Linear Stability Analysis of a Supercritical Water Loop driven by Natural Convection

G.B. Koren, August 2010

PNR 131-2010-007

Nomenclature

Roman letters

A	Cross-sectional area	m^2
D_H	Hydraulic diameter	m
f	Darcy-Weisbach friction factor	-
g	Gravitational acceleration	$\mathrm{m/s^2}$
G	Mass flux	$\mathrm{kg/\left(m^2\,s\right)}$
h	Specific enthalpy	J/kg
K	Local pressure drop coefficient	-
L_C	Length of core	m
$\dot{m}$	Mass flow	kg/s
N	Number of grid points	-
N_dh	Dimensionless enthalpy jump	-
N_sub	Subcooling number	-
p	Pressure	Pa
P	Heated perimeter	m
$\dot{q}$	Heat flow	W
q'	Linear heat flow	W/m
q''	Heat flux	$\mathrm{W/m^2}$
q'''	Volumetric heat flow	$\mathrm{W/m^3}$
Re	Reynolds number	-
S	Wetted perimeter	m
t	Time	s
T	Temperature	°C
u	Velocity	m/s
z	Spatial coordinate	m

Greek letters

ϵ	Roughness	m
θ	Angle between flow and horizontal	rad
λ	Eigenvalue	1/s
μ	Dynamic viscosity	Pa s
ρ	Density	$\mathrm{kg/m^3}$
ω	Angular frequency	rad/s

Subscripts

x_i	Value at point i
x_{in}	Value at core inlet
x_{out}	Value at core outlet
x_{pc}	Value at pseudo-critical conditions

Other

$\overline{x}$	Steady-state variable
x'	Perturbation variable

Abbreviations

HPLWR	High Performance Light Water Reactor
NIST	National Institute of Standards and Technology
EOS	Equation of State
SCWR	Supercritical Water Reactor

Abstract

The HPLWR (High Performance Light Water Reactor) is the European version of the SCWR (Supercritical Water Reactor) and is one of the Generation IV concepts that have enhanced safety, improved efficiency and less nuclear waste compared to current nuclear reactors.

A possible way to enhance the safety is by using natural convection as the driving mechanism for the coolant flow. Natural convection is especially interesting because of the large differences in density occurring under supercritical conditions. This is safer because of its independence of mechanical systems (i.e. pumps, which are used in forced convection loops).

The goal of this project was to investigate the linear stability of a one-dimensional, simplified version of the HPLWR (without the power-density feedback, but with constant power) around the steady-state solution for a range of operational conditions. A code was written based on one-dimensional equations for mass, energy and momentum transport. The code successfully predicted the steady-state behaviour of a system. The results were benchmarked with data from literature. The code works for natural convection loops as well as forced convection systems. The stability plots do not agree with literature and are therefore considered to be incorrect.

Table of Contents

Chapter 1

Introduction

1.1 HPLWR

The HPLWR (High Performance Light Water Reactor) is the European version of the SCWR (Supercritical Water Reactor) and is one of the Generation IV concepts that have enhanced safety, improved efficiency and less nuclear waste compared to current nuclear reactors (Generation IV International Forum, 2002). The working fluid of the HPLWR is water. The system pressure of 25 MPa is above the critical pressure for water. This means that above a certain temperature or specific enthalpy, the water will be a supercritical fluid (see figure 1.1).

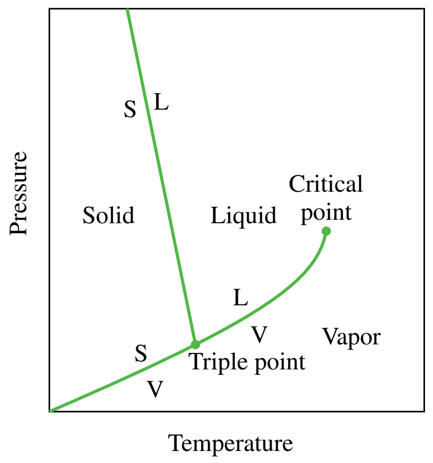


Figure 1.1: Schematic diagram of the phase plot for a fluid that expends upon freezing (such as water). Figure is adopted from Moran and Shapiro (2006).

The thermodynamic efficiency of the HPLWR is estimated to be 44% (Squarer et al., 2003). This high thermal efficiency can be obtained because of the high temperature at the outlet of the core. In the most recent HPLWR design the system has a three-pass core to protect the materials from peaks in temperature (Hofmeister et al., 2007). A schematic view of the three-pass core is given in figure 1.2.

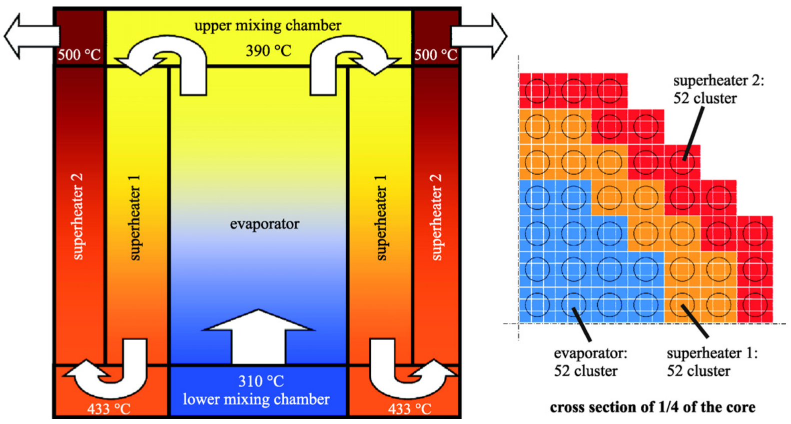


Figure 1.2: Schematic diagram of the arrangement of the three-pass core of the HPLWR design. Figure has been adopted from Fischer et al. (2009).

1.2 Natural convection loop

A possible way to enhance the safety is by making use of natural convection. This is especially interesting because of the large differences in densities occurring under supercritical conditions (see figure 1.3). The difference in density in the riser section and downcomer section (see figure 1.4 for these sections) of the loop is the driving force for the motion as will be explained below.

