjslingtube.Value = 1;

```

```
//Intemidient steps
```

```
//Tri-atomic gases
```

```

_kg.Value = 1e-6;
_kfa.Value = 1e-6;

```

```

_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e-6;
_ppn.Value = 1e-6;
_cp.Value = 1e-6;

//Platen superheater direct radiation adjustment
_phiradloss.Value = 1e-6;

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Coefficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

_mfg.Value = 0;

_epiG.Value = 0;

if(_Foulingoption.Value == 1) //User specified fouling resistance
{
    _tfoulingfinal.Value = _tfouling.Value;
}

else
{
    _tfoulingfinal.Value = _tfoulingfinal.Value;
}

//Surface emissivity always 0.8 in superheats

_episurface.Value = 0.8;

```

```

//Adjust the tube surface emissivity if there s a slag screen in front of the
radiant super heater

if(_slagscreenconsideration.Value == 1)
{
    _episurface.Value = 0.68;

    _xangslag.Value = 1.15484 * Math.Pow((_sslslag.Value / _Dslag.Value),
-0.850475);

    _xsslslag.Value = 1 - Math.Pow((1 - _xangslag.Value),
_nslagdeep.Value);
}

else
{
    _episurface.Value = 0.8;
}

//Identify the correct A

if(_Fueltype.Value == 0) //Default case: Bituminos or Anthracite Coal
{
    _A.Value = 0.4;
}

else if(_Fueltype.Value == 1) //Fuel oil and Gas
{
    _A.Value = 0.3;
}

else if(_Fueltype.Value == 2) //Lignite
{
    _A.Value = 0.5;
}

else
{
    _A.Value = 0.3; //Use default value
}

_HEeffectivenessext.Value = _HEeffectiveness.Value;

_HEeffectivenessint.Value = _HEeffectiveness.Value;

```

```

if(_InternalConvectionCorrelation.Value == 1) //Dittus-Boelter
{
    _HEeffectivenessint.Value = _HEeffectiveness.Value * 0.023;
}

else
{
    _HEeffectivenessint.Value = _HEeffectiveness.Value;
}

}

//do post simulation cleanup here
public override void Cleanup()
{
}

//script main execution function - called every cycle
public override void Execute(double Time)
{
    //Common Geometries

    H_diff = Math.Abs(_Hci.Value - _Hce.Value); //Calculate the difference in height
    and it is used for calculating the the floor tube area(assuming trapizium cross section)

    _Aci.Value = _wci.Value * _Hci.Value; //Heat exchanger cavity inlet open duct
    area

    _Ace.Value = _wce.Value * _Hce.Value; //Heat exchanger cavity outlet open
    duct area

    _Aopen.Value = (_Aci.Value + _Ace.Value) / 2; //Take average area (assuming
    trapizium cross section)

    _Dhwall.Value = (4 * _Aopen.Value) / (_wci.Value + _Hci.Value + _wce.Value +
    _Hce.Value); //Calculate the effective hydraulic diameter of the duct wall

    if(_HXdiscretization.Value == 0) //Bulk / not discretized
    {
        _Lc.Value = (_Nltotal.Value + 1) * _sl.Value; //Heat exchanger cavity
        length (only the region that is covered by the flue gas CV which is just the flue gas surrounding the
        steam tubes)
    }
}

```

```

else if(_HXdiscretization.Value == 1) //Discretized pass by pass
{
    _Lc.Value = (_Npp.Value + 1) * _sl.Value; //Heat exchanger cavity
length (only the region that is covered by the flue gas CV which is just the flue gas surrounding the
steam tubes)
}

else //Assume not discretized
{
    _Lc.Value = (_Nltotal.Value + 1) * _sl.Value; //Heat exchanger cavity
length (only the region that is covered by the flue gas CV which is just the flue gas surrounding the
steam tubes)
}

_Lc.Value = (_Npp.Value + 1) * _sl.Value; //Heat exchanger cavity length (only
the region that is covered by the flue gas CV which is just the flue gas surrounding the steam
tubes)

_LpctoLc.Value = _Lpc.Value / _Lc.Value; //Ratio between flue gas pre-cavity
length and cavity length and it's used for gas radiation adjustment

L_floor = Math.Pow(Math.Pow(H_diff, 2) + Math.Pow(_Lc.Value, 2), 0.5);
//Floor tube length

_Aductwall.Value = _Lc.Value * ((_wci.Value + _wce.Value) + (_Hci.Value +
_Hce.Value)); //Caity side water wall area

if(_Pass.Value == 0) //Default case, no pass (tower type boiler) / back pass
{
    _Aroof.Value = 1e-6;

    _Afloor.Value = 1e-6;
}

else if(_Pass.Value == 1) //Convective pass
{
    _Aroof.Value = ((_wci.Value + _wce.Value) / 2) * _Lc.Value -
(_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4);

    if(_HEtype == 1)
    {
        _Aductwall.Value = _wci.Value * _Hci.Value + (_Hci.Value +
_Hce.Value) * (Math.Pow(Math.Pow((Math.Abs(_wce.Value - _wci.Value) / 2), 2) +

```

```

Math.Pow(_Lc.Value, 2) , 0.5));

        _Afloor.Value = 1e-6;

    }

    else
    {
        _Aductwall.Value = (_Hci.Value + _Hce.Value) *
(Math.Pow(Math.Pow((Math.Abs(_wce.Value - _wci.Value) / 2), 2) + Math.Pow(_Lc.Value, 2) , 0.5));

        _Afloor.Value = ((_wci.Value + _wce.Value) / 2) * L_floor;

    }

}

else //Use default case
{
    _Aroof.Value = 1e-6;

    _Afloor.Value = 1e-6;

}

_L.Value = (Math.PI / 2) * _Do.Value;

_Vc.Value = _Aopen.Value * _Lc.Value - (_Ntodd.Value * (_Npp.Value / 2) +
_Nteven.Value * (_Npp.Value / 2)) * Math.PI * (Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value;

_Lfg.Value = _Vc.Value / _Aopen.Value;

_Pfg.Value = (4 * _Aopen.Value) / _Do.Value; //Adjusting the characteristic
length of the flue gas CV so that the Re calculated in flue gas CV uses Do as the flow path length

_Atubeouter.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
(_Npp.Value / 2)) * _Lpath.Value * Math.PI * _Do.Value;

_Di.Value = _Do.Value - 2 * _twall.Value;

_Atubeinner.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
(_Npp.Value / 2)) * _Lpath.Value * Math.PI * _Di.Value;

_Nfwt.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
(_Npp.Value / 2));

if(_Npp.Value % 2 == 1) //For the case where there is odd number of parallel

```


path

```

    {
        _Vc.Value = _Aopen.Value * _Lc.Value - (_Ntodd.Value *
        ((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) * Math.PI * (Math.Pow(_Do.Value,
        2) / 4) * _Lpath.Value;

        _Lfg.Value = _Vc.Value / _Aopen.Value;

        _Pfg.Value = (4 * _Aopen.Value) / _Do.Value; //Adjusting the
characteristic length of the flue gas CV so that the Re calculated in flue gas CV uses Do as the flow
path length

        _Atubeouter.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
        _Nteven.Value * ((_Npp.Value - 1) / 2)) * _Lpath.Value * Math.PI * _Do.Value;

        _Di.Value = _Do.Value - 2 * _twall.Value;

        _Atubeinner.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
        _Nteven.Value * ((_Npp.Value - 1) / 2)) * _Lpath.Value * Math.PI * _Di.Value;

        _Nfwf.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
        _Nteven.Value * ((_Npp.Value - 1) / 2));
    }

    else
    {
        _Vc.Value = _Vc.Value;
    }

    _Rt.Value = _st.Value / _Do.Value;

    _Rl.Value = _sl.Value / _Do.Value;

    //Void fraction, see [1] Eq. (9) and (10)

    if(_Rl.Value >= 1)
    {
        _psi.Value = 1 - (Math.PI / (4 * _Rt.Value));
    }

    else
    {
        _psi.Value = 1 - (Math.PI / (4 * _Rt.Value * _Rl.Value));
    }

```

```
//Case Specific
```

```
//Default values
```

```
_ufraction.Value = _Rt.Value / (_Rt.Value - 1); //Use for adjusting flue gas  
velocity see [2] Fig. 2
```

```
_fal.Value = (280 * Math.PI * (Math.Pow(Math.Pow(_Rl.Value, 0.5) - 0.6, 2) +  
0.75)) / ((4 * _Rt.Value * _Rl.Value - Math.PI) * Math.Pow(_Rt.Value, 1.6)); //See [2] Eq. (4)
```

```
_zetao.Value = 1 / Math.Pow(_Rt.Value, 2); //See [2] Eq (19)
```

```
_Nr.Value = _Nltotal.Value; //See [2] Fig 2.
```

```
if(_tubebankarrangement.Value == 1) //Staggered tube bank  
{
```

```
    _sd.Value = Math.Pow(((0.5 * _st.Value) * (0.5 * _st.Value)) +  
(_sl.Value * _sl.Value), 0.5); //See [1] Fig. 2
```

```
    _Rd.Value = _sd.Value / _Do.Value;
```

```
    _checkpoint1.Value = 0.5 * Math.Pow(2 * _Rt.Value + 1, 0.5);
```

```
    _fat.Value = 2.5 + (1.2 / Math.Pow(_Rt.Value - 0.85, 1.08)) + 0.4 *  
Math.Pow((_Rl.Value / _Rt.Value) - 1, 3) - 0.01 * Math.Pow((_Rt.Value / _Rl.Value) - 1, 3); //See [2]  
Eq. (12)
```

```
    _fa.Value = 1 + (2 / (3 * _Rl.Value)); //See [1] Eq. (18)
```

```
//Mean Beam Length
```

```
    _sHT.Value = 3.6 * ((((_st.Value * _sl.Value) / 2) -  
((Math.Pow(_Do.Value, 2) * Math.PI) / 8)) * 1) / ((Math.PI * (_Do.Value / 2)) * 1));
```

```
    if(_Rl.Value < checkpoint1) // Gas flow in diagonal  
    {
```

```
        _zetao.Value = Math.Pow((2 * (_Rd.Value - 1)) / (_Rt.Value  
* (_Rt.Value - 1)), 2); //See [2] Eq. (20)
```

```
        _fal.Value = (280 * Math.PI *  
(Math.Pow(Math.Pow(_Rl.Value, 0.5) - 0.6, 2) + 0.75)) / ((4 * _Rt.Value * _Rl.Value - Math.PI) *  
Math.Pow(_Rd.Value, 1.6)); //See [2] Eq. (9)
```

```
        _ufraction.Value = _Rt.Value / (2 * (_Rd.Value - 1)); //See  
[2] Fig. 2
```

```

        _Nr.Value = _Nltotal.Value - 1; //See [2] Fig. 2
    }

    else
    {
        _zetao.Value = _zetao.Value;
    }

}

else if(_tubebankarrangement.Value == 0) //In-line tube bank
{
    _sd.Value = 0;

    _Rd.Value = 0;

    _fat.Value = (0.22 + 1.2 * (Math.Pow(1 - (0.94 / _RI.Value), 0.6) /
Math.Pow(_Rt.Value - 0.85, 1.3))) * Math.Pow(10, 0.47 * ((_RI.Value / _Rt.Value) - 1.5)) + (0.03 *
(_Rt.Value - 1) * (_RI.Value - 1)); //See [2] Eq.(6)

    _fa.Value = 1 + ((0.7 * ((_RI.Value / _Rt.Value) - 0.3)) /
(Math.Pow(_psi.Value, 1.5) * (((_RI.Value / _Rt.Value) + 0.7) * ((_RI.Value / _Rt.Value) + 0.7))));
//See [1] Eq. (17)

    //Mean Beam Length

    _sHT.Value = 3.6 * (((_st.Value * _sl.Value) - ((Math.Pow(_Do.Value,
2) * Math.PI) / 4)) * 1) / (Math.PI * _Do.Value * 1));
}

else
{
    _sd.Value = 999;
}

//fnt calc

if(_Nr.Value < 10)
{
    if(_Nr.Value >= 5)
    {
        _fnt.Value = _zetao.Value * ((1 / _Nr.Value) - 0.1); //See [2]
Eq. (17)
    }
}

```

```

        else
        {
            _fnt.Value = 0;
        }
    }

    else
    {
        _fnt.Value = 0; //See [2] Eq. (18)
    }

    //Calculate Direct radiation correction coefficient

    _phiradloss.Value = Math.Pow(Math.Pow(_Lc.Value / _st.Value, 2) + 1, 0.5) -
    (_Lc.Value / _st.Value); //See [3] Eq.(6-8)

    if(_HEtype.Value == 0) //Default case
    {
        _AHTouter.Value = _Atubeouter.Value;

        _HTareacorrection.Value = 1;

        if(_Foulingoption.Value == 1) //User specified fouling resistance
        {
            _tfoulingfinal.Value = _tfoulingfinal.Value;
        }

        else
        {
            if(_Fueltype.Value == 1) //Fuel oil and Gas
            {
                _tfoulingfinal.Value = 0.0026;
            }

            else
            {
                _tfoulingfinal.Value = 0.0043;
            }
        }
    }
}

```

```

else if(_HEtype.Value == 1) //Platen
{
    _AHTouter.Value = 2 * _Lpath.Value * _Lc.Value * _Ntodd.Value;
//See [3] Eq. (6-45)

    _HTareacorreption.Value = (Math.PI * _Do.Value) / (2 * _sl.Value);
//See [3] Eq. (6-46)

    if(_Foulingoption.Value == 1) //User specified fouling resistance
    {
        _tfoulingfinal.Value = _tfoulingfinal.Value;
    }

    else
    {
        if(_Slagtype.Value == 0) //Half slagging coal
        {
            if(_Sootbloweroption.Value == 0) //With soot
            blowing
            {
                _tfoulingfinal.Value = 5.27497e-6 *
                _Tfg.Value + 1.47684e-3;
            }

            else //Without Soot blowing
            {
                _tfoulingfinal.Value = 1.01224e-5 *
                _Tfg.Value - 8.99667e-4;
            }
        }

        else if(_Slagtype.Value == 1) //Non-slagging coal
        {
            _tfoulingfinal.Value = 3.85e-6 * _Tfg.Value +
            7.46885e-4;
        }

        else //Oil or Gas withsoot blowing
        {
            _tfoulingfinal.Value = 2.77299e-5 * _Tfg.Value -
            1.01451e-2;
        }
    }
}

```

```

else //Use Default Value
{
    _AHTouter.Value = _Atubeouter.Value;

    _HTareacorreption.Value = 1;

    if(_Foulingoption.Value == 1) //User specified fouling resistance
    {
        _tfoulingfinal.Value = _tfoulingfinal.Value;
    }

    else
    {
        if(_Fueltype.Value == 1) //Fuel oil and Gas
        {
            _tfoulingfinal.Value = 0.0026;
        }

        else
        {
            _tfoulingfinal.Value = 0.0043;
        }
    }
}

//Fin calculation

//Circular fin case

if(_Df.Value > _Do.Value)
{
    _Nfinperpass.Value = Math.Round(_Lpath.Value / _sfin.Value) + 1;

    _Nfin.Value = _Nfinperpass.Value * _Nfwf.Value;

    _Afin.Value = _Nfin.Value * ((Math.PI * _Df.Value *
_finthickness.Value) + 2 * ((Math.PI * (Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4));

    _Atubeouter.Value = (_Atubeouter.Value - (Math.PI * _Do.Value *
_finthickness * _Nfin)) + _Afin.Value;
}

```

```

_AHTouter.Value = _Atubeouter.Value;

_Vc.Value = _Aopen.Value * _Lc.Value - (((Math.PI *
(Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4) * _finthickness.Value * _Nfin.Value +
(_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the flue gas CV

_Lfg.Value = _Vc.Value / _Aopen.Value; //Update flue gas CV length
with the changed flue gas CV volume

if(_Npp.Value % 2 == 1) //For the case where there is odd number of
parallel path
{
_Vc.Value = _Aopen.Value * _Lc.Value - (((Math.PI *
(Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4) * _finthickness.Value * _Nfin.Value +
(_Ntodd.Value * ((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the flue gas CV

_Lfg.Value = _Vc.Value / _Aopen.Value; //Update flue gas
CV length with the changed flue gas CV volume
}

else
{
_Vc.Value = _Vc.Value;
}

_phifin.Value = ((_Df.Value / _Do.Value) - 1) * (1 + 0.35 *
Math.Log(_Df.Value / _Do.Value)); //See [3] Eq. (7-13)
}

else
{
_Afin.Value = 0;
}

if(_Lf.Value > _Do.Value)
{
if(_bf.Value < _Do.Value)
{
_Afin.Value = 0;
}

else
{

```

```

        _Nfinperpass.Value = Math.Round(_Lpath.Value /
_sfin.Value) + 1;

        _Nfin.Value = _Nfinperpass.Value * _Nfwt.Value;

        _Afin.Value = _Nfin.Value * ( (2 * (_Lf.Value + _bf.Value) *
_finthickness.Value) + (2 * (_Lf.Value * _bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4)))));

        _Atubeouter.Value = (_Atubeouter.Value - (Math.PI *
_Do.Value * _finthickness * _Nfin)) + _Afin.Value;

        _AHTouter.Value = _Atubeouter.Value;

        _Vc.Value = _Aopen.Value * _Lc.Value - ((2 * (_Lf.Value *
_bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4))) * _finthickness.Value * _Nfin.Value +
(_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the flue gas CV

        _Lfg.Value = _Vc.Value / _Aopen.Value; //Update flue gas
CV length with the changed flue gas CV volume

        if(_Npp.Value % 2 == 1) //For the case where there is odd
number of parallel path
        {
            _Vc.Value = _Aopen.Value * _Lc.Value - ((2 *
(_Lf.Value * _bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4))) * _finthickness.Value *
_Nfin.Value + (_Ntodd.Value * ((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) *
Math.PI * (Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the flue gas
CV

            _Lfg.Value = _Vc.Value / _Aopen.Value; //Update
flue gas CV length with the changed flue gas CV volume
        }

        else
        {
            _Vc.Value = _Vc.Value;
        }

        _phifinprime.Value = 1.28 * (_bf.Value / _Do.Value) *
Math.Pow((_Lf.Value / _bf.Value) - 0.2, 0.5); //See [3] Eq. (7-14)

        _phifin.Value = (_phifinprime.Value - 1) * (1 + 0.35 *
Math.Log(_phifinprime.Value)); //See [3] Eq. (7-14)
    }

```



```

    }

    else
    {
        _Afin.Value = 0;
    }

    //Calculate Wall heat transfer geometries

    for(int i = 0; i < _Nwall.Value ; i++)
    {
        _Lwall[i].Value = _twall.Value / _Nwall.Value;

        _Awall[i].Value = Math.PI * (_Ntodd.Value * (_Npp.Value / 2) +
        _Nteven.Value * (_Npp.Value / 2)) * _Lpath.Value * (_Do.Value - 2 * (i + 1) * _Lwall[i].Value);
    }

    //Check for High Ash loading case

    if(_HighAshLoading == 1)
    {
        //Initialise constants

        //b0 Coefficients
        _coeffb0[0].Value = 0.130;
        _coeffb0[1].Value = 0.595;
        _coeffb0[2].Value = 0.275;

        //b1 Coefficients
        _coeffb1[0].Value = 0.000265;
        _coeffb1[1].Value = -0.000150;
        _coeffb1[2].Value = -0.000115;

        //KG Coefficients
        _coeffkG[0].Value = 0.0;
        _coeffkG[1].Value = 0.824;
        _coeffkG[2].Value = 25.907;

        //Calculate Coefficient A
        _coeffA.Value = 3 / (2 * rhoash * _dfa.Value);

        //Calculate Qabs using emperical relation
        _Qfracabs.Value = 0.275 * Math.Pow((_dfa.Value / 1e-6), 0.298) -
0.305;

```

```

        //Calculate Qscattering using emperical realtion
        _Qfracscatter.Value = (6.2188e-3 - 1.0492e-2 * (_dfa.Value / 1e-6) +
7.287e-3 * Math.Pow((_dfa.Value / 1e-6), 2) - 2.1925e-5 * Math.Pow((_dfa.Value / 1e-6), 3)) /
(1.851e-1 - 2.0405e-3 * Math.Pow((_dfa.Value / 1e-6), 2) + 6.254e-4 * Math.Pow((_dfa.Value /
1e-6), 3));

        //Calculate Coefficient gamma
        _coeffgamma.Value = Math.Pow( 1 + ((2 * _Qfracscatter.Value) /
_Qfracabs.Value), 0.5);

        //Calculate Coefficient Beta
        _coeffbeta.Value = (_coeffgamma.Value -1) / (_coeffgamma.Value +
1);

    }

    else
    {
        _mfg.Value = _mfg.Value;
    }

    //Process Flue Gas data

    List<double> fractions = new List<double>();

    foreach(IPS.Properties.Double val in _MassFractions)
    {
        fractions.Add(val.Value);
    }
    double[] MassFractions = fractions.ToArray();

    MixedFluids fluid = (MixedFluids)_FluidDataReference.Value;

    //Create an array for flue gas mass fraction

    double[] Mfg = new double[fractions.Count];

    double[] fgMolarMass = new double[fractions.Count - 2];

    double[] Pureconstituent = new double[fractions.Count];

    //Pre-loop clean

    _mfg.Value = 0;

```

```

for(int i = 1; i < (fractions.Count - 1); i++)
{
    Mfg[i] = MassFractions[i] / (1 - (MassFractions[0] +
MassFractions[6]));

    _mfg.Value += MassFractions[i] * _mproducts.Value;

    Pureconstituent[i] = 1;

    fgMolarMass[i - 1] = fluid.GetMolarMass(Pureconstituent,
Parent.Identifier.Value);

    Pureconstituent[i] = 0;
}

//Calculate tri-atomic gases volume and partial fraction

//Initialise

double[] gammafg = new double[fractions.Count - 2];

double[] ppfg = new double[fractions.Count - 2];

MMfg = fluid.GetMolarMass(Mfg, Parent.Identifier.Value);

for(int i = 0; i < (fractions.Count - 2); i++)
{
    gammafg[i] = (Mfg[i + 1] * MMfg) / fgMolarMass[i];

    ppfg[i] = _pamb.Value * gammafg[i];
}

_gammaH2O.Value = gammafg[1];

_gammaman.Value = gammafg[0] + gammafg[1] + gammafg[4];

_ppH2O.Value = ppfg[1];

_ppn.Value = ppfg[0] + ppfg[1] + ppfg[4];

if(HighAshLoading == 1)
{

```

```

//Get Flue gas density
_rhofginHT.Value = fluid.GetDensity(_Tfg.Value, _pamb.Value, Mfg,
Parent.Identifier.Value);

//Calculate Flue gas volume flow rate
_Vfg.Value = _mfg.Value / _rhofginHT.Value;

//Calculate Lp
_Lp.Value = (_mproducts.Value * _MassFractions[0].Value) /
_Vfg.Value;

//Calcaulate Coefficient ai and epsilon G

//Pre-loop clean up
_epiG.Value = 0;

for(int i = 0; i < _coeffa.Count; i++)
{
    _coeffa[i].Value = _coeffb0[i].Value + _coeffb1[i].Value *
_Tfg.Value;

    _epiG.Value += _coeffa[i].Value * (1 - Math.Exp(-1 *
_coeffkG[i].Value * _ppn.Value * 1e-5 * _sHT.Value));
}

//Calculate K emisivity of Gas
_KeG.Value = -1 * (Math.Log(1 - _epiG.Value) / _sHT.Value);

//Calculate the optical thickness of gas particulate dispersion
_phiGandP.Value = (_Qfracabs.Value * _coeffA.Value * _Lp.Value +
_KeG.Value) * _sHT.Value * _coeffgamma.Value;

//Calculate effective gas-particulate emissivity
_epsilonfg.Value = (1 - _coeffbeta.Value) * ((1 - Math.Exp(-1 *
_phiGandP.Value)) / (1 + _coeffbeta.Value * Math.Exp(-1 * _phiGandP.Value)));
}

else //From Gurwich method
{
    _kg.Value = (((7.8 + 16 * _gammaH2O.Value) / (3.16 *

```

```

Math.Pow((( _ppn.Value / Math.Pow(10, 6)) * _sHT.Value), 0.5))) - 1) * (1 - 0.37 * (_Tfg.Value /
1000));

TA_rho fg = fluid.GetDensity(273.15, 101325, Mfg,
Parent.Identifier.Value);

_kfa.Value = (48350 * TA_rho fg) / Math.Pow((Math.Pow(_Tfg.Value,
2) * Math.Pow((_dfa.Value / Math.Pow(10, -6)), 2)), 0.333333333333333333);

_krad.Value = _kg.Value * _gamman.Value + _kfa.Value *
MassFractions[0];

_epsilonfg.Value = 1 - Math.Exp((_krad.Value * -1) * (_pamb.Value /
Math.Pow(10, 6)) * _sHT.Value);

}

//Radiation bypass, Radiation absorded and Back Radiation

_Qradpass.Value = _Qraddirect.Value * (1 - _epsilonfg.Value) *
(_phiradloss.Value / _beta.Value);

_Qradabs.Value = _Qraddirect.Value - _Qradpass.Value;

_Qradback.Value = 0.5 * sigmao * _epsilonfg.Value * _Ace.Value *
Math.Pow(_Tfg.Value, 4);

_Qradbacknet.Value = _Qbackrad.Value - _Qradback.Value ;

//Radiation bypass, Radiation absorded and Back Radiation with Slag Screen

if(_slagscreenconsideration == 1)
{
_Qradabs.Value = _xsslag.Value * _Qraddirect.Value;

_Qradpass.Value = _Qraddirect.Value - _Qradabs.Value;

_Qradback.Value = 0.5 * sigmao * _epsilonfg.Value * _Ace.Value *
Math.Pow(_Tfg.Value, 4);

_Qradbacknet.Value = _Qbackrad.Value - _Qradback.Value ;

}

else

```

```
{
    _Tfg.Value = _Tfg.Value;
}
```

```
//Heat transfer coefficient due to gas radiation
```

```
_hgasrad.Value = sigmao * ((_episurface.Value + 1) / 2) * _epsilonfg.Value *
Math.Pow(_Tfg.Value, 3) * ((1 - Math.Pow(_Twalloutertube.Value / _Tfg.Value, 4)) / (1 -
_Twalloutertube.Value / _Tfg.Value)); //See [3] Eq. (6-31)
```

```
//Apply Pre-cavity correction to hgasrad
```

```
_hgasradadj.Value = _hgasrad.Value * (1 + _A.Value * Math.Pow(_Tpc.Value /
1000, 0.25) * Math.Pow(_LpctoLc.Value, 0.07)); //See [3] Eq.(6-33)
```

```
//Heat transfer coefficient due to gas radiation to the duct wall
```

```
_hgasradtowall.Value = sigmao * ((_episurface.Value + 1) / 2) *
_epsilonfg.Value * Math.Pow(_Tfg.Value, 3) * ((1 - Math.Pow(_Tductwalloutertube.Value /
_Tfg.Value, 4)) / (1 - (_Tductwalloutertube.Value / _Tfg.Value))); //See [3] Eq. (6-31)
```

```
//Apply Pre-cavity correction to hgasradtowall
```

```
_hgasradadjtowall.Value = _hgasradtowall.Value * (1 + _A.Value *
Math.Pow(_Tpc.Value / 1000, 0.25) * Math.Pow(_LpctoLc.Value, 0.07)); //See [3] Eq.(6-33)
```

```
//Heat transfer coefficient due to gas radiation
```

```
_hgasradslingtube.Value = sigmao * ((_episurface.Value + 1) / 2) *
_epsilonfg.Value * Math.Pow(_Tfg.Value, 3) * ((1 - Math.Pow(_Tslingtubewalloutertube.Value /
_Tfg.Value, 4)) / (1 - (_Tslingtubewalloutertube.Value / _Tfg.Value))); //See [3] Eq. (6-31)
```

```
//Apply Pre-cavity correction to hgasrad
```

```
_hgasradadjslingtube.Value = _hgasradslingtube.Value * (1 + _A.Value *
Math.Pow(_Tpc.Value / 1000, 0.25) * Math.Pow(_LpctoLc.Value, 0.07)); //See [3] Eq.(6-33)
```

```
//Get Fluid properties from the mean Flue Gas temperature and mean wall
temperature
```

```
_rho fg.Value = fluid.GetDensity(_Tfg.Value, _pamb.Value, MassFractions,
Parent.Identifier.Value);
```

```
_lambda fg.Value = fluid.GetConductivity(_Tfg.Value, _pamb.Value,
MassFractions, Parent.Identifier.Value);
```

```

_cp.Value = fluid.GetSpecificHeat(_Tfg.Value, _pamb.Value, MassFractions,
Parent.Identifier.Value);

_mu.Value = fluid.GetViscosity(_Tfg.Value, _pamb.Value, MassFractions,
Parent.Identifier.Value);

if(_Twallmean.Value <= 0)
{
    _Twallmean.Value = 300;
}

else
{
    _Twallmean.Value = _Twallmean.Value;
}

_muw.Value = fluid.GetViscosity(_Twallmean.Value, _pamb.Value,
MassFractions, Parent.Identifier.Value);

if ( _ufg.Value != 0.0 )
{
    _Refg.Value = (_rhofg.Value * _ufg.Value * _Do.Value *
Math.Sin(_thetaAOA.Value)) / _mu.Value;
    _Refgmax.Value = _Refg.Value * _ufraction.Value; //Used for
calculating pressure loss in tube bank using Gnielinski's method, see [2]
    _Reheat.Value = _Refg.Value * (Math.PI / 2) * (1 / _psi.Value);
//Used for calculating Nusselt number using Gnielinski's method, see [1]
    _ufgmax.Value = _ufg.Value * _ufraction.Value; //Used for
calculating pressure loss in tube bank using Gnielinski's method, see [2]
    _ufgheat.Value = _ufg.Value * (Math.PI / 2) * (1 / _psi.Value); //Used
for calculating Nusselt number using Gnielinski's method, see [2]
}
else
{
    _Refg.Value = _Refg.Value;
}

if ( _lambdafg.Value != 0.0 )
{
    _Prfg.Value = (_cp.Value * _mu.Value) / _lambdafg.Value;
}
else
{
    _Prfg.Value = _Prfg.Value;
}

```

```

    }

    //Friction coefficient calculation

    //Calculate zetal

    _zetal.Value = _fal.Value / _Refgmax.Value; //See [2] Eq. (3) and (8)

    //Calculate case specific coefficients

    if(_Ntodd.Value == _Nteven.Value) //In-line tube bank
    {
        _F.Value = 1 - Math.Exp((( _Refgmax.Value + 1000) * -1) / 2000);
//See Eq. [2] Eq. (2b)

        _zetat.Value = _fat.Value / Math.Pow(_Refgmax.Value, 0.1 *
(_Rl.Value / _Rt.Value)); //See [2] Eq. (5)

    }

    else if(_Ntodd.Value != _Nteven.Value) //Staggered tube bank
    {
        _F.Value = 1 - Math.Exp((( _Refgmax.Value + 200) * -1) / 1000); //See
Eq. [2] Eq. (7b)

        _zetat.Value = _fat.Value / Math.Pow(_Refgmax.Value, 0.25); //See
[2] Eq. (11)

    }

    else // Default to be In-line
    {
        _F.Value = 1 - Math.Exp((( _Refgmax.Value + 1000) * -1) / 2000);

        _zetat.Value = _fat.Value / Math.Pow(_Refgmax.Value, 0.1 *
(_Rl.Value / _Rt.Value));

    }

    //Calculate coefficients that's depending on viscosities

    _fzt.Value = Math.Pow(_muw.Value / _mu.Value, 0.14); // See [2] Eq. (14)

    if(_Nr.Value < 10)
    {

```



```

        _fznl.Value = Math.Pow(_muw.Value / _mu.Value, (0.57 *
Math.Pow(_Nr.Value / 10, 0.25)) / Math.Pow(((4 * _Rt.Value * _Rl.Value / Math.PI) - 1) *
_Refgmax.Value, 0.25)); //See [2] Eq. (15)
    }

    else
    {
        _fznl.Value = Math.Pow(_muw.Value / _mu.Value, 0.57 /
Math.Pow(((4 * _Rt.Value * _Rl.Value / Math.PI) - 1) * _Refgmax.Value, 0.25)); //See [2] Eq. (16)
    }

    //Calculate friction coefficient (drag coefficient)

    _zeta.Value = _zetat.Value * _fznl.Value + (_zetat.Value * _fzt.Value +
_fnt.Value) * _F.Value; //See [2] Eq. (21) and (22)

    //Calculate kforward and pressure loss in tube bank

    _kforward.Value = _zeta.Value * _Npp.Value * Math.Pow(_ufraction.Value, 2);
//See [2] Eq. (1)

    //Collect flue gas

    _Deltap.Value = _kforward.Value * ((_rho fg.Value * Math.Pow(_ufg.Value, 2)) /
2); //See [2] Eq. (1)

    //Calculate Nu0lam per row

    _Nu0lam.Value = 0.664 * Math.Pow(_Reheat.Value, 0.5) *
Math.Pow(_Prfg.Value, 0.3333333333); //See [1] Eq. (12)

    //Calculate Nu0turb per row

    _Nu0turb.Value = (0.037 * Math.Pow(_Reheat.Value, 0.8) * _Prfg.Value) / (1 +
2.443 * Math.Pow(_Reheat.Value, -0.1) * (Math.Pow(_Prfg.Value, 0.6666666666) - 1)); //See [1]
Eq. (13)

    //Calculate Nu per row

    _Nu0row.Value = 0.3 + Math.Pow((_Nu0lam.Value * _Nu0lam.Value) +
(_Nu0turb.Value * _Nu0turb.Value), 0.5); //See [1] Eq. (11)

    //Calculate Nu0 for the entire tube bundle

    if(_Nltotal.Value < 10)

```

```

{
    _Nu0.Value = ((1 + ((_Nltotal.Value - 1) * _fa.Value)) / _Nltotal.Value)
* _Nu0row.Value; //See [1] Eq.(21)
}

else
{
    _Nu0.Value = _fa.Value * _Nu0row.Value; //See [1] Eq. (8)
}

// Flue gas Temperature Gradient correction factor

_K.Value = Math.Pow(_Tfg.Value / _Twallmean.Value, _n.Value); //See [1] Eq.

