1D Heat Flux code explained

Jack Taylor

April 15, 2020

Version Log	Change log	Date
1	First Issue of Document	15/04/2020

1 Introduction

The purpose of the one-dimensional (1D) heat flux code is to model the coolant flow over high heat flux Plasma Facing Components (PFC), in this case the diverter. The code is separated into multiple files where the parameters, functions and their mathematical foundations within each are explained in this document. The purpose of the code is to produce a steady state result of flow conditions within the diverter.

1.1 Applicable documents

2 List of Symbols and associated code ID

Table 2 details the parameters used in the code and their physical definitions.

Parameter	code ID	Units	Comments
P_{input}	input_pressure	Pa	Coolant inlet pressure
T_{input}	input_temperature	K	Coolant inlet temperature
u_{input}	v_r_input	m/s	Coolant inlet velocity, varies from 5-50 in steps of 5
$\dot{Q}_{input}$	input_power	W/m^2	Input power per unit area
$\dot{m}$	q_r	kg/s	Mass flow rate, varies from 5-50 in steps of 5
X	section_0	m	X coordinate of the copper panel
Y	section_1	m	Y coordinate of the copper panel
th_{copper}	h	m	Thickness of the copper panel
ε	epsi	m	Coolant channel surface roughness

Table 1: User input parameters

Parameter	code ID	Units	Comments
ρ_{input}	input_rho	kg/m^3	Input density, calculated using CoolProp, input_pressure and input_temperature
ρ	D	kg/m^3	Density downstream of input, calculated using CoolProp, input_pressure and input_temperature
h	h_1	W/Km^2	Film coefficient in W/Km^2
h_{film}	htc_0	W/K	Film coefficient in W/K
h_{linear}	h_f	m	Linear head loss in pipe
u	v_s, v_secc	m/s	Velocity of bulk flow downstream of inlet at a given point, list of velocities
$T_{coolant}$	T_ref	K	Temperature of coolant
T_{copper}	T_metal	m	Temperature of copper plate
$P_{coolant}$	P_secc	Pa	Pressure of coolant
h_{total}	hf_tot	K	Total head loss of pipe, sum of h_f
$\dot{Q}$	input_power	W	Input power
R_e	Re	-	Reynolds number at each discretised point
P_r	Pr	-	Prandtl number at each discretised point
N_u	Nu	-	Nusselt number at each discretised point
A_g	Ag	m^2	Area of annulus channel at a given discretised point
A_{PFC}	A, Alist	m^2	Surface area of copper panel on PFC side between two discretised points
t_g	tg	m	Coolant gap thickness
D_H	dh	m	Hydraulic Diameter
l	deltaz	m	Considered length
L	-	m	Characteristic length, not used explicitly in the code as, dependant on its use, is defined via another length parameter
t_g	tg	m	Coolant gap thickness
f	fricc	-	Moody friction coefficient
c_p	C	-	Specific heat capacity, calculated using CoolProp, Pressure and Temperature
ν	V/D	-	Kinematic Viscosity, calculated using CoolProp, Pressure, Temperature and density
μ	V	-	Dynamic Viscosity, calculated using CoolProp, Pressure and Temperature

Table 2: Output parameters

3 Code breakdown

The code is separated into 5 files containing multiple classes, where *CHICA 1.3.py* dictates the order that classes are called and implemented. The current flow of classes being called within each file is shown in Figure 1. As can be seen in Figure 1, *Setup.py* and *Runner.py* receive inputs and pass outputs to *CHICA 1.3.py*, calling *Non_Dimensional.py* and *Coolant_Geometry.py* to perform calculations. These additional files have been separated to allow easy replacement/additional tools to be implemented, dependant on the flow conditions being considered, without disrupting the main flow of the code.

The classes within *Setup.py* and *Runner.py* are split between those that output the "first guess" computation of the flow conditions at each point and those that output the converged results. Sections 3.1 to 3.5 detail the contents of each file and its function in the 1D heat flux code.

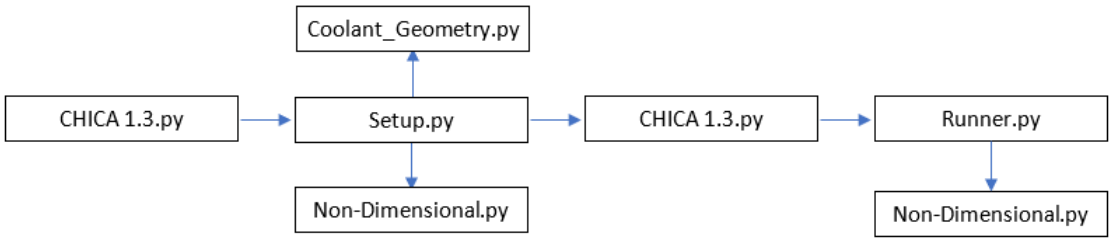

Figure 1: Code Flow

3.1 CHICA 1.3

As explained above, *CHICA 1.3.py* defines the order that classes are called as well as requiring user inputs for initial flow conditions and considered geometry, these detailed in Table 1. The name and order that classes are called in *CHICA 1.3.py* is detailed below.