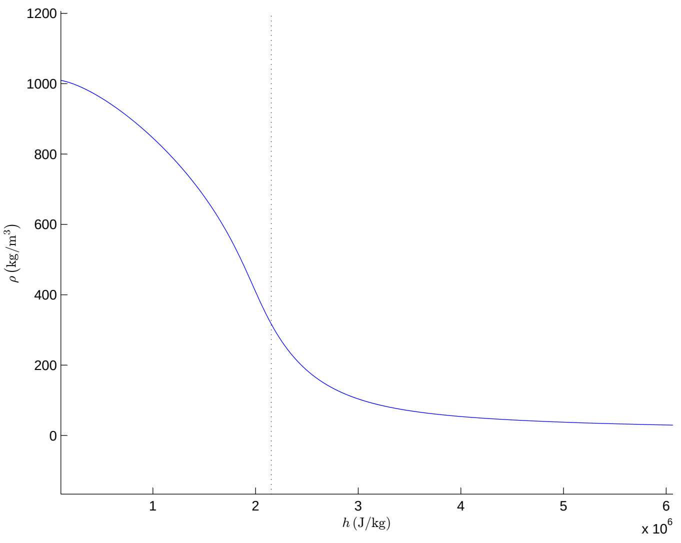


Figure 1.3: Plot of ρ as a function of h for water. The pseudo critical enthalpy $h_{\mathrm{pc}} = 2.1529 \cdot 10^6$ J/kg is indicated by the vertical dotted line. The data is taken from NIST

The principle of natural convection in the SCWR works as follows (see figure 1.4). The enthalpy at the entrance of the core, h_{in}, is assumed to be lower than h_{pc}, the pseudo critical enthalpy. When the core is active, the enthalpy in the core will increase (due to heat addition). The consequence for the density can be seen in figure 1.3: it decreases. When the hot fluid passes the cooler, its enthalpy will decrease again and therefore the density increases. This means that the mass of the fluid in the downcomer is larger than the mass in the riser and therefore a mass flow is induced. Because of this continuous mechanism, no pump is needed for the circulation of the fluid.

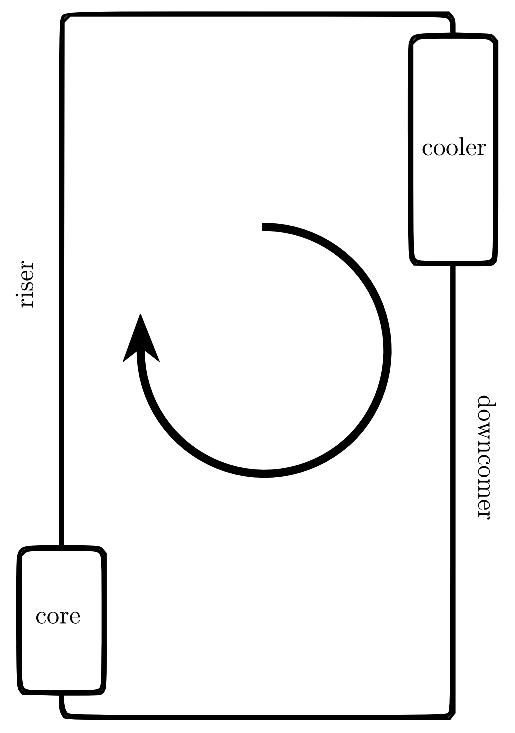


Figure 1.4: Schematic diagram of natural convection loop. Direction of flow is clockwise as indicated by the arrow. Riser, downcomer, core and cooler are indicated.

The reason that a natural convection loop is considered to be safer than a forced convection loop, is that it is not dependent on a mechanical system as is the case for forced convection loops. In a forced convection loop a pump is needed to maintain the flow. If the pump breaks down, serious accidents can happen because of the over-heating of the core material, which could ultimately lead to a meltdown. The natural convection loop is driven by the heating in the core, therefore the fluid will always be in motion if the core is active.

When the inflow to the core decreases T_{out} is higher than normal and therefore the density in the core and riser section decreases (see figure 1.3). The lower density in the riser means a larger difference in density between the downcomer and the riser. This results in more flow into the core. This makes that T_{out} now decreases and the denisty in the core and riser section increases. This mechanism makes the natural convection loop inherently safe.

1.3 Stability

A system is said to be stable when it returns to its state after a perturbation from that state. This depends on whether the initial perturbation grows or decays in time. This overall behaviour of a system depends on the relative strength of the underlying mechanisms and time scales that are associated with these mechanisms

Instabilities can be classified into two categories being static and dynamic. Static instabilities occur when, for instance, for a certain pressure drop different mass flows can exist. This instability is also called a Ledinegg instability. Static instabilities can be predicted with steady-state equations. Whether a system has dynamic instabilities can be solved by using a time-dependent approach.

Van Bragt (1998) makes the following distinction of instabilities. Type I instabilities are low-frequency and due to the gravitational pressure drop over the riser section. Type II instabilities are of higher-frequency and are caused by frictional pressure losses. These instabilities are

A stability plot is a plot that shows regions of stable operation and unstable operation. Usually there are dimensionless numbers on the axes of such plots. The dimensionless numbers used in this thesis are defined in section 3.2.1. The Type I an Type II instabilities can be observed in a stability plot as the two noses of the stability line. The upper left nose corresponds to a Type I instability and the lower nose corresponds to a Type II instability.

1.4 Outline

The goal of this thesis is to implement a code that can perform stability calculations for a supercritical water loop driven by natural convection. In the next chapter, the equations needed for the steady-state and stability calculations are derived. Chapter 3 focuses on the eigenvalues and their interpretation. After this, a chapter follows which is devoted to the implementation process. After this chapter, the output of the code will be benchmarked and conclusions can be drawn about the correctness of the code. Recommendations for future research are given afterwards. The nomenclature and bibliography have been placed respectively at the inside front and inside back cover of this thesis for the convenience of the reader.

Chapter 2

Mathematical Foundation

This chapter starts with the governing equations, first in their continuous form and finally in discrete-space, continuous-time form. The boundary conditions are described in the second section. The third section is devoted to the steady-state equations. And finally, the fourth section presents the linear perturbed equations. Together, this forms the mathematical foundation for calculating the stability of a system.

2.1 Governing equations

In this section the mass, energy and momentum equation are presented in their one-dimensional, constant cross-section form. They can also be found (except for the slightly different notation) in, for instance, Ambrosini and Sharabi (2008). First the mass equation will be presented, secondly the energy equation and finally the momentum eqation. This corresponds to the order in which they are solved to obtain the steady-state solution later on.

After presenting the continuous equations, they will be discretized. The reason for doing this in such an early stage is that this assures better consistency later on (i.e. between the steady-state solution and the perturbed form of the equations, discussed in sections 2.3 and 2.4 respectively). An upwind discretisation was selected for this task, since it is numerically stable and does not allow the zigzag patterns (which can be described as the one-dimensional analog of the two-dimensional checkerboard patterns) that are permitted by for instance the central difference approximation (Patankar, 1980).

The following recipe has been used to discretize the equations (x denotes a variable that appears in the equations and that varies as a function of z. This last criterion excludes A, P and D_H which are constants and t which is independent of z)

- derivatives with respect to space are discretized using an upwind scheme $\frac{\partial x}{\partial z} \to \frac{x_i - x_{i-1}}{\Delta z}$. This is indeed upwind, since the flow is defined to be in positive z direction
- partial derivatives with respect to time are turned into total derivatives while the function that it acts on gets a subscript denoting its position $\frac{\partial x}{\partial t} \to \frac{\mathrm{d}x_i}{\mathrm{d}t}$
- other functions of z appearing in the equations also get a subscript denoting their position $x \to x_i$

The discrete form is the form that will be used and referred to on numerous occasions in this report.