//Calculate Nu with temperature gradient adjustments

_Nu.Value = _Nu0.Value * _K.Value; //See [1] Eq. 23

//Calculate heat transfer coefficient due to convection

_hconv.Value = (_Nu.Value * _lambdafg.Value) / _L.Value; //See [1] Eq. (14)

//Combine h

_hcombine.Value = _HEffectivenessext.Value * (_hgasradadj.Value +
_HTareacorrection.Value * _hconv.Value); //See [3] Eq. (6-43)

//Calculate h for the duct wall

if(_ConvectionCorrelationtosurroundingwalls == 1) //Flat Plates
{
    _Rewall.Value = (_rho fg.Value * _u fg.Value * _Lc.Value *
Math.Sin(_thetaAOA.Value)) / _mu.Value;

    if(_Rewall.Value < 5e5 ) //Assume fully developed Laminar Flow
    {
        _Nuwall.Value = 0.664 * Math.Pow(_Rewall.Value, 0.5) *
Math.Pow(_Prfg.Value, 0.33333333);

        _hconvwall.Value = (_Nuwall.Value * lambdafg.Value) /
_Lc.Value;

        _hcombinewall.Value = _hgasradadjtowall.Value +
_hconvwall.Value;
    }
}

```

```

    }

    else //Assume fully developed turbulent Flow
    {
        _Nuwall.Value = 0.037 *
Math.Pow(_Rewall.Value, 0.8) * Math.Pow(_Prfg.Value, 0.33333333);

        _hconvwall.Value = (_Nuwall.Value * lambdafg.Value) /
_Lc.Value;

        _hcombinewall.Value = _hgasradadjtowell.Value +
_hconvwall.Value;
    }
}

else //Square Duct Internal Force Convection
{
    _Rewall.Value = (_rho fg.Value * _ufg.Value * _Dhwall.Value *
Math.Sin(_thetaAOA.Value)) / _mu.Value;

    if(_Rewall.Value <= 2800) //Laminar Flow in a Square Duct
    {
        DuctWHratio = ((_Hci.Value + _Hce.Value) / 2) /
((_wci.Value + _wce.Value) / 2);

        //Linearly interpolate Nu

        if(DuctWHratio <= 2)
        {
            _Nuwall.Value = 2.98 + ((3.39 - 2.98) / (2 - 1)) *
(DuctWHratio - 1);

            _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

            _hcombinewall.Value = _hgasradadjtowell.Value
+ _hconvwall.Value;
        }

        else if(DuctWHratio > 2 && DuctWHratio <= 3)
        {
            _Nuwall.Value = 3.39 + ((3.96 - 3.39) / (3 - 2)) *
(DuctWHratio - 2);

```

```

_lambdafg.Value) / _Dhwall.Value;

+ _hconvwall.Value;

}

else if(DuctWHratio > 3 && DuctWHratio <= 4)
{
    _Nuwall.Value = 3.96 + ((4.44 - 3.96) / (4 - 3)) *

(DuctWHratio - 3);

_lambdafg.Value) / _Dhwall.Value;

+ _hconvwall.Value;

}

else if(DuctWHratio > 4 && DuctWHratio <= 6)
{
    _Nuwall.Value = 5.14 + ((5.14 - 4.44) / (6 - 4)) *

(DuctWHratio - 4);

_lambdafg.Value) / _Dhwall.Value;

+ _hconvwall.Value;

}

else if(DuctWHratio > 6 && DuctWHratio <= 8)
{
    _Nuwall.Value = 5.14 + ((5.6 - 5.14) / (8 - 6)) *

(DuctWHratio - 6);

_lambdafg.Value) / _Dhwall.Value;

+ _hconvwall.Value;

}

else
{

```

```

Heat exchanger script
_Nuwall.Value = 5.6 + ((7.54 - 5.6) / (999 - 8)) *
(DuctWHratio - 8);

_lambda_dfg.Value) / _Dhwall.Value;

_hconvwall.Value = (_Nuwall.Value *

_hcombinewall.Value = _hgasradadjtowall.Value
+ _hconvwall.Value;
    }
}

else
{
    if(_NuCorrelationTypesurroundingwalls == 1)
//Dittus-Boelter
    {
        _Nuwall.Value = 0.023 *
Math.Pow(_Rewall.Value, 0.8) * Math.Pow(_Prfg.Value, 0.3);

        _hconvwall.Value = (_Nuwall.Value *
_lambda_dfg.Value) / _Dhwall.Value;

        _hcombinewall.Value = _hgasradadjtowall.Value
+ _hconvwall.Value;
    }

    else //Use Gnielinski's Correlation
    {
        _fwall.Value = Math.Pow((0.79 *
Math.Log(_Rewall.Value) - 1.64), -2);

        _Nuwall.Value = ((_fwall.Value / 8) *
(_Rewall.Value - 1000) * _Prfg.Value) / (1 + 12.7 * Math.Pow((_fwall.Value / 8), 0.5) *
(Math.Pow(_Prfg.Value, 0.666666666666) - 1));

        _hconvwall.Value = (_Nuwall.Value *
_lambda_dfg.Value) / _Dhwall.Value;

        _hcombinewall.Value = _hgasradadjtowall.Value
+ _hconvwall.Value;
    }
}
}

```

```

if(_Afin.Value != 0)
{
    //Calculate X for fin

    _Xfin.Value = _phifin.Value * (_Do.Value / 2) * Math.Pow((2
* _hcombine.Value) / (_lambdafin.Value * _finthickness.Value), 0.5); //See [3] Eq. (7-12)

    //Calculate fin efficiency

    _etaf.Value = Math.Tanh(_Xfin.Value) / _Xfin.Value; //See [3] Eq.
(7-12)

    //Calculate overall fin efficiency

    _etao.Value = 1 + (1 - _etaf.Value) * (_Afin.Value /
_Atubeouter.Value); //See [3] Eq. (7-7)

    //Adjust equivalent fouling layer thickness

    _tfoulingfinal.Value = _tfouling.Value / _etao.Value; //See [3] Eq.
(7-8)
}

else
{
    //Calculate X for fin

    _Xfin.Value = 1e-6;

    //Calculate fin efficiency

    _etaf.Value = 1e-6;

    //Calculate overall fin efficiency

    _etao.Value = 1;

}

}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{

```

```

        Execute(0.0);
    }

    //any processing you want to do while solving steady state
    public override void ExecuteSteadyState()
    {
        Execute(0.0);
    }

    //any processing you want to do after steady state
    public override void ExecuteAfterSteadyState()
    {
        Execute(0.0);
    }

    //add code here that you want to execute
    //when an associated button is pushed
    public override void HandleObjectEvent(object sender)
    {
    }

    //constructor initialises parameters
    public Script()
    {
        //Inialialise
        //Inputs

        //Geometrical

        //Pre-cavity
        _Lpc = new IPS.Properties.Double();

        //Flue gas Cavity
        _Hci = new IPS.Properties.Double();
        _wci = new IPS.Properties.Double();
        _Hce = new IPS.Properties.Double();
        _wce = new IPS.Properties.Double();

        //Heat exchanger tubes
        _Lpath = new IPS.Properties.Double();
        _Do = new IPS.Properties.Double();
        _twall = new IPS.Properties.Double();
        _tfouling = new IPS.Properties.Double();
    }

```

```

_Ntodd = new IPS.Properties.Double();
_Nteven = new IPS.Properties.Double();
_Nltotal = new IPS.Properties.Double();
_st = new IPS.Properties.Double();
_sl = new IPS.Properties.Double();
_Npp = new IPS.Properties.Double();
_thetaAOA = new IPS.Properties.Double();
_Nwall = new IPS.Properties.Double();

//Slag Screen Geometry
_Dslag = new IPS.Properties.Double();
_sslag = new IPS.Properties.Double();
_nslagdeep = new IPS.Properties.Double();

//Fins
_Df = new IPS.Properties.Double();
_Lf = new IPS.Properties.Double();
_bf = new IPS.Properties.Double();
_etaf = new IPS.Properties.Double();
_finthickness = new IPS.Properties.Double();
_sfin = new IPS.Properties.Double();
_lambdafin = new IPS.Properties.Double();

//Others
_beta = new IPS.Properties.Double();
_tubebankarrangement = new IPS.Properties.Double();
_slagscreenconsideration = new IPS.Properties.Double();
_Fueltype = new IPS.Properties.Double();
_HEtype = new IPS.Properties.Double();
_Pass = new IPS.Properties.Double();
_HighAshLoading = new IPS.Properties.Double();
_HEeffectiveness = new IPS.Properties.Double();
_InternalConvectionCorrelation = new IPS.Properties.Double();
_ConvectionCorrelationtosurroundingwalls = new IPS.Properties.Double();
_NuCorrelationTypesurroundingwalls = new IPS.Properties.Double();
_Foulingoption = new IPS.Properties.Double();
_Sootbloweroption = new IPS.Properties.Double();
_Slagtype = new IPS.Properties.Double();
_HXdiscretization = new IPS.Properties.Double();

//Thermal or fluid properties

```



```
//Flue gas side
_Qraddirect = new IPS.Properties.Double();
_Qbackrad = new IPS.Properties.Double();
_FluidDataReference = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReference.Types = new Type[] {typeof(MixedFluids)};
_MassFractions = new IPS.Properties.List<IPS.Properties.Double>();
_Tfg = new IPS.Properties.Double();
_Tpc = new IPS.Properties.Double();
_pamb = new IPS.Properties.Double();
_ufg = new IPS.Properties.Double();
_Twalloutertube = new IPS.Properties.Double();
_Twallmean = new IPS.Properties.Double();
_Tductwalloutertube = new IPS.Properties.Double();
_Tslingtubewalloutertube = new IPS.Properties.Double();
_dfa = new IPS.Properties.Double();
_n = new IPS.Properties.Double();
```

```
//High ash loading Inputs
_mproducts = new IPS.Properties.Double();
```

```
//Outputs
```

```
//Thermal or fluid properties
```

```
//Flue gas side
_Refg = new IPS.Properties.Double();
_Refgmax = new IPS.Properties.Double();
_Reheat = new IPS.Properties.Double();
_Rewall = new IPS.Properties.Double();
_Prfg = new IPS.Properties.Double();
_rho fg = new IPS.Properties.Double();
_lambda fg = new IPS.Properties.Double();
_mu = new IPS.Properties.Double();
_muw = new IPS.Properties.Double();
```

```
//Geometry(pre-simulation data process)
```

```
//Flue Gas Cavity CV
_Aci = new IPS.Properties.Double();
_Ace = new IPS.Properties.Double();
_Aopen = new IPS.Properties.Double();
```

```

_Vc = new IPS.Properties.Double();
_Lfg = new IPS.Properties.Double();
_Pfg = new IPS.Properties.Double();
_Lc = new IPS.Properties.Double();
_LpctoLc = new IPS.Properties.Double();
_Aductwall = new IPS.Properties.Double();
_Aroof = new IPS.Properties.Double();
_Afloor = new IPS.Properties.Double();

//Heat exchanger tubes
_Atubeouter = new IPS.Properties.Double();
_HTareacorrection = new IPS.Properties.Double();
_AHTouter = new IPS.Properties.Double();
_Atubeinner = new IPS.Properties.Double();
_Di = new IPS.Properties.Double();
_Dhwall = new IPS.Properties.Double();
_Nr = new IPS.Properties.Double();
_Nfwt = new IPS.Properties.Double();
_sd = new IPS.Properties.Double();
_Rt = new IPS.Properties.Double();
_Rl = new IPS.Properties.Double();
_Rd = new IPS.Properties.Double();
_ufraction = new IPS.Properties.Double();
_Lwall = new IPS.Properties.List<IPS.Properties.Double>();
_Awall = new IPS.Properties.List<IPS.Properties.Double>();
_psi = new IPS.Properties.Double();
_Nfinperpass = new IPS.Properties.Double();
_Nfin = new IPS.Properties.Double();
_Afin = new IPS.Properties.Double();
_etaf = new IPS.Properties.Double();
_etao = new IPS.Properties.Double();
_Xfin = new IPS.Properties.Double();
_phifin = new IPS.Properties.Double();
_phifinprime = new IPS.Properties.Double();
_sHT = new IPS.Properties.Double();
_L = new IPS.Properties.Double();
_episurface = new IPS.Properties.Double();

//Thermal and fluid properties

//Radiation Adjustment due to Slag Screen
_xangslag = new IPS.Properties.Double();
_xsslag = new IPS.Properties.Double();

```

```
//Radiations
_Qradabs = new IPS.Properties.Double();
_Qradpass = new IPS.Properties.Double();
_Qradback = new IPS.Properties.Double();
_Qradbacknet = new IPS.Properties.Double();
```

```
//Tube bank pressure loss
_Deltap = new IPS.Properties.Double();
_zeta = new IPS.Properties.Double();
_zetal = new IPS.Properties.Double();
_fal = new IPS.Properties.Double();
_fat = new IPS.Properties.Double();
_checkpoint1 = new IPS.Properties.Double();
_zetat = new IPS.Properties.Double();
_fznl = new IPS.Properties.Double();
_fzt = new IPS.Properties.Double();
_fnt = new IPS.Properties.Double();
_zetao = new IPS.Properties.Double();
_F = new IPS.Properties.Double();
_kforward = new IPS.Properties.Double();
```

```
//Heat transfer coefficient
```

```
//Gas Radiation
_hgasrad = new IPS.Properties.Double();
_A = new IPS.Properties.Double();
_hgasradadj = new IPS.Properties.Double();
```

```
//Tube bank extenal surface convective heat transfer unfing Gnlniskis method
```

```
_fa = new IPS.Properties.Double();
_Nu0lam = new IPS.Properties.Double();
_Nu0turb = new IPS.Properties.Double();
_Nu0row = new IPS.Properties.Double();
_Nu0 = new IPS.Properties.Double();
_K = new IPS.Properties.Double();
_Nu = new IPS.Properties.Double();
_hconv = new IPS.Properties.Double();
_hcombine = new IPS.Properties.Double();
```

```
//Gas Radiation to the duct wall
_hgasradtowall = new IPS.Properties.Double();
_hgasradadjtowall = new IPS.Properties.Double();
```

see [1]

```

//Convective heat transfer to the surrounding duct walls
_ffield = new IPS.Properties.Double();
_Nuwall = new IPS.Properties.Double();
_hconvwall = new IPS.Properties.Double();
_hcombinewall = new IPS.Properties.Double();

//Gas Radiation to the sling/hanger tubes
_hgasradslingtube = new IPS.Properties.Double();
_hgasradadslingtube = new IPS.Properties.Double();

//Others

//Effectiveness
_HEeffectivenessext = new IPS.Properties.Double();
_HEeffectivenessint = new IPS.Properties.Double();

//Intemidient steps

//Tri-atomic gases
_kg = new IPS.Properties.Double();
_kfa = new IPS.Properties.Double();
_krad = new IPS.Properties.Double();
_epsilonfg = new IPS.Properties.Double();
_gammaH2O = new IPS.Properties.Double();
_gamman = new IPS.Properties.Double();
_ppH2O = new IPS.Properties.Double();
_ppn = new IPS.Properties.Double();
_cp = new IPS.Properties.Double();

//Platen superheater direct radiation adjustment
_phiradloss = new IPS.Properties.Double();

//High Ash Loading Outputs
_rhofginHT = new IPS.Properties.Double();
_mfg = new IPS.Properties.Double();
_Vfg = new IPS.Properties.Double();

//High Ash Loading Ceofficient Check
_coeffa = new IPS.Properties.Double();
_Qfracabs = new IPS.Properties.Double();
_Qfracscatter = new IPS.Properties.Double();
_coeffgamma = new IPS.Properties.Double();

```

```

_coeffbeta = new IPS.Properties.Double();
_Lp = new IPS.Properties.Double();
_epiG = new IPS.Properties.Double();
_coeffa = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb0 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb1 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffkG = new IPS.Properties.List<IPS.Properties.Double>();
_KeG = new IPS.Properties.Double();
_phiGandP = new IPS.Properties.Double();

//Fouling layer thickness passed to the component
_tfoulingfinal = new IPS.Properties.Double();

//Additional parameters to check with C-schedule
_ufgmax = new IPS.Properties.Double();
_ufgheat = new IPS.Properties.Double();

//Unit
//Inputs

//Geometrical

//Pre-cavity
_Lpc.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Flue gas Cavity
_Hci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_wci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_wce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Heat exchanger tubes
_Lpath.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Do.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_twall.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_tfouling.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_st.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_sl.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_thetaAOA.DisplayUnitGroup = IPS.Units.UnitGroup.Angle;

//Slag Screen Geometry
_Dslag.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_sslag.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

```

```

//Fins
_Df.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Lf.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_bf.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_finthickness.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_sfin.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_lambdafin.DisplayUnitGroup = IPS.Units.UnitGroup.Thermal_Conductivity;

//Thermal or fluid properties

//Flue gas side
_Qraddirect.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qbackrad.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Tpc.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tfg.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_pamb.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
_ufg.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
_Twalloutertube.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twallmean.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tductwalloutertube.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tslingtubewalloutertube.DisplayUnitGroup =
IPS.Units.UnitGroup.Temperature;
_dfa.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//High ash loading Inputs
_mproducts.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Outputs

//Thermal or fluid properties

//Flue gas side
_rhofg.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_lambdafg.DisplayUnitGroup = IPS.Units.UnitGroup.Thermal_Conductivity;
_mu.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;
_muw.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;

//Geometry(pre-simulation data process)

//Flue Gas Cavity CV
_Aci.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

```

```

_Ace.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aopen.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Vc.DisplayUnitGroup = IPS.Units.UnitGroup.Volume;
_Lfg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Pfg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Lc.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Aductwall.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aroof.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Afloor.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

```

```
//Heat exchanger tubes
```

```

_Atubeouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_AHTouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Atubeinner.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Di.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Dhwall.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Afin.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_sHT.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_L.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

```

```
//Thermal and fluid properties
```

```
//Radiations
```

```

_Qradabs.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradpass.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradback.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradbacknet.DisplayUnitGroup = IPS.Units.UnitGroup.Power;

```

```
//Tube bank pressure loss
```

```
_Deltap.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
```

```
//Tube bank extenal surface convective heat transfer using Gnlinskis method
```

see [1]

```

_hgasrad.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hgasradadj.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hconv.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hcombine.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;

```

```
//Convective heat transfer to the surrounding duct walls
```

```
_hgasradtowall.DisplayUnitGroup =
```

```

IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
    _hgasradadjtowell.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
    _hconvwall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
    _hcombinewall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;

    //Gas Radiation to the sling/hanger tubes
    _hgasradslingtube.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
    _hgasradadjslingtube.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;


    //Others


    //Intemidient steps


    //Tri-atomic gases
    _ppH2O.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
    _ppn.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;


    //High Ash Loading Outputs
    _rhoH2O.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
    _mfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
    _Vfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Volume;


    //Fouling layer thickness passed to the component
    _tfoulingfinal.DisplayUnitGroup = IPS.Units.UnitGroup.Length;


    //Additional parameters to check with C-schedule
    _ufgmax.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
    _ufgheat.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;


    //Initial values
    //Inputs


    //Geometrical


    //Pre-cavity

```



```

_Lpc.Value = 1e-6;

//Flue gas Cavity
_Hci.Value = 1e-6;
_wci.Value = 1e-6;
_Hce.Value = 1e-6;
_wce.Value = 1e-6;

//Heat exchanger tubes
_Lpath.Value = 1e-6;
_Do.Value = 1e-6;
_twall.Value = 1e-6;
_tfouling.Value = 1e-6;
_Ntodd.Value = 1;
_Nteven.Value = 1;
_Nltotal.Value = 1;
_st.Value = 1e-6;
_sl.Value = 1e-6;
_Npp.Value = 1;
_thetaAOA.Value = Math.PI / 2; //Default value
_Nwall.Value = 1;

//Slag Screen Geometry
_Dslag.Value = 1e-6;
_sslag.Value = 1e-6;
_nslagdeep.Value = 1e-6;

//Fins
_Df.Value = 1e-6;
_Lf.Value = 1e-6;
_bf.Value = 1e-6;
_etaf.Value = 1;
_finthickness.Value = 1e-6;
_sfin.Value = 1e-6;
_lambdafin.Value = 1e-6;

//Others
_beta.Value = 1; //Default, no re-radiation, see [3]
_tubebankarrangement.Value = 0; //Default, inline
_slagscreenconsideration.Value = 0; //Default, no slag screen
_Fueltype.Value = 0; //Default Lignite
_HEtype.Value = 0; //Default, Convective
_Pass.Value = 0; //Default, back/no pass
_HighAshLoading.Value = 0; //Default, Do not consider
_HEeffectiveness.Value = 1; //Default, 100% heat exchanger effectiveness

```

```

_InternalConvectionCorrelation.Value = 0; //Default, Gnielinski
_ConvectionCorrelationtosurroundingwalls.Value = 0; //Default, Internal duct
flow with retangular cross-section

```

```

_NuCorrelationTypesurroundingwalls.Value = 0; //Default, Gnielinski
_Foulingoption.Value = 0; //Default, use the correlation in [3]
_Sootbloweroption.Value = 0; //Default, with soot blower, see [3]
_Slagtype.Value = 0; //Default, Half slagging coal, see [3]
_HXdiscretization.Value = 0;

```

```

//Thermal or fluid properties

```

```

//Flue gas side

```

```

_Qraddirect.Value = 1e-6;
_Qbackrad.Value = 1e-6;
_Tpc.Value = 300; // 26.85 degree by default
_Tfg.Value = 300; // 26.85 degree by default
_pamb.Value = 101325; // Atm pressure by default
_ufg.Value= 1e-6;
_Twalloutertube.Value = 1e-6;
_Twallmean.Value = 1e-6;
_Tductwalloutertube.Value = 1e-6;
_Tslingtubewalloutertube.Value = 1e-6;
_dfa.Value = 16e-6; // High volitle coal by default
_n.Value = 0;

```

```

//High ash loading Inputs

```

```

_mproducts.Value = 0;

```

```

//Outputs

```

```

//Thermal or fluid properties

```

```

///Flue gas side

```

```

_Refg.Value = 1e3;
_Refgmax.Value = 1e3;
_Reheat.Value = 1e3;
_Rewall.Value = 1e3;
_Prfg.Value = 1;
_rhofg.Value = 1e3;
_lambdafg.Value = 1e-6;
_mu.Value = 1e-6;
_muw.Value = 1e-6;

```

```

//Geometry(pre-simulation data process)

//Flue Gas Cavity CV
_Aci.Value = 1e-6;
_Ace.Value = 1e-6;
_Aopen.Value = 1e-6;
_Vc.Value = 1e-6;
_Lfg.Value = 1e-6;
_Pfg.Value = 1e-6;
_Lc.Value = 1e-6;
_LpctoLc.Value = 1e-6;
_Aductwall.Value = 1e-6;
_Aroof.Value = 1e-6;
_Afloor.Value = 1e-6;

//Heat exchanger tubes
_Atubeouter.Value = 1e-6;
_HTareacorrection.Value = 1; //Default value
_AHTouter.Value = 1e-6;
_Atubeinner.Value = 1e-6;
_Di.Value = 1e-6;
_Dhwall.Value = 1e-6;
_Nr.Value = 1;
_Nfwt.Value = 1;
_sd.Value = 1e-6;
_Rt.Value = 1e-6;
_Rl.Value = 1e-6;
_Rd.Value = 1e-6;
_ufraction.Value = 1e-6;
_psi.Value = 1e-6;
_Nfinperpass.Value = 1e-6;
_Nfin.Value = 1e-6;
_Afin.Value = 1e-6;
_etaf.Value = 1e-6;
_etao.Value = 1e-6;
_Xfin.Value = 1e-6;
_phifin.Value = 1e-6;
_phifinprime.Value = 1e-6;
_sHT.Value = 1e-6;
_L.Value = 1e-6;
_episurface.Value = 0.8; // Default value

//Thermal and fluid properties

```

```
//Radiation Adjustment due to Slag Screen
_xangslag.Value = 1e-6;
_xsslag.Value = 1e-6;
```

```
//Radiations
_Qradabs.Value = 1e-6;
_Qradpass.Value = 1e-6;
_Qradback.Value = 1e-6;
_Qradbacknet.Value = 1e-6;
```

```
//Tube bank pressure loss
_Deltap.Value = 1e-6;
_zeta.Value = 1e-6;
_zetal.Value = 1e-6;
_fal.Value = 1e-6;
_fat.Value = 1e-6;
_checkpoint1.Value = 1e-6;
_zetat.Value = 1e-6;
_fznl.Value = 1e-6;
_fzt.Value = 1e-6;
_fnt.Value = 1e-6;
_zetao.Value = 1e-6;
_F.Value = 1e-6;
_kforward.Value = 1e-6;
```

```
//Heat transfer coefficient
```

```
//Gas Radiation
_hgasrad.Value = 1;
_A.Value = 0.4; //Default Value
_hgasradadj.Value = 1;
```

```
//Tube bank extenal surface convective heat transfer unfing Gnlinskis method
```

```
_fa.Value = 1e-3;
_Nu0lam.Value = 1e-3;
_Nu0turb.Value = 1e-3;
_Nu0row.Value = 1e-3;
_Nu0.Value = 1e-3;
_K.Value = 1e-6;
_Nu.Value = 1e-6;
_hconv.Value = 1;
_hcombine.Value = 1;
```

see [1]

```
//Gas Radiation to the duct wall
_hgasradtowall.Value = 1;
_hgasradadjtowall.Value = 1;

//Convective heat transfer to the surrounding duct walls
_fwall.Value = 1e-6;
_Nuwall.Value = 1e-6;
_hconvwall.Value = 1;
_hcombinewall.Value = 1;

//Gas Radiation to the sling/hanger tubes
_hgasradslingtube.Value = 1;
_hgasradadjslingtube.Value = 1;

//Others

//Effectiveness
_HEeffectivenessext.Value = 1;
_HEeffectivenessint.Value = 1;

//Constants
sigmao = 5.67 * Math.Pow(10, -8);

//Intemidient steps

//Tri-atomic gases
_kg.Value = 1e-6;
_kfa.Value = 1e-6;
_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e3;
_ppn.Value = 1e3;
MMfg = 1e-6;
_cp.Value = 1e3;

//High Ash Loading constants
rhoash = 2200;

//Platen superheater direct radiation adjustment
_phiradloss.Value = 1e-6;
```

```

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_mfg.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Coefficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_epiG.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

//Fouling layer thickness passed to the component
_tfoulingfinal.Value = 1e-6;

//Additional parameters to check with C-schedule
_ufgmax.Value = 10;
_ufgheat.Value = 10;

DuctWHratio = 1;

}

//property declarations to make
//parameters visible to outside world
//Inputs

//Geometrical

//Pre-cavity
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Pre-cavity Length")]
[GridOrder(11)]
public IPS.Properties.Double Lpc
{
    get
    {
        return _Lpc;
    }
}

```

```

    }
}

//Flue gas Cavity

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Inlet Height")]
[GridOrder(12)]
public IPS.Properties.Double Hci
{
    get
    {
        return _Hci;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Inlet Width")]
[GridOrder(13)]
public IPS.Properties.Double wci
{
    get
    {
        return _wci;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Outlet Height")]
[GridOrder(14)]
public IPS.Properties.Double Hce
{
    get
    {
        return _Hce;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Outlet Width")]
[GridOrder(15)]
public IPS.Properties.Double wce
{
    get
    {

```

```

        return _wce;
    }
}

//Heat exchanger tubes

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Single Tube Path Length")]
[GridOrder(16)]
public IPS.Properties.Double Lpath
{
    get
    {
        return _Lpath;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Outer Diameter")]
[GridOrder(17)]
public IPS.Properties.Double Do
{
    get
    {
        return _Do;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Wall Thickness")]
[GridOrder(18)]
public IPS.Properties.Double twall
{
    get
    {
        return _twall;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Equivalent Thickness to Ash Deposition thermal Resistance on External
Heat Exchanger Surface")]
[GridOrder(19)]
public IPS.Properties.Double tfouling
{

```



```

        get
        {
            return _tfouling;
        }
    }

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Transverse Tubes in Odd Row")]
[GridOrder(20)]
public IPS.Properties.Double Ntodd
{
    get
    {
        return _Ntodd;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Transverse Tubes in Even Row")]
[GridOrder(21)]
public IPS.Properties.Double Nteven
{
    get
    {
        return _Nteven;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Total Number of Longitudinal Tubes in Heat Exchanger")]
[GridOrder(22)]
public IPS.Properties.Double Nltotal
{
    get
    {
        return _Nltotal;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Bank Transverse Pitch")]
[GridOrder(23)]
public IPS.Properties.Double st
{
    get

```

```

        {
            return _st;
        }
    }

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Bank Longitudinal Pitch")]
[GridOrder(24)]
public IPS.Properties.Double sl
{
    get
    {
        return _sl;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Tubes in Parallel Path")]
[GridOrder(25)]
public IPS.Properties.Double Npp
{
    get
    {
        return _Npp;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Gas Flow Oblique Angle Relative to Tube Bank")]
[GridOrder(26)]
public IPS.Properties.Double thetaAOA
{
    get
    {
        return _thetaAOA;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Nodes in Tube Wall")]
[GridOrder(27)]
public IPS.Properties.Double Nwall
{
    get
    {

```