(1) *setup.initial_setup*

(2) *solver.initial*

(3) *setup.looper_setup*

(4) *solver.looper*

This allows the "first guess" to be computed via *initial*, followed by cycling over the computation via *looper* until converged and a steady state result achieved.

3.1.1 CoolProp

CoolProp is a python module that calculates flow properties dependant on user defined flow parameters and the fluid to be evaluated. Where CoolProp is required, temperature and pressure are

used as inputs to compute flow conditions. Below is the general format of the tool.

CoolProp.PropSI([Parameter to be computed], [Input parameter$_1$], [Parameter value$_1$], [Input parameter$_2$], [Parameter value$_2$], [Fluid to be evaluated])

3.2 Coolant_Geometry.py

The *coolant_geometry* class contains two functions, *a_r_poloidal* and *a_r_toroidal*, where each handles purely poloidal and purely toroidal flow respectively. The functions require the physical geometry of the PFC and outputs the geometry parameters of the coolant channels. The code expects discretised geometry parameters and uses these as the basis for discretising the coolant channel.

3.2.1 Theory

In this section the calculation for each parameter used in *coolant_geometry* is explained.

Coolant gap thickness, t_g
The coolant gap thickness is computed at the diverter inlet and kept constant along the length of the channel.

$$\dot{V} = \frac{\dot{m}}{\rho} = uA_g$$

$$A_g = \pi(X_{inlet} + t_g)^2 - \pi X_{inlet}^2$$

$$t_g^2 + 2X_{inlet}t_g - \frac{\dot{m}}{\rho u \pi} = 0$$

Solve for t_g.

Coolant channel area, A_g
The area of the channel gap is computed at each node.

$$A_{g,i} = \pi(X_i + t_g)^2 - \pi X_i^2$$

Hydraulic diameter, D_H
The hydraulic diameter allows for handling non perfectly circular geometry in standard piping equations. The current version of the code treats the coolant geometry as an annulus, so the hydraulic diameter is defined as below.

$$D_H = 2 \times t_g$$

PFC surface area, A_{PFC}
Computed using the area of a Frustum, removing the area of the top and bottom circles.

$$A_{PFC,i} = \pi(X_i + X_{i+1})\sqrt{(X_i - X_{i+1})^2 + (Y_i - Y_{i+1})^2}$$

Input power, $\dot{Q}$
The units of power input, $\dot{Q}_{input}$, are [W/m^2]. To derive the power input in Watts the following equation is applied, where the term $\frac{X}{X+th_{copper}}$ scales the power dependant on the thickness of the copper plate.

$$\dot{Q} = \dot{Q}_{input} A_{PFC} \frac{X}{X + th_{copper}}$$

3.3 Non_Dimensional.py

The *non_dimensional* class contains two functions, *nusselt* and *film_coeff*. *nusselt* returns; N_u, R_e, P_r, v, f, h_{linear} and requires the flow conditions and channel geometry at the point being considered, this being; $\dot{m}$, $T_{coolant}$, A_g, P_{input}, D_H, ε and l. *film_coeff* returns the heat transfer coefficient h and requires; D_H, P_{input}, $T_{coolant}$ and N_u as inputs.

3.3.1 Theory

In this section the calculation for each parameter used in *non_dimensional* is explained.

Bulk velocity, u

$$u = \frac{\dot{m}}{\rho A_g}$$

Reynolds number, R_e
Reynolds number considers the ratio of inertial to viscous forces in a fluid. For this case, the considered length L is defined by the Hydraulic diameter D_H.

$$R_e = \frac{\rho u L}{\nu}$$

Prandtl number, P_r
Prandtl number considers the ratio of momentum diffusivity to thermal diffusivity in a fluid.

$$P_r = \frac{\mu c_p}{k}$$

Moody friction factor, f
Moody friction factor is calculated using the Colebrook equation below.

$$\frac{1}{\sqrt{f}} = -2log_{10}\left(\frac{\epsilon/D}{3.7} + \frac{2.51}{R_e\sqrt{f}}\right)$$