2.1.1 Mass equation

The one-dimensional, constant cross-section mass equation is

$$\frac{\partial \rho}{\partial t} + \frac{\partial G}{\partial z} = 0. \tag{2.1}$$

Applying the procedure stated above yields the discrete form of the mass equation

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} + \frac{G_i - G_{i-1}}{\Delta z} = 0. \tag{2.2}$$

2.1.2 Energy equation

The one-dimensional, constant cross-section energy equation is

$$\frac{\partial \rho h}{\partial t} + \frac{\partial G h}{\partial z} = q'' \left(\frac{P}{A}\right), \tag{2.3}$$

where the source term is on the right side of the equals sign. In this form, the potential energy and kinetic energy have been disregarded. In discrete form this is

$$\frac{\mathrm{d}\rho_i h_i}{\mathrm{d}t} + \frac{G_i h_i - G_{i-1} h_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right). \tag{2.4}$$

2.1.3 Momentum equation

The one-dimensional, constant cross-section momentum equation is

$$\frac{\partial G}{\partial t} + \frac{\partial}{\partial z}\left(\frac{G^2}{\rho}\right) + \frac{\partial p}{\partial z} = -g\rho \sin\theta - \frac{1}{2}\frac{G^2}{\rho}\left[\frac{f}{D_{\mathrm{H}}} + \sum_j K_j \delta\left(z - z_j\right)\right], \tag{2.5}$$

where θ is the counterclockwise angle between the direction of flow and the horizontal. D_{H} is the hydraulic diameter of the channel, defined as $D_{\mathrm{H}} = \frac{4A}{S}$. Again the source term has been placed on the right side of the equals sign. The source term containing the sine function ($-g\rho \sin\theta$) can be either positive or negative, and therefore respectively a source or sink of momentum. The friction term is always negative and therefore a sink term for momentum (as is expected by the nature of friction).

Although it might seem contradictory to neglect the potential energy from the energy equation but to keep the gravity term in the momentum equation, this actually makes perfect sense. The gravity term is essential in the momentum equation because it is the driving force for the motion, it is the mechanism that is responsable for flow in the system. The gravitational potential energy however, is negligable compared to the other terms in the energy equation (Todreas and Kazimi, 1990).

In order to model the Darcy-Weisbach friction factor f, the Haaland approximation is used (Haaland, 1983)

$$f = \left[-1.8 \log\left(\left(\frac{\epsilon/D_{\mathrm{H}}}{3.7}\right)^{1.11} + \frac{6.9}{\mathrm{Re}}\right)\right]^{-2}, \qquad \text{for} \quad \mathrm{Re} > 3\cdot 10^4. \tag{2.6}$$

It must be noted that the Haaland approximation is valid only in the specified range for the Reynolds number which is defined as $\mathrm{Re} = \frac{GD_\mathrm{H}}{\mu}$. The expected values for Re are larger than $3 \cdot 10^4$.

The discrete form of the momentum equation is

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{G_i^2}{\rho_i} - \frac{G_{i-1}^2}{\rho_{i-1}}\right) + \frac{p_i - p_{i-1}}{\Delta z} = -g\rho_i \sin\theta_i - \frac{1}{2}\frac{G_i^2}{\rho_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.7}$$

The Haaland equation in discrete form, for the friction factor f_i, does not change except that it now includes the discrete Reynolds number, which is $\mathrm{Re}_i = \frac{G_i D_\mathrm{H}}{\mu_i}$.

2.1.4 Equation of state

The three independent variables that describe the system are the mass flux G, the specific enthalpy h and pressure p. The density is not an independent variable since, in this work, it is completely defined by the enthalpy: $\rho = \rho(h)$ or, in its discrete form, $\rho_i = \rho(h_i)$. As was already mentioned in the introduction, the data used for estimating ρ as a function of h is taken from NIST. The equation of state (EOS) is in general a function of enthalpy and pressure, but in the regime of interest ($p = 25\,\mathrm{MPa}$) the pressure happens to have almost no influence on ρ. This simplifies the analysis from one that has 4 equations (mass, energy, momentum and EOS) and 4 independent variables (G, h, p and ρ), into an anlysis of 3 equations (mass, energy, momentum) and 3 independent variables (G, h and p).

2.2 Boundary conditions

In the previous section the governing equations were presented. There is still an important ingredient missing however: the boundary conditions[1]. The boundary conditions represent the geometry and operational conditions, while the equations represent the governing physics. The boundary conditions can be chosen to model a natural convection loop such as described in chapter 1, but it is also possible to model, for instance, a vertical pipe with an applied pressure drop (e.g. by a pump). In the following, the appropriate parameters for the natural convection loop will be desribed. It should be kept in mind, however, that the boundary condiontions are not restricted to loop geometries.

The operational parameters are the variables that can be set without changing the setup itself. These are the input enthalpy h_in, the heat flux of the core q'' and the applied pressure drop Δp. Obviously, in the case of a natural convection loop Δp equals zero.

The natural convection loop has a core and riser section and a cooler and downcomer section (see figure 2.1). They are represented by the variable θ in the momentum equation (equation 2.5). For the riser section, θ equals $\frac{1}{2}\pi$, and because $\sin\left(\frac{1}{2}\pi\right) = 1$ this corresponds to a decelerating force as can be seen in equation 2.5. For the downcomer, $\theta = \frac{3}{2}\pi$, and therefore $\sin\theta$ equals -1 corresponding to an accelerating force. For a horizontal tube θ equals π or 2π, for both of which the sine function is equal to zero and therefore the gravity does not affect that part of the system.

[1]Initial conditions are not specified, because they are unnecessary since the system of equations will not be solved in time.

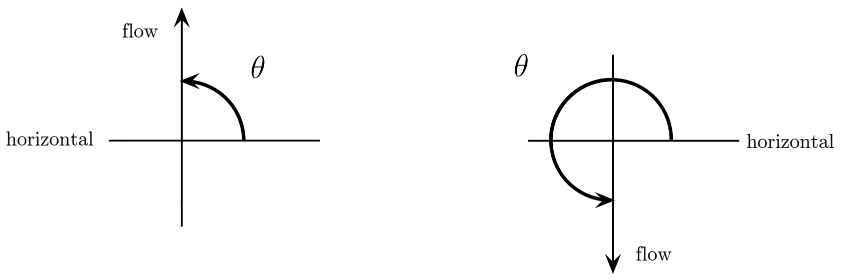


Figure 2.1: Schematic drawings of the direction of flow and associated angle θ for riser section (left) and downcomer (right). In the riser, flow is upwards and, therefore, $\theta = \frac{1}{2}\pi$. The riser is a sink for momentum. In the downcomer section, flow is downwards and, therefore, $\theta = \frac{3}{2}\pi$. The downcomer is a source of momentum.