```

        return _Nwall;
    }
}

//Slag Screen Geometry

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slag Screen Tube Outer Diameter")]
[GridOrder(28)]
public IPS.Properties.Double Dslag
{
    get
    {
        return _Dslag;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slag Screen Tube Transverse Pitch")]
[GridOrder(29)]
public IPS.Properties.Double sslag
{
    get
    {
        return _sslag;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Rows Deep of The Slag Screen")]
[GridOrder(30)]
public IPS.Properties.Double nslagdeep
{
    get
    {
        return _nslagdeep;
    }
}

//Fins

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Diameter(For Circular Fins)")]
[GridOrder(31)]
public IPS.Properties.Double Df

```

```

{
    get
    {
        return _Df;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Length(For Rectangular Fins)")]
[GridOrder(32)]
public IPS.Properties.Double Lf
{
    get
    {
        return _Lf;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Width(For Rectangular Fins)")]
[GridOrder(33)]
public IPS.Properties.Double bf
{
    get
    {
        return _bf;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Thickness")]
[GridOrder(34)]
public IPS.Properties.Double finthickness
{
    get
    {
        return _finthickness;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Pitch")]
[GridOrder(35)]
public IPS.Properties.Double sfin
{

```

```

        get
        {
            return _sfin;
        }
    }

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Thermal Conductivity")]
[GridOrder(36)]
public IPS.Properties.Double lambdafin
{
    get
    {
        return _lambdafin;
    }
}

```

```
//Others
```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Platen Superheater Reradiation Factor")]
[GridOrder(37)]
public IPS.Properties.Double beta
{
    get
    {
        return _beta;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Arrangement (0: Inline / Default case, 1: Staggered)")]
[GridOrder(38)]
public IPS.Properties.Double tubebankarrangement
{
    get
    {
        return _tubebankarrangement;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slag Screen Consideration (0: No Slag Screen / Default case, 1: With Slag
Screen)")]

```

```
[GridOrder(39)]
public IPS.Properties.Double slagscreenconsideration
{
    get
    {
        return _slagscreenconsideration;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel Type (0: Bituminous Coal, 1: Fuel oil and Gas, 2: Lignite)")]
[GridOrder(40)]
public IPS.Properties.Double Fueltype
{
    get
    {
        return _Fueltype;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Type (0: Convective, 1: Platen)")]
[GridOrder(41)]
public IPS.Properties.Double HETYPE
{
    get
    {
        return _HETYPE;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Location (0: Back Pass / No Pass, 1: Convective Pass)")]
[GridOrder(42)]
public IPS.Properties.Double Pass
{
    get
    {
        return _Pass;
    }
}
```

```
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("High Ash Loading Effect(0: Do Not Considers, 1: Considers)")]
[GridOrder(43)]
```

```

public IPS.Properties.Double HighAshLoading
{
    get
    {
        return _HighAshLoading;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Effectiveness")]
[GridOrder(44)]
public IPS.Properties.Double HEeffectiveness
{
    get
    {
        return _HEeffectiveness;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Internal Force Convection Correlation in The Steam Tubes(0: Gnielinski,
1: Dittus-Boelter)")]
[GridOrder(45)]
public IPS.Properties.Double InternalConvectionCorrelation
{
    get
    {
        return _InternalConvectionCorrelation;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas to Water Wall Convection Correlation Consideration(0: Internal
Duct Flow, 1: Flat Plate)")]
[GridOrder(46)]
public IPS.Properties.Double ConvectionCorrelationtosurroundingwalls
{
    get
    {
        return _ConvectionCorrelationtosurroundingwalls;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Internal Force Convection Correlation in The Duct(0: Gnielinski, 1:

```

```

Dittus-Boelter)"]
    [GridOrder(47)]
    public IPS.Properties.Double NuCorrelationTypesurroundingwalls
    {
        get
        {
            return _NuCorrelationTypesurroundingwalls;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Fouling Resistance Consideration (0: Use Default Correaltions, 1: User
Specified Fouling Resistance)")]
    [GridOrder(48)]
    public IPS.Properties.Double Foulingoption
    {
        get
        {
            return _Foulingoption;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Soot Blower Consideration(0: With Soot Blower, 1: Without Soot
Blower)")]
    [GridOrder(49)]
    public IPS.Properties.Double Sootbloweroption
    {
        get
        {
            return _Sootbloweroption;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Slagging Source Consideration(0: Half-slagging Coal, 1: Non-slagging
Coal, 2: Oil or Gas)")]
    [GridOrder(50)]
    public IPS.Properties.Double Slagtype
    {
        get
        {
            return _Slagtype;
        }
    }

```



```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Discretization(0: Not discretized/Bulk, 1: Discretized pass
by pass)")]
[GridOrder(51)]
public IPS.Properties.Double HXdiscretization
{
    get
    {
        return _HXdiscretization;
    }
}

//Thermal or fluid properties

//Flue gas side

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Direct Radiation")]
[GridOrder(52)]
public IPS.Properties.Double Qraddirect
{
    get
    {
        return _Qraddirect;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Back Radiation from Previous Heat Exchanger")]
[GridOrder(53)]
public IPS.Properties.Double Qbackrad
{
    get
    {
        return _Qbackrad;
    }
}

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Fluid")]
[GridOrder(54)]
public IPS.Properties.DataReferenceProperty FluidDataReference
{
    get

```

```

        {
            return _FluidDataReference;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Flue Gas Mass fractions")]
    [GridOrder(55)]
    public IPS.Properties.List<IPS.Properties.Double> MassFractions
    {
        get
        {
            return _MassFractions;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Mean Flue Gas Temperature")]
    [GridOrder(56)]
    public IPS.Properties.Double Tfg
    {
        get
        {
            return _Tfg;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Flue Gas Temperature in The Pre-Cavity")]
    [GridOrder(57)]
    public IPS.Properties.Double Tpc
    {
        get
        {
            return _Tpc;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Ambient Pressure in The Heat Exchanger")]
    [GridOrder(58)]
    public IPS.Properties.Double pamb
    {
        get
        {

```

```

        return _pamb;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Open-Duct Velocity")]
[GridOrder(59)]
public IPS.Properties.Double ufg
{
    get
    {
        return _ufg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Outer Tube Wall Temperature")]
[GridOrder(60)]
public IPS.Properties.Double Twalloutertube
{
    get
    {
        return _Twalloutertube;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Tube Wall Temperature")]
[GridOrder(61)]
public IPS.Properties.Double Twallmean
{
    get
    {
        return _Twallmean;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Duct Wall Temperature")]
[GridOrder(62)]
public IPS.Properties.Double Tductwalloutertube
{
    get
    {
        return _Tductwalloutertube;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Sling Tube Outer Surface Temperature")]
[GridOrder(63)]
public IPS.Properties.Double Tslingtubewalloutertube
{
    get
    {
        return _Tslingtubewalloutertube;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Ash Particle Size")]
[GridOrder(64)]
public IPS.Properties.Double dfa
{
    get
    {
        return _dfa;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Temperature Gradient adjustification factor")]
[GridOrder(65)]
public IPS.Properties.Double n
{
    get
    {
        return _n;
    }
}

//High ash loading Inputs

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Combustion Product Mass Flow Rate")]
[GridOrder(66)]
public IPS.Properties.Double mproducts
{
    get
    {

```

```

        return _mproducts;
    }
}

//Outputs

//Thermal or fluid properties

//Flue gas side

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Open Duct Reynold's Number at Mean Flue Gas Temperature")]
[GridOrder(67)]
public IPS.Properties.Double Refg
{
    get
    {
        return _Refg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Interstitial Reynold's Number at Mean Flue Gas Temperature")]
[GridOrder(68)]
public IPS.Properties.Double Refgmax
{
    get
    {
        return _Refgmax;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Reynold's Number Use for Heat Transfer Calculation at Mean
Flue Gas Temperature")]
[GridOrder(69)]
public IPS.Properties.Double Reheat
{
    get
    {
        return _Reheat;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Reynold's Number Use for Heat Transfer Calculation to The
Ducts")]

```

```

[GridOrder(70)]
public IPS.Properties.Double Rewall
{
    get
    {
        return _Rewall;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Prandtl Number at Mean Flue Gas Temperature")]
[GridOrder(71)]
public IPS.Properties.Double Prfg
{
    get
    {
        return _Prfg;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Density at Mean Flue Gas Temperature")]
[GridOrder(72)]
public IPS.Properties.Double rhofg
{
    get
    {
        return _rhofg;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Thermal Conductivity at Mean Flue Gas Temperature")]
[GridOrder(73)]
public IPS.Properties.Double lambdafg
{
    get
    {
        return _lambdafg;
    }
}

```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Flue Gas at Mean Flue Gas Temperature")]
[GridOrder(74)]
public IPS.Properties.Double mu
{
    get
    {
        return _mu;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Flue Gas at Mean Wall Temperature")]
[GridOrder(75)]
public IPS.Properties.Double muw
{
    get
    {
        return _muw;
    }
}
```

```
//Geometry(pre-simulation data process)
```

```
//Flue Gas Cavity CV
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Inlet Area")]
[GridOrder(76)]
public IPS.Properties.Double Aci
{
    get
    {
        return _Aci;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Exit Area")]
[GridOrder(77)]
public IPS.Properties.Double Ace
{
    get
    {
```

```

        return _Ace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Mean Open Duct Area")]
[GridOrder(78)]
public IPS.Properties.Double Aopen
{
    get
    {
        return _Aopen;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Volume")]
[GridOrder(79)]
public IPS.Properties.Double Vc
{
    get
    {
        return _Vc;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Length")]
[GridOrder(80)]
public IPS.Properties.Double Lfg
{
    get
    {
        return _Lfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Characteristic Length(Circumference)")]
[GridOrder(81)]
public IPS.Properties.Double Pfg
{
    get
    {
        return _Pfg;
    }
}

```



```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Physical Length")]
[GridOrder(82)]
public IPS.Properties.Double Lc
{
    get
    {
        return _Lc;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Pre-cavity to Heat Exchanger Length Ratio")]
[GridOrder(83)]
public IPS.Properties.Double LpctoLc
{
    get
    {
        return _LpctoLc;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Surrounding Duct Wall Area")]
[GridOrder(84)]
public IPS.Properties.Double Aductwall
{
    get
    {
        return _Aductwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Roof Area")]
[GridOrder(85)]
public IPS.Properties.Double Aroof
{
    get
    {
        return _Aroof;
    }
}

```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Floor Area")]
```

```
[GridOrder(86)]
```

```
public IPS.Properties.Double Afloor
```

```
{
```

```
    get
```

```
    {
```

```
        return _Afloor;
```

```
    }
```

```
}
```

```
//Heat exchanger tubes
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Tube Outer Surface Area")]
```

```
[GridOrder(87)]
```

```
public IPS.Properties.Double Atubeouter
```

```
{
```

```
    get
```

```
    {
```

```
        return _Atubeouter;
```

```
    }
```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Heat Transfer Area Correction Factor (Only != 1 in The Case of Platen)")]
```

```
[GridOrder(88)]
```

```
public IPS.Properties.Double HTareacorection
```

```
{
```

```
    get
```

```
    {
```

```
        return _HTareacorection;
```

```
    }
```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("External Heat Transfer Area (Only Differ from Tube Outer Surface Area in  
The Case of Platen)")]
```

```
[GridOrder(89)]
```

```
public IPS.Properties.Double AHTouter
```

```
{
```

```
    get
```

```
    {
```

```

        return _AHTouter;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Inner Surface Area")]
[GridOrder(90)]
public IPS.Properties.Double Atubeinner
{
    get
    {
        return _Atubeinner;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Inner Diameter")]
[GridOrder(91)]
public IPS.Properties.Double Di
{
    get
    {
        return _Di;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Open Duct Effective Hydraulic Diameter")]
[GridOrder(92)]
public IPS.Properties.Double Dhwall
{
    get
    {
        return _Dhwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tube Rows")]
[GridOrder(93)]
public IPS.Properties.Double Nr
{
    get
    {
        return _Nr;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tubes with Flowing Water")]
[GridOrder(94)]
public IPS.Properties.Double Nfwt
{
    get
    {
        return _Nfwt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Diagonal Pitch")]
[GridOrder(95)]
public IPS.Properties.Double sd
{
    get
    {
        return _sd;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Transverse Pitch to Diameter Ratio")]
[GridOrder(96)]
public IPS.Properties.Double Rt
{
    get
    {
        return _Rt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Longitual Pitch to Diameter Ratio")]
[GridOrder(97)]
public IPS.Properties.Double RI
{
    get
    {
        return _RI;
    }
}

```

```

}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Diagonal Pitch to Diameter Ratio")]
[GridOrder(98)]
public IPS.Properties.Double Rd
{
    get
    {
        return _Rd;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Interstitial Gas Velocity Fraction")]
[GridOrder(99)]
public IPS.Properties.Double ufraction
{
    get
    {
        return _ufraction;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Wall Heat Transfer Lengths")]
[GridOrder(100)]
public IPS.Properties.List<IPS.Properties.Double> Lwall
{
    get
    {
        return _Lwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Wall Heat Transfer Areas")]
[GridOrder(101)]
public IPS.Properties.List<IPS.Properties.Double> Awall
{
    get
    {
        return _Awall;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Void Fraction")]
[GridOrder(102)]
public IPS.Properties.Double psi
{
    get
    {
        return _psi;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Fins per Single Tube")]
[GridOrder(103)]
public IPS.Properties.Double Nfinperpass
{
    get
    {
        return _Nfinperpass;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Total Number of Fins")]
[GridOrder(104)]
public IPS.Properties.Double Nfin
{
    get
    {
        return _Nfin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fin Area")]
[GridOrder(105)]
public IPS.Properties.Double Afin
{
    get
    {
        return _Afin;
    }
}

```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fin Efficiency")]
[GridOrder(106)]
public IPS.Properties.Double etaf
{
    get
    {
        return _etaf;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Overall Fin Efficiency")]
[GridOrder(107)]
public IPS.Properties.Double etao
{
    get
    {
        return _etao;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("X fin")]
[GridOrder(108)]
public IPS.Properties.Double Xfin
{
    get
    {
        return _Xfin;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi Fin")]
[GridOrder(109)]
public IPS.Properties.Double phifin
{
    get
    {
        return _phifin;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```

[DisplayName("Phi Fin Prime")]
[GridOrder(110)]
public IPS.Properties.Double phifinprime
{
    get
    {
        return _phifinprime;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Mean Beam Length")]
[GridOrder(111)]
public IPS.Properties.Double sHT
{
    get
    {
        return _sHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Characteristic Length")]
[GridOrder(112)]
public IPS.Properties.Double L
{
    get
    {
        return _L;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Surface Emissivity")]
[GridOrder(113)]
public IPS.Properties.Double episurface
{
    get
    {
        return _episurface;
    }
}

//Thermal and fluid properties

```



```
//Radiation Adjustment due to Slag Screen
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Angular Coefficient of the Slag Screen")]
[GridOrder(114)]
public IPS.Properties.Double xangslag
{
    get
    {
        return _xangslag;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Overall Radiation Heat Transfer Adjustment Coefficient with Slag
Screen")]
[GridOrder(115)]
public IPS.Properties.Double xsslag
{
    get
    {
        return _xsslag;
    }
}
```

```
//Radiations
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Direct Radiation Absorbtion rate")]
[GridOrder(116)]
public IPS.Properties.Double Qradabs
{
    get
    {
        return _Qradabs;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Bypassed Radiation Transfer Rate")]
[GridOrder(117)]
public IPS.Properties.Double Qradpass
{
    get
    {
```

```

        return _Qradpass;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Back Radiation Transfer Rate")]
[GridOrder(118)]
public IPS.Properties.Double Qradback
{
    get
    {
        return _Qradback;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Net Back Radiation Transfer Rate into The Flue Gas Body")]
[GridOrder(119)]
public IPS.Properties.Double Qradbacknet
{
    get
    {
        return _Qradbacknet;
    }
}

//Tube bank pressure loss

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Pressure Loss in The Tube Bank")]
[GridOrder(120)]
public IPS.Properties.Double Deltap
{
    get
    {
        return _Deltap;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Bank Friction Coefficient")]
[GridOrder(121)]
public IPS.Properties.Double zeta
{
    get

```

```

        {
            return _zeta;
        }
    }

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Drag Coefficient in Laminar Flow")]
[GridOrder(122)]
public IPS.Properties.Double zetaI
{
    get
    {
        return _zetaI;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f a laminar")]
[GridOrder(123)]
public IPS.Properties.Double faI
{
    get
    {
        return _faI;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f a Turbulent")]
[GridOrder(124)]
public IPS.Properties.Double faT
{
    get
    {
        return _faT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flow Direction Checking Criteria")]
[GridOrder(125)]
public IPS.Properties.Double checkpoint1
{
    get

```

```

        {
            return _checkpoint1;
        }
    }

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Drag Coefficient in Turbulent Flow")]
[GridOrder(126)]
public IPS.Properties.Double zetat
{
    get
    {
        return _zetat;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f z n, laminar")]
[GridOrder(127)]
public IPS.Properties.Double fznI
{
    get
    {
        return _fznI;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f z, turbulent")]
[GridOrder(128)]
public IPS.Properties.Double fzt
{
    get
    {
        return _fzt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f n turbulent")]
[GridOrder(129)]
public IPS.Properties.Double fnt
{
    get
    {

```

```

        return _fnt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Zeta o")]
[GridOrder(130)]
public IPS.Properties.Double zetao
{
    get
    {
        return _zetao;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("F")]
[GridOrder(131)]
public IPS.Properties.Double F
{
    get
    {
        return _F;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Bundle Overall Drag Coefficient")]
[GridOrder(132)]
public IPS.Properties.Double kforward
{
    get
    {
        return _kforward;
    }
}

//Heat transfer coefficient

//Gas Radiation

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation")]
[GridOrder(133)]
public IPS.Properties.Double hgasrad

```

```

{
    get
    {
        return _hgasrad;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Emperial Constant: A (0: Bitumious or Anthracite/ default, 1: Fuel oil and
gas, 2: Lignite)")]
[GridOrder(134)]
public IPS.Properties.Double A
{
    get
    {
        return _A;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted Heat Transfer Coefficient due to Gas Radiation")]
[GridOrder(135)]
public IPS.Properties.Double hgasradadj
{
    get
    {
        return _hgasradadj;
    }
}

//Tube bank extenal surface convective heat transfer using Gnlinskis method see [1]

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("fa")]
[GridOrder(136)]
public IPS.Properties.Double fa
{
    get
    {
        return _fa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under laminar regime")]

```

```

[GridOrder(137)]
public IPS.Properties.Double Nu0lam
{
    get
    {
        return _Nu0lam;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under turbulent regime")]
[GridOrder(138)]
public IPS.Properties.Double Nu0turb
{
    get
    {
        return _Nu0turb;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row")]
[GridOrder(139)]
public IPS.Properties.Double Nu0row
{
    get
    {
        return _Nu0row;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu over entire tube bank")]
[GridOrder(140)]
public IPS.Properties.Double Nu0
{
    get
    {
        return _Nu0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Temperature Gradient Correction Factor")]
[GridOrder(141)]

```

```

public IPS.Properties.Double K
{
    get
    {
        return _K;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Nassault Number")]
[GridOrder(142)]
public IPS.Properties.Double Nu
{
    get
    {
        return _Nu;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Coefficient from Convection")]
[GridOrder(143)]
public IPS.Properties.Double hconv
{
    get
    {
        return _hconv;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Combined Flue Gas Heat Transfer Coefficient")]
[GridOrder(144)]
public IPS.Properties.Double hcombine
{
    get
    {
        return _hcombine;
    }
}

//Gas Radiation to the duct wall

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation to the Duct Wall")]

```



```

[GridOrder(145)]
public IPS.Properties.Double hgasradowall
{
    get
    {
        return _hgasradowall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted Heat Transfer Coefficient due to Gas Radiation to the Duct
Wall")]
[GridOrder(146)]
public IPS.Properties.Double hgasradadjtowall
{
    get
    {
        return _hgasradadjtowall;
    }
}

//Convective heat transfer to the surrounding duct walls

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Friction Coefficient used in Heat Transfer to the Duct Wall")]
[GridOrder(147)]
public IPS.Properties.Double fwall
{
    get
    {
        return _fwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Nassault Number to The Duct Wall")]
[GridOrder(148)]
public IPS.Properties.Double Nuwall
{
    get
    {
        return _Nuwall;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Coefficient to The Duct Wall")]
[GridOrder(149)]
public IPS.Properties.Double hconvwall
{
    get
    {
        return _hconvwall;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Combined Flue Gas Heat Transfer Coefficient to The Duct Wall")]
[GridOrder(150)]
public IPS.Properties.Double hcombinewall
{
    get
    {
        return _hcombinewall;
    }
}

```

//Gas Radiation to the sling/hanger tubes

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation to the Sling Tube Outer
Surface")]
[GridOrder(151)]
public IPS.Properties.Double hgasradslingtube
{
    get
    {
        return _hgasradslingtube;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted Heat Transfer Coefficient due to Gas Radiation to the Sling Tube
Outer Surface")]
[GridOrder(152)]
public IPS.Properties.Double hgasradadslingtube
{
    get
    {
        return _hgasradadslingtube;
    }
}

```

```

    }
}

//Others

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Effectiveness Applied to The External Surface")]
[GridOrder(153)]
public IPS.Properties.Double HEeffectivenessext
{
    get
    {
        return _HEeffectivenessext;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Effectiveness Applied to The Internal Surface")]
[GridOrder(154)]
public IPS.Properties.Double HEeffectivenessint
{
    get
    {
        return _HEeffectivenessint;
    }
}

//Intemidient steps

//Tri-atomic gases

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kg")]
[GridOrder(155)]
public IPS.Properties.Double kg
{
    get
    {
        return _kg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kfa")]
[GridOrder(156)]

```

```

public IPS.Properties.Double kfa
{
    get
    {
        return _kfa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("krad")]
[GridOrder(157)]
public IPS.Properties.Double krad
{
    get
    {
        return _krad;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("epsilonfg")]
[GridOrder(158)]
public IPS.Properties.Double epsilonfg
{
    get
    {
        return _epsilonfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Specific Volume of H2O")]
[GridOrder(159)]
public IPS.Properties.Double gammaH2O
{
    get
    {
        return _gammaH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-atomic Specific Volume")]
[GridOrder(160)]
public IPS.Properties.Double gamman

```

```

{
    get
    {
        return _gamman;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("H2O Partial Pressure")]
[GridOrder(161)]
public IPS.Properties.Double ppH2O
{
    get
    {
        return _ppH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-Atomic Partial Pressure")]
[GridOrder(162)]
public IPS.Properties.Double ppn
{
    get
    {
        return _ppn;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Constant Pressure Specific Heat at Mean Temperature")]
[GridOrder(163)]
public IPS.Properties.Double cp
{
    get
    {
        return _cp;
    }
}

//Platen superheater direct radiation adjustment

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Radition Loss Correction Factor")]
[GridOrder(164)]
public IPS.Properties.Double phiradloss

```

```

{
    get
    {
        return _phiradloss;
    }
}

//High ash loading volume flow rate

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas Density in The Heat Exchanger")]
[GridOrder(165)]
public IPS.Properties.Double rhofginHT
{
    get
    {
        return _rhofginHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Mass Flow Rate")]
[GridOrder(166)]
public IPS.Properties.Double mfg
{
    get
    {
        return _mfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Volume Flow Rate")]
[GridOrder(167)]
public IPS.Properties.Double Vfg
{
    get
    {
        return _Vfg;
    }
}

//High ash loading effect coefficient check

[PropertyUsage(UseProperty.RESULT)]

```

```

[DisplayName("A in High Ash Loading Calculation")]
[GridOrder(168)]
public IPS.Properties.Double coeffA
{
    get
    {
        return _coeffA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q abs in High Ash Loading Calculation")]
[GridOrder(169)]
public IPS.Properties.Double Qfracabs
{
    get
    {
        return _Qfracabs;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q scatter in High Ash Loading Calculation")]
[GridOrder(170)]
public IPS.Properties.Double Qfracscatter
{
    get
    {
        return _Qfracscatter;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Gamma in High Ash Loading Calculation")]
[GridOrder(171)]
public IPS.Properties.Double coeffgamma
{
    get
    {
        return _coeffgamma;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Beta in High Ash Loading Calculation")]

```

```

[GridOrder(172)]
public IPS.Properties.Double coeffbeta
{
    get
    {
        return _coeffbeta;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Lp in High Ash Loading Calculation")]
[GridOrder(173)]
public IPS.Properties.Double Lp
{
    get
    {
        return _Lp;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Epsilon G in High Ash Loading Calculation")]
[GridOrder(174)]
public IPS.Properties.Double epiG
{
    get
    {
        return _epiG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("a in High Ash Loading Calculation")]
[GridOrder(175)]
public IPS.Properties.List<IPS.Properties.Double> coeffa
{
    get
    {
        return _coeffa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b0 in High Ash Loading Calculation")]
[GridOrder(176)]

```



```

public IPS.Properties.List<IPS.Properties.Double> coeffb0
{
    get
    {
        return _coeffb0;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b1 in High Ash Loading Calculation")]
[GridOrder(177)]
public IPS.Properties.List<IPS.Properties.Double> coeffb1
{
    get
    {
        return _coeffb1;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KG in High Ash Loading Calculation")]
[GridOrder(178)]
public IPS.Properties.List<IPS.Properties.Double> coeffkG
{
    get
    {
        return _coeffkG;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KeG in High Ash Loading Calculation")]
[GridOrder(179)]
public IPS.Properties.Double KeG
{
    get
    {
        return _KeG;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi G+P in High Ash Loading Calculation")]
[GridOrder(180)]

```

```

public IPS.Properties.Double phiGandP
{
    get
    {
        return _phiGandP;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Final Equivalent Fouling Layer Thickness")]
[GridOrder(181)]
public IPS.Properties.Double tfoulingfinal
{
    get
    {
        return _tfoulingfinal;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Interstitial Velocity")]
[GridOrder(182)]
public IPS.Properties.Double ufgmax
{
    get
    {
        return _ufgmax;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Velocity")]
[GridOrder(183)]
public IPS.Properties.Double ufgheat
{
    get
    {
        return _ufgheat;
    }
}
}

```

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll;
//css_ref Flownex.dll;

using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.Components;
using Flownex.DataReferences.Materials;

//Reference
//Heat Atlas VDI G7 [1]
//Heat Atlas VDI L1 [2]
//MEC4117Z Textbook [3]
//Heat Atlas VDI G6 [4]

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Geometrical

    //Pre-cavity
    IPS.Properties.Double _Lpc;

    //Flue gas Cavity
    IPS.Properties.Double _Lc;
    IPS.Properties.Double _Hci;
    IPS.Properties.Double _wci;
    IPS.Properties.Double _Hce;
    IPS.Properties.Double _wce;

    //Heat exchanger tubes
    IPS.Properties.Double _Lpath;
    IPS.Properties.Integer _nsections;
    IPS.Properties.Double _Dotube;
    IPS.Properties.Double _twalltube;
    IPS.Properties.Double _tfouling;
    IPS.Properties.Double _Nt;

```

```

IPS.Properties.Double _Nteff;
IPS.Properties.Double _Nltotal;
IPS.Properties.Double _st;
IPS.Properties.Double _Npp;
IPS.Properties.Double _thetaAOA;
IPS.Properties.Double _Nwall;

//Others
IPS.Properties.Double _beta;
IPS.Properties.Double _Fueltype;
IPS.Properties.Double _HighAshLoading;
IPS.Properties.Double _HEeffectiveness;
IPS.Properties.Double _InternalConvectionCorrelation;
IPS.Properties.Double _ConvectionCorrelationtosurroundingwalls;
IPS.Properties.Double _NuCorrelationTypesurroundingwalls;
IPS.Properties.Double _Foulingoption;
IPS.Properties.Double _Sootbloweroption;
IPS.Properties.Double _Slagtype;

//Thermal or fluid properties

//Flue gas side
IPS.Properties.Double _Qraddirect;
IPS.Properties.Double _Qbackrad;
IPS.Properties.DataReferenceProperty _FluidDataReference;
IPS.Properties.List<IPS.Properties.Double> _MassFractions;
IPS.Properties.Double _Tfg;
IPS.Properties.Double _Tpc;
IPS.Properties.Double _pamb;
IPS.Properties.Double _ufg;
IPS.Properties.Double _Twalloutertube;
IPS.Properties.Double _Twallmean;
IPS.Properties.Double _Tductwalloutertube;
IPS.Properties.Double _dfa;
IPS.Properties.Double _n;

//High ash loading Inputs
IPS.Properties.Double _mproducts;

//Outputs

//Thermal or fluid properties

//Flue gas side

```

```
IPS.Properties.Double _Refg;  
IPS.Properties.Double _Reheat;  
IPS.Properties.Double _Rewall;  
IPS.Properties.Double _Prfg;  
IPS.Properties.Double _rho fg;  
IPS.Properties.Double _lambda fg;  
IPS.Properties.Double _mu;  
IPS.Properties.Double _muw;  
  
//Geometry(pre-simulation data process)  
  
//Flue Gas Cavity CV  
IPS.Properties.Double _Aci;  
IPS.Properties.Double _Ace;  
IPS.Properties.Double _Aopen;  
IPS.Properties.Double _Vc;  
IPS.Properties.Double _Lfg;  
IPS.Properties.Double _Pfg;  
IPS.Properties.Double _LpctoLc;  
  
//Heat exchanger tubes  
IPS.Properties.Double _Atubeouter;  
IPS.Properties.Double _AHTouter;  
IPS.Properties.Double _Atubeinner;  
IPS.Properties.Double _Aprojected;  
IPS.Properties.Double _Arearatio;  
IPS.Properties.Double _Ditube;  
IPS.Properties.Double _Dhwall;  
IPS.Properties.Double _Do;  
IPS.Properties.Double _Di;  
IPS.Properties.Double _twall;  
IPS.Properties.Double _Nr;  
IPS.Properties.Double _Nfwt;  
IPS.Properties.Double _Rt;  
IPS.Properties.List<IPS.Properties.Double> _Lwall;  
IPS.Properties.List<IPS.Properties.Double> _Awall;  
IPS.Properties.Double _psi;  
IPS.Properties.Double _sHT;  
IPS.Properties.Double _L;  
IPS.Properties.Double _episurface;  
  
//Thermal and fluid properties
```

```

//Radiations
IPS.Properties.Double _Qradabs;
IPS.Properties.Double _Qradpass;
IPS.Properties.Double _Qradback;
IPS.Properties.Double _Qradbacknet;

//Heat transfer coefficients

//Gas Radiation
IPS.Properties.Double _hgasrad;
IPS.Properties.Double _A;
IPS.Properties.Double _hgasradadj;

//Tube bank extenal surface convective heat transfer using Gnlinskis method see [1]
IPS.Properties.Double _fa;
IPS.Properties.Double _Nu0lam;
IPS.Properties.Double _Nu0turb;
IPS.Properties.Double _Nu0row;
IPS.Properties.Double _Nu0;
IPS.Properties.Double _K;
IPS.Properties.Double _Nu;
IPS.Properties.Double _hconv;
IPS.Properties.Double _hcombine;

//Gas Radiation to the duct wall
IPS.Properties.Double _hgasradtowall;
IPS.Properties.Double _hgasradadjtowall;

//Convective heat transfer to the surrounding duct walls
IPS.Properties.Double _fwall;
IPS.Properties.Double _Nuwall;
IPS.Properties.Double _hconvwall;
IPS.Properties.Double _hcombinewall;

//Others

//Effectiveness
IPS.Properties.Double _HEeffectivenessext;
IPS.Properties.Double _HEeffectivenessint;