Where f is computed for turbulent flow in rough pipes by the Goudar-Sonnad approximation below. This approximation is valid for all Reynolds numbers.

$$\frac{1}{\sqrt{f}} = a\left(\ln\frac{d}{r} + D_{CFA}\right)$$

$$a = \frac{2}{\ln 10}$$

$$d = \frac{\ln 10 R_e}{5.2}$$

$$b = \frac{\epsilon / D_H}{3.7}$$

$$s = bd + \ln d$$

$$r = s^{\frac{s}{s+1}}$$

$$m = bd + \ln \frac{d}{r}$$

$$p = \ln \frac{r}{m}$$

$$D_{LA} = p \frac{m}{m+1}$$

$$D_{CFA} = D_{LA} \left(1 + \frac{p/2}{(m+1)^2 + (p/3)(2m-1)} \right)$$

Nusselt number, N_u

Nusselt number considers the ratio of convective heat transfer to conductive heat transfer at the boundary of a fluid.

$$N_u = \frac{hL}{k}$$

Where the approximation for turbulent fluid flows in pipes, the Gnielinski correlation, is shown below and is used to compute the Nusselt number.

$$N_u = \frac{(f/8) R_e P_r}{1.07 + 12.7\sqrt{f/8}(P_r^{2/3} - 1)}$$

The validity criteria for this approximation are:

$$0.5 < P_r < 2000$$

$$3000 < R_e < 5 \times 10^6$$

3.4 Setup.py

The class *setup* contains two functions, *initial_setup* and *looper_setup*, where these functions initialise the input and output sets for the solvers in *Runner.py* as well as calculating the initial flow conditions and geometry parameters through implementing *non_dimensional* and *coolant_geometry*. No calculations are performed in the body of this code outside those called from other classes.

3.5 Runner.py

The *runner* class contains two functions *initial* and *looper*, where *initial* performs an initial calculation of the temperature at each node based on constant pressure, updating the pressure term retrospectively based on the temperature at each discretised point. The *looper* function takes this pressure output and recalculates the flow conditions at each discretised point, again updating the output pressure. When the pressure delta at each point between loops is $< 10^{-8}$ the result is considered converged and so the therefore a steady state result.

The calculation steps are as follows:

initial

(1) Compute $T_{coolant,i+1}$ at each node, where $c_{p,i}$ is calculated using *CoolProp*, $T_{coolant,i}$ and $P_{coolant,i}$:

$$T_{coolant,i+1} = T_{coolant,i} + \frac{\dot{Q}_i}{c_{p,i}\dot{m}}$$

(2) Implement *non_dimensional* to return at each node $[N_u, R_e, P_r, h_{linear}, u]_{i+1}$
(3) Compute at each node $T_{metal,i+1}$

$$T_{metal,i+1} = T_{coolant,i+1} + \frac{\dot{Q}_i}{h_{i+1}A_{PFC,i}}$$

(4) Implement *film_coeff* and compute h_film at each node

$$h_{film,i} = h_i A_{PFC,i}$$

(5) Compute $P_{coolant}$ at each node, where ρ_i is calculated using *CoolProp*, $T_{coolant,i}$ and $P_{coolant,i}$:

$$P_{coolant,i+1} = P_{coolant,i} + \frac{\rho_i}{2}(u_i^2 - u_{i+1}^2) + \rho_i g(y_i - y_{i+1}) - \rho_i g h_{1,i+1}$$

This provides the first computation of the flow conditions at each node.

looper

looper repeats the steps of *initial*, taking the pressure computed in step (5) as input for step (1), as opposed to the constant pressure input of *initial*. The final pressure is then compared to this input pressure, checking against the *Error* pass criteria below. *Error* $> 10^{-8}$ will restart the computation with the most recent pressure output as input for the new computation. For *Error* $< 10^{-8}$, the final pressure distribution is considered converged and so the result steady state.

$$Error = \sqrt{\left(1 - \frac{P_{in}}{P_{out}}\right)^2}$$

4 Future updates

The code is currently in very early stages and requires further updates to be applicable for industrial use. The following points have been highlighted as areas for improvement.

- *initial* & *looper*: Reduce these into a single function, does not need to be separated for a "first guess" followed by separate cycling function.
- Geometry: The current assumption of an annulus is not useful, needs to be updated to consider more appropriate geometry i.e. discrete piping, micro channels, multiple discrete panels as opposed to a continuous swept geometry.
- Time stepping: Include time stepping to capture transient effects.
- CFD and FEA: Allow for CFD and FEA inputs
- Add in references to this document