Last but not least, periodic boundary conditions apply to all variables in space. In mathematical form

$$x_i = x_{i+N}, \qquad \forall\ i. \tag{2.8}$$

Here x represents any function of z, and z is considered to be defined from $-\infty$ to $+\infty$. If the interval on which z is defined is restricted to $[0, N]$ then the aforementioned equality (equation 2.8) reduces to $x_0 = x_N$. Note that this is just a matter of definition and the physical interpretation is the same.

2.3 Steady-state equations

In this section the equations for calculating the steady-state solution are derived from the discrete governing equations.

Below, an overview is given of the procedure that has been applied (x_i denotes one of the variables G_i, h_i, p_i and ρ_i).

- derivatives with respect to time are set equal to zero $\frac{\mathrm{d}x_i}{\mathrm{d}t} \to 0$
- for consistency throughout the report the steady-state variables receive an overbar $x_i \to \overline{x}_i$

The resulting set of equations can be used to obtain the steady-state solution when initial conditions are given. This will be further explained in section 4.5.

2.3.1 Mass equation

The time-dependent mass equation (equation 2.2) is repeated for convenience

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} + \frac{G_i - G_{i-1}}{\Delta z} = 0. \tag{2.9}$$

Applying the procedure mentioned above gives the very simple equation

$$\overline{G}_i = \overline{G}_{i-1}, \quad \text{or equivalently} \quad \overline{G} = \text{constant}. \tag{2.10}$$

In the remainder of this thesis the notation without the index is favored because this is consistent with the fact that other constants in the equations are also given without index.

2.3.2 Energy equation

The time-dependent energy equation (equation 2.4) is repeated for convenience

$$\frac{\mathrm{d}\rho_i h_i}{\mathrm{d}t} + \frac{G_i h_i - G_{i-1} h_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right). \tag{2.11}$$

Using the rules stated above and using the steady-state mass equation (equation 2.10) yields

$$\overline{G}\frac{\overline{h}_i - \overline{h}_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right). \tag{2.12}$$

Solving for $\overline{h}_i$ yields

$$\overline{h}_i = q_i'' \left(\frac{P}{A}\right) \frac{\Delta z}{\overline{G}} + \overline{h}_{i-1}. \tag{2.13}$$

2.3.3 Momentum equation

The time-dependent momentum equation (equation 2.7) is repeated here for convenience

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{G_i^2}{\rho_i} - \frac{G_{i-1}^2}{\rho_{i-1}}\right) + \frac{p_i - p_{i-1}}{\Delta z} = -g\rho_i \sin\theta_i - \frac{1}{2}\frac{G_i^2}{\rho_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.14}$$

Again applying the procedure and the steady-state mass equation (equation 2.10) yields

$$\frac{\overline{G}^2}{\Delta z}\left(\frac{1}{\overline{\rho}_i} - \frac{1}{\overline{\rho}_{i-1}}\right) + \frac{\overline{p}_i - \overline{p}_{i-1}}{\Delta z} = -g\overline{\rho}_i \sin\theta_i - \frac{1}{2}\frac{\overline{G}^2}{\overline{\rho}_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.15}$$

This can be rewritten into an expression for $\overline{p}_i$

$$\overline{p}_i = \overline{G}^2\left(\frac{1}{\overline{\rho}_{i-1}} - \frac{1}{\overline{\rho}_i}\right) - \Delta z\left(g\overline{\rho}_i \sin\theta_i + \frac{1}{2}\frac{\overline{G}^2}{\overline{\rho}_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]\right) + \overline{p}_{i-1}. \tag{2.16}$$

2.4 Perturbed equations

In this section the time-dependent equations will be decomposed in a way that results in three linear equations of perturbations. These equations will be used for the linear stability analysis, as described in chapter 3.

The three variables that describe the system (G, h and p) have been decomposed as follows

$$G_i \to \overline{G} + G_i', \tag{2.17}$$

$$h_i \to \overline{h}_i + h_i', \tag{2.18}$$

$$p_i \to \overline{p}_i + p_i'. \tag{2.19}$$

In this notation the variables with a bar are the steady-state variables (consistent with the notation in the previous section) and the variables with a prime are perturbations[2]. The steady-state variables are constant over time and can vary in space, the perturbations can vary both in time and in space. Note that the steady-state mass equation (equation 2.10) is already present in equation 2.17 because $\overline{G}$ has no index.

The variables that are a function of either G, h or p have also been decomposed

$$\rho_i \to \overline{\rho}_i + \rho_i'. \tag{2.20}$$

So in total, four variables have been decomposed. Although the friction factor f depends, via the Reynolds number, on both G and h (through μ, as can be seen in its definition on page 7), f has not been decomposed because a variation in G or h results in only a minor variation in f.

The expression $\rho' \approx \overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}} h'$ has been used to eliminate ρ' in favor of h'. $\frac{\mathrm{d}\rho}{\mathrm{d}h}$ is also a function of h and the bar on the function means that it is to be evaluated for the steady-state value of the enthalpy: $\overline{h}$

The decomposition is combined with a linearization in perturbations. The following simple rules desribe this process (x and χ denote one of the four variables that are being decomposed.)

- derivative with respect to time of a steady-state variable is equal to zero $\frac{\mathrm{d}\overline{x}}{\mathrm{d}t} \rightarrow 0$
- second order terms have been neglected $x'\chi' \rightarrow 0$

The desired outcome of this analysis is a set of equations that is linear in perturbations of the three system variables (G', h' and p').

2.4.1 Mass equation

The time-dependent mass equation (equation 2.2) is repeated below for convenience

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} + \frac{G_i - G_{i-1}}{\Delta z} = 0. \tag{2.21}$$

The steady-state mass equation is already present in equation 2.17, and is therefore not needed here in its explicit form. Applying the above described procedure on the time-dependent mass equation yields

$$\frac{\mathrm{d}\rho_i'}{\mathrm{d}t} + \frac{G_i' - G_{i-1}'}{\Delta z} = 0, \tag{2.22}$$

where ρ_i' can be eliminated by using $\rho_i' \approx \overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\Big|_i h_i'$. Performing this elimination and changing the order of some of the terms yields

$$\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\Bigg|_i \frac{\mathrm{d}h_i'}{\mathrm{d}t} = \frac{1}{\Delta z} G_{i-1}' - \frac{1}{\Delta z} G_i', \tag{2.23}$$

which is in the desired form.