//Constants
double sigmao;

```

```

//Intermediate steps

//Tri-atomic gases
IPS.Properties.Double _kg;
IPS.Properties.Double _kfa;
IPS.Properties.Double _krad;
IPS.Properties.Double _epsilonfg;
IPS.Properties.Double _gammaH2O;
IPS.Properties.Double _gamman;
IPS.Properties.Double _ppH2O;
IPS.Properties.Double _ppn;
double MMfg;
double TA_rho fg;
double H_diff;
double L_floor;
double theta floor;
IPS.Properties.Double _cp;

//High Ash Loading constants
double rhoash;

//High Ash Loading Outputs
IPS.Properties.Double _rho fg in HT;
IPS.Properties.Double _mfg;
IPS.Properties.Double _Vfg;

//High Ash Loading Coefficient Check
IPS.Properties.Double _coeffA;
IPS.Properties.Double _Qfracabs;
IPS.Properties.Double _Qfracscatter;
IPS.Properties.Double _coeffgamma;
IPS.Properties.Double _coeffbeta;
IPS.Properties.Double _Lp;
IPS.Properties.Double _epiG;
IPS.Properties.List<IPS.Properties.Double> _coeffa;
IPS.Properties.List<IPS.Properties.Double> _coeffb0;
IPS.Properties.List<IPS.Properties.Double> _coeffb1;
IPS.Properties.List<IPS.Properties.Double> _coeffkG;
IPS.Properties.Double _KeG;
IPS.Properties.Double _phi Gand P;

//Fouling layer thickness passed to the component
IPS.Properties.Double _tfoulingfinal;

//Intermediate Step for Equivalent Geometry

```

```
double Lactual;
```

```
double DuctWHratio;
```

```
//Report Tube Wall Conductive Heat Transfer Temperatures
```

```
IPS.Properties.Double _Twallfouling;
```

```
IPS.Properties.Double _Twall1;
```

```
IPS.Properties.Double _Twall2;
```

```
IPS.Properties.Double _Twall3;
```

```
IPS.Properties.Double _Twallfoulingresult;
```

```
IPS.Properties.Double _Twall1result;
```

```
IPS.Properties.Double _Twall2result;
```

```
IPS.Properties.Double _Twall3result;
```

```
//do pre simulation initialisation here
```

```
public override void Initialise()
```

```
{
```

```
    //Pre-simulation cleaning to prevent Nan and infinities from previous  
simulation interfering current simulation
```

```
    //Outputs
```

```
    //Thermal and fluid properties
```

```
    //Radiations
```

```
    _Qradabs.Value = 1e-6;
```

```
    _Qradpass.Value = 1e-6;
```

```
    _Qradback.Value = 1e-6;
```

```
    _Qradbacknet.Value = 1e-6;
```

```
    //Heat transfer coefficients
```

```
    //Gas Radiation
```

```
    _hgasrad.Value = 1;
```

```
    _A.Value = 1e-6;
```

```
    _hgasradadj.Value = 1;
```

```
    //Tube bank external surface convective heat transfer using Gnlinskis method
```

see [1]

```
    _fa.Value = 1e-6;
```

```
    _Nu0lam.Value = 1e-6;
```

```
    _Nu0turb.Value = 1e-6;
```

```
    _Nu0row.Value = 1e-6;
```



```
_Nu0.Value = 1e-6;
_K.Value = 1e-6;
_Nu.Value = 1e-6;
_hconv.Value = 1;
_hcombine.Value = 1;

//Gas Radiation to the duct wall
_hgasradtowall.Value = 1;
_hgasradadjtowall.Value = 1;

//Convective heat transfer to the surrounding duct walls
_fwall.Value = 1e-6;
_Nuwall.Value = 1e-6;
_hconvwall.Value = 1;
_hcombinewall.Value = 1;

//Intemidient steps

//Tri-atomic gases
_kg.Value = 1e-6;
_kfa.Value = 1e-6;
_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e-6;
_ppn.Value = 1e-6;
_cp.Value = 1e-6;

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Ceofficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

_mfg.Value = 0;
```

```
_epiG.Value = 0;

_tfoulingfinal.Value = _tfouling.Value;

//Report Tube Wall Conductive Heat Transfer Temperatures
_Twallfoulingresult.Value = 273.15;
_Twall1result.Value = 273.15;
_Twall2result.Value = 273.15;
_Twall3result.Value = 273.15;

//Surface emissivity of the Slag Screen

_episurface.Value = 0.68;

//Identify the correct A

if(_Fueltype == 0) //Default case: Bituminos or Anthracite Coal
{
    _A.Value = 0.4;
}

else if(_Fueltype == 1) //Fuel oil and Gas
{
    _A.Value = 0.3;
}

else if(_Fueltype == 2) //Lignite
{
    _A.Value = 0.5;
}

else
{
    _A.Value = 0.3; //Use default value
}

_HEeffectivenessext.Value = _HEeffectiveness.Value;

_HEeffectivenessint.Value = _HEeffectiveness.Value;

if(_InternalConvectionCorrelation == 1) //Dittus-Boelter
{
    _HEeffectivenessint.Value = _HEeffectiveness.Value * 0.023;
}
```

```

else
{
    _HEffectivenessint.Value = _HEffectiveness.Value;
}

double[] Lsections = new double[_nsections.Value];

double[] nbranches = new double[_nsections.Value];

Lactual = 0;

for(int i = 0; i < _nsections.Value; i++)
{
    Lsections[i] = (_Lpath.Value / _nsections.Value);

    nbranches[i] = (_Nt.Value / _Nteff.Value) - ((_Nt.Value / _Nteff.Value)
/ _nsections.Value) * i;

    Lactual += Lsections[i] * nbranches[i];
}

_Do.Value = _Dotube.Value * Math.Pow(Lactual / _Lpath.Value, 0.5);

_Ditube.Value = _Dotube.Value - 2 * _twalltube.Value;

_Di.Value = _Ditube.Value * Math.Pow(Lactual / _Lpath.Value, 0.5);

_twall.Value = (_Do.Value - _Di.Value) / 2;

}

//do post simulation cleanup here
public override void Cleanup()
{
}

//script main execution function - called every cycle
public override void Execute(double Time)
{
    //Common Geometries

    _Aci.Value = _wci.Value * _Hci.Value; //Heat exchanger cavity inlet open duct
area

```

```

_duct_area.Value = _wce.Value * _Hce.Value; //Heat exchanger cavity outlet open
duct area

_Aopen.Value = (_Aci.Value + _Ace.Value) / 2; //Take average area (assuming
trapizium cross section)

_Dhwall.Value = (4 * _Aopen.Value) / (_wci.Value + _Hci.Value + _wce.Value +
_Hce.Value); //Calculate the effective hydraulic diameter of the duct wall

_Aprojected.Value = Lactual * _Dotube.Value * _Nteff.Value;

_Arearratio.Value = _Aprojected.Value / _Aopen.Value ;

_LpctoLc.Value = _Lpc.Value / _Lc.Value; //Ratio between flue gas pre-cavity
length and cavity length and it's used for gas radiation adjustment

_L.Value = (Math.PI / 2) * _Do.Value;

_Vc.Value = _Aopen.Value * _Lc.Value - (Math.PI * (Math.Pow(_Dotube.Value,
2) / 4)) * Lactual * _Nteff.Value;

_Lfg.Value = _Vc.Value / _Aopen.Value;

_Pfg.Value = (4 * _Aopen.Value) / _Do.Value; //Adjusting the characteristic
length of the flue gas CV so that the Re calculated in flue gas CV uses Do as the flow path length

_Atubeouter.Value = Lactual * Math.PI * _Dotube.Value * _Nteff.Value;

_Atubeinner.Value = Lactual * Math.PI * _Ditube.Value * _Nteff.Value;

_Nfwt.Value = (_Nteff.Value * _Npp.Value);

_Rt.Value = _st.Value / _Do.Value;

//Void fraction, see [4] Eq. (7)

_psi.Value = 1 - (Math.PI / (4 * _Rt.Value));

//Mean Beam Length

_sHT.Value = 3.6 * (_Vc.Value / _Atubeouter.Value);

//Case Specific

//Default values

```

```

_Nr.Value = _Nltotal.Value; //See [2] Fig 2.

_fa.Value = 1; //See [1] Eq. (17)

//Calculate Direct radiation correction coefficient

_AHTouter.Value = _Atubeouter.Value;

//Calculate Fouling Resistance using correlations in [3]

if(_Foulingoption.Value == 1) //User specified fouling resistance
{
    _tfoulingfinal.Value = _tfoulingfinal.Value;
}

else
{
    if(_Slagtype.Value == 0) //Half slagging coal
    {
        if(_Sootbloweroption.Value == 0) //With soot blowing
        {
            _tfoulingfinal.Value = 5.27497e-6 * _Tfg.Value +
1.47684e-3;
        }

        else //Without Soot blowing
        {
            _tfoulingfinal.Value = 1.01224e-5 * _Tfg.Value -
8.99667e-4;
        }
    }

    else if(_Slagtype.Value == 1) //Non-slagging coal
    {
        _tfoulingfinal.Value = 3.85e-6 * _Tfg.Value + 7.46885e-4;
    }

    else //Oil or Gas withsoot blowing
    {
        _tfoulingfinal.Value = 2.77299e-5 * _Tfg.Value -
1.01451e-2;
    }
}

```

```

//Fin calculation

//Calculate Wall heat transfer geometries

for(int i = 0; i < _Nwall.Value ; i++)
{
    _Lwall[i].Value = _twall.Value / _Nwall.Value;

    _Awall[i].Value = Math.PI * (_Nteff.Value * _Npp.Value) *
_Lpath.Value * (_Do.Value - 2 * (i + 1) * _Lwall[i].Value);
}

//Check for High Ash loading case

if(_HighAshLoading == 1)
{
    //Initialise constants

    //b0 Coefficients
    _coeffb0[0].Value = 0.130;
    _coeffb0[1].Value = 0.595;
    _coeffb0[2].Value = 0.275;

    //b1 Coefficients
    _coeffb1[0].Value = 0.000265;
    _coeffb1[1].Value = -0.000150;
    _coeffb1[2].Value = -0.000115;

    //KG Coefficients
    _coeffkG[0].Value = 0.0;
    _coeffkG[1].Value = 0.824;
    _coeffkG[2].Value = 25.907;

    //Calculate Coefficient A
    _coeffA.Value = 3 / (2 * rhoash * _dfa.Value);

    //Calculate Qabs using emperical relation
    _Qfracabs.Value = 0.275 * Math.Pow((_dfa.Value / 1e-6), 0.298) -
0.305;

    //Calculate Qscattering using emperical realtion
    _Qfracscatter.Value = (6.2188e-3 - 1.0492e-2 * (_dfa.Value / 1e-6) +
7.287e-3 * Math.Pow((_dfa.Value / 1e-6), 2) - 2.1925e-5 * Math.Pow((_dfa.Value / 1e-6), 3)) /
(1.851e-1 - 2.0405e-3 * Math.Pow((_dfa.Value / 1e-6), 2) + 6.254e-4 * Math.Pow((_dfa.Value /
1e-6), 3));

```

```

//Calculate Coefficient gamma
_coeffgamma.Value = Math.Pow( 1 + ((2 * _Qfracscatter.Value) /
_Qfracabs.Value), 0.5);

//Calculate Coefficient Beta
_coeffbeta.Value = (_coeffgamma.Value - 1) / (_coeffgamma.Value +
1);

}

else
{
    _mfg.Value = _mfg.Value;
}

if(_mproducts.Value != 0)
{
    //Process Flue Gas data

    List<double> fractions = new List<double>();

    foreach(IPS.Properties.Double val in _MassFractions)
    {
        fractions.Add(val.Value);
    }
    double[] MassFractions = fractions.ToArray();

    MixedFluids fluid = (MixedFluids)_FluidDataReference.Value;

    //Create an array for flue gas mass fraction

    double[] Mfg = new double[fractions.Count];

    double[] fgMolarMass = new double[fractions.Count - 2];

    double[] Pureconstituent = new double[fractions.Count];

    //Pre-loop clean

    _mfg.Value = 0;

    for(int i = 1; i < (fractions.Count - 1); i++)
    {

```

```

Mfg[i] = MassFractions[i] / (1 - (MassFractions[0] +
MassFractions[6]));

_mfg.Value += MassFractions[i] * _mproducts.Value;

Pureconstituent[i] = 1;

fgMolarMass[i - 1] = fluid.GetMolarMass(Pureconstituent,
Parent.Identifier.Value);

Pureconstituent[i] = 0;
}

//Calculate tri-atomic gases volume and partial fraction

//Initialise

double[] gammafg = new double[fractions.Count - 2];

double[] ppfg = new double[fractions.Count - 2];

MMfg = fluid.GetMolarMass(Mfg, Parent.Identifier.Value);

for(int i = 0; i < (fractions.Count - 2); i++)
{
    gammafg[i] = (Mfg[i + 1] * MMfg) / fgMolarMass[i];

    ppfg[i] = _pamb.Value * gammafg[i];
}

_gammaH2O.Value = gammafg[1];

_gamman.Value = gammafg[0] + gammafg[1] + gammafg[4];

_ppH2O.Value = ppfg[1];

_ppn.Value = ppfg[0] + ppfg[1] + ppfg[4];

if(HighAshLoading == 1)
{

    //Get Flue gas density

    _rhofginHT.Value = fluid.GetDensity(_Tfg.Value,
_pamb.Value, Mfg, Parent.Identifier.Value);

```



```

//Calculate Flue gas volume flow rate

_Vfg.Value = _mfg.Value / _rhoHfHT.Value;

//Calculate Lp
_Lp.Value = (_mproducts.Value * _MassFractions[0].Value)
/ _Vfg.Value;

//Calculate Coefficient ai and epsilon G

//Pre-loop clean up

_epiG.Value = 0;

for(int i = 0; i < _coeffa.Count; i++)
{
    _coeffa[i].Value = _coeffb0[i].Value +
    _coeffb1[i].Value * _Tfg.Value;

    _epiG.Value += _coeffa[i].Value * (1 -
Math.Exp(-1 * _coeffkG[i].Value * _ppn.Value * 1e-5 * _sHT.Value));
}

//Calculate K emisivity of Gas
_KeG.Value = -1 * (Math.Log(1 - _epiG.Value) /
_sHT.Value);

//Calculate the optical thickness of gas particulate
dispersion
_phiGandP.Value = (_Qfracabs.Value * _coeffA.Value *
_Lp.Value + _KeG.Value) * _sHT.Value * _coeffgamma.Value;

//Calculate effective gas-particulate emissivity
_epsilonfg.Value = (1 - _coeffbeta.Value) * ((1 -
Math.Exp(-1 * _phiGandP.Value)) / (1 + _coeffbeta.Value * Math.Exp(-1 * _phiGandP.Value)));
}

else //From Gurwich method
{
    _kg.Value = (((7.8 + 16 * _gammaH2O.Value) / (3.16 *
Math.Pow(((_ppn.Value / Math.Pow(10, 6)) * _sHT.Value), 0.5))) - 1) * (1 - 0.37 * (_Tfg.Value /
1000)));
}

```

[illegible]

```
//Apply Pre-cavity correction to hgasradtowall
```

```
    _hgasradadjtowall.Value = _hgasradtowall.Value * (1 + _A.Value *  
Math.Pow(_Tpc.Value / 1000, 0.25) * Math.Pow(_LpctoLc.Value, 0.07)); //See [3] Eq.(6-33)
```

```
//Get Fluid properties from the mean Flue Gas temperature and  
mean wall temperature
```

```
    _rho fg.Value = fluid.GetDensity(_Tfg.Value, _pamb.Value,  
MassFractions, Parent.Identifier.Value);
```

```
    _lambda fg.Value = fluid.GetConductivity(_Tfg.Value, _pamb.Value,  
MassFractions, Parent.Identifier.Value);
```

```
    _cp.Value = fluid.GetSpecificHeat(_Tfg.Value, _pamb.Value,  
MassFractions, Parent.Identifier.Value);
```

```
    _mu.Value = fluid.GetViscosity(_Tfg.Value, _pamb.Value,  
MassFractions, Parent.Identifier.Value);
```

```
    if(_Twallmean.Value <= 0)  
    {  
        _Twallmean.Value = 300;  
    }
```

```
    else  
    {  
        _Twallmean.Value = _Twallmean.Value;  
    }
```

```
    _muw.Value = fluid.GetViscosity(_Twallmean.Value, _pamb.Value,  
MassFractions, Parent.Identifier.Value);
```

```
    if ( _ufg.Value != 0.0 )  
    {  
        _Refg.Value = (_rho fg.Value * _ufg.Value * _Do.Value *  
Math.Sin(_thetaAOA.Value)) / _mu.Value;
```

```
        _Reheat.Value = _Refg.Value * (Math.PI / 2) * (1 /  
_psi.Value); //Used for calculating Nusselt number using Gnielinski's method, see [1]
```

```
    }  
    else  
    {  
        _Refg.Value = _Refg.Value;  
    }
```

```

if ( _lambdafg.Value != 0.0 )
{
    _Prfg.Value = ( _cp.Value * _mu.Value ) / _lambdafg.Value;
}
else
{
    _Prfg.Value = _Prfg.Value;
}

//Calculate Nu0lam per row

_Nu0lam.Value = 0.664 * Math.Pow(_Reheat.Value, 0.5) *
Math.Pow(_Prfg.Value, 0.3333333333); //See [1] Eq. (12)

//Calculate Nu0turb per row

_Nu0turb.Value = (0.037 * Math.Pow(_Reheat.Value, 0.8) *
_Prfg.Value) / (1 + 2.443 * Math.Pow(_Reheat.Value, -0.1) * (Math.Pow(_Prfg.Value,
0.6666666666) - 1)); //See [1] Eq. (13)

//Calculate Nu per row

_Nu0row.Value = 0.3 + Math.Pow((_Nu0lam.Value * _Nu0lam.Value)
+ (_Nu0turb.Value * _Nu0turb.Value), 0.5); //See [1] Eq. (11)

//Calculate Nu0 for the entire tube bundle

if(_Nltotal.Value < 10)
{
    _Nu0.Value = ((1 + ((_Nltotal.Value - 1) * _fa.Value)) /
_Nltotal.Value) * _Nu0row.Value; //See [1] Eq.(21)
}

else
{
    _Nu0.Value = _fa.Value * _Nu0row.Value; //See [1] Eq. (8)
}

// Flue gas Temperature Gradient correction factor

_K.Value = Math.Pow(_Tfg.Value / _Twallmean.Value, _n.Value);

//See [1] Eq. 26

//Calculate Nu with temperature gradient adjustments

```

```
_Nu.Value = _Nu0.Value * _K.Value; //See [1] Eq. 23
```

```
//Calculate heat transfer coefficient due to convection
```

```
_hconv.Value = (_Nu.Value * _lambdafg.Value) / _L.Value; //See [1]
```

Eq. (14)

```
//Combine h
```

```
_hcombine.Value = _HEffectivenessext.Value * (_hgasradadj.Value +  
_hconv.Value); //See [3] Eq. (6-43)
```

```
//Calculate h for the duct wall
```

```
if(_ConvectionCorrelationtosurroundingwalls == 1) //Flat Plates  
{
```

```
    _Rewall.Value = (_rho fg.Value * _ufg.Value * _Lc.Value *  
Math.Sin(_thetaAOA.Value)) / _mu.Value;
```

```
    if(_Rewall.Value < 5e5 ) //Assume fully developed Laminar  
Flow
```

```
{
```

```
    _Nuwall.Value = 0.664 *  
Math.Pow(_Rewall.Value, 0.5) * Math.Pow(_Prfg.Value, 0.33333333);
```

```
    _hconvwall.Value = (_Nuwall.Value *  
lambdafg.Value) / _Lc.Value;
```

```
    _hcombinewall.Value = _hgasradadjtowall.Value  
+ _hconvwall.Value;
```

```
}
```

```
else //Assume fully developed turbulent Flow
```

```
{
```

```
    _Nuwall.Value = 0.037 *  
Math.Pow(_Rewall.Value, 0.8) * Math.Pow(_Prfg.Value, 0.33333333);
```

```
    _hconvwall.Value = (_Nuwall.Value *  
lambdafg.Value) / _Lc.Value;
```

```
    _hcombinewall.Value = _hgasradadjtowall.Value  
+ _hconvwall.Value;
```

```
}
```

```
}
```

```

else //Square Duct Internal Force Convection
{
    _Rewall.Value = (_rho fg.Value * _ufg.Value * _Dhwall.Value
* Math.Sin(_thetaAOA.Value)) / _mu.Value;

    if(_Rewall.Value <= 2800) //Laminar Flow in a Square Duct
    {
        DuctWHratio = ((_Hci.Value + _Hce.Value) / 2) /
((_wci.Value + _wce.Value) / 2);

        //Linearly interpolate Nu

        if(DuctWHratio <= 2)
        {
            _Nuwall.Value = 2.98 + ((3.39 - 2.98) /
(2 - 1)) * (DuctWHratio - 1);

            _hconvwall.Value = (_Nuwall.Value *
lambda fg.Value) / _Dhwall.Value;

            _hcombinewall.Value =
_hgasradadjtowall.Value + _hconvwall.Value;

        }

        else if(DuctWHratio > 2 && DuctWHratio <= 3)
        {
            _Nuwall.Value = 3.39 + ((3.96 - 3.39) /
(3 - 2)) * (DuctWHratio - 2);

            _hconvwall.Value = (_Nuwall.Value *
lambda fg.Value) / _Dhwall.Value;

            _hcombinewall.Value =
_hgasradadjtowall.Value + _hconvwall.Value;

        }

        else if(DuctWHratio > 3 && DuctWHratio <= 4)
        {
            _Nuwall.Value = 3.96 + ((4.44 - 3.96) /
(4 - 3)) * (DuctWHratio - 3);

            _hconvwall.Value = (_Nuwall.Value *
lambda fg.Value) / _Dhwall.Value;

```

```

_hgasradadjtowell.Value + _hconvwall.Value;
}

else if(DuctWHratio > 4 && DuctWHratio <= 6)
{
    _Nuwall.Value = 5.14 + ((5.14 - 4.44) /
(6 - 4)) * (DuctWHratio - 4);

    _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

    _hgasradadjtowell.Value + _hconvwall.Value;
}

else if(DuctWHratio > 6 && DuctWHratio <= 8)
{
    _Nuwall.Value = 5.14 + ((5.6 - 5.14) /
(8 - 6)) * (DuctWHratio - 6);

    _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

    _hgasradadjtowell.Value + _hconvwall.Value;
}

else
{
    _Nuwall.Value = 5.6 + ((7.54 - 5.6) /
(999 - 8)) * (DuctWHratio - 8);

    _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

    _hgasradadjtowell.Value + _hconvwall.Value;
}

else
{
    if(_NuCorrelationTypesurroundingwalls == 1)
//Dittus-Boelter

```

Sling tube screen script

```
{
    _Nuwall.Value = 0.023 *
Math.Pow(_Rewall.Value, 0.8) * Math.Pow(_Prfg.Value, 0.3);

    _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

    _hcombinewall.Value =
_hgasradadjtowell.Value + _hconvwall.Value;
}

else //Use Gnielinski's Correlation
{
    _fwall.Value = Math.Pow((0.79 *
Math.Log(_Rewall.Value) - 1.64), -2);

    _Nuwall.Value = ((_fwall.Value / 8) *
(_Rewall.Value - 1000) * _Prfg.Value) / (1 + 12.7 * Math.Pow((_fwall.Value / 8), 0.5) *
(Math.Pow(_Prfg.Value, 0.666666666666) - 1));

    _hconvwall.Value = (_Nuwall.Value *
lambdafg.Value) / _Dhwall.Value;

    _hcombinewall.Value =
_hgasradadjtowell.Value + _hconvwall.Value;
}

}

//Report Tube Wall Conductive Heat Transfer Temperatures

_Twallfouling.Value = _Twalloutertube.Value;

_Twallfoulingresult.Value = _Twallfouling.Value;

_Twall1result.Value = _Twall1.Value;

_Twall2result.Value = _Twall2.Value;

_Twall3result.Value = _Twall3.Value;

}

else
```



```

        {
            _mproducts.Value = _mproducts.Value;
        }

    }

    //any processing you want to do before steady state
    public override void ExecuteBeforeSteadyState()
    {
        Execute(0.0);
    }

    //any processing you want to do while solving steady state
    public override void ExecuteSteadyState()
    {
        Execute(0.0);
    }

    //any processing you want to do after steady state
    public override void ExecuteAfterSteadyState()
    {
        Execute(0.0);
    }

    //add code here that you want to execute
    //when an associated button is pushed
    public override void HandleObjectEvent(object sender)
    {
    }

    //constructor initialises parameters
    public Script()
    {
        //Inilialise
        //Inputs

        //Geometrical

        //Pre-cavity
        _Lpc = new IPS.Properties.Double();

        //Flue gas Cavity
        _Lc = new IPS.Properties.Double();
    }

```

```

_Hci = new IPS.Properties.Double();
_wci = new IPS.Properties.Double();
_Hce = new IPS.Properties.Double();
_wce = new IPS.Properties.Double();

//Heat exchanger tubes
_Lpath = new IPS.Properties.Double();
_nsections = new IPS.Properties.Integer();
_Dotube = new IPS.Properties.Double();
_twalltube = new IPS.Properties.Double();
_tfouling = new IPS.Properties.Double();
_Nt = new IPS.Properties.Double();
_Nteff = new IPS.Properties.Double();
_Nltotal = new IPS.Properties.Double();
_st = new IPS.Properties.Double();
_Npp = new IPS.Properties.Double();
_thetaAOA = new IPS.Properties.Double();
_Nwall = new IPS.Properties.Double();

//Others
_beta = new IPS.Properties.Double();
_Fueltype = new IPS.Properties.Double();
_HighAshLoading = new IPS.Properties.Double();
_HEffectiveness = new IPS.Properties.Double();
_InternalConvectionCorrelation = new IPS.Properties.Double();
_ConvectionCorrelationtosurroundingwalls = new IPS.Properties.Double();
_NuCorrelationTypesurroundingwalls = new IPS.Properties.Double();
_Foulingoption = new IPS.Properties.Double();
_Sootbloweroption = new IPS.Properties.Double();
_Slagtype = new IPS.Properties.Double();

//Thermal or fluid properties

//Flue gas side
_Qraddirect = new IPS.Properties.Double();
_Qbackrad = new IPS.Properties.Double();
_FluidDataReference = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReference.Types = new Type[]{typeof(MixedFluids)};
_MassFractions = new IPS.Properties.List<IPS.Properties.Double>();
_Tfg = new IPS.Properties.Double();
_Tpc = new IPS.Properties.Double();
_pamb = new IPS.Properties.Double();
_ufg = new IPS.Properties.Double();
_Twalloutertube = new IPS.Properties.Double();

```

```

_Twallmean = new IPS.Properties.Double();
_Tductwalloutertube = new IPS.Properties.Double();
_dfa = new IPS.Properties.Double();
_n = new IPS.Properties.Double();

//High ash loading Inputs
_mproducts = new IPS.Properties.Double();

//Outputs

//Thermal or fluid properties

//Flue gas side
_Refg = new IPS.Properties.Double();
_Reheat = new IPS.Properties.Double();
_Rewall = new IPS.Properties.Double();
_Prfg = new IPS.Properties.Double();
_rhofg = new IPS.Properties.Double();
_lambdafg = new IPS.Properties.Double();
_mu = new IPS.Properties.Double();
_muw = new IPS.Properties.Double();

//Geometry(pre-simulation data process)

//Flue Gas Cavity CV
_Aci = new IPS.Properties.Double();
_Ace = new IPS.Properties.Double();
_Aopen = new IPS.Properties.Double();
_Vc = new IPS.Properties.Double();
_Lfg = new IPS.Properties.Double();
_Pfg = new IPS.Properties.Double();
_LpctoLc = new IPS.Properties.Double();

//Heat exchanger tubes
_Atubeouter = new IPS.Properties.Double();
_AHTouter = new IPS.Properties.Double();
_Atubeinner = new IPS.Properties.Double();
_Aprojected = new IPS.Properties.Double();
_Arearatio = new IPS.Properties.Double();
_Ditube = new IPS.Properties.Double();
_Dhwall = new IPS.Properties.Double();

```

```

_Do = new IPS.Properties.Double();
_Di = new IPS.Properties.Double();
_twall = new IPS.Properties.Double();
_Nr = new IPS.Properties.Double();
_Nfwt = new IPS.Properties.Double();
_Rt = new IPS.Properties.Double();
_Lwall = new IPS.Properties.List<IPS.Properties.Double>();
_Awall = new IPS.Properties.List<IPS.Properties.Double>();
_psi = new IPS.Properties.Double();
_sHT = new IPS.Properties.Double();
_L = new IPS.Properties.Double();
_episurface = new IPS.Properties.Double();

```

```
//Thermal and fluid properties
```

```
//Radiations
```

```

_Quadabs = new IPS.Properties.Double();
_Quadpass = new IPS.Properties.Double();
_Quadback = new IPS.Properties.Double();
_Quadbacknet = new IPS.Properties.Double();

```

```
//Tube bank pressure loss
```

```
//Heat transfer coefficient
```

```
//Gas Radiation
```

```

_hgasrad = new IPS.Properties.Double();
_A = new IPS.Properties.Double();
_hgasradadj = new IPS.Properties.Double();

```

```
//Tube bank extenal surface convective heat transfer using Gnlniskis method
```

see [1]

```

_fa = new IPS.Properties.Double();
_Nu0lam = new IPS.Properties.Double();
_Nu0turb = new IPS.Properties.Double();
_Nu0row = new IPS.Properties.Double();
_Nu0 = new IPS.Properties.Double();
_K = new IPS.Properties.Double();
_Nu = new IPS.Properties.Double();
_hconv = new IPS.Properties.Double();
_hcombine = new IPS.Properties.Double();

```

```
//Gas Radiation to the duct wall
```

```

_hgasradtowall = new IPS.Properties.Double();
_hgasradadjtowall = new IPS.Properties.Double();

//Convective heat transfer to the surrounding duct walls
_fwall = new IPS.Properties.Double();
_Nuwall = new IPS.Properties.Double();
_hconvwall = new IPS.Properties.Double();
_hcombinewall = new IPS.Properties.Double();


//Others

//Effectiveness
_HEeffectivenessext = new IPS.Properties.Double();
_HEeffectivenessint = new IPS.Properties.Double();


//Intemidient steps

//Tri-atomic gases
_kg = new IPS.Properties.Double();
_kfa = new IPS.Properties.Double();
_krad = new IPS.Properties.Double();
_epsilonfg = new IPS.Properties.Double();
_gammaH2O = new IPS.Properties.Double();
_gamman = new IPS.Properties.Double();
_ppH2O = new IPS.Properties.Double();
_ppn = new IPS.Properties.Double();
_cp = new IPS.Properties.Double();


//High Ash Loading Outputs
_rhofginHT = new IPS.Properties.Double();
_mfg = new IPS.Properties.Double();
_Vfg = new IPS.Properties.Double();


//High Ash Loading Coefficient Check
_coffA = new IPS.Properties.Double();
_Qfracabs = new IPS.Properties.Double();
_Qfracscatter = new IPS.Properties.Double();
_coffgamma = new IPS.Properties.Double();
_coffbeta = new IPS.Properties.Double();
_Lp = new IPS.Properties.Double();
_epiG = new IPS.Properties.Double();
_coffA = new IPS.Properties.List<IPS.Properties.Double>();

```

```

_coeffb0 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb1 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffkG = new IPS.Properties.List<IPS.Properties.Double>();
_KeG = new IPS.Properties.Double();
_phiGandP = new IPS.Properties.Double();

//Fouling layer thickness passed to the component
_tfoulingfinal = new IPS.Properties.Double();