2.4.2 Energy equation

The time-dependent energy equation (equation 2.4) and the steady-state energy equation in its unsolved form (equation 2.12) are repeated below for reference.

$$\frac{\mathrm{d}\rho_i h_i}{\mathrm{d}t} + \frac{G_i h_i - G_{i-1} h_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right), \tag{2.24}$$

$$\overline{G}\frac{\overline{h}_i - \overline{h}_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right). \tag{2.25}$$

[2]Note that the discretization scheme for perturbations, x', is **not** strictly upwind since it is allowed for perturbations to be negative. The flow itself ($\overline{x} + x'$), however, stays in positive z direction because the perturbations are small compared to the steady-state variables $\overline{x}$. Therefore the discretization scheme is still upwind.

Decomposing the time-dependent equation yields

$$\frac{\mathrm{d}\left(\overline{\rho}_i h_i' + \rho_i' \overline{h}_i\right)}{\mathrm{d}t} + \frac{\overline{G}\,\overline{h}_i + \overline{G}h_i' + G_i'\overline{h}_i - \overline{G}\,\overline{h}_{i-1} - \overline{G}h_{i-1}' - G_{i-1}'\overline{h}_{i-1}}{\Delta z} = q_i'' \left(\frac{P}{A}\right). \tag{2.26}$$

Subtracting the steady-state equation gives

$$\frac{\mathrm{d}\left(\overline{\rho}_i h_i' + \rho_i' \overline{h}_i\right)}{\mathrm{d}t} + \frac{\overline{G}h_i' + G_i'\overline{h}_i - \overline{G}h_{i-1}' - G_{i-1}'\overline{h}_{i-1}}{\Delta z} = 0. \tag{2.27}$$

This equation can be simplified by writing out the temporal derivative and again applying $\rho_i' \approx \left.\frac{\mathrm{d}\rho}{\mathrm{d}h}\right|_i h_i'$. At the same time the terms have been somewhat manipulated, yielding

$$\left(\overline{\rho}_i + \overline{h}_i \left.\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\right|_i\right) \frac{\mathrm{d}h_i'}{\mathrm{d}t} = \frac{\overline{h}_{i-1}}{\Delta z} G_{i-1}' - \frac{\overline{h}_i}{\Delta z} G_i' + \frac{\overline{G}}{\Delta z} h_{i-1}' - \frac{\overline{G}}{\Delta z} h_i'. \tag{2.28}$$

Again, the resulting equation is in its desired, linear form.

2.4.3 Momentum equation

Below, the time-dependent and steady-state momentum equations are repeated (equations 2.7 and 2.15 respectively)

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{G_i^2}{\rho_i} - \frac{G_{i-1}^2}{\rho_{i-1}}\right) + \frac{p_i - p_{i-1}}{\Delta z} = -g\rho_i \sin\theta_i - \frac{1}{2}\frac{G_i^2}{\rho_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right], \tag{2.29}$$

$$\frac{\overline{G}^2}{\Delta z}\left(\frac{1}{\overline{\rho}_i} - \frac{1}{\overline{\rho}_{i-1}}\right) + \frac{\overline{p}_i - \overline{p}_{i-1}}{\Delta z} = -g\overline{\rho}_i \sin\theta_i - \frac{1}{2}\frac{\overline{G}^2}{\overline{\rho}_i}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.30}$$

Decomposing the time-dependent equation yields

$$\frac{\mathrm{d}G_i'}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{\left(\overline{G} + G_i'\right)^2}{\overline{\rho}_i + \rho_i'} - \frac{\left(\overline{G} + G_{i-1}'\right)^2}{\overline{\rho}_{i-1} + \rho_{i-1}'}\right) + \frac{\overline{p}_i + p_i' - \overline{p}_{i-1} - p_{i-1}'}{\Delta z} =$$
$$- g\left(\overline{\rho}_i - \rho_i'\right)\sin\theta_i - \frac{1}{2}\frac{\left(\overline{G} + G_i'\right)^2}{\overline{\rho}_i + \rho_i'}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.31}$$

Subtracting equation 2.30 yields

$$\frac{\mathrm{d}G_i'}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{\left(\overline{G} + G_i'\right)^2}{\overline{\rho}_i + \rho_i'} - \frac{\overline{G}^2}{\overline{\rho}_i}\right) - \frac{1}{\Delta z}\left(\frac{\left(\overline{G} + G_{i-1}'\right)^2}{\overline{\rho}_{i-1} + \rho_{i-1}'} - \frac{\overline{G}^2}{\overline{\rho}_{i-1}}\right) + \frac{p_i' - p_{i-1}'}{\Delta z} =$$
$$- g\rho_i' \sin\theta_i - \frac{1}{2}\left(\frac{\left(\overline{G} + G_i'\right)^2}{\overline{\rho}_i + \rho_i'} - \frac{\overline{G}^2}{\overline{\rho}_i}\right)\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \tag{2.32}$$

Notice that in this equation the three expressions between parentheses are almost identical. The first and third are actually identical, while the second expression differs only in the position index. The terms can be rewritten by writing out the quadratic expression and using the Taylor expansion $\frac{1}{\overline{x}+x'} \approx \frac{1}{\overline{x}} - \frac{x'}{\overline{x}^2}$

$$\frac{\left(\overline{G} + G_i'\right)^2}{\overline{\rho}_i + \rho_i'} - \frac{\overline{G}^2}{\overline{\rho}_i} \approx \frac{\overline{G}^2 + 2\overline{G}G_i'}{\overline{\rho}_i} - \frac{\overline{G}^2 \rho_i'}{\overline{\rho}_i^2} - \frac{\overline{G}^2}{\overline{\rho}_i} = \frac{2\overline{G}G_i'}{\overline{\rho}_i} - \frac{\overline{G}^2 \rho_i'}{\overline{\rho}_i^2}. \tag{2.33}$$

Inserting this result back into equation 2.32 for the corresponding position index yields

$$\frac{\mathrm{d}G_i'}{\mathrm{d}t} + \frac{1}{\Delta z}\left(\frac{2\overline{G}G_i'}{\overline{\rho}_i} - \frac{\overline{G}^2\rho_i'}{\overline{\rho}_i^2}\right) - \frac{1}{\Delta z}\left(\frac{2\overline{G}G_{i-1}'}{\overline{\rho}_{i-1}} - \frac{\overline{G}^2\rho_{i-1}'}{\overline{\rho}_{i-1}^2}\right) + \frac{p_i' - p_{i-1}'}{\Delta z} = \\ - g\rho_i'\sin\theta_i - \frac{1}{2}\left(\frac{2\overline{G}G_i'}{\overline{\rho}_i} - \frac{\overline{G}^2\rho_i'}{\overline{\rho}_i^2}\right)\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]. \quad (2.34)$$

Finally the three ρ_i' terms and the single ρ_{i-1}' term will be replaced by $\left.\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\right|_i h_i'$ and $\left.\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\right|_{i-1} h_{i-1}'$ respectively. Also, the terms in the equation are somewhat shuffled to obtain a form that is convenient for later purposes.

$$\frac{\mathrm{d}G_i'}{\mathrm{d}t} = \frac{2\overline{G}}{\Delta z\,\overline{\rho}_{i-1}}G_{i-1}' - \left(\frac{2}{\Delta z} + \left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]\right)\frac{\overline{G}}{\overline{\rho}_i}G_i' - \frac{\overline{G}^2}{\Delta z\,\overline{\rho}_{i-1}^2}\left.\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\right|_{i-1} h_{i-1}' \\ + \left(\frac{\overline{G}^2}{\Delta z\,\overline{\rho}_i^2} - g\sin\theta_i + \frac{1}{2}\frac{\overline{G}^2}{\overline{\rho}_i^2}\left[\frac{f_i}{D_\mathrm{H}} + \frac{K_i}{\Delta z}\right]\right)\left.\overline{\frac{\mathrm{d}\rho}{\mathrm{d}h}}\right|_i h_i' + \frac{1}{\Delta z}p_{i-1}' - \frac{1}{\Delta z}p_i'. \quad (2.35)$$

Chapter 3

Linear Stability Analysis

In this chapter the linear equations which were derived in the previous chapter will be used to formulate a criterion for stability in terms of eigenvalues.

3.1 Matrix equation

The set of three linear equations (equations 2.23, 2.28 and 2.35) can be written as the single matrix differential equation

$$\mathsf{A}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathsf{B}\boldsymbol{x}, \tag{3.1}$$

where $\boldsymbol{x}$ is a vector of length $3N$ that contains G'_i, h'_i and p'_i for $0 < i \leq N$. A and B are $3N$-by-$3N$ matrices whose elements can be obtained straightforwardly from equations 2.23, 2.28 and 2.35. Below the structure of matrix A is given. The mass equation corresponds to the first N rows, the energy equation corresponds to rows $N+1$ to $2N$ and the momentum equation fills up the rows from $2N+1$ to $3N$.

$$\mathsf{A} = \begin{bmatrix}
0 & \cdots & 0 & a_{1,N} & & & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & & \ddots & & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & & & a_{N,2N} & 0 & \cdots & 0 \\
 & & & & & & & & \\
0 & \cdots & 0 & a_{N+1,N} & & & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & & \ddots & & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & & & a_{2N,2N} & 0 & \cdots & 0 \\
 & & & & & & & & \\
a_{2N+1,1} & & & 0 & \cdots & 0 & 0 & \cdots & 0 \\
 & \ddots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
 & & a_{3N,N} & 0 & \cdots & 0 & 0 & \cdots & 0
\end{bmatrix}. \tag{3.2}$$

Studying the structure of A reveals that this matrix is singular. Therefore the inverse of A does not exist. This has some implications that will be discussed in the next section.

3.2 Generalized eigenvalue problem

A first step in solving the matrix differential equation (equation 3.1) can be inserting $\boldsymbol{x} = \boldsymbol{\xi} e^{\lambda t}$ as suggested in Boyce and DiPrima (2004). Performing the insertion and calculating the derivative yields $\mathsf{A}\lambda\boldsymbol{\xi}e^{\lambda t} = \mathsf{B}\boldsymbol{\xi}e^{\lambda t}$, which can be simplified by dividing through $e^{\lambda t}$ (which is always allowed since the exponential function never equals zero). This results in the generalized eigenvalue problem

$$\mathsf{A}\lambda\boldsymbol{\xi} = \mathsf{B}\boldsymbol{\xi}. \tag{3.3}$$

So, finding a solution to the set of three differential equations (equations 2.23, 2.28 and 2.35) is equivalent to solving this generalized eigenvalue problem. Now the implications of A being singular are more apparent, since A^{-1} does not exist, the generalized eigenvalue problem can not be converted to the normal eigenvalue problem[1]. The singular generalized eigenvalue problem can be solved by the QZ algorithm (Moler and Stewart, 1973). The implementation of the calculation of the eigenvalue for the singular generalized eigenvalue problem in MATLAB will be described in section 4.6.

In order to have a better understanding of the meaning of the eigenvector $\boldsymbol{\xi}$ and the eigenvalue λ, the following decomposition was made

$$\xi_k = \alpha_k e^{i\beta_k} \qquad \text{where } \alpha_k = |\xi_k| \text{ and } \beta_k = \angle\xi_k \tag{3.4}$$

$$\lambda = a + ib \qquad \text{where } a = \mathrm{Re}\{\lambda\} \text{ and } b = \mathrm{Im}\{\lambda\} \tag{3.5}$$

Now, $\boldsymbol{x}$ can be written as

$$x_k = \alpha_k e^{at} e^{i(\beta_k + bt)} \tag{3.6}$$

Here the physical interpretation of the eigenvector and eigenvalue can be seen. α_k is a non-negative constant that multiplies the remaining part of the expression. β_k is an inital phase shift, it is of no importance here, since the system will not be solved in time. b is the angular frequency of oscillation ω. a is the most important parameter for this thesis: if $a > 0$, then the magnitude is growing in time and therefore the system is unstable, if $a < 0$, the magnitude is decreasing and the system is stable. Note that in this analysis no assumptions have been made on the form of either $\boldsymbol{\xi}$ or λ, which means that the conclusions are valid for any complex $\boldsymbol{\xi}$ or λ that will be found.

Criterion: if the real part of an eigenvalue is positive, the system is unstable, if the real part of the eigenvalue is negative, then the system is stable.

3.2.1 Dimensionless numbers

After presenting the criterion for stability, the dimensionless numbers that facilitate easy comparison of stability plots for different setups and from different sources are next

$$N_{\mathrm{sub}} \equiv \frac{h_{\mathrm{pc}} - h_{\mathrm{in}}}{h_{\mathrm{pc}}} \tag{3.7}$$

$$N_{\mathrm{dh}} \equiv \frac{\dot{q}}{GAh_{\mathrm{pc}}} \tag{3.8}$$

The definition of N_{dh} is the same as N_{PCH} in Marcel et al. (2009) when h_{pc} is taken as the reference enthalpy in the definition of N_{PCH}. The value used for h_{pc} for water is: $h_{\mathrm{pc}} = 2.1529 \cdot 10^6$ J/kg.

[1]Note that dividing equation 3.3 by λ and multiplying by B^{-1} is not allowed as well, because λ can be zero.

Because an enthalpy smaller than 0 does not exist, it is impossible for N_{sub} to be larger than 1, as can be seen from its definition (equation 3.7). The range of N_{dh} is also restricted, because only positive values for $\dot{q}$ and G are considered (see equation 3.8).

Chapter 4

Implementation

In this chapter, the implementation of the theory, as presented in the previous chapters, is described. First, the general thought behind the coding and the resulting structure are discussed. After this, some more specific topics are covered such as the modeling of certain fluid properties using splines and the algorithms that have been used for the steady-state solution and the stability analysis.