//Report Tube Wall Conductive Heat Transfer Temperatures
_Twallfouling = new IPS.Properties.Double();
_Twall1 = new IPS.Properties.Double();
_Twall2 = new IPS.Properties.Double();
_Twall3 = new IPS.Properties.Double();
_Twallfoulingresult = new IPS.Properties.Double();
_Twall1result = new IPS.Properties.Double();
_Twall2result = new IPS.Properties.Double();
_Twall3result = new IPS.Properties.Double();

//Unit
//Inputs

//Geometrical

//Pre-cavity
_Lpc.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Flue gas Cavity
_Lc.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_wci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Hce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_wce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Heat exchanger tubes
_Lpath.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Dotube.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_twalltube.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_tfouling.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_st.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_thetaAOA.DisplayUnitGroup = IPS.Units.UnitGroup.Angle;

```

```
//Thermal or fluid properties

//Flue gas side
_Qraddirect.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qbackrad.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Tpc.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tfg.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_pamb.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
_ufg.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
_Twalloutertube.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twallmean.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tductwalloutertube.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_dfa.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//High ash loading Inputs
_mproducts.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Outputs

//Thermal or fluid properties

//Flue gas side
_rhofg.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_lambdafg.DisplayUnitGroup = IPS.Units.UnitGroup.Thermal_Conductivity;
_mu.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;
_muw.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;

//Geometry(pre-simulation data process)

//Flue Gas Cavity CV
_Aci.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Ace.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aopen.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Vc.DisplayUnitGroup = IPS.Units.UnitGroup.Volume;
_Lfg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Pfg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Heat exchanger tubes
_Atubeouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_AHTouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Atubeinner.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aprojected.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
```

```

_Ditube.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Dhwall.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Do.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Di.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_twall.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_sHT.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_L.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

```

```
//Thermal and fluid properties
```

```
//Radiations
```

```

_Qradabs.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradpass.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradback.DisplayUnitGroup = IPS.Units.UnitGroup.Power;
_Qradbacknet.DisplayUnitGroup = IPS.Units.UnitGroup.Power;

```

```
//Tube bank extenal surface convective heat transfer using Gnlinskis method
```

see [1]

```

_hgasrad.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hgasradadj.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hconv.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hcombine.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;

```

```
//Convective heat transfer to the surrounding duct walls
```

```

_hgasradtowall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hgasradadjtowall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hconvwall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hcombinewall.DisplayUnitGroup =
IPS.Units.UnitGroup.Heat_Transfer_Coefficient;

```

```
//Others
```

```
//Intemidient steps
```

```
//Tri-atomic gases
```

```
_ppH2O.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
```



```

_ppn.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

//High Ash Loading Outputs
_rhofinHT.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_mfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_Vfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Volume;

//Fouling layer thickness passed to the component
_tfoulingfinal.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Report Tube Wall Conductive Heat Transfer Temperatures
_Twallfouling.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall1.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall2.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall3.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twallfoulingresult.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall1result.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall2result.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Twall3result.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;

//Initial values
//Inputs

//Geometrical

//Pre-cavity
_Lpc.Value = 1e-6;

//Flue gas Cavity
_Lc.Value = 1e-6;
_Hci.Value = 1e-6;
_wci.Value = 1e-6;
_Hce.Value = 1e-6;
_wce.Value = 1e-6;

//Heat exchanger tubes
_Lpath.Value = 1e-6;
_nsections.Value = 1;
_Dotube.Value = 1e-6;
_twalltube.Value = 1e-6;
_tfouling.Value = 1e-6;
_Nt.Value = 1;

```

```

_Nteff.Value = 1;
_Nltotal.Value = 1;
_st.Value = 1e-6;
_Npp.Value = 1;
_thetaAOA.Value = Math.PI / 2; //Default value
_Nwall.Value = 3;

//Others
_beta.Value = 1; //Default, no re-radiation, see [3]
_Fueltype.Value = 0; //Default Lignite
_HighAshLoading.Value = 0; //Default, Do not consider
_HEffectiveness.Value = 1; //Default, 100% heat exchanger effectiveness
_InternalConvectionCorrelation.Value = 0; //Default, Gnielinski
_ConvectionCorrelationtosurroundingwalls.Value = 0; //Default, Internal duct
flow with retangular cross-section
_NuCorrelationTypesurroundingwalls.Value = 0; //Default, Gnielinski
_Foulingoption.Value = 0; //Default, use the correlation in [3]
_Sootbloweroption.Value = 0; //Default, with soot blower, see [3]
_Slagtype.Value = 0; //Default, Half slagging coal, see [3]

//Thermal or fluid properties

//Flue gas side
_Qraddirect.Value = 1e-6;
_Qbackrad.Value = 1e-6;
_Tpc.Value = 300; // 26.85 degree by default
_Tfg.Value = 300; // 26.85 degree by default
_pamb.Value = 101325; // Atm pressure by default
_ufg.Value = 1e-6;
_Twalloutertube.Value = 1e-6;
_Twallmean.Value = 1e-6;
_Tductwalloutertube.Value = 1e-6;
_dfa.Value = 16e-6; // High volitle coal by default
_n.Value = 0;

//High ash loading Inputs
_mproducts.Value = 0;

//Outputs

//Thermal or fluid properties

```

```

///Flue gas side
_Refg.Value = 1e3;
_Reheat.Value = 1e3;
_Reheat.Value = 1e3;
_Prfg.Value = 1;
_rhofg.Value = 1e3;
_lambdafg.Value = 1e-6;
_mu.Value = 1e-6;
_muw.Value = 1e-6;

//Geometry(pre-simulation data process)

//Flue Gas Cavity CV
_Aci.Value = 1e-6;
_Ace.Value = 1e-6;
_Aopen.Value = 1e-6;
_Vc.Value = 1e-6;
_Lfg.Value = 1e-6;
_Pfg.Value = 1e-6;
_LpctoLc.Value = 1e-6;

//Heat exchanger tubes
_Atubeouter.Value = 1e-6;
_AHTouter.Value = 1e-6;
_Atubeinner.Value = 1e-6;
_Aprojected.Value = 1e-6;
_Arearatio.Value = 1e-6;
_Ditube.Value = 1e-6;
_Dhwall.Value = 1e-6;
_Do.Value = 1e-6;
_Di.Value = 1e-6;
_twall.Value = 1e-6;
_Nr.Value = 1;
_Nfwt.Value = 1;
_Rt.Value = 1e-6;
_psi.Value = 1e-6;
_sHT.Value = 1e-6;
_L.Value = 1e-6;
_episurface.Value = 0.68;// Default value

//Thermal and fluid properties

//Radiations
_Qradabs.Value = 1e-6;
_Qradpass.Value = 1e-6;

```

```
_Qradback.Value = 1e-6;
_Qradbacknet.Value = 1e-6;

//Heat transfer coefficient

//Gas Radiation
_hgasrad.Value = 1e-6;
_A.Value = 0.4; //Default Value
_hgasradadj.Value = 1e-6;

//Convective heat transfer by Gnlinski method in VDI
_fa.Value = 1e-3;
_Nu0lam.Value = 1e-3;
_Nu0turb.Value = 1e-3;
_Nu0row.Value = 1e-3;
_Nu0.Value = 1e-3;
_K.Value = 1e-6;
_Nu.Value = 1e-6;
_hconv.Value = 1;
_hcombine.Value = 1;

//Gas Radiation to the duct wall
_hgasradtowall.Value = 1;
_hgasradadjtowall.Value = 1;

//Convective heat transfer to the surrounding duct walls
_fwall.Value = 1e-6;
_Nuwall.Value = 1e-6;
_hconvwall.Value = 1;
_hcombinewall.Value = 1;


//Others

//Effectiveness
_HEeffectivenessext.Value = 1;
_HEeffectivenessint.Value = 1;

//Constants
sigmao = 5.67 * Math.Pow(10, -8);

//Intemidient steps
```

```
//Tri-atomic gases
_kg.Value = 1e-6;
_kfa.Value = 1e-6;
_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e3;
_ppn.Value = 1e3;
MMfg = 1e-6;
_cp.Value = 1e3;

//High Ash Loading constants
rhoash = 2200;

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_mfg.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Coefficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_epiG.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

//Fouling layer thickness passed to the component
_tfoulingfinal.Value = 1e-6;

DuctWHratio = 1;

//Report Tube Wall Conductive Heat Transfer Temperatures
_Twallfouling.Value = 300;
_Twall1.Value = 300;
_Twall2.Value = 300;
_Twall3.Value = 300;
_Twallfoulingresult.Value = 300;
_Twall1result.Value = 300;
_Twall2result.Value = 300;
_Twall3result.Value = 300;
```

```

}

//property declarations to make
//parameters visible to outside world
//Inputs

//Geometrical

//Pre-cavity
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Pre-cavity Length")]
[GridOrder(11)]
public IPS.Properties.Double Lpc
{
    get
    {
        return _Lpc;
    }
}

//Flue gas Cavity

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Physical Length")]
[GridOrder(12)]
public IPS.Properties.Double Lc
{
    get
    {
        return _Lc;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Inlet Height")]
[GridOrder(13)]
public IPS.Properties.Double Hci
{
    get
    {
        return _Hci;
    }
}

```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Inlet Width")]
[GridOrder(14)]
public IPS.Properties.Double wci
{
    get
    {
        return _wci;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Outlet Height")]
[GridOrder(15)]
public IPS.Properties.Double Hce
{
    get
    {
        return _Hce;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Cavity Outlet Width")]
[GridOrder(16)]
public IPS.Properties.Double wce
{
    get
    {
        return _wce;
    }
}

//Heat exchanger tubes

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Total Slag Screen Tube Path Length")]
[GridOrder(17)]
public IPS.Properties.Double Lpath
{
    get
    {
        return _Lpath;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Slag Screen Tube Vertical Branches")]
[GridOrder(18)]
public IPS.Properties.Integer nsections
{
    get
    {
        return _nsections;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slag Screen Tube Outer Diameter")]
[GridOrder(19)]
public IPS.Properties.Double Dotube
{
    get
    {
        return _Dotube;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slag Screen Tube Wall Thickness")]
[GridOrder(20)]
public IPS.Properties.Double twalltube
{
    get
    {
        return _twalltube;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Equivalent Thickness to Ash Deposition thermal Resistance on External
Heat Exchanger Surface")]
[GridOrder(21)]
public IPS.Properties.Double tfouling
{
    get
    {
        return _tfouling;
    }
}

```



```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Transverse Tubes")]
[GridOrder(22)]
public IPS.Properties.Double Nt
{
    get
    {
        return _Nt;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Effective Transverse Tubes")]
[GridOrder(23)]
public IPS.Properties.Double Nteff
{
    get
    {
        return _Nteff;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Total Number of Longitudinal Tubes in Heat Exchanger")]
[GridOrder(24)]
public IPS.Properties.Double Nltotal
{
    get
    {
        return _Nltotal;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Bank Transverse Pitch")]
[GridOrder(25)]
public IPS.Properties.Double st
{
    get
    {
        return _st;
    }
}

```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Tubes in Parallel Path")]
[GridOrder(27)]
public IPS.Properties.Double Npp
{
    get
    {
        return _Npp;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Gas Flow Oblique Angle Relative to Tube Bank")]
[GridOrder(28)]
public IPS.Properties.Double thetaAOA
{
    get
    {
        return _thetaAOA;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Nodes in Tube Wall")]
[GridOrder(29)]
public IPS.Properties.Double Nwall
{
    get
    {
        return _Nwall;
    }
}

//Others

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Platen Superheater Reradiation Factor")]
[GridOrder(30)]
public IPS.Properties.Double beta
{
    get

```

```

        {
            return _beta;
        }
    }

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fuel Type (0: Bituminous Coal, 1: Fuel oil and Gas, 2: Lignite)")]
[GridOrder(31)]
public IPS.Properties.Double Fueltype
{
    get
    {
        return _Fueltype;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("High Ash Loading Effect(0: Do Not Considers, 1: Considers)")]
[GridOrder(32)]
public IPS.Properties.Double HighAshLoading
{
    get
    {
        return _HighAshLoading;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Effectiveness")]
[GridOrder(33)]
public IPS.Properties.Double HEeffectiveness
{
    get
    {
        return _HEeffectiveness;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Internal Force Convection Correlation(0: Gnielinski, 1: Dittus-Boelter)")]
[GridOrder(34)]
public IPS.Properties.Double InternalConvectionCorrelation
{
    get
    {

```

```

        return _InternalConvectionCorrelation;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas to Water Wall Convection Correlation Consideration(0: Internal
Duct Flow, 1: Flat Plate)")]
[GridOrder(35)]
public IPS.Properties.Double ConvectionCorrelationtosurroundingwalls
{
    get
    {
        return _ConvectionCorrelationtosurroundingwalls;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Internal Force Convection Correlation in The Duct(0: Gnielinski, 1:
Dittus-Boelter)")]
[GridOrder(36)]
public IPS.Properties.Double NuCorrelationTypesurroundingwalls
{
    get
    {
        return _NuCorrelationTypesurroundingwalls;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fouling Resistance Consideration (0: Use Default Correaltions, 1: User
Specified Fouling Resistance)")]
[GridOrder(37)]
public IPS.Properties.Double Foulingoption
{
    get
    {
        return _Foulingoption;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Soot Blower Consideration(0: With Soot Blower, 1: Without Soot
Blower)")]
[GridOrder(38)]
public IPS.Properties.Double Sootbloweroption

```

```

{
    get
    {
        return _Sootbloweroption;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Slagging Source Consideration(0: Half-slagging Coal, 1: Non-slagging
Coal, 2: Oil or Gas)")]
[GridOrder(39)]
public IPS.Properties.Double Slagtype
{
    get
    {
        return _Slagtype;
    }
}

//Thermal or fluid properties

//Flue gas side

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Direct Radiation")]
[GridOrder(40)]
public IPS.Properties.Double Qraddirect
{
    get
    {
        return _Qraddirect;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Back Radiation from Previous Heat Exchanger")]
[GridOrder(41)]
public IPS.Properties.Double Qbackrad
{
    get
    {
        return _Qbackrad;
    }
}

```

```

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Fluid")]
[GridOrder(42)]
public IPS.Properties.DataReferenceProperty FluidDataReference
{
    get
    {
        return _FluidDataReference;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Mass fractions")]
[GridOrder(43)]
public IPS.Properties.List<IPS.Properties.Double> MassFractions
{
    get
    {
        return _MassFractions;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Flue Gas Temperature")]
[GridOrder(44)]
public IPS.Properties.Double Tfg
{
    get
    {
        return _Tfg;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Temperature in The Pre-Cavity")]
[GridOrder(45)]
public IPS.Properties.Double Tpc
{
    get
    {
        return _Tpc;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]

```

```

[DisplayName("Ambient Pressure in The Heat Exchanger")]
[GridOrder(46)]
public IPS.Properties.Double pamb
{
    get
    {
        return _pamb;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Open-Duct Velocity")]
[GridOrder(47)]
public IPS.Properties.Double ufg
{
    get
    {
        return _ufg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Outer Tube Wall Temperature")]
[GridOrder(48)]
public IPS.Properties.Double Twalloutertube
{
    get
    {
        return _Twalloutertube;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Tube Wall Temperature")]
[GridOrder(49)]
public IPS.Properties.Double Twallmean
{
    get
    {
        return _Twallmean;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Duct Wall Temperature")]

```

```

[GridOrder(50)]
public IPS.Properties.Double Tductwalloutertube
{
    get
    {
        return _Tductwalloutertube;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Ash Particle Size")]
[GridOrder(51)]
public IPS.Properties.Double dfa
{
    get
    {
        return _dfa;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Temperature Gradient adjustification factor")]
[GridOrder(52)]
public IPS.Properties.Double n
{
    get
    {
        return _n;
    }
}

//High ash loading Inputs

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Combustion Product Mass Flow Rate")]
[GridOrder(53)]
public IPS.Properties.Double mproducts
{
    get
    {
        return _mproducts;
    }
}

```



```
//Outputs
```

```
//Thermal or fluid properties
```

```
//Flue gas side
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Open Duct Reynold's Number at Mean Flue Gas Temperature")]
[GridOrder(54)]
public IPS.Properties.Double Refg
{
    get
    {
        return _Refg;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Reynold's Number Use for Heat Transfer Calculation at Mean
Flue Gas Temperature")]
[GridOrder(55)]
public IPS.Properties.Double Reheat
{
    get
    {
        return _Reheat;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Reynold's Number Use for Heat Transfer Calculation to The
Ducts")]
[GridOrder(56)]
public IPS.Properties.Double Rewall
{
    get
    {
        return _Rewall;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Prandtl Number at Mean Flue Gas Temperature")]
[GridOrder(57)]
```

```

public IPS.Properties.Double Prfg
{
    get
    {
        return _Prfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Density at Mean Flue Gas Temperature")]
[GridOrder(58)]
public IPS.Properties.Double rhofg
{
    get
    {
        return _rhofg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Thermal Conductivity at Mean Flue Gas Temperature")]
[GridOrder(59)]
public IPS.Properties.Double lambdafg
{
    get
    {
        return _lambdafg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Flue Gas at Mean Flue Gas Temperature")]
[GridOrder(60)]
public IPS.Properties.Double mu
{
    get
    {
        return _mu;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Flue Gas at Mean Wall Temperature")]
[GridOrder(61)]
public IPS.Properties.Double muw

```

```

{
    get
    {
        return _muw;
    }
}

//Geometry(pre-siulation data process)

//Flue Gas Cavity CV

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Inlet Area")]
[GridOrder(62)]
public IPS.Properties.Double Aci
{
    get
    {
        return _Aci;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Exit Area")]
[GridOrder(63)]
public IPS.Properties.Double Ace
{
    get
    {
        return _Ace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Cavity Mean Open Duct Area")]
[GridOrder(64)]
public IPS.Properties.Double Aopen
{
    get
    {
        return _Aopen;
    }
}

[PropertyUsage(UseProperty.RESULT)]

```

```
[DisplayName("Flue Gas Cavity Volume")]
```

```
[GridOrder(65)]
```

```
public IPS.Properties.Double Vc
```

```
{
```

```
    get
```

```
    {
```

```
        return _Vc;
```

```
    }
```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Flue Gas Cavity Length")]
```

```
[GridOrder(66)]
```

```
public IPS.Properties.Double Lfg
```

```
{
```

```
    get
```

```
    {
```

```
        return _Lfg;
```

```
    }
```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Flue Gas Cavity Characteristic Length(Circumference)")]
```

```
[GridOrder(67)]
```

```
public IPS.Properties.Double Pfg
```

```
{
```

```
    get
```

```
    {
```

```
        return _Pfg;
```

```
    }
```

```
}
```

```
[PropertyUsage(UseProperty.RESULT)]
```

```
[DisplayName("Pre-cavity to Heat Exchanger Length Ratio")]
```

```
[GridOrder(68)]
```

```
public IPS.Properties.Double LpctoLc
```

```
{
```

```
    get
```

```
    {
```

```
        return _LpctoLc;
```

```
    }
```

```
}
```

```
//Heat exchanger tubes
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Outer Surface Area")]
[GridOrder(69)]
public IPS.Properties.Double Atubeouter
{
    get
    {
        return _Atubeouter;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("External Heat Transfer Area (Only Differ from Tube Outer Surface Area in
The Case of Platen)")]
[GridOrder(70)]
public IPS.Properties.Double AHTouter
{
    get
    {
        return _AHTouter;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Inner Surface Area")]
[GridOrder(71)]
public IPS.Properties.Double Atubeinner
{
    get
    {
        return _Atubeinner;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Slag Screen Projected Surface Area")]
[GridOrder(72)]
public IPS.Properties.Double Aprojected
{
    get
    {
        return _Aprojected;
    }
}
```

```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Slag Screen Area Ratio")]
[GridOrder(73)]
public IPS.Properties.Double Areatatio
{
    get
    {
        return _Areatatio;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Slag Screen Tube Inner Diameter")]
[GridOrder(74)]
public IPS.Properties.Double Ditube
{
    get
    {
        return _Ditube;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Open Duct Effective Hydraulic Diameter")]
[GridOrder(75)]
public IPS.Properties.Double Dhwall
{
    get
    {
        return _Dhwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Slag Screen Outer Diameter")]
[GridOrder(76)]
public IPS.Properties.Double Do
{
    get
    {
        return _Do;
    }
}

```

```

}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Slag Screen Inner Diameter")]
[GridOrder(77)]
public IPS.Properties.Double Di
{
    get
    {
        return _Di;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Slag Screen Wall Tickness")]
[GridOrder(78)]
public IPS.Properties.Double twall
{
    get
    {
        return _twall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tube Rows")]
[GridOrder(79)]
public IPS.Properties.Double Nr
{
    get
    {
        return _Nr;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tubes with Flowing Water")]
[GridOrder(80)]
public IPS.Properties.Double Nfwt
{
    get
    {
        return _Nfwt;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Transverse Pitch to Diameter Ratio")]
[GridOrder(81)]
public IPS.Properties.Double Rt
{
    get
    {
        return _Rt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Tube Wall Heat Transfer Lengths")]
[GridOrder(82)]
public IPS.Properties.List<IPS.Properties.Double> Lwall
{
    get
    {
        return _Lwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Tube Wall Heat Transfer Areas")]
[GridOrder(83)]
public IPS.Properties.List<IPS.Properties.Double> Awall
{
    get
    {
        return _Awall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Void Fraction")]
[GridOrder(84)]
public IPS.Properties.Double psi
{
    get
    {
        return _psi;
    }
}

```



```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Slag Screen Mean Beam Length")]
[GridOrder(85)]
public IPS.Properties.Double sHT
{
    get
    {
        return _sHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Characteristic Length")]
[GridOrder(86)]
public IPS.Properties.Double L
{
    get
    {
        return _L;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Surface Emissivity")]
[GridOrder(87)]
public IPS.Properties.Double episurface
{
    get
    {
        return _episurface;
    }
}

//Thermal and fluid properties

//Radiations

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Direct Radiation Absorbtion rate")]
[GridOrder(88)]
public IPS.Properties.Double Qradabs
{
    get
    {

```

```

        return _Qradabs;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Bypassed Radiation Transfer Rate")]
[GridOrder(89)]
public IPS.Properties.Double Qradpass
{
    get
    {
        return _Qradpass;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Back Radiation Transfer Rate")]
[GridOrder(90)]
public IPS.Properties.Double Qradback
{
    get
    {
        return _Qradback;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Net Back Radiation Transfer Rate into The Flue Gas Body")]
[GridOrder(91)]
public IPS.Properties.Double Qradbacknet
{
    get
    {
        return _Qradbacknet;
    }
}

//Heat transfer coefficient

//Gas Radiation

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation")]
[GridOrder(92)]
public IPS.Properties.Double hgasrad

```

```

{
    get
    {
        return _hgasrad;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Emperial Constant: A (0: Bitumious or Anthracite/ default, 1: Fuel oil and
gas, 2: Lignite)")]

```

```

[GridOrder(93)]
public IPS.Properties.Double A
{
    get
    {
        return _A;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted Heat Transfer Coefficient due to Gas Radiation")]
[GridOrder(94)]

```

```

public IPS.Properties.Double hgasradadj
{
    get
    {
        return _hgasradadj;
    }
}

```

//Convective heat transfer by Gnlinskis method in VDI

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("fa")]
[GridOrder(95)]

```

```

public IPS.Properties.Double fa
{
    get
    {
        return _fa;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under laminar regime")]

```

```

[GridOrder(96)]
public IPS.Properties.Double Nu0lam
{
    get
    {
        return _Nu0lam;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under turbulent regime")]
[GridOrder(97)]
public IPS.Properties.Double Nu0turb
{
    get
    {
        return _Nu0turb;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row")]
[GridOrder(98)]
public IPS.Properties.Double Nu0row
{
    get
    {
        return _Nu0row;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu over entire tube bank")]
[GridOrder(99)]
public IPS.Properties.Double Nu0
{
    get
    {
        return _Nu0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Temperature Gradient Correction Factor")]
[GridOrder(100)]

```

```

public IPS.Properties.Double K
{
    get
    {
        return _K;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Nassault Number")]
[GridOrder(101)]
public IPS.Properties.Double Nu
{
    get
    {
        return _Nu;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Coefficient from Convection")]
[GridOrder(102)]
public IPS.Properties.Double hconv
{
    get
    {
        return _hconv;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Combined Flue Gas Heat Transfer Coefficient")]
[GridOrder(103)]
public IPS.Properties.Double hcombine
{
    get
    {
        return _hcombine;
    }
}

//Gas Radiation to the duct wall

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation to the Duct Wall")]

```

```
[GridOrder(104)]
public IPS.Properties.Double hgasradowall
{
    get
    {
        return _hgasradowall;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Adjusted Heat Transfer Coefficient due to Gas Radiation to the Duct
Wall")]
```

```
[GridOrder(105)]
public IPS.Properties.Double hgasradadjtoward
{
    get
    {
        return _hgasradadjtoward;
    }
}
```

```
//Convective heat transfer to the surrounding duct walls
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Friction Coefficient used in Heat Transfer to the Duct Wall")]
[GridOrder(106)]
public IPS.Properties.Double fwall
{
    get
    {
        return _fwall;
    }
}
```

```
[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Nassault Number to The Duct Wall")]
[GridOrder(107)]
public IPS.Properties.Double Nuwall
{
    get
    {
        return _Nuwall;
    }
}
```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Coefficient to The Duct Wall")]
[GridOrder(108)]
public IPS.Properties.Double hconvwall
{
    get
    {
        return _hconvwall;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Combined Flue Gas Heat Transfer Coefficient to The Duct Wall")]
[GridOrder(109)]
public IPS.Properties.Double hcombinewall
{
    get
    {
        return _hcombinewall;
    }
}

//Others

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Effectiveness Applied to The External Surface")]
[GridOrder(110)]
public IPS.Properties.Double HEeffectivenessext
{
    get
    {
        return _HEeffectivenessext;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Effectiveness Applied to The Internal Surface")]
[GridOrder(111)]
public IPS.Properties.Double HEeffectivenessint
{
    get
    {
        return _HEeffectivenessint;
    }
}

```

```
//Intermediate steps
```

```
//Tri-atomic gases
```

```
[PropertyUsage(UseProperty.RESULT)]  
[DisplayName("kg")]  
[GridOrder(112)]  
public IPS.Properties.Double kg  
{  
    get  
    {  
        return _kg;  
    }  
}
```

```
[PropertyUsage(UseProperty.RESULT)]  
[DisplayName("kfa")]  
[GridOrder(113)]  
public IPS.Properties.Double kfa  
{  
    get  
    {  
        return _kfa;  
    }  
}
```

```
[PropertyUsage(UseProperty.RESULT)]  
[DisplayName("krad")]  
[GridOrder(114)]  
public IPS.Properties.Double krad  
{  
    get  
    {  
        return _krad;  
    }  
}
```

```
[PropertyUsage(UseProperty.RESULT)]  
[DisplayName("epsilonfg")]  
[GridOrder(115)]  
public IPS.Properties.Double epsilonfg  
{  
    get  
    {
```



```
        return _epsilonfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Specific Volume of H2O")]
[GridOrder(116)]
public IPS.Properties.Double gammaH2O
{
    get
    {
        return _gammaH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-atomic Specific Volume")]
[GridOrder(117)]
public IPS.Properties.Double gamman
{
    get
    {
        return _gamman;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("H2O Partial Pressure")]
[GridOrder(118)]
public IPS.Properties.Double ppH2O
{
    get
    {
        return _ppH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-Atomic Partial Pressure")]
[GridOrder(119)]
public IPS.Properties.Double ppn
{
    get
    {
        return _ppn;
    }
}
```

```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Constant Pressure Specific Heat at Mean Temperature")]
[GridOrder(120)]
public IPS.Properties.Double cp
{
    get
    {
        return _cp;
    }
}

//High ash loading volume flow rate

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas Density in The Heat Exchanger")]
[GridOrder(121)]
public IPS.Properties.Double rhofginHT
{
    get
    {
        return _rhofginHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Mass Flow Rate")]
[GridOrder(122)]
public IPS.Properties.Double mfg
{
    get
    {
        return _mfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Volume Flow Rate")]
[GridOrder(123)]
public IPS.Properties.Double Vfg
{
    get
    {

```

```

        return _Vfg;
    }
}

//High ash loading effect coefficient check

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("A in High Ash Loading Calculation")]
[GridOrder(124)]
public IPS.Properties.Double coeffA
{
    get
    {
        return _coeffA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q abs in High Ash Loading Calculation")]
[GridOrder(125)]
public IPS.Properties.Double Qfracabs
{
    get
    {
        return _Qfracabs;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q scatter in High Ash Loading Calculation")]
[GridOrder(126)]
public IPS.Properties.Double Qfracscatter
{
    get
    {
        return _Qfracscatter;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Gamma in High Ash Loading Calculation")]
[GridOrder(127)]
public IPS.Properties.Double coeffgamma
{
    get

```

```

        {
            return _coeffgamma;
        }
    }

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Beta in High Ash Loading Calculation")]
[GridOrder(128)]
public IPS.Properties.Double coeffbeta
{
    get
    {
        return _coeffbeta;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Lp in High Ash Loading Calculation")]
[GridOrder(129)]
public IPS.Properties.Double Lp
{
    get
    {
        return _Lp;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Epsilon G in High Ash Loading Calculation")]
[GridOrder(130)]
public IPS.Properties.Double epiG
{
    get
    {
        return _epiG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("a in High Ash Loading Calculation")]
[GridOrder(131)]
public IPS.Properties.List<IPS.Properties.Double> coeffa
{
    get
    {

```

```

        return _coeffa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b0 in High Ash Loading Calculation")]
[GridOrder(132)]
public IPS.Properties.List<IPS.Properties.Double> coeffb0
{
    get
    {
        return _coeffb0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b1 in High Ash Loading Calculation")]
[GridOrder(133)]
public IPS.Properties.List<IPS.Properties.Double> coeffb1
{
    get
    {
        return _coeffb1;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KG in High Ash Loading Calculation")]
[GridOrder(134)]
public IPS.Properties.List<IPS.Properties.Double> coeffkG
{
    get
    {
        return _coeffkG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KeG in High Ash Loading Calculation")]
[GridOrder(135)]
public IPS.Properties.Double KeG
{
    get
    {

```

```

        return _KeG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi G+P in High Ash Loading Calculation")]
[GridOrder(136)]
public IPS.Properties.Double phiGandP
{
    get
    {
        return _phiGandP;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Final Equivalent Fouling Layer Thickness")]
[GridOrder(137)]
public IPS.Properties.Double tfoulingfinal
{
    get
    {
        return _tfoulingfinal;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Carbon Deposition Layer Component Temperature")]
[GridOrder(138)]
public IPS.Properties.Double Twallfouling
{
    get
    {
        return _Twallfouling;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Wall Section 1 Conductive Component Temperature")]
[GridOrder(139)]
public IPS.Properties.Double Twall1
{
    get
    {
        return _Twall1;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Wall Section 2 Conductive Component Temperature")]
[GridOrder(140)]
public IPS.Properties.Double Twall2
{
    get
    {
        return _Twall2;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Wall Section 3 Conductive Component Temperature")]
[GridOrder(141)]
public IPS.Properties.Double Twall3
{
    get
    {
        return _Twall3;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Carbon Deposition Layer Temperature")]
[GridOrder(142)]
public IPS.Properties.Double Twallfoulingresult
{
    get
    {
        return _Twallfoulingresult;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Wall Section 1 Mean Temperature")]
[GridOrder(143)]
public IPS.Properties.Double Twall1result
{
    get
    {
        return _Twall1result;
    }
}

```

```
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Wall Section 2 Mean Temperature")]
[GridOrder(144)]
public IPS.Properties.Double Twall2result
{
    get
    {
        return _Twall2result;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Wall Section 3 Mean Temperature")]
[GridOrder(145)]
public IPS.Properties.Double Twall3result
{
    get
    {
        return _Twall3result;
    }
}