4.1 Structure

While working on the code it was decided to switch from several MATLAB scripts containing all the code and all the data for one specific calculation to a more convenient structure: a single script with various interlinking, modular functions for several different calculations. This structure is possible, because a lot of calculations have the same building blocks. For instance: the problem of calculating the pressure drop for a certain mass flow and calculating the mass flow for which the pressure drop is zero, shares a lot of similarities; the latter is coded as an extended version of the algorithm that solves the first problem (see section 4.5).

The benefits are obvious because it means less code and therefore less work. This approach also means that updating of the code is easier, since every block of code appears only once. From a users perspective the single script is also more convenient because in this script any calculation can be started.

A disadvantage is that all the functions have their own separate workspace. If, for instance a variable from one function is needed in another, the function containing the variable needs to be called. As a consequence the lists of input and output arguments tend to be longer than desired. To keep things as orderly as possible it was decided to divide the variables that are likely to be changed from the variables that are kept constant most of the time. The first can be altered easily from the main script and the latter are defined in the functions themselves and can be changed by editing the functions.

4.2 Fluid properties

The values for ρ, μ and $\frac{\mathrm{d}\rho}{\mathrm{d}h}$ are stored in separate functions. These functions all have h as input, where h can be a single value (scalar) or a list of values (vector). The data functions return the value(s) corresponding to the input. If, for instance, the density of water at $h = 1.3 \cdot 10^6$ J/kg is needed it can be obtained, simply by executing `estrho(1.3E6,'water')`.

The functions are cubic splines, based on a set of data points taken from NIST. Data for ρ and μ can be simply read off from the NIST tables and $\frac{\mathrm{d}\rho}{\mathrm{d}h}$ is obtained by a central difference approximation of the data for ρ, performed by Sanders (2009). 39 data points have been used for each of the three functions. Since the variation is largest near the pseudo critical point (see figure 1.3), the data points in this interval were taken closer together. This resulted in smooth functions as can be seen in figure 4.1.

The data has been extrapolated outside its original domain. This is simply done by assigning the value at the lower boundary (call this value $x_{\mathrm{min}} = x(h_{\mathrm{min}})$, where $x(h)$ is the variable that is to be estimated) to all enthalpies smaller than h_{min} and x_{max} (defined in an analog way) to all enthalpies larger than h_{max}. For some variables, the extrapolated data seemed to be a more reasonable approximation to the actual values than for others (in the sense that the resulting plot is more smooth). This can be seen in figure 4.1 where the spline functions for $\frac{\mathrm{d}\rho}{\mathrm{d}h}$ of Freon R-23 and μ of water are plotted.

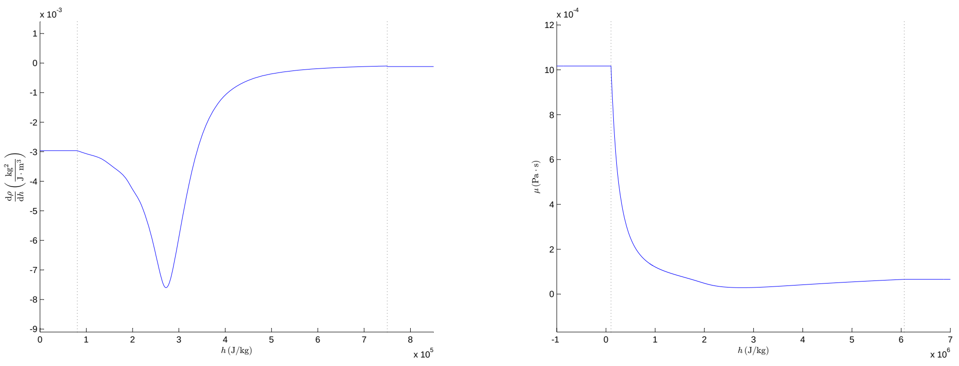


Figure 4.1: Plots of cubic splines used to estimate $\frac{\mathrm{d}\rho}{\mathrm{d}h}$ of Freon R-23 (left) and μ of water (right) as a function of enthalpy h. The boundaries (outside which the extrapolated data was used) are indicated by the vertical, dotted lines.

When extrapolated data is used, a warning message appears in the main screen. When this warning is given it should be kept in mind that for some variables the data will just be fine and for others the approximation will not have the desired accuracy. It should be realized however, that although the data might not be suitable for a final answer of some problem, it can be perfectly used in an iteration process where approximate values will suffice. In section 4.5 an iteration process will be described that can use extrapolated data to converge to points inside the boundary. In such cases the use of extrapolated data does not alter the final accuracy and prevents the code from crashing in its process.

4.3 Variables

All variables and relations were implemented according to the variables and equations in the previous chapters. In some cases there were some minor changes, however. While coding it was observed, for example, that using the volumetric heat flow q''' and the effective gravity g_{eff} (defined below) is more convenient than working with the heat flux G and the angle between the direction of flow and the horizontal θ.

The sine term in the momentum equation (equation 2.16) has not been implemented as such. Instead, the $g \sin\theta$ term is represented by the single term `gEff`. The numerical values that it can take on are in the range of -9.81 to 9.81. The advantage is that there is no need for calculating the sine and that there is no need for defining a θ. At the same time it reduces the number of output terms of functions. The primary reason is making the code better readable and understandable, computational advantages are of minor importance.

4.4 Flexible input

As stated before, it was observed during the coding that the volumetric heat flow q''' is a convenient variable to work with. Another observation was that when comparing data from different sources, it is likely to encounter various ways of specifying the heat flow, being either the power $\dot{q}$, linear power q' , the heat flux q'' or the volumetric heat flow q'''. Those quantities are defined as

$$q' = \frac{\dot{q}}{L_{\text{C}}} \tag{4.1a}$$

$$q'' = \frac{\dot{q}}{P \cdot L_{\text{C}}} \tag{4.1b}$$

$$q''' = \frac{\dot{q}}{A \cdot L_{\text{C}}}. \tag{4.1c}$$

To be consistent in the use of volumetric heat flow throughout the code and facilitate easy comparison with literature, a small part was added to the code, where the `heatinputform` and `heatoutputform` can be set. Possible values are the strings `'power'`, `'linear'`, `'flux'` and `'volumetric'`. The variable `heatinput` is given a numerical value, corresponding to the `heatinputform`. Using rewritten versions of equations 4.1 the code was implemented.

The main reason for writing this piece of code was that, despite the simplicity of the mathematics involved, it was experienced that when continuously deriving the relations and subsequently converting quantities an error is easily made and relatively hard to detect and therefore costing a lot of valuable time.

The same has been done for the mass flow. `flowinputform` and `flowoutputform` can be set to either `'mflow'` or `'mflux'`. The variable `flowinput` is used in an analog way to `heatinput`.

Different models for the friction factor appear in the literature. The Haaland approximation (given in equation 2.6) is used used by Gómez (2008) and Sanders (2009) for instance. A constant value is also often used for f (in fact it is used in Ambrosini and Sharabi (2008) that will be used for benchmarking purposes later on). Another popular alternative is the Blasius relation (Janssen and Warmoeskerken, 2006)

$$f = 0.316\,\text{Re}^{-0.25} \qquad \left(4000 < \text{Re} < 10^5\right) \tag{4.2}$$

To enhance the possibilities of comparing calculated results with data from literature, these friction models have also been incorporated in the code. The variables `fmodel` and `fconstant` were added to the code for this purpose. In the case that `fmodel` equals `'haaland'`, then the variable `fconstant` is the relative roughness $\left(\frac{\epsilon}{D_H}\right.$ in equation 2.6$\left.\right)$. When `fmodel` equals `'constant'`, then `fconstant` is the value of the friction factor, f itself. For `fmodel = 'blasius'` the Blasius relation is used and the value of `fconstant` is ignored.

4.5 Steady-state solution

The steady-state solution can be found as follows. First an initial mass flux is guessed. Next, for a given input enthalpy, the variation of the enthalpy over the loop corresponding to the guessed mass flux can be calculated using equation 2.13. After this, the pressure variation over the loop can be calculated using equation 2.16. Now, the net pressure drop over the loop is compared to the pressure drop that is specified[1]. Depending on the difference between the specified pressure drop and the calculated pressure drop a new mass flux is guessed. This procedure is continued until the calculated pressure drop is equal to the specified pressure drop to within the reach of some specified tolerance. This method is called the shooting method.

An essential ingredient for the shooting method is an algorithm that can come up with improved guesses for the mass flux. Several possibilities where considered for this task and finally the secant method was selected. The secant method can be described as the discrete counterpart of Newton's method. In Newton's method the derivative and the amplitude of a function at some point are used to find a zero-crossing (see figure 4.2). The secant method is used here because there is no closed expression for the derivative of the function (in this case the pressure). In the secant method two points are needed to make a linear approximation of the derivative (therefore there are two initial guesses needed as input for the secant method).

The advantage of the secant method is its simplicity and high speed of convergence. A disadvantage of using the secant method is that there could be situations where the secant method does not converge at all. To prevent the code from going on for ever in such cases, a second stop criterion was implemented. If the number of iteration steps exceeds `maxstep`, then the code stops and a message is given in the main screen. Although preventing the computer from crashing is nice, it does not yield the zero-crossing that was needed.

[1]Natural convection is implemented as a special case of forced convection, namely when the applied pressure drop equals zero. Therefore the code is not limited to use on natural convection loops only. This not only means that the code can be used in more situations, it also enlarges the set of data from literature that can be used for benchmarking.

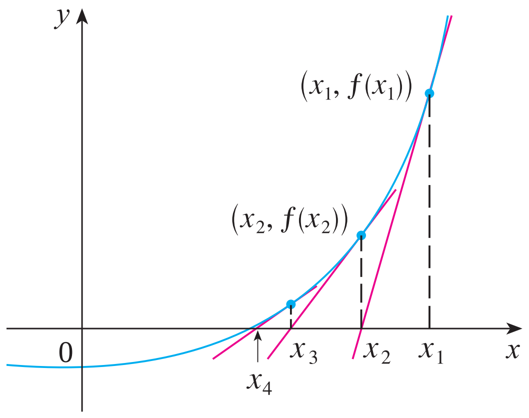


Figure 4.2: Diagram explaining the algorithm of Newton's method. The objective is to approximate the zero-crossing of the function $f(x)$. The initial guess is x_1. Evaluating the function and its derivative for x_1 yields respectively $f(x_1)$ and $f'(x_1)$. A straight line with slope equal to $f'(x_1)$ and that crosses $f(x_1)$ has x_2 as zero-crossing. This way an improved guess can be obtained. This method is repeated until the desired accuracy is obtained (i.e. when $f(x_n) <$ accuracy). Figure adopted from Stewart (2008).

To be able to find this, the parameters `G1` and `G2` (the first two mass fluxes that are used for the secant method algorithm) that are defined in the function `forcedconv` can be tuned. The converging/diverging behaviour of the algorithm depends on those values. A good guess can be obtained by using the function `ledinegg`. This function plots the pressure drop over a system as a function of the mass flow or mass flux. From this graph the location of the zero-crossing can be estimated. Using values close to this zero-crossing for `G1` and `G2` will likely solve the problem.

4.6 Eigenvalues

The generalized eigenvalue problem (equation 3.3) can be solved by the MATLAB function `eig` (MATLAB Function Reference)[2]. It has been observed, however, that not all of the values that MATLAB returns are actually eigenvalues[3].

In order to select the actual eigenvalues from the set of values given by MATLAB, the function `eig` can be instructed to give the corresponding eigenvectors. This is done by executing

```
>> [V,D] = eig(B,A);
```

where `A` and `B` are defined according to matrices A and B in equation (equation 3.3). This produces a diagonal matrix `D` of generalized eigenvalues and a full matrix `V` whose columns are the corresponding eigenvectors such that

$$\texttt{A*V*D = B*V.} \tag{4.3}$$

[2]Before the function can be used, the matriced A and B from equation 3.3 must be defined in MATLAB. In order to check for mistakes in defining the matrices the `spy` function has been used as described in appendix D

[3]The reason why MATLAB returns those values is not clear, but might have something to do with the fact that the `eig` command can also be used for the normal eigenvalue problem. In the normal eigenvalue problem the number of eigenvalues equals the size of the square-matrix. In the singular generalized eigenvalue problem there can be less than $3N$ eigenvalues for two $3N$-by-$3N$ matrices, but for some reason MATLAB always returns $3N$. This strange behaviour was also found for a simple test problem that is shown in appendix A.

The actual eigenvalues can now be filtered from the wrong eigenvalues by checking whether the above-mentioned equality is satisfied. If not, than those values can be disregarded because if they do not satisfy equation 4.3, than they are not eigenvalues. This is a computationally expensive method since the eigenvectors would otherwise not been necessary. Also a lot of memory is wasted in storing the eigenvalues as matrices instead of vectors. Last but not least the three matrix multiplications affect the speed of the code. For this method to work there also needs to be a tolerance defined since the equality in equation 4.3 might not be met completely (due to round-off errors). This introduces an unwanted dependency to this method, since there is no exact criterion, but a choice for the tolerance has to be made.

Because of all the disadvantages given above, an alternative method to filter the eigenvalues has been used instead. It was noted that the incorrect values have a large magnitude (most of them are equal to `Inf` or `-Inf`, this is also demonstrated for the test problem in appendix A). To filter the non-physical eigenvalues, all values with magnitude larger then a certain limit (for instance 10^{10}) have been disregarded. Values smaller than this limit have been considered to be genuine eigenvalues. Calculating and filtering the eigenvalues is now done in two simple lines.

```
>> E = eig(B,A);
>> E = E(abs(E)<1e10);
```

Now, `E` is a vector containing the eigenvalues