}
```



```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll;
//css_ref Flownex.dll;

using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.Components;
using Flownex.DataReferences.Materials;

//Reference
//Heat Atlas VDI G7 [1]
//Heat Atlas VDI L1 [2]
//MEC4117Z Textbook [3]

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Geometrical

    //Air Cavity
    IPS.Properties.Double _Hci;
    IPS.Properties.Double _wci;
    IPS.Properties.Double _Hce;
    IPS.Properties.Double _wce;

    //Heat exchanger tubes
    IPS.Properties.Double _Lpath;
    IPS.Properties.Double _Do;
    IPS.Properties.Double _twall;
    IPS.Properties.Double _tfouling;
    IPS.Properties.Double _Ntodd;
    IPS.Properties.Double _Nteven;
    IPS.Properties.Double _Nltotal;
    IPS.Properties.Double _st;
    IPS.Properties.Double _sl;
    IPS.Properties.Double _Npp;
    IPS.Properties.Double _thetaAOA;

```

```
IPS.Properties.Double _Nwall;

//Fins
IPS.Properties.Double _Df;
IPS.Properties.Double _Lf;
IPS.Properties.Double _bf;
IPS.Properties.Double _finthickness;
IPS.Properties.Double _sfin;
IPS.Properties.Double _lambdafin;

//Others
IPS.Properties.Double _tubebankarrangement;
IPS.Properties.Double _HEtype;
IPS.Properties.Double _Pass;
IPS.Properties.Double _HighAshLoading;

//Thermal or fluid properties

//Flue gas side
IPS.Properties.DataReferenceProperty _FluidDataReference;
IPS.Properties.List<IPS.Properties.Double> _MassFractions;
IPS.Properties.Double _Tfg;
IPS.Properties.Double _pfg;
IPS.Properties.Double _Twalloutertube;
IPS.Properties.Double _Twallmean;
IPS.Properties.Double _dfa;
IPS.Properties.Double _n;

//Air side
IPS.Properties.DataReferenceProperty _FluidDataReferenceair;
IPS.Properties.List<IPS.Properties.Double> _MassFractionsair;
IPS.Properties.Double _Tair;
IPS.Properties.Double _pair;
IPS.Properties.Double _uair;

//High ash loading Inputs
IPS.Properties.Double _mproducts;

//Outputs

//Thermal or fluid properties

//Air side
```

```

IPS.Properties.Double _Reair;
IPS.Properties.Double _Reairmax;
IPS.Properties.Double _Reheat;
IPS.Properties.Double _Prair;
IPS.Properties.Double _rhoair;
IPS.Properties.Double _lambdaair;
IPS.Properties.Double _mu;
IPS.Properties.Double _muw;

//Geometry(pre-simulation data process)

//Air Cavity CV
IPS.Properties.Double _Aci;
IPS.Properties.Double _Ace;
IPS.Properties.Double _Aopen;
IPS.Properties.Double _Vc;
IPS.Properties.Double _Lair;
IPS.Properties.Double _Pair;
IPS.Properties.Double _Lc;
IPS.Properties.Double _Aductwall;
IPS.Properties.Double _Aroof;
IPS.Properties.Double _Afloor;

//Heat exchanger tubes
IPS.Properties.Double _Atubeouter;
IPS.Properties.Double _AHTouter;
IPS.Properties.Double _Atubeinner;
IPS.Properties.Double _Di;
IPS.Properties.Double _Nr;
IPS.Properties.Double _Nfw;
IPS.Properties.Double _sd;
IPS.Properties.Double _Rt;
IPS.Properties.Double _Rl;
IPS.Properties.Double _Rd;
IPS.Properties.Double _ufraction;
IPS.Properties.List<IPS.Properties.Double> _Lwall;
IPS.Properties.List<IPS.Properties.Double> _Awall;
IPS.Properties.Double _psi;
IPS.Properties.Double _Nfinperpass;
IPS.Properties.Double _Nfin;
IPS.Properties.Double _Afin;
IPS.Properties.Double _etaf;
IPS.Properties.Double _etao;
IPS.Properties.Double _Xfin;
IPS.Properties.Double _phifin;

```

```
IPS.Properties.Double _phifinprime;  
IPS.Properties.Double _sHT;  
IPS.Properties.Double _L;  
IPS.Properties.Double _episurface;
```

```
//Thermal and fluid properties
```

```
//Tube bank pressure loss  
IPS.Properties.Double _Deltap;  
IPS.Properties.Double _zeta;  
IPS.Properties.Double _zetat;  
IPS.Properties.Double _fal;  
IPS.Properties.Double _fat;  
IPS.Properties.Double _checkpoint1;  
IPS.Properties.Double _zetat;  
IPS.Properties.Double _fznl;  
IPS.Properties.Double _fzt;  
IPS.Properties.Double _fnt;  
IPS.Properties.Double _zetao;  
IPS.Properties.Double _F;  
IPS.Properties.Double _kforward;
```

```
//Heat transfer coefficients
```

```
//Gas Radiation  
IPS.Properties.Double _hgasrad;
```

```
//Convective heat transfer by Gnlinskis method in VDI
```

```
IPS.Properties.Double _fa;  
IPS.Properties.Double _Nu0lam;  
IPS.Properties.Double _Nu0turb;  
IPS.Properties.Double _Nu0row;  
IPS.Properties.Double _Nu0;  
IPS.Properties.Double _K;  
IPS.Properties.Double _Nu;  
IPS.Properties.Double _hconv;
```

```
//Others
```

```
//Constants
```

```

double sigmao;

//Intermediate steps

//Tri-atomic gases
IPS.Properties.Double _kg;
IPS.Properties.Double _kfa;
IPS.Properties.Double _krad;
IPS.Properties.Double _epsilonfg;
IPS.Properties.Double _gammaH2O;
IPS.Properties.Double _gamman;
IPS.Properties.Double _ppH2O;
IPS.Properties.Double _ppn;
double MMfg;
double TA_rhofg;
double H_diff;
double L_floor;
double thetafloor;
IPS.Properties.Double _cp;

//High Ash Loading constants
double rhoash;

//High Ash Loading Outputs
IPS.Properties.Double _rhofginHT;
IPS.Properties.Double _mfg;
IPS.Properties.Double _Vfg;

//High Ash Loading Coefficient Check
IPS.Properties.Double _coeffA;
IPS.Properties.Double _Qfracabs;
IPS.Properties.Double _Qfracscatter;
IPS.Properties.Double _coeffgamma;
IPS.Properties.Double _coeffbeta;
IPS.Properties.Double _Lp;
IPS.Properties.Double _epiG;
IPS.Properties.List<IPS.Properties.Double> _coeffa;
IPS.Properties.List<IPS.Properties.Double> _coeffb0;
IPS.Properties.List<IPS.Properties.Double> _coeffb1;
IPS.Properties.List<IPS.Properties.Double> _coeffkG;
IPS.Properties.Double _KeG;
IPS.Properties.Double _phiGandP;

//Fouling layer thickness passed to the component

```

```

IPS.Properties.Double _tfoulingfinal;

//Additional parameters to check with C-schedule
IPS.Properties.Double _uairmax;
IPS.Properties.Double _uairheat;

//do pre simulation initialisation here
public override void Initialise()
{
    //Pre-simulation cleaning to prevent Nan and infinities from previous
simulation interfering current simulation

    //Outputs

    //Thermal and fluid properties

    //Tube bank pressure loss
    _Deltap.Value = 1e-6;
    _zeta.Value = 1e-6;
    _zetat.Value = 1e-6;
    _fal.Value = 1e-6;
    _fat.Value = 1e-6;
    _checkpoint1.Value = 1e-6;
    _zetat.Value = 1e-6;
    _fznl.Value = 1e-6;
    _fzt.Value = 1e-6;
    _fnt.Value = 1e-6;
    _zetao.Value = 1e-6;
    _F.Value = 1e-6;
    _kforward.Value = 1e-6;

    //Heat transfer coefficients

    //Gas Radiation
    _hgasrad.Value = 1e-6;

    //Convective heat transfer by Gnlinskis method in VDI
    _fa.Value = 1e-6;
    _Nu0lam.Value = 1e-6;
    _Nu0turb.Value = 1e-6;
    _Nu0row.Value = 1e-6;
    _Nu0.Value = 1e-6;
    _K.Value = 1e-6;

```

```

_Nu.Value = 1e-6;
_hconv.Value = 1e-6;

//Intermediate steps

//Tri-atomic gases
_kg.Value = 1e-6;
_kfa.Value = 1e-6;
_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e-6;
_ppn.Value = 1e-6;
_cp.Value = 1e-6;

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Coefficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

_mfg.Value = 0;

_epiG.Value = 0;

_tfoulingfinal.Value = _tfouling.Value;

}

//do post simulation cleanup here
public override void Cleanup()
{
}

```

```

//script main execution function - called every cycle
public override void Execute(double Time)
{

    //Surface emissivity always 0.8 in superheats

    _episurface.Value = 0.8;

    //Common Geometries

    H_diff = Math.Abs(_Hci.Value - _Hce.Value); //Calculate the difference in height
    and it is used for calculating the floor tube area (assuming trapezium cross section)

    _Aci.Value = _wci.Value * _Lpath.Value; //Heat exchanger cavity inlet open
duct area

    _Ace.Value = _wce.Value * _Lpath.Value; //Heat exchanger cavity outlet open
duct area

    _Aopen.Value = (_Aci.Value + _Ace.Value) / 2; //Take average area (assuming
trapezium cross section)

    _Lc.Value = (Hci.Value + _Hce.Value) / 2; //Heat exchanger cavity length (only
the region that is covered by the flue gas CV which is just the flue gas surrounding the steam
tubes)

    L_floor = Math.Pow(Math.Pow(H_diff, 2) + Math.Pow(_Lc.Value, 2), 0.5);
//Floor tube length

    _Aductwall.Value = _Lpath.Value * ((_wci.Value + _wce.Value) + (_Hci.Value +
_Hce.Value)); //Cavity side water wall area

    if(_Pass.Value == 0) //Default case, no pass (tower type boiler) / back pass
    {
        _Aroof.Value = 1e-6;

        _Afloor.Value = 1e-6;
    }

    else if(_Pass.Value == 1) //Convective pass
    {
        _Aroof.Value = ((_wci.Value + _wce.Value) / 2) * _Lc.Value -
        (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
        (Math.Pow(_Do.Value, 2) / 4);
    }
}

```



```

        if(_HEtype == 1)
        {
            _Aductwall.Value = _wci.Value * _Hci.Value + (_Hci.Value +
            _Hce.Value) * (Math.Pow(Math.Pow((Math.Abs(_wce.Value - _wci.Value) / 2), 2) +
            Math.Pow(_Lc.Value, 2), 0.5));

            _Afloor.Value = 1e-6;

        }

        else
        {
            _Aductwall.Value = (_Hci.Value + _Hce.Value) *
            (Math.Pow(Math.Pow((Math.Abs(_wce.Value - _wci.Value) / 2), 2) + Math.Pow(_Lc.Value, 2), 0.5));

            _Afloor.Value = ((_wci.Value + _wce.Value) / 2) * L_floor;

        }
    }

    else //Use default case
    {
        _Aroof.Value = 1e-6;

        _Afloor.Value = 1e-6;

    }

    _L.Value = (Math.PI / 2) * _Do.Value;

    _Vc.Value = _Aopen.Value * _Lc.Value - (_Ntodd.Value * (_Npp.Value / 2) +
    _Nteven.Value * (_Npp.Value / 2)) * Math.PI * (Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value;

    _Lair.Value = _Vc.Value / _Aopen.Value;

    _Pair.Value = (4 * _Aopen.Value) / _Do.Value; //Adjusting the characteristic
length of the air CV so that the Re calculated in air CV uses Do as the flow path length

    _Atubeouter.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
    (_Npp.Value / 2)) * _Lpath.Value * Math.PI * _Do.Value;

    _Di.Value = _Do.Value - 2 * _twall.Value;

    _Atubeinner.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
    (_Npp.Value / 2)) * _Lpath.Value * Math.PI * _Di.Value;

```

```

        _Nfwt.Value = (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value *
(_Npp.Value / 2));

        if(_Npp.Value % 2 == 1) //For the case where there is odd number of parallel
path
        {
            _Vc.Value = _Aopen.Value * _Lc.Value - (_Ntodd.Value *
((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) * Math.PI * (Math.Pow(_Do.Value,
2) / 4) * _Lpath.Value;

            _Lair.Value = _Vc.Value / _Aopen.Value;

            _Pair.Value = (4 * _Aopen.Value) / _Do.Value; //Adjusting the
characteristic length of the air CV so that the Re calculated in air CV uses Do as the flow path
length

            _Atubeouter.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
_Nteven.Value * ((_Npp.Value - 1) / 2)) * _Lpath.Value * Math.PI * _Do.Value;

            _Di.Value = _Do.Value - 2 * _twall.Value;

            _Atubeinner.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
_Nteven.Value * ((_Npp.Value - 1) / 2)) * _Lpath.Value * Math.PI * _Di.Value;

            _Nfwt.Value = (_Ntodd.Value * ((_Npp.Value + 1) / 2) +
_Nteven.Value * ((_Npp.Value - 1) / 2));
        }

        else
        {
            _Vc.Value = _Vc.Value;
        }

        _Rt.Value = _st.Value / _Do.Value;

        _Rl.Value = _sl.Value / _Do.Value;

        //Void fraction, see [1] Eq. (9) and (10)

        if(_Rl.Value >= 1)
        {
            _psi.Value = 1 - (Math.PI / (4 * _Rt.Value));
        }

```

```

else
{
    _psi.Value = 1 - (Math.PI / (4 * _Rt.Value * _Rl.Value));
}

//Case Specific

//Default values

_ufraction.Value = _Rt.Value / (_Rt.Value - 1); //Use for adjusting air velocity
see [2] Fig. 2

_fal.Value = (280 * Math.PI * (Math.Pow(Math.Pow(_Rl.Value, 0.5) - 0.6, 2) +
0.75)) / ((4 * _Rt.Value * _Rl.Value - Math.PI) * Math.Pow(_Rt.Value, 1.6)); //See [2] Eq. (4)

_zetao.Value = 1 / Math.Pow(_Rt.Value, 2); //See [2] Eq (19)

_Nr.Value = _Nltotal.Value; //See [2] Fig 2.

if(_tubebankarrangement.Value == 1) //Staggered tube bank
{
    _sd.Value = Math.Pow(((0.5 * _st.Value) * (0.5 * _st.Value)) +
(_sl.Value * _sl.Value), 0.5); //See [1] Fig. 2

    _Rd.Value = _sd.Value / _Do.Value;

    _checkpoint1.Value = 0.5 * Math.Pow(2 * _Rt.Value + 1, 0.5);

    _fat.Value = 2.5 + (1.2 / Math.Pow(_Rt.Value - 0.85, 1.08)) + 0.4 *
Math.Pow((_Rl.Value / _Rt.Value) - 1, 3) - 0.01 * Math.Pow((_Rt.Value / _Rl.Value) - 1, 3); //See [2]
Eq. (12)

    _fa.Value = 1 + (2 / (3 * _Rl.Value)); //See [1] Eq. (18)

    //Mean Beam Length

    _sHT.Value = (4 * _Atubeinner.Value * _Lpath.Value) /
(_Atubeinner.Value);

    if(_Rl.Value < checkpoint1) // Gas flow in diagonal
    {
        _zetao.Value = Math.Pow((2 * (_Rd.Value - 1)) / (_Rt.Value
* (_Rt.Value - 1)), 2); //See [2] Eq. (20)

        _fal.Value = (280 * Math.PI *

```

```
(Math.Pow(Math.Pow(_RI.Value, 0.5) - 0.6, 2) + 0.75)) / ((4 * _Rt.Value * _RI.Value - Math.PI) *
Math.Pow(_Rd.Value, 1.6)); //See [2] Eq. (9)
```

[2] Fig. 2

```
_ufraction.Value = _Rt.Value / (2 * (_Rd.Value - 1)); //See
```

```
_Nr.Value = _Nltotal.Value - 1; //See [2] Fig. 2
```

```
}
```

```
else
```

```
{
```

```
_zetao.Value = _zetao.Value;
```

```
}
```

```
}
```

```
else if(_tubebankarrangement.Value == 0) //In-line tube bank
```

```
{
```

```
_sd.Value = 0;
```

```
_Rd.Value = 0;
```

```
_fat.Value = (0.22 + 1.2 * (Math.Pow(1 - (0.94 / _RI.Value), 0.6) /
Math.Pow(_Rt.Value - 0.85, 1.3))) * Math.Pow(10, 0.47 * ((_RI.Value / _Rt.Value) - 1.5)) + (0.03 *
(_Rt.Value - 1) * (_RI.Value - 1)); //See [2] Eq.(6)
```

```
_fa.Value = 1 + ((0.7 * ((_RI.Value / _Rt.Value) - 0.3)) /
(Math.Pow(_psi.Value, 1.5) * (((_RI.Value / _Rt.Value) + 0.7) * ((_RI.Value / _Rt.Value) + 0.7))));
//See [1] Eq. (17)
```

```
}
```

```
else
```

```
{
```

```
_sd.Value = 999;
```

```
}
```

```
//fnt calc
```

```
if(_Nr.Value < 10)
```

```
{
```

```
if(_Nr.Value >= 5)
```

```
{
```

```
_fnt.Value = _zetao.Value * ((1 / _Nr.Value) - 0.1); //See [2]
```

```
}
```

Eq. (17)

```

        else
        {
            _fnt.Value = 0;
        }
    }

    else
    {
        _fnt.Value = 0; //See [2] Eq. (18)
    }

    _AHTouter.Value = _Atubeouter.Value;

    //Fin calculation

    //Circular fin case

    if(_Df.Value > _Do.Value)
    {
        _Nfinperpass.Value = Math.Round(_Lpath.Value / _sfin.Value) + 1;

        _Nfin.Value = _Nfinperpass.Value * _Nfwf.Value;

        _Afin.Value = _Nfin.Value * ((Math.PI * _Df.Value *
        _finthickness.Value) + 2 * ((Math.PI * (Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4));

        _Atubeouter.Value = (_Atubeouter.Value - (Math.PI * _Do.Value *
        _finthickness * _Nfin)) + _Afin.Value;

        _AHTouter.Value = _Atubeouter.Value;

        _Vc.Value = _Aopen.Value * _Lc.Value - (((Math.PI *
        (Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4) * _finthickness.Value * _Nfin.Value +
        (_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
        (Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the air CV

        _Lair.Value = _Vc.Value / _Aopen.Value; //Update air CV length with
the changed air CV volume

        if(_Npp.Value % 2 == 1) //For the case where there is odd number of
parallel path
        {
            _Vc.Value = _Aopen.Value * _Lc.Value - (((Math.PI *
            (Math.Pow(_Df.Value, 2) - Math.Pow(_Do.Value, 2))) / 4) * _finthickness.Value * _Nfin.Value +

```

```

(_Ntodd.Value * ((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the air CV

_Lair.Value = _Vc.Value / _Aopen.Value; //Update air CV
length with the changed air CV volume
    }

    else
    {
        _Vc.Value = _Vc.Value;
    }

    _phifin.Value = ((_Df.Value / _Do.Value) - 1) * (1 + 0.35 *
Math.Log(_Df.Value / _Do.Value)); //See [3] Eq. (7-13)
    }

    else
    {
        _Afin.Value = 0;
    }

    if(_Lf.Value > _Do.Value)
    {
        if(_bf.Value < _Do.Value)
        {
            _Afin.Value = 0;
        }

        else
        {
            _Nfinperpass.Value = Math.Round(_Lpath.Value /
_sfin.Value) + 1;

            _Nfin.Value = _Nfinperpass.Value * _Nfwt.Value;

            _Afin.Value = _Nfin.Value * ( (2 * (_Lf.Value + _bf.Value) *
_finthickness.Value) + (2 * (_Lf.Value * _bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4)))));

            _Atubeouter.Value = (_Atubeouter.Value - (Math.PI *
_Do.Value * _finthickness * _Nfin)) + _Afin.Value;

            _AHTouter.Value = _Atubeouter.Value;

            _Vc.Value = _Aopen.Value * _Lc.Value - ((2 * (_Lf.Value *
_bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4))) * _finthickness.Value * _Nfin.Value +

```

```
(_Ntodd.Value * (_Npp.Value / 2) + _Nteven.Value * (_Npp.Value / 2)) * Math.PI *
(Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the flue gas CV
```

```
_Lair.Value = _Vc.Value / _Aopen.Value; //Update flue gas
CV length with the changed air CV volume
```

```
if(_Npp.Value % 2 == 1) //For the case where there is odd
number of parallel path
```

```
{
    _Vc.Value = _Aopen.Value * _Lc.Value - ((2 *
(_Lf.Value * _bf.Value - ((Math.PI * Math.Pow(_Do.Value, 2)) / 4))) * _finthickness.Value *
_Nfin.Value + (_Ntodd.Value * ((_Npp.Value + 1) / 2) + _Nteven.Value * ((_Npp.Value - 1) / 2)) *
Math.PI * (Math.Pow(_Do.Value, 2) / 4) * _Lpath.Value); //Subtract fin volumes from the air CV
```

```
_Lair.Value = _Vc.Value / _Aopen.Value;
//Update air CV length with the changed air CV volume
}
```

```
else
```

```
{
```

```
_Vc.Value = _Vc.Value;
```

```
}
```

```
_phifinprime.Value = 1.28 * (_bf.Value / _Do.Value) *
Math.Pow((_Lf.Value / _bf.Value) - 0.2, 0.5); //See [3] Eq. (7-14)
```

```
_phifin.Value = (_phifinprime.Value - 1) * (1 + 0.35 *
Math.Log(_phifinprime.Value)); //See [3] Eq. (7-14)
}
```

```
}
```

```
else
```

```
{
```

```
_Afin.Value = 0;
```

```
}
```

```
//Calculate Wall heat transfer geometries
```

```
for(int i = 0; i < _Nwall.Value ; i++)
```

```
{
```

```
_Lwall[i].Value = _twall.Value / _Nwall.Value;
```

```
_Awall[i].Value = Math.PI * (_Ntodd.Value * (_Npp.Value / 2) +
_Nteven.Value * (_Npp.Value / 2)) * _Lpath.Value * (_Do.Value - 2 * (i + 1) * _Lwall[i].Value);
```

```

}

//Check for High Ash loading case

if(_HighAshLoading == 1)
{
    //Initialise constants

    //b0 Coefficients
    _coeffb0[0].Value = 0.130;
    _coeffb0[1].Value = 0.595;
    _coeffb0[2].Value = 0.275;

    //b1 Coefficients
    _coeffb1[0].Value = 0.000265;
    _coeffb1[1].Value = -0.000150;
    _coeffb1[2].Value = -0.000115;

    //KG Coefficients
    _coeffkG[0].Value = 0.0;
    _coeffkG[1].Value = 0.824;
    _coeffkG[2].Value = 25.907;

    //Calculate Coefficient A
    _coeffA.Value = 3 / (2 * rhoash * _dfa.Value);

    //Calculate Qabs using emperical relation
    _Qfracabs.Value = 0.275 * Math.Pow((_dfa.Value / 1e-6), 0.298) -
0.305;

    //Calculate Qscattering using emperical realtion
    _Qfracscatter.Value = (6.2188e-3 - 1.0492e-2 * (_dfa.Value / 1e-6) +
7.287e-3 * Math.Pow((_dfa.Value / 1e-6), 2) - 2.1925e-5 * Math.Pow((_dfa.Value / 1e-6), 3)) /
(1.851e-1 - 2.0405e-3 * Math.Pow((_dfa.Value / 1e-6), 2) + 6.254e-4 * Math.Pow((_dfa.Value /
1e-6), 3));

    //Calculate Coefficient gamma
    _coeffgamma.Value = Math.Pow( 1 + ((2 * _Qfracscatter.Value) /
_Qfracabs.Value), 0.5);

    //Calculate Coefficient Beta
    _coeffbeta.Value = (_coeffgamma.Value -1) / (_coeffgamma.Value +
1);
}

```



```

else
{
    _mfg.Value = _mfg.Value;
}

//Process Flue Gas data

List<double> fractions = new List<double>();
List<double> fractionsair = new List<double>();

foreach(IPS.Properties.Double val in _MassFractions)
{
    fractions.Add(val.Value);
    fractionsair.Add(val.Value);
}
double[] MassFractions = fractions.ToArray();
double[] MassFractionsair = fractionsair.ToArray();

MixedFluids fluid = (MixedFluids)_FluidDataReference.Value;
MixedFluids fluidair = (MixedFluids)_FluidDataReferenceair.Value;

//Create an array for flue gas mass fraction

double[] Mfg = new double[fractions.Count];

double[] fgMolarMass = new double[fractions.Count - 2];

double[] Pureconstituent = new double[fractions.Count];

//Pre-loop clean

_mfg.Value = 0;

for(int i = 1; i < (fractions.Count - 1); i++)
{
    Mfg[i] = MassFractions[i] / (1 - (MassFractions[0] +
MassFractions[6]));

    _mfg.Value += MassFractions[i] * _mproducts.Value;

    Pureconstituent[i] = 1;

    fgMolarMass[i - 1] = fluid.GetMolarMass(Pureconstituent,

```

```

Parent.Identifier.Value);

        Pureconstituent[i] = 0;
    }

    //Calculate tri-atomic gases volume and partial fraction

    //Initialise

    double[] gammafg = new double[fractions.Count - 2];

    double[] ppfg = new double[fractions.Count - 2];

    MMfg = fluid.GetMolarMass(Mfg, Parent.Identifier.Value);

    for(int i = 0; i < (fractions.Count - 2); i++)
    {
        gammafg[i] = (Mfg[i + 1] * MMfg) / fgMolarMass[i];

        ppfg[i] = _pfg.Value * gammafg[i];
    }

    _gammaH2O.Value = gammafg[1];

    _gamman.Value = gammafg[0] + gammafg[1] + gammafg[4];

    _ppH2O.Value = ppfg[1];

    _ppn.Value = ppfg[0] + ppfg[1] + ppfg[4];

    if(HighAshLoading == 1)
    {

        //Get Flue gas density

        _rhofginHT.Value = fluid.GetDensity(_Tfg.Value, _pfg.Value, Mfg,
Parent.Identifier.Value);

        //Calculate Flue gas volume flow rate

        _Vfg.Value = _mfg.Value / _rhofginHT.Value;

        //Calculate Lp
        _Lp.Value = (_mproducts.Value * _MassFractions[0].Value) /
_Vfg.Value;

```

[illegible]

```

        _epsilonfg.Value = 1 - Math.Exp((_krad.Value * -1) * (_pfg.Value /
Math.Pow(10, 6)) * _sHT.Value);

    }

    //Heat transfer coefficient due to gas radiation

    _hgasrad.Value = 0.8 * sigmao * ((_episurface.Value + 1) / 2) *
_epsilonfg.Value * Math.Pow(_Tfg.Value, 3) * ((1 - Math.Pow(_Twalloutertube.Value / _Tfg.Value,
4)) / (1 - (_Twalloutertube.Value / _Tfg.Value))); //See [3] Eq. (6-31)

    //Get Fluid properties from the mean Air temperature and mean wall
temperature

    _rhoair.Value = fluidair.GetDensity(_Tair.Value, _pair.Value, MassFractionsair,
Parent.Identifier.Value);

    _lambdaair.Value = fluidair.GetConductivity(_Tair.Value, _pair.Value,
MassFractionsair, Parent.Identifier.Value);

    _cp.Value = fluidair.GetSpecificHeat(_Tair.Value, _pair.Value, MassFractionsair,
Parent.Identifier.Value);

    _mu.Value = fluidair.GetViscosity(_Tair.Value, _pair.Value, MassFractionsair,
Parent.Identifier.Value);

    if(_Twallmean.Value <= 0)
    {
        _Twallmean.Value = 300;
    }

    else
    {
        _Twallmean.Value = _Twallmean.Value;
    }

    _muw.Value = fluidair.GetViscosity(_Twallmean.Value, _pair.Value,
MassFractionsair, Parent.Identifier.Value);

    if ( _uair.Value != 0.0 )
    {
        _Reair.Value = (_rhoair.Value * _uair.Value * _Do.Value *
Math.Sin(_thetaAOA.Value)) / _mu.Value;
        _Reairmax.Value = _Reair.Value * _ufraction.Value; //Used for
calculating pressure loss in tube bank using Gnielinski's method, see [2]

```

```

        _Reheat.Value = _Reair.Value * (Math.PI / 2) * (1 / _psi.Value);
//Used for calculating Nusselt number using Gnielinski's method, see [1]
        _uairmax.Value = _uair.Value * _ufraction.Value; //Used for
calculating pressure loss in tube bank using Gnielinski's method, see [2]
        _uairheat.Value = _uair.Value * (Math.PI / 2) * (1 / _psi.Value);
//Used for calculating Nusselt number using Gnielinski's method, see [2]
    }
    else
    {
        _Reair.Value = _Reair.Value;
    }

    if ( _lambdaair.Value != 0.0 )
    {
        _Prair.Value = ( _cp.Value * _mu.Value ) / _lambdaair.Value;
    }
    else
    {
        _Prair.Value = _Prair.Value;
    }

    //Friction coefficient calculation

    //Calculate zetal

    _zetat.Value = _fat.Value / _Reairmax.Value; //See [2] Eq. (3) and (8)

    //Calculate case specific coefficients

    if( _Ntodd.Value == _Nteven.Value ) //In-line tube bank
    {
        _F.Value = 1 - Math.Exp((( _Reairmax.Value + 1000 ) * -1 ) / 2000);
//See Eq. [2] Eq. (2b)

        _zetat.Value = _fat.Value / Math.Pow( _Reairmax.Value, 0.1 *
( _Rl.Value / _Rt.Value )); //See [2] Eq. (5)
    }

    else if( _Ntodd.Value != _Nteven.Value ) //Staggered tube bank
    {
        _F.Value = 1 - Math.Exp((( _Reairmax.Value + 200 ) * -1 ) / 1000); //See
Eq. [2] Eq. (7b)

        _zetat.Value = _fat.Value / Math.Pow( _Reairmax.Value, 0.25 ); //See

```

[2] Eq. (11)

```

    }

    else // Default to be In-line
    {
        _F.Value = 1 - Math.Exp(((_Reairmax.Value + 1000) * -1) / 2000);

        _zetat.Value = _fat.Value / Math.Pow(_Reairmax.Value, 0.1 *
(_Rl.Value / _Rt.Value));

    }

    //Calculate coefficients that's depending on viscosities

    _fzt.Value = Math.Pow(_muw.Value / _mu.Value, 0.14); // See [2] Eq. (14)

    if(_Nr.Value < 10)
    {
        _fznl.Value = Math.Pow(_muw.Value / _mu.Value, (0.57 *
Math.Pow(_Nr.Value / 10, 0.25)) / Math.Pow(((4 * _Rt.Value * _Rl.Value / Math.PI) - 1) *
_Reairmax.Value, 0.25)); //See [2] Eq. (15)
    }

    else
    {
        _fznl.Value = Math.Pow(_muw.Value / _mu.Value, 0.57 /
Math.Pow(((4 * _Rt.Value * _Rl.Value / Math.PI) - 1) * _Reairmax.Value, 0.25)); //See [2] Eq. (16)
    }

    //Calculate friction coefficient (drag coefficient)

    _zeta.Value = _zetat.Value * _fznl.Value + (_zetat.Value * _fzt.Value +
_fnt.Value) * _F.Value; //See [2] Eq. (21) and (22)

    //Calculate kforward and pressure loss in tube bank

    _kforward.Value = _zeta.Value * _Npp.Value * Math.Pow(_ufraction.Value, 2);
//See [2] Eq. (1)

    //Collect air

    _Deltap.Value = _kforward.Value * ((_rhoair.Value * Math.Pow(_uair.Value, 2))
/ 2); //See [2] Eq. (1)

```

```
//Calculate Nu0lam per row
```

```
    _Nu0lam.Value = 0.664 * Math.Pow(_Reheat.Value, 0.5) *  
Math.Pow(_Prair.Value, 0.3333333333); //See [1] Eq. (12)
```

```
//Calculate Nu0turb per row
```

```
    _Nu0turb.Value = (0.037 * Math.Pow(_Reheat.Value, 0.8) * _Prair.Value) / (1 +  
2.443 * Math.Pow(_Reheat.Value, -0.1) * (Math.Pow(_Prair.Value, 0.6666666666) - 1)); //See [1]  
Eq. (13)
```

```
//Calculate Nu per row
```

```
    _Nu0row.Value = 0.3 + Math.Pow((_Nu0lam.Value * _Nu0lam.Value) +  
(_Nu0turb.Value * _Nu0turb.Value), 0.5); //See [1] Eq. (11)
```

```
//Calculate Nu0 for the entire tube bundle
```

```
    if(_Nltotal.Value < 10)  
    {  
        _Nu0.Value = ((1 + ((_Nltotal.Value - 1) * _fa.Value)) / _Nltotal.Value)  
* _Nu0row.Value; //See [1] Eq.(21)  
    }  
  
    else  
    {  
        _Nu0.Value = _fa.Value * _Nu0row.Value; //See [1] Eq. (8)  
    }
```

```
//Air Temperature Gradient correction factor
```

```
    _K.Value = Math.Pow(_Tfg.Value / _Twallmean.Value, _n.Value); //See [1] Eq.
```

26

```
//Calculate Nu with temperature gradient adjustments
```

```
    _Nu.Value = _Nu0.Value * _K.Value; //See [1] Eq. 23
```

```
//Calculate heat transfer coefficient due to convection
```

```
    _hconv.Value = 0.8 * (_Nu.Value * _lambdaair.Value) / _L.Value; //See [1] Eq.
```

(14)

```
    if(_Afin.Value != 0)  
    {
```

```

//Calculate X for fin

_Xfin.Value = _phifin.Value * (_Do.Value / 2) * Math.Pow((2
*_hconv.Value) / (_lambdafin.Value * _finthickness.Value), 0.5); //See [3] Eq. (7-12)

//Calculate fin efficiency

_etaf.Value = Math.Tanh(_Xfin.Value) / _Xfin.Value; //See [3] Eq.
(7-12)

//Calculate overall fin efficiency

_etao.Value = 1 + (1 - _etaf.Value) * (_Afin.Value /
_Atubeouter.Value); //See [3] Eq. (7-7)

//Adjust equivalent fouling layer thickness

_tfoulingfinal.Value = _tfouling.Value / _etao.Value; //See [3] Eq.
(7-8)
    }

    else
    {
        //Calculate X for fin

        _Xfin.Value = 1e-6;

        //Calculate fin efficiency

        _etaf.Value = 1e-6;

        //Calculate overall fin efficiency

        _etao.Value = 1;

    }

}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{
    Execute(0.0);
}

```



```

//any processing you want to do while solving steady state
public override void ExecuteSteadyState()
{
    Execute(0.0);
}

//any processing you want to do after steady state
public override void ExecuteAfterSteadyState()
{
    Execute(0.0);
}

//add code here that you want to execute
//when an associated button is pushed
public override void HandleObjectEvent(object sender)
{
}

//constructor initialises parameters
public Script()
{
    //Initialise
    //Inputs

    //Geometrical

    //Air Cavity
    _Hci = new IPS.Properties.Double();
    _wci = new IPS.Properties.Double();
    _Hce = new IPS.Properties.Double();
    _wce = new IPS.Properties.Double();

    //Heat exchanger tubes
    _Lpath = new IPS.Properties.Double();
    _Do = new IPS.Properties.Double();
    _twall = new IPS.Properties.Double();
    _tfouling = new IPS.Properties.Double();
    _Ntodd = new IPS.Properties.Double();
    _Nteven = new IPS.Properties.Double();
    _Nltotal = new IPS.Properties.Double();
    _st = new IPS.Properties.Double();
    _sl = new IPS.Properties.Double();
}

```

```

_Npp = new IPS.Properties.Double();
_thetaAOA = new IPS.Properties.Double();
_Nwall = new IPS.Properties.Double();

//Fins
_Df = new IPS.Properties.Double();
_Lf = new IPS.Properties.Double();
_bf = new IPS.Properties.Double();
_etaf = new IPS.Properties.Double();
_finthickness = new IPS.Properties.Double();
_sfin = new IPS.Properties.Double();
_lambdafin = new IPS.Properties.Double();

//Others
_tubebankarrangement = new IPS.Properties.Double();
_Hetype = new IPS.Properties.Double();
_Pass = new IPS.Properties.Double();
_HighAshLoading = new IPS.Properties.Double();

//Thermal or fluid properties

//Flue gas side
_FluidDataReference = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReference.Types = new Type[]{typeof(MixedFluids)};
_MassFractions = new IPS.Properties.List<IPS.Properties.Double>();
_Tfg = new IPS.Properties.Double();
_pfg = new IPS.Properties.Double();
_Twalloutertube = new IPS.Properties.Double();
_Twallmean = new IPS.Properties.Double();
_dfa = new IPS.Properties.Double();
_n = new IPS.Properties.Double();

//Air side
_FluidDataReferenceair = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReferenceair.Types = new Type[]{typeof(MixedFluids)};
_MassFractionsair = new IPS.Properties.List<IPS.Properties.Double>();
_Tair = new IPS.Properties.Double();
_pair = new IPS.Properties.Double();
_uair = new IPS.Properties.Double();

//High ash loading Inputs
_mproducts = new IPS.Properties.Double();

//Outputs

```

```

//Thermal or fluid properties

//Air side
_Reair = new IPS.Properties.Double();
_Reairmax = new IPS.Properties.Double();
_Reheat = new IPS.Properties.Double();
_Prair = new IPS.Properties.Double();
_rhoair = new IPS.Properties.Double();
_lambdaair = new IPS.Properties.Double();
_mu = new IPS.Properties.Double();
_muw = new IPS.Properties.Double();

//Geometry(pre-simulation data process)

//Air Cavity CV
_Aci = new IPS.Properties.Double();
_Ace = new IPS.Properties.Double();
_Aopen = new IPS.Properties.Double();
_Vc = new IPS.Properties.Double();
_Lair = new IPS.Properties.Double();
_Pair = new IPS.Properties.Double();
_Lc = new IPS.Properties.Double();
_Aductwall = new IPS.Properties.Double();
_Aroof = new IPS.Properties.Double();
_Afloor = new IPS.Properties.Double();

//Heat exchanger tubes
_Atubeouter = new IPS.Properties.Double();
_AHTouter = new IPS.Properties.Double();
_Atubeinner = new IPS.Properties.Double();
_Di = new IPS.Properties.Double();
_Nr = new IPS.Properties.Double();
_Nfwt = new IPS.Properties.Double();
_sd = new IPS.Properties.Double();
_Rt = new IPS.Properties.Double();
_Rl = new IPS.Properties.Double();
_Rd = new IPS.Properties.Double();
_ufraction = new IPS.Properties.Double();
_Lwall = new IPS.Properties.List<IPS.Properties.Double>();
_Awall = new IPS.Properties.List<IPS.Properties.Double>();
_psi = new IPS.Properties.Double();
_Nfinperpass = new IPS.Properties.Double();
_Nfin = new IPS.Properties.Double();

```

```

_Afin = new IPS.Properties.Double();
_etaf = new IPS.Properties.Double();
_etao = new IPS.Properties.Double();
_Xfin = new IPS.Properties.Double();
_phifin = new IPS.Properties.Double();
_phifinprime = new IPS.Properties.Double();
_sHT = new IPS.Properties.Double();
_L = new IPS.Properties.Double();
_episurface = new IPS.Properties.Double();

//Thermal and fluid properties

//Tube bank pressure loss
_Deltap = new IPS.Properties.Double();
_zeta = new IPS.Properties.Double();
_zetal = new IPS.Properties.Double();
_fal = new IPS.Properties.Double();
_fat = new IPS.Properties.Double();
_checkpoint1 = new IPS.Properties.Double();
_zetat = new IPS.Properties.Double();
_fznl = new IPS.Properties.Double();
_fzt = new IPS.Properties.Double();
_fnt = new IPS.Properties.Double();
_zetao = new IPS.Properties.Double();
_F = new IPS.Properties.Double();
_kforward = new IPS.Properties.Double();

//Heat transfer coefficient

//Gas Radiation
_hgasrad = new IPS.Properties.Double();

//Convective heat transfer by Gnlinski in VDI
_fa = new IPS.Properties.Double();
_Nu0lam = new IPS.Properties.Double();
_Nu0turb = new IPS.Properties.Double();
_Nu0row = new IPS.Properties.Double();
_Nu0 = new IPS.Properties.Double();
_K = new IPS.Properties.Double();
_Nu = new IPS.Properties.Double();
_hconv = new IPS.Properties.Double();

```

```

//Others

//Intemidient steps

//Tri-atomic gases
_kg = new IPS.Properties.Double();
_kfa = new IPS.Properties.Double();
_krad = new IPS.Properties.Double();
_epsilonfg = new IPS.Properties.Double();
_gammaH2O = new IPS.Properties.Double();
_gamman = new IPS.Properties.Double();
_ppH2O = new IPS.Properties.Double();
_ppn = new IPS.Properties.Double();
_cp = new IPS.Properties.Double();

//High Ash Loading Outputs
_rhofginHT = new IPS.Properties.Double();
_mfg = new IPS.Properties.Double();
_Vfg = new IPS.Properties.Double();

//High Ash Loading Coefficient Check
_coeffA = new IPS.Properties.Double();
_Qfracabs = new IPS.Properties.Double();
_Qfracscatter = new IPS.Properties.Double();
_coeffgamma = new IPS.Properties.Double();
_coeffbeta = new IPS.Properties.Double();
_Lp = new IPS.Properties.Double();
_epiG = new IPS.Properties.Double();
_coeffA = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb0 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffb1 = new IPS.Properties.List<IPS.Properties.Double>();
_coeffkG = new IPS.Properties.List<IPS.Properties.Double>();
_KeG = new IPS.Properties.Double();
_phiGandP = new IPS.Properties.Double();

//Fouling layer thickness passed to the component
_tfoulingfinal = new IPS.Properties.Double();

//Additional parameters to check with C-schedule
_uairmax = new IPS.Properties.Double();
_uairheat = new IPS.Properties.Double();

//Unit

```

```
//Inputs
```

```
//Geometrical
```

```
//Air Cavity
```

```
_Hci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_wci.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_Hce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_wce.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
```

```
//Heat exchanger tubes
```

```
_Lpath.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_Do.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_twall.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_tfouling.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_st.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_sl.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_thetaAOA.DisplayUnitGroup = IPS.Units.UnitGroup.Angle;
```

```
//Fins
```

```
_Df.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_Lf.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_bf.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_finthickness.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_sfin.DisplayUnitGroup = IPS.Units.UnitGroup.Length;  
_lambdafin.DisplayUnitGroup = IPS.Units.UnitGroup.Thermal_Conductivity;
```

```
//Thermal or fluid properties
```

```
//Flue gas side
```

```
_Tfg.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;  
_pfg.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;  
_Twalloutertube.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;  
_Twallmean.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;  
_dfa.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
```

```
//Air side
```

```
_Tair.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;  
_pair.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;  
_uair.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
```

```
//High ash loading Inputs
```

```
_mproducts.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
```

```
//Outputs
```

```
//Thermal or fluid properties
```

```
//Air side
```

```
_rhoair.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_lambdaair.DisplayUnitGroup = IPS.Units.UnitGroup.Thermal_Conductivity;
_mu.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;
_muw.DisplayUnitGroup = IPS.Units.UnitGroup.Viscosity_Dynamic;
```

```
//Geometry(pre-simulation data process)
```

```
//Air Cavity CV
```

```
_Aci.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Ace.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aopen.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Vc.DisplayUnitGroup = IPS.Units.UnitGroup.Volume;
_Lair.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Pair.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Lc.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Aductwall.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Aroof.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Afloor.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
```

```
//Heat exchanger tubes
```

```
_Atubeouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_AHTouter.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Atubeinner.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Di.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Afin.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_sHT.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_L.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
```

```
//Thermal and fluid properties
```

```
//Tube bank pressure loss
```

```
_Deltap.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
```

```

//Heat transfer coefficient
_hgasrad.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;
_hconv.DisplayUnitGroup = IPS.Units.UnitGroup.Heat_Transfer_Coefficient;

//Others

//Intemidient steps

//Tri-atomic gases
_ppH2O.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
_ppn.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

//High Ash Loading Outputs
_rhofginHT.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_mfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_Vfg.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Volume;

//Fouling layer thickness passed to the component
_tfoulingfinal.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Additional parameters to check with C-schedule
_uairmax.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
_uairheat.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;

//Initial values
//Inputs

//Geometrical

//Air Cavity
_Hci.Value = 1e-6;
_wci.Value = 1e-6;
_Hce.Value = 1e-6;
_wce.Value = 1e-6;

//Heat exchanger tubes
_Lpath.Value = 1e-6;
_Do.Value = 1e-6;
_twall.Value = 1e-6;
_tfouling.Value = 1e-6;

```



```

_Ntodd.Value = 1;
_Nteven.Value = 1;
_Nltotal.Value = 1;
_st.Value = 1e-6;
_sl.Value = 1e-6;
_Npp.Value = 1;
_thetaAOA.Value = Math.PI / 2; //Default value
_Nwall.Value = 1;

//Fins
_Df.Value = 1e-6;
_Lf.Value = 1e-6;
_bf.Value = 1e-6;
_etaf.Value = 1;
_finthickness.Value = 1e-6;
_sfin.Value = 1e-6;
_lambdafin.Value = 1e-6;

//Others
_tubebankarrangement.Value = 0; //Default Value
_HEtype.Value = 0; //Default value
_Pass.Value = 0; //Default value
_HighAshLoading.Value = 0; //Default value

//Thermal or fluid properties

//Flue gas side
_Tfg.Value = 300; // 26.85 degree by default
_pfg.Value = 101325; // Atm pressure by default
_Twalloutertube.Value = 1e-6;
_Twallmean.Value = 1e-6;
_dfa.Value = 16e-6; // High volatile coal by default
_n.Value = 0;

//Air side
_Tair.Value = 300; // 26.85 degree by default
_pair.Value = 101325; // Atm pressure by default
_uair.Value = 1e-6;

//High ash loading Inputs
_mproducts.Value = 0;

//Outputs

```

```
//Thermal or fluid properties

///Air side
_Reair.Value = 1e3;
_Reairmax.Value = 1e3;
_Reheat.Value = 1e3;
_Prair.Value = 1;
_rhoair.Value = 1e3;
_lambdaair.Value = 1e-6;
_mu.Value = 1e-6;
_muw.Value = 1e-6;

//Geometry(pre-simulation data process)

//Air Cavity CV
_Aci.Value = 1e-6;
_Ace.Value = 1e-6;
_Aopen.Value = 1e-6;
_Vc.Value = 1e-6;
_Lair.Value = 1e-6;
_Pair.Value = 1e-6;
_Lc.Value = 1e-6;
_Aductwall.Value = 1e-6;
_Aroof.Value = 1e-6;
_Afloor.Value = 1e-6;

//Heat exchanger tubes
_Atubeouter.Value = 1e-6;
_AHTouter.Value = 1e-6;
_Atubeinner.Value = 1e-6;
_Di.Value = 1e-6;
_Nr.Value = 1;
_Nfwt.Value = 1;
_sd.Value = 1e-6;
_Rt.Value = 1e-6;
_Rl.Value = 1e-6;
_Rd.Value = 1e-6;
_ufraction.Value = 1e-6;
_psi.Value = 1e-6;
_Nfinperpass.Value = 1e-6;
_Nfin.Value = 1e-6;
_Afin.Value = 1e-6;
_etaf.Value = 1e-6;
```

```
_etao.Value = 1e-6;
_Xfin.Value = 1e-6;
_phifin.Value = 1e-6;
_phifinprime.Value = 1e-6;
_sHT.Value = 1e-6;
_L.Value = 1e-6;
_episurface.Value = 0.8;// Default value

//Thermal and fluid properties

//Tube bank pressure loss
_Deltap.Value = 1e-6;
_zeta.Value = 1e-6;
_zetal.Value = 1e-6;
_fal.Value = 1e-6;
_fat.Value = 1e-6;
_checkpoint1.Value = 1e-6;
_zetat.Value = 1e-6;
_fznl.Value = 1e-6;
_fzt.Value = 1e-6;
_fnt.Value = 1e-6;
_zetao.Value = 1e-6;
_F.Value = 1e-6;
_kforward.Value = 1e-6;

//Heat transfer coefficient

//Gas Radiation
_hgasrad.Value = 1e-6;

//Convective heat transfer by Gnlinski method in VDI
_fa.Value = 1e-3;
_Nu0lam.Value = 1e-3;
_Nu0turb.Value = 1e-3;
_Nu0row.Value = 1e-3;
_Nu0.Value = 1e-3;
_K.Value = 1e-6;
_Nu.Value = 1e-6;
_hconv.Value = 1e-6;

//Others
```

```
//Constants
sigmao = 5.67 * Math.Pow(10, -8);

//Intemidient steps

//Tri-atomic gases
_kg.Value = 1e-6;
_kfa.Value = 1e-6;
_krad.Value = 1e-6;
_epsilonfg.Value = 1e-6;
_gammaH2O.Value = 1e-6;
_gamman.Value = 1e-6;
_ppH2O.Value = 1e3;
_ppn.Value = 1e3;
MMfg = 1e-6;
_cp.Value = 1e3;

//High Ash Loading constants
rhoash = 2200;

//High Ash Loading Outputs
_rhofginHT.Value = 1e-6;
_mfg.Value = 1e-6;
_Vfg.Value = 1e-6;

//High Ash Loading Ceofficient Check
_coeffA.Value = 1e-6;
_Qfracabs.Value = 1e-6;
_Qfracscatter.Value = 1e-6;
_coeffgamma.Value = 1e-6;
_coeffbeta.Value = 1e-6;
_Lp.Value = 1e-6;
_epiG.Value = 1e-6;
_KeG.Value = 1e-6;
_phiGandP.Value = 1e-6;

//Fouling layer thickness passed to the component
_tfoulingfinal.Value = 1e-6;

//Additional parameters to check with C-schedule
_uairmax.Value = 10;
_uairheat.Value = 10;
```

```

}

//property declarations to make
//parameters visible to outside world
//Inputs

//Geometrical

//Air Cavity

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Cavity Inlet Height")]
[GridOrder(11)]
public IPS.Properties.Double Hci
{
    get
    {
        return _Hci;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Cavity Inlet Width")]
[GridOrder(12)]
public IPS.Properties.Double wci
{
    get
    {
        return _wci;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Cavity Outlet Height")]
[GridOrder(13)]
public IPS.Properties.Double Hce
{
    get
    {
        return _Hce;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Cavity Outlet Width")]
[GridOrder(14)]
public IPS.Properties.Double wce
{
    get
    {
        return _wce;
    }
}

//Heat exchanger tubes

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Single Tube Path Length")]
[GridOrder(15)]
public IPS.Properties.Double Lpath
{
    get
    {
        return _Lpath;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Outer Diameter")]
[GridOrder(16)]
public IPS.Properties.Double Do
{
    get
    {
        return _Do;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Wall Thickness")]
[GridOrder(17)]
public IPS.Properties.Double twall
{
    get
    {
        return _twall;
    }
}

```

```

    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Equivalent Thickness to Ash Deposition thermal Resistance on External
Heat Exchanger Surface")]
    [GridOrder(18)]
    public IPS.Properties.Double tfouling
    {
        get
        {
            return _tfouling;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Number of Transverse Tubes in Odd Row")]
    [GridOrder(19)]
    public IPS.Properties.Double Ntodd
    {
        get
        {
            return _Ntodd;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Number of Transverse Tubes in Even Row")]
    [GridOrder(20)]
    public IPS.Properties.Double Nteven
    {
        get
        {
            return _Nteven;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Total Number of Longitudinal Tubes in Heat Exchanger")]
    [GridOrder(21)]
    public IPS.Properties.Double Nltotal
    {
        get
        {
            return _Nltotal;
        }
    }

```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Bank Transverse Pitch")]
[GridOrder(22)]
public IPS.Properties.Double st
{
    get
    {
        return _st;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Bank Longitudinal Pitch")]
[GridOrder(23)]
public IPS.Properties.Double sl
{
    get
    {
        return _sl;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Tubes in Parallel Path")]
[GridOrder(24)]
public IPS.Properties.Double Npp
{
    get
    {
        return _Npp;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Gas Flow Oblique Angle Relative to Tube Bank")]
[GridOrder(25)]
public IPS.Properties.Double thetaAOA
{
    get
    {
        return _thetaAOA;
    }
}

```



```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Number of Nodes in Tube Wall")]
[GridOrder(26)]
public IPS.Properties.Double Nwall
{
    get
    {
        return _Nwall;
    }
}

```

```
//Fins
```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Diameter(For Circular Fins)")]
[GridOrder(27)]
public IPS.Properties.Double Df
{
    get
    {
        return _Df;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Length(For Rectangular Fins)")]
[GridOrder(28)]
public IPS.Properties.Double Lf
{
    get
    {
        return _Lf;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Width(For Rectangular Fins)")]
[GridOrder(29)]
public IPS.Properties.Double bf
{
    get
    {
        return _bf;
    }
}

```

```

}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Thickness")]
[GridOrder(30)]
public IPS.Properties.Double finthickness
{
    get
    {
        return _finthickness;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Pitch")]
[GridOrder(31)]
public IPS.Properties.Double sfin
{
    get
    {
        return _sfin;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Fin Thermal Conductivity")]
[GridOrder(32)]
public IPS.Properties.Double lambdafin
{
    get
    {
        return _lambdafin;
    }
}

//Others

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Tube Arrangement (0: Inline / Default case, 1: Staggered)")]
[GridOrder(33)]
public IPS.Properties.Double tubebankarrangement
{
    get
    {

```

```

        return _tubebankarrangement;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Type (0: Convective, 1: Platen)")]
[GridOrder(34)]
public IPS.Properties.Double HEtype
{
    get
    {
        return _HEtype;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Heat Exchanger Location (0: Back Pass / No Pass, 1: Convective Pass)")]
[GridOrder(35)]
public IPS.Properties.Double Pass
{
    get
    {
        return _Pass;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("High Ash Loading Effect(0: Do Not Considers, 1: Considers)")]
[GridOrder(36)]
public IPS.Properties.Double HighAshLoading
{
    get
    {
        return _HighAshLoading;
    }
}

//Thermal or fluid properties

//Flue gas side

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Flue Gas")]
[GridOrder(37)]
public IPS.Properties.DataReferenceProperty FluidDataReference

```

```

{
    get
    {
        return _FluidDataReference;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Mass fractions")]
[GridOrder(38)]
public IPS.Properties.List<IPS.Properties.Double> MassFractions
{
    get
    {
        return _MassFractions;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Flue Gas Temperature")]
[GridOrder(39)]
public IPS.Properties.Double Tfg
{
    get
    {
        return _Tfg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Pressure in The Heat Exchanger")]
[GridOrder(40)]
public IPS.Properties.Double pfg
{
    get
    {
        return _pfg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Outer Tube Wall Temperature")]
[GridOrder(41)]
public IPS.Properties.Double Twalloutertube
{

```

```

        get
        {
            return _Twalloutertube;
        }
    }

    [PropertyUsage(Property.DYNAMIC)]
    [DisplayName("Mean Tube Wall Temperature")]
    [GridOrder(42)]
    public IPS.Properties.Double Twallmean
    {
        get
        {
            return _Twallmean;
        }
    }

    [PropertyUsage(Property.DYNAMIC)]
    [DisplayName("Ash Particle Size")]
    [GridOrder(43)]
    public IPS.Properties.Double dfa
    {
        get
        {
            return _dfa;
        }
    }

    [PropertyUsage(Property.DYNAMIC)]
    [DisplayName("Temperature Gradient justification factor")]
    [GridOrder(44)]
    public IPS.Properties.Double n
    {
        get
        {
            return _n;
        }
    }

    //Air side

    [PropertyUsage(Property.STATIC)]
    [DisplayName("Air")]
    [GridOrder(45)]
    public IPS.Properties.DataReferenceProperty FluidDataReferenceair

```

```

{
    get
    {
        return _FluidDataReferenceair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Mass fractions")]
[GridOrder(46)]
public IPS.Properties.List<IPS.Properties.Double> MassFractionsair
{
    get
    {
        return _MassFractionsair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Air Temperature")]
[GridOrder(47)]
public IPS.Properties.Double Tair
{
    get
    {
        return _Tair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Pressure in The Heat Exchanger")]
[GridOrder(48)]
public IPS.Properties.Double pair
{
    get
    {
        return _pair;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mean Air Velocity")]
[GridOrder(49)]
public IPS.Properties.Double uair
{

```

```

        get
        {
            return _uair;
        }
    }

//High ash loading Inputs

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Combustion Products Mass Flow Rate")]
[GridOrder(50)]
public IPS.Properties.Double mproducts
{
    get
    {
        return _mproducts;
    }
}

//Outputs

//Thermal or fluid properties

//Air side

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Open Duct Reynold's Number at Mean Flue Gas Temperature")]
[GridOrder(51)]
public IPS.Properties.Double Reair
{
    get
    {
        return _Reair;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Interstitial Reynold's Number at Mean Air Temperature")]
[GridOrder(52)]
public IPS.Properties.Double Reairmax
{
    get
    {

```

```

        return _Reairmax;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Reynold's Number Use for Heat Transfer Calculation at Mean Air
Temperature")]
[GridOrder(53)]
public IPS.Properties.Double Reheat
{
    get
    {
        return _Reheat;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Prandtl Number at Mean Air Temperature")]
[GridOrder(54)]
public IPS.Properties.Double Prair
{
    get
    {
        return _Prair;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Density at Mean Air Temperature")]
[GridOrder(55)]
public IPS.Properties.Double rhoair
{
    get
    {
        return _rhoair;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Thermal Conductivity at Mean Air Temperature")]
[GridOrder(56)]
public IPS.Properties.Double lambdaair
{
    get
    {

```



```

        return _lambdaair;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Flue Gas at Mean Air Temperature")]
[GridOrder(57)]
public IPS.Properties.Double mu
{
    get
    {
        return _mu;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Dynamic Viscosity of Air at Mean Wall Temperature")]
[GridOrder(58)]
public IPS.Properties.Double muw
{
    get
    {
        return _muw;
    }
}

//Geometry(pre-simulation data process)

//Air Cavity CV

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Inlet Area")]
[GridOrder(59)]
public IPS.Properties.Double Aci
{
    get
    {
        return _Aci;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Exit Area")]
[GridOrder(60)]
public IPS.Properties.Double Ace

```

```

{
    get
    {
        return _Ace;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Mean Open Duct Area")]
[GridOrder(61)]
public IPS.Properties.Double Aopen
{
    get
    {
        return _Aopen;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Volume")]
[GridOrder(62)]
public IPS.Properties.Double Vc
{
    get
    {
        return _Vc;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Length")]
[GridOrder(63)]
public IPS.Properties.Double Lair
{
    get
    {
        return _Lair;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Cavity Characteristic Length(Circumference)")]
[GridOrder(64)]
public IPS.Properties.Double Pair
{

```

```

        get
        {
            return _Pair;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Air Cavity Physical Length")]
    [GridOrder(65)]
    public IPS.Properties.Double Lc
    {
        get
        {
            return _Lc;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Surrounding Duct Wall Area")]
    [GridOrder(66)]
    public IPS.Properties.Double Aductwall
    {
        get
        {
            return _Aductwall;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Roof Area")]
    [GridOrder(67)]
    public IPS.Properties.Double Aroof
    {
        get
        {
            return _Aroof;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Floor Area")]
    [GridOrder(68)]
    public IPS.Properties.Double Afloor
    {
        get

```

```

        {
            return _Afloor;
        }
    }

```

```
//Heat exchanger tubes
```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Outer Surface Area")]
[GridOrder(69)]
public IPS.Properties.Double Atubeouter
{
    get
    {
        return _Atubeouter;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("External Heat Transfer Area (Only Differ from Tube Outer Surface Area in
The Case of Platen)")]
[GridOrder(70)]
public IPS.Properties.Double AHTouter
{
    get
    {
        return _AHTouter;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Inner Surface Area")]
[GridOrder(71)]
public IPS.Properties.Double Atubeinner
{
    get
    {
        return _Atubeinner;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Inner Diameter")]
[GridOrder(72)]
public IPS.Properties.Double Di

```

```

{
    get
    {
        return _Di;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tube Rows")]
[GridOrder(73)]
public IPS.Properties.Double Nr
{
    get
    {
        return _Nr;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Number of Tubes with Flowing Water")]
[GridOrder(74)]
public IPS.Properties.Double Nfwt
{
    get
    {
        return _Nfwt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Diagonal Pitch")]
[GridOrder(75)]
public IPS.Properties.Double sd
{
    get
    {
        return _sd;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Transverse Pitch to Diameter Ratio")]
[GridOrder(76)]
public IPS.Properties.Double Rt
{

```

```

        get
        {
            return _Rt;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Longitunal Pitch to Diameter Ratio")]
    [GridOrder(77)]
    public IPS.Properties.Double RI
    {
        get
        {
            return _RI;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Diagonal Pitch to Diameter Ratio")]
    [GridOrder(78)]
    public IPS.Properties.Double Rd
    {
        get
        {
            return _Rd;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Interstitial Gas Velocity Fraction")]
    [GridOrder(79)]
    public IPS.Properties.Double ufraction
    {
        get
        {
            return _ufraction;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Tube Wall Heat Transfer Lengths")]
    [GridOrder(80)]
    public IPS.Properties.List<IPS.Properties.Double> Lwall
    {
        get

```

```

        {
            return _Lwall;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Tube Wall Heat Transfer Areas")]
    [GridOrder(81)]
    public IPS.Properties.List<IPS.Properties.Double> Awall
    {
        get
        {
            return _Awall;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Void Fraction")]
    [GridOrder(82)]
    public IPS.Properties.Double psi
    {
        get
        {
            return _psi;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Number of Fins per Single Tube")]
    [GridOrder(83)]
    public IPS.Properties.Double Nfinperpass
    {
        get
        {
            return _Nfinperpass;
        }
    }

    [PropertyUsage(UseProperty.RESULT)]
    [DisplayName("Total Number of Fins")]
    [GridOrder(84)]
    public IPS.Properties.Double Nfin
    {
        get
        {

```

```

        return _Nfin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fin Area")]
[GridOrder(85)]
public IPS.Properties.Double Afin
{
    get
    {
        return _Afin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Fin Efficiency")]
[GridOrder(86)]
public IPS.Properties.Double etaf
{
    get
    {
        return _etaf;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Overall Fin Efficiency")]
[GridOrder(87)]
public IPS.Properties.Double etao
{
    get
    {
        return _etao;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("X fin")]
[GridOrder(88)]
public IPS.Properties.Double Xfin
{
    get
    {
        return _Xfin;
    }
}

```



```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi Fin")]
[GridOrder(89)]
public IPS.Properties.Double phifin
{
    get
    {
        return _phifin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Phi Fin Prime")]
[GridOrder(90)]
public IPS.Properties.Double phifinprime
{
    get
    {
        return _phifinprime;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Exchanger Mean Beam Length")]
[GridOrder(91)]
public IPS.Properties.Double sHT
{
    get
    {
        return _sHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Characteristic Length")]
[GridOrder(92)]
public IPS.Properties.Double L
{
    get
    {
        return _L;
    }
}

```

```

}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Surface Emissivity")]
[GridOrder(93)]
public IPS.Properties.Double episurface
{
    get
    {
        return _episurface;
    }
}

//Thermal and fluid properties

//Tube bank pressure loss

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Pressure Loss in The Tube Bank")]
[GridOrder(94)]
public IPS.Properties.Double Deltap
{
    get
    {
        return _Deltap;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Bank Friction Coefficient")]
[GridOrder(95)]
public IPS.Properties.Double zeta
{
    get
    {
        return _zeta;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Drag Coefficient in Laminar Flow")]
[GridOrder(96)]
public IPS.Properties.Double zetal
{
    get

```

```

        {
            return _zetat;
        }
    }

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f a laminar")]
[GridOrder(97)]
public IPS.Properties.Double fal
{
    get
    {
        return _fal;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f a Turbulent")]
[GridOrder(98)]
public IPS.Properties.Double fat
{
    get
    {
        return _fat;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flow Direction Checking Criteria")]
[GridOrder(99)]
public IPS.Properties.Double checkpoint1
{
    get
    {
        return _checkpoint1;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Drag Coefficient in Turbulent Flow")]
[GridOrder(100)]
public IPS.Properties.Double zetat
{
    get

```

```

        {
            return _zetat;
        }
    }

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f z n, laminar")]
[GridOrder(101)]
public IPS.Properties.Double fznI
{
    get
    {
        return _fznI;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f z, turbulent")]
[GridOrder(102)]
public IPS.Properties.Double fzt
{
    get
    {
        return _fzt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("f n turbulent")]
[GridOrder(103)]
public IPS.Properties.Double fnt
{
    get
    {
        return _fnt;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Zeta o")]
[GridOrder(104)]
public IPS.Properties.Double zetao
{
    get
    {

```

```

        return _zetao;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("F")]
[GridOrder(105)]
public IPS.Properties.Double F
{
    get
    {
        return _F;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tube Bundle Overall Drag Coefficient")]
[GridOrder(106)]
public IPS.Properties.Double kforward
{
    get
    {
        return _kforward;
    }
}

//Heat transfer coefficient

//Gas Radiation

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Heat Transfer Coefficient due to Gas Radiation")]
[GridOrder(107)]
public IPS.Properties.Double hgasrad
{
    get
    {
        return _hgasrad;
    }
}

//Convective heat transfer by Gnlinskis method in VDI

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("fa")]

```

```

[GridOrder(108)]
public IPS.Properties.Double fa
{
    get
    {
        return _fa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under laminar regime")]
[GridOrder(109)]
public IPS.Properties.Double Nu0lam
{
    get
    {
        return _Nu0lam;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row under turbulent regime")]
[GridOrder(110)]
public IPS.Properties.Double Nu0turb
{
    get
    {
        return _Nu0turb;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu per row")]
[GridOrder(111)]
public IPS.Properties.Double Nu0row
{
    get
    {
        return _Nu0row;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Nu over entire tube bank")]
[GridOrder(112)]

```

```

public IPS.Properties.Double Nu0
{
    get
    {
        return _Nu0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Temperature Gradient Correction Factor")]
[GridOrder(113)]
public IPS.Properties.Double K
{
    get
    {
        return _K;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Nassault Number")]
[GridOrder(114)]
public IPS.Properties.Double Nu
{
    get
    {
        return _Nu;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Heat Transfer Coefficient from Convection")]
[GridOrder(115)]
public IPS.Properties.Double hconv
{
    get
    {
        return _hconv;
    }
}

//Others

//Intemidient steps

```

```
//Tri-atomic gases

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kg")]
[GridOrder(116)]
public IPS.Properties.Double kg
{
    get
    {
        return _kg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("kfa")]
[GridOrder(117)]
public IPS.Properties.Double kfa
{
    get
    {
        return _kfa;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("krad")]
[GridOrder(118)]
public IPS.Properties.Double krad
{
    get
    {
        return _krad;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("epsilonfg")]
[GridOrder(119)]
public IPS.Properties.Double epsilonfg
{
    get
    {
        return _epsilonfg;
    }
}
```



```

}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Specific Volume of H2O")]
[GridOrder(120)]
public IPS.Properties.Double gammaH2O
{
    get
    {
        return _gammaH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-atomic Specific Volume")]
[GridOrder(121)]
public IPS.Properties.Double gamman
{
    get
    {
        return _gamman;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("H2O Partial Pressure")]
[GridOrder(122)]
public IPS.Properties.Double ppH2O
{
    get
    {
        return _ppH2O;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tri-Atomic Partial Pressure")]
[GridOrder(123)]
public IPS.Properties.Double ppn
{
    get
    {
        return _ppn;
    }
}

```

```

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Constant Pressure Specific Heat at Mean Temperature")]
[GridOrder(124)]
public IPS.Properties.Double cp
{
    get
    {
        return _cp;
    }
}

//High ash loading volume flow rate

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue gas Density in The Heat Exchanger")]
[GridOrder(125)]
public IPS.Properties.Double rhofginHT
{
    get
    {
        return _rhofginHT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Mass Flow Rate")]
[GridOrder(126)]
public IPS.Properties.Double mfg
{
    get
    {
        return _mfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Volume Flow Rate")]
[GridOrder(127)]
public IPS.Properties.Double Vfg
{
    get
    {
        return _Vfg;
    }
}

```

```

}

//High ash loading effect coefficient check

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("A in High Ash Loading Calculation")]
[GridOrder(128)]
public IPS.Properties.Double coeffA
{
    get
    {
        return _coeffA;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q abs in High Ash Loading Calculation")]
[GridOrder(129)]
public IPS.Properties.Double Qfracabs
{
    get
    {
        return _Qfracabs;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Q scatter in High Ash Loading Calculation")]
[GridOrder(130)]
public IPS.Properties.Double Qfracscatter
{
    get
    {
        return _Qfracscatter;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Gamma in High Ash Loading Calculation")]
[GridOrder(131)]
public IPS.Properties.Double coeffgamma
{
    get
    {
        return _coeffgamma;
    }
}

```

```

    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Beta in High Ash Loading Calculation")]
[GridOrder(132)]
public IPS.Properties.Double coeffbeta
{
    get
    {
        return _coeffbeta;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Lp in High Ash Loading Calculation")]
[GridOrder(133)]
public IPS.Properties.Double Lp
{
    get
    {
        return _Lp;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Epsilon G in High Ash Loading Calculation")]
[GridOrder(134)]
public IPS.Properties.Double epiG
{
    get
    {
        return _epiG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("a in High Ash Loading Calculation")]
[GridOrder(135)]
public IPS.Properties.List<IPS.Properties.Double> coeffa
{
    get
    {
        return _coeffa;
    }
}

```

```

}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b0 in High Ash Loading Calculation")]
[GridOrder(136)]
public IPS.Properties.List<IPS.Properties.Double> coeffb0
{
    get
    {
        return _coeffb0;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("b1 in High Ash Loading Calculation")]
[GridOrder(137)]
public IPS.Properties.List<IPS.Properties.Double> coeffb1
{
    get
    {
        return _coeffb1;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KG in High Ash Loading Calculation")]
[GridOrder(138)]
public IPS.Properties.List<IPS.Properties.Double> coeffkG
{
    get
    {
        return _coeffkG;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("KeG in High Ash Loading Calculation")]
[GridOrder(139)]
public IPS.Properties.Double KeG
{
    get
    {
        return _KeG;
    }
}

```

```

}

[PropertyUsage(Property.RESULT)]
[DisplayName("Phi G+P in High Ash Loading Calculation")]
[GridOrder(140)]
public IPS.Properties.Double phiGandP
{
    get
    {
        return _phiGandP;
    }
}

[PropertyUsage(Property.RESULT)]
[DisplayName("Final Equivalent Fouling Layer Thickness")]
[GridOrder(141)]
public IPS.Properties.Double tfoulingfinal
{
    get
    {
        return _tfoulingfinal;
    }
}

[PropertyUsage(Property.RESULT)]
[DisplayName("Air Interstitial Velocity")]
[GridOrder(142)]
public IPS.Properties.Double ufgmax
{
    get
    {
        return _uairmax;
    }
}

[PropertyUsage(Property.RESULT)]
[DisplayName("Air Heat Transfer Velocity")]
[GridOrder(143)]
public IPS.Properties.Double ufgheat
{
    get
    {
        return _uairheat;
    }
}

```

}

```

//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
//css_ref IPS.Units.dll;
//css_ref IPS.Utilities.CSharp.dll;
//css_ref Flownex.dll;

using System;
using IPS.Properties;
using IPS.Scripting;
using Flownex.Components;
using Flownex.DataReferences.Materials;

//Reference
//Heat Atlas VDI G7 [1]
//Heat Atlas VDI L1 [2]
//MEC4117Z Textbook [3]

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs

    //Geometrical

    //Rotory Air Heater Length
    IPS.Properties.Double _LAH;

    //Rotor speed
    IPS.Properties.Double _nrotor;

    //Fluid Flow Areas
    IPS.Properties.Double _Air;
    IPS.Properties.Double _Afg;

    //Total Heat Transfer Surface Area
    IPS.Properties.Double _AHtotal;

    //Compartment Geometry
    IPS.Properties.Double _ncompartment;
    IPS.Properties.Double _Dhcompartment;
    IPS.Properties.Double _DAHouter;
    IPS.Properties.Double _DAHinner;

```



```
IPS.Properties.Double _Agap;
IPS.Properties.Double _LcN;
IPS.Properties.Double _Lc0;

//Others
IPS.Properties.Double _Kgap;
IPS.Properties.Double _CorrelationType;
IPS.Properties.Double _Airleakageoptions;
IPS.Properties.Double _Airleakagepercentage;

//Non-uniform sweeping
IPS.Properties.Double _zetasweep;

//Area fraction
IPS.Properties.Double _fracxfg;
IPS.Properties.Double _fracxair;

//Thermal or fluid properties

//Air Inlet
IPS.Properties.Double _mairin;
IPS.Properties.DataReferenceProperty _FluidDataReference;
IPS.Properties.List<IPS.Properties.Double> _MassFractions;

//Temperatures
IPS.Properties.Double _Tairin;
IPS.Properties.Double _Tairout;
IPS.Properties.Double _TairAvg;

//Pressures
IPS.Properties.Double _pfg;
IPS.Properties.Double _pair;

//Outputs

//Geometry(pre-simulation data process)

//Characteristic Length
IPS.Properties.Double _Pfg;
IPS.Properties.Double _Pair;

//Compartment Geometry
IPS.Properties.Double _VcompartmentAvg;

//Velocities
```

```

IPS.Properties.Double _UNT;
IPS.Properties.Double _UOT;
IPS.Properties.Double _fdischarge;

//Air Leakage
IPS.Properties.Double _mairleakgap;
IPS.Properties.Double _mairleakcarryover;
IPS.Properties.Double _mairleakall;

//Thermal or fluid properties
IPS.Properties.Double _rhoairin;
IPS.Properties.Double _rhoairout;
IPS.Properties.Double _hairin;
IPS.Properties.Double _hairout;
IPS.Properties.Double _haireffective;
IPS.Properties.Double _Taireffective;

//Heat transfer coefficients

//Coefficient c
IPS.Properties.Double _Coeffc;

//Overall Coefficient
IPS.Properties.Double _Coeffall;

//Heat Transfer Area
IPS.Properties.Double _AHfg;
IPS.Properties.Double _AHair;

double UnoverLcN;
double U0overLc0;
double Squaresum;

//do pre simulation initialisation here
public override void Initialise()
{

}

//do post simulation cleanup here
public override void Cleanup()
{
}

```

```

//script main execution function - called every cycle
public override void Execute(double Time)
{
    //Process Air data

    List<double> fractions = new List<double>();

    foreach(IPS.Properties.Double val in _MassFractions)
    {
        fractions.Add(val.Value);
    }
    double[] MassFractions = fractions.ToArray();

    MixedFluids fluid = (MixedFluids)_FluidDataReference.Value;

    //Calculate Geometry Outputs

    _Pfg.Value = (4 * _Afg) / _Dhcompartment.Value;

    _Pair.Value = (4 * _Air) / _Dhcompartment.Value;

    _VcompartmentAvg.Value = _LAH.Value * Math.PI *
Math.Pow((_Dhcompartment.Value / 2), 2);

    _AHfg.Value = _fracxfg.Value * _AHTotal.Value;

    _AHair.Value = _fracxair.Value * _AHTotal.Value;

    //Enthalpy and Density of Air at Inlet and Exit

    _rhoairin.Value = fluid.GetDensity(_Tairin.Value, _pair.Value, MassFractions,
Parent.Identifier.Value);

    _rhoairout.Value = fluid.GetDensity(_Tairout.Value, _pair.Value, MassFractions,
Parent.Identifier.Value);

    _hairin.Value = fluid.GetEnthalpy(_Tairin.Value, _pair.Value, MassFractions,
Parent.Identifier.Value);

    _hairout.Value = fluid.GetEnthalpy(_Tairout.Value, _pair.Value, MassFractions,
Parent.Identifier.Value);

    //Air Leakage Calculation

```

```

if(_Airleakageoptions.Value == 0) //Default Case, Air Leakage Accounted by
Percentage
{
    _mairleakall.Value = (_Airleakagepercentage.Value / 100) *
    _mairin.Value;

    _Taireffective.Value = _TairAvg.Value;
}

else //Calculate the Air Leakage
{
    _mairleakgap.Value = _Kgap.Value * _Agap.Value * Math.Pow((2 *
Math.Abs(_pair.Value - _pfg.Value) * _rhoairin.Value), 0.5);

    _UNT.Value = _nrotor.Value * (Math.PI / 30) * (_DAHouter / 2);

    _U0T.Value = _nrotor.Value * (Math.PI / 30) * (_DAHinner / 2);

    UnoverLcN = _UNT.Value / _LcN.Value;

    U0overLc0 = _U0T.Value / _Lc0.Value;

    Squaresum = (UnoverLcN * UnoverLcN) + (U0overLc0 * U0overLc0);

    _fdischarge.Value = Math.Pow(0.5 * Squaresum, 0.5);

    _mairleakcarryover.Value = _rhoairout.Value *
_VcompartmentAvg.Value * _ncompartment.Value * _fdischarge.Value;

    _mairleakall.Value = _mairleakgap.Value + _mairleakcarryover.Value;

    _haireffective.Value = (_mairleakgap.Value * _hairin.Value +
    _mairleakcarryover.Value * _hairout.Value) / _mairleakall.Value;

    _Taireffective.Value = fluid.GetTemperatureFromPH(_pair.Value,
    _haireffective.Value, MassFractions, Parent.Identifier.Value);

}

//Assign coefficient C based on Rotational Speed
if(_nrotor.Value == 0.5)
{
    _Coeffc.Value = 0.85;
}

```

```

else if(_nrotor.Value == 1)
{
    _Coeffc.Value = 0.97;
}

else if(_nrotor.Value == 1.5)
{
    _Coeffc.Value = 1;
}

//Calculate overall coefficient applied to the convective compoents
if(_CorrelationType.Value == 0)
{
    _Coeffall.Value = _zetasweep.Value * _Coeffc.Value;
}

else if(_CorrelationType.Value == 1)
{
    _Coeffall.Value = 0.023 * _zetasweep.Value * _Coeffc.Value;
}

}

//any processing you want to do before steady state
public override void ExecuteBeforeSteadyState()
{
    Execute(0.0);
}

//any processing you want to do while solving steady state
public override void ExecuteSteadyState()
{
    Execute(0.0);
}

//any processing you want to do after steady state
public override void ExecuteAfterSteadyState()
{
    Execute(0.0);
}

//add code here that you want to execute
//when an associated button is pushed
public override void HandleObjectEvent(object sender)
{

```

```
}
```

```
//constructor initialises parameters
```

```
public Script()
```

```
{
```

```
    //Initialise
```

```
    //Inputs
```

```
    //Geometrical
```

```
    //Rotory Air Heater Length
```

```
    _LAH = new IPS.Properties.Double();
```

```
    //Rotor speed
```

```
    _nrotor = new IPS.Properties.Double();
```

```
    //Fluid Flow Areas
```

```
    _Air = new IPS.Properties.Double();
```

```
    _Afg = new IPS.Properties.Double();
```

```
    //Total Heat Transfer Surface Area
```

```
    _AHtotal = new IPS.Properties.Double();
```

```
    //Compartment Geometry
```

```
    _ncompartment = new IPS.Properties.Double();
```

```
    _Dhcompartment = new IPS.Properties.Double();
```

```
    _DAHouter = new IPS.Properties.Double();
```

```
    _DAHinner = new IPS.Properties.Double();
```

```
    _Agap = new IPS.Properties.Double();
```

```
    _LcN = new IPS.Properties.Double();
```

```
    _LcO = new IPS.Properties.Double();
```

```
    //Others
```

```
    _Kgap = new IPS.Properties.Double();
```

```
    _CorrelationType = new IPS.Properties.Double();
```

```
    _Airleakageoptions = new IPS.Properties.Double();
```

```
    _Airleakagepercentage = new IPS.Properties.Double();
```

```
    //Non-uniform sweeping
```

```
    _zetasweep = new IPS.Properties.Double();
```

```
    //Area fraction
```

```

_fracxfg = new IPS.Properties.Double();
_fracxair = new IPS.Properties.Double();

//Thermal or fluid properties

//Air Inlet
_mairin = new IPS.Properties.Double();
_FluidDataReference = new IPS.Properties.DataReferenceProperty(null);
_FluidDataReference.Types = new Type[] {typeof(MixedFluids)};
_MassFractions = new IPS.Properties.List<IPS.Properties.Double>();

//Temperatures
_Tairin = new IPS.Properties.Double();
_Tairout = new IPS.Properties.Double();
_TairAvg = new IPS.Properties.Double();

//Pressures
_pfg = new IPS.Properties.Double();
_pair = new IPS.Properties.Double();

//Outputs

//Geometry(pre-simulation data process)

//Characteristic Length
_Pfg = new IPS.Properties.Double();
_Pair = new IPS.Properties.Double();

//Compartment Geometry
_VcompartmentAvg = new IPS.Properties.Double();

//Velocities
_UNT = new IPS.Properties.Double();
_UOT = new IPS.Properties.Double();
_fdischarge = new IPS.Properties.Double();

//Air Leakage
_mairleakgap = new IPS.Properties.Double();
_mairleakcarryover = new IPS.Properties.Double();
_mairleakall = new IPS.Properties.Double();

//Thermal or fluid properties
_rhoairin = new IPS.Properties.Double();
_rhoairout = new IPS.Properties.Double();

```

```

_hairin = new IPS.Properties.Double();
_hairout = new IPS.Properties.Double();
_haireffective = new IPS.Properties.Double();
_Taireffective = new IPS.Properties.Double();

//Heat transfer coefficients

//Coefficient c
_Coeffc = new IPS.Properties.Double();

//Overall Coefficient
_Coeffall = new IPS.Properties.Double();

//Heat Transfer Area
_AHfg = new IPS.Properties.Double();
_AHair = new IPS.Properties.Double();

//Unit
//Inputs

//Geometrical

//Rotory Air Heater Length
_LAH.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Fluid Flow Areas
_Aair.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_Afg.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

//Total Heat Transfer Surface Area
_AHtotal.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

//Compartment Hydraulic Diameter
_Dhcompartment.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_DAHouter.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_DAHinner.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Agap.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_LcN.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_LcO.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Thermal or fluid properties

//Air Inlet

```



```

_mairin.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Temperatures
_Tairin.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_Tairout.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;
_TairAvg.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;

//Pressures
_pfg.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;
_pair.DisplayUnitGroup = IPS.Units.UnitGroup.Pressure_Stress;

//Outputs

//Geometry(pre-simulation data process)

//Characteristic Length
_Pfg.DisplayUnitGroup = IPS.Units.UnitGroup.Length;
_Pair.DisplayUnitGroup = IPS.Units.UnitGroup.Length;

//Compartment Geometry
_VcompartmentAvg.DisplayUnitGroup = IPS.Units.UnitGroup.Volume;

//Velocities
_UNT.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;
_UOT.DisplayUnitGroup = IPS.Units.UnitGroup.Velocity;

//Air Leakage
_mairleakgap.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mairleakcarryover.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;
_mairleakall.DisplayUnitGroup = IPS.Units.UnitGroup.Flow_Rate_Mass;

//Thermal or fluid properties
_rhoairin.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_rhoairout.DisplayUnitGroup = IPS.Units.UnitGroup.Density;
_hairin.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_hairout.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_haireffective.DisplayUnitGroup = IPS.Units.UnitGroup.Specific_Energy;
_Taireffective.DisplayUnitGroup = IPS.Units.UnitGroup.Temperature;

//Heat Transfer Area
_AHfg.DisplayUnitGroup = IPS.Units.UnitGroup.Area;
_AHair.DisplayUnitGroup = IPS.Units.UnitGroup.Area;

```

```
//Initial values
//Inputs

//Geometrical

//Rotory Air Heater Length
_LAH.Value = 1e-6;

//Rotor speed
_nrotor.Value = 1e-6;

//Fluid Flow Areas
_Aair.Value = 1e-6;
_Afg.Value = 1e-6;

//Total Heat Transfer Surface Area
_AHtotal.Value = 1e-6;

//Compartment Hydraulic Diameter
_ncompartment.Value = 0;
_Dhcompartment.Value = 1e-6;
_DAHouter.Value = 1e-6;
_DAHinner.Value = 1e-6;
_Agap.Value = 1e-6;
_LcN.Value = 1e-6;
_LcO.Value = 1e-6;

//Others
_Kgap.Value = 0.4;
_CorrelationType.Value = 0;
_Airleakageoptions.Value = 0; //Default Value
_Airleakagepercentage.Value = 1e-6;

//Non-uniform sweeping
_zetasweep.Value = 1e-6;

//Area fraction
_fracxfg.Value = 1;
_fracxair.Value = 1;

//Thermal or fluid properties
```

```
//Air Inlet
_mairin.Value = 1e-6;

//Temperatures
_Tairin.Value = 273.15;
_Tairout.Value = 273.15;
_TairAvg.Value = 273.15;

//Pressures
_pfg.Value = 1000;
_pair.Value = 1000;

//Outputs

//Geometry(pre-simulation data process)

//Characteristic Length
_Pfg.Value = 1;
_Pair.Value = 1;

//Compartment Geometry
_VcompartmentAvg.Value = 10;

//Velocities
_UNT.Value = 0.315;
_UOT.Value = 0.08;
_fdischarge.Value = 1e-6;

//Air Leakage
_mairleakgap.Value = 8;
_mairleakcarryover.Value = 10;
_mairleakall.Value = 18;

//Thermal or fluid properties
_rhoairin.Value = 1;
_rhoairout.Value = 1;
_hairin.Value = 1e-6;
_hairout.Value = 1e-6;
_haireffective.Value = 1e-6;
_Taireffective.Value = 273.15;

//Heat transfer coefficients

//Coefficient c
```

```

        _Coeffc.Value = 0.8;

        //Overall Coefficient
        _Coeffall.Value = 0.6;

        //Heat Transfer Area
        _AHfg.Value = 70;
        _AHair.Value = 57;

    }

    //property declarations to make
    //parameters visible to outside world
    //Inputs

    //Geometrical

    //Rotory Air Heater Length
    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Air Heater Length")]
    [GridOrder(11)]
    public IPS.Properties.Double LAH
    {
        get
        {
            return _LAH;
        }
    }

    //Rotor speed
    [PropertyUsage(UseProperty.DYNAMIC)]
    [DisplayName("Air Heater Rotational Velocity in rpm")]
    [GridOrder(12)]
    public IPS.Properties.Double nrotor
    {
        get
        {
            return _nrotor;
        }
    }

    //Fluid Flow Areas

    [PropertyUsage(UseProperty.DYNAMIC)]

```

```

[DisplayName("Air Flow Area")]
[GridOrder(13)]
public IPS.Properties.Double Aair
{
    get
    {
        return _Aair;
    }
}

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Flue Gas Flow Area")]
[GridOrder(14)]
public IPS.Properties.Double Afg
{
    get
    {
        return _Afg;
    }
}

//Total Heat Transfer Surface Area

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Total Heating Area")]
[GridOrder(15)]
public IPS.Properties.Double AHtotal
{
    get
    {
        return _AHtotal;
    }
}

//Compartment Geometry

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Number of Compartment per Radial Plane")]
[GridOrder(16)]
public IPS.Properties.Double ncompartment
{
    get
    {
        return _ncompartment;
    }
}

```

```

}

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Compartment Hydraulic Diameter")]
[GridOrder(17)]
public IPS.Properties.Double Dhcompartment
{
    get
    {
        return _Dhcompartment;
    }
}

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Compartment Furthest From the Rotor Centre")]
[GridOrder(18)]
public IPS.Properties.Double DAHouter
{
    get
    {
        return _DAHouter;
    }
}

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Compartment Closest From the Rotor Centre")]
[GridOrder(19)]
public IPS.Properties.Double DAHinner
{
    get
    {
        return _DAHinner;
    }
}

[PropertyUsage(Property.DYNAMIC)]
[DisplayName("Air Leak Gap Area")]
[GridOrder(20)]
public IPS.Properties.Double Agap
{
    get
    {
        return _Agap;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Width of The Furthest Compartment")]
[GridOrder(21)]
public IPS.Properties.Double LcN
{
    get
    {
        return _LcN;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Width of The Closest Compartment")]
[GridOrder(22)]
public IPS.Properties.Double Lc0
{
    get
    {
        return _Lc0;
    }
}

```

//Others

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Discharge Coefficient (K)")]
[GridOrder(23)]
public IPS.Properties.Double Kgap
{
    get
    {
        return _Kgap;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Nu Correlation (0: Gnielinski, 1: Dittus-Boelter)")]
[GridOrder(24)]
public IPS.Properties.Double CorrelationType
{
    get
    {

```

```

        return _CorrelationType;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Leakage Option (0: By Total In Feed Air Mass Percentage / Default, 1:
Calculate The Air Leakage Flow Rate)")]
[GridOrder(25)]
public IPS.Properties.Double Airleakageoptions
{
    get
    {
        return _Airleakageoptions;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Leakage by Total In Feed Air Mass(in %)")]
[GridOrder(26)]
public IPS.Properties.Double Airleakagepercentage
{
    get
    {
        return _Airleakagepercentage;
    }
}

//Non-uniform sweeping

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Non-Uniformaty Sweeping Coefficient")]
[GridOrder(27)]
public IPS.Properties.Double zetasweep
{
    get
    {
        return _zetasweep;
    }
}

//Area fraction

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Flue Gas Side Heat Tranfer Area Fraction")]
[GridOrder(28)]

```



```

public IPS.Properties.Double fracxfg
{
    get
    {
        return _fracxfg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Side Heat Tranfer Area Fraction")]
[GridOrder(29)]
public IPS.Properties.Double fracxair
{
    get
    {
        return _fracxair;
    }
}

//Thermal or fluid properties

//Air Inlet

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Mass Flow Rate")]
[GridOrder(30)]
public IPS.Properties.Double mairin
{
    get
    {
        return _mairin;
    }
}

[PropertyUsage(UseProperty.STATIC)]
[DisplayName("Fluid")]
[GridOrder(31)]
public IPS.Properties.DataReferenceProperty FluidDataReference
{
    get
    {
        return _FluidDataReference;
    }
}

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Mass fractions")]
[GridOrder(32)]
public IPS.Properties.List<IPS.Properties.Double> MassFractions
{
    get
    {
        return _MassFractions;
    }
}

//Temperatures
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Inlet Temperature")]
[GridOrder(33)]
public IPS.Properties.Double Tairin
{
    get
    {
        return _Tairin;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Air Exit Temperature")]
[GridOrder(34)]
public IPS.Properties.Double Tairout
{
    get
    {
        return _Tairout;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Average Air Temperature")]
[GridOrder(35)]
public IPS.Properties.Double TairAvg
{
    get
    {
        return _TairAvg;
    }
}

```

```

//Pressures
[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Average Flue Gas Pressure")]
[GridOrder(36)]
public IPS.Properties.Double pfg
{
    get
    {
        return _pfg;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Average Air Pressure")]
[GridOrder(37)]
public IPS.Properties.Double pair
{
    get
    {
        return _pair;
    }
}

//Outputs

//Geometry(pre-simulation data process)

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Side Characteristic Length")]
[GridOrder(38)]
public IPS.Properties.Double Pfg
{
    get
    {
        return _Pfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Side Characteristic Length")]
[GridOrder(39)]
public IPS.Properties.Double Pair
{
    get
    {

```

```

        return _Pair;
    }
}

//Heat transfer coefficients

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Average Compartment Volume")]
[GridOrder(40)]
public IPS.Properties.Double VcompartmentAvg
{
    get
    {
        return _VcompartmentAvg;
    }
}

//Velocities

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tangential Velocity of the Furthest Compartment")]
[GridOrder(41)]
public IPS.Properties.Double UNT
{
    get
    {
        return _UNT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Tangential Velocity of the Closest Compartment")]
[GridOrder(42)]
public IPS.Properties.Double UOT
{
    get
    {
        return _UOT;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Average Compartment Discharge Frequency (f_discharge)")]
[GridOrder(43)]

```

```

public IPS.Properties.Double fdischarge
{
    get
    {
        return _fdischarge;
    }
}

//Air Leakage

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Leakage Through Gap")]
[GridOrder(44)]
public IPS.Properties.Double mairleakgap
{
    get
    {
        return _mairleakgap;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Carryover Air Leakage")]
[GridOrder(45)]
public IPS.Properties.Double mairleakcarryover
{
    get
    {
        return _mairleakcarryover;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Overall Air Leakage")]
[GridOrder(46)]
public IPS.Properties.Double mairleakall
{
    get
    {
        return _mairleakall;
    }
}

//Thermal or fluid properties

```

```

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("In Feed Air Density")]
[GridOrder(47)]
public IPS.Properties.Double rhoairin
{
    get
    {
        return _rhoairin;
    }
}

[PropertyUsage(UseProperty.DYNAMIC)]
[DisplayName("Exit Air Density")]
[GridOrder(48)]
public IPS.Properties.Double rhoairout
{
    get
    {
        return _rhoairout;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Enthalpy of the In Feed Air")]
[GridOrder(49)]
public IPS.Properties.Double hairin
{
    get
    {
        return _hairin;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Enthalpy of the Exit Air")]
[GridOrder(50)]
public IPS.Properties.Double hairout
{
    get
    {
        return _hairout;
    }
}

[PropertyUsage(UseProperty.RESULT)]

```

```

[DisplayName("Effective Air Enthalpy")]
[GridOrder(51)]
public IPS.Properties.Double haireffective
{
    get
    {
        return _haireffective;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Effective Air Temperature")]
[GridOrder(52)]
public IPS.Properties.Double Taireffective
{
    get
    {
        return _Taireffective;
    }
}

//Heat transfer coefficients

//Coefficient c

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Rotating Speed Coefficient C")]
[GridOrder(53)]
public IPS.Properties.Double Coeffc
{
    get
    {
        return _Coeffc;
    }
}

//Overall Coefficient

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Overall Coefficient Applied to the Heat Tranfer Component")]
[GridOrder(54)]
public IPS.Properties.Double Coeffall
{
    get
    {

```

```
        return _Coeffall;
    }
}

//Heat Transfer Area

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Flue Gas Heat Transfer Area")]
[GridOrder(55)]
public IPS.Properties.Double AHfg
{
    get
    {
        return _AHfg;
    }
}

[PropertyUsage(UseProperty.RESULT)]
[DisplayName("Air Heat Transfer Area")]
[GridOrder(56)]
public IPS.Properties.Double AHair
{
    get
    {
        return _AHair;
    }
}
}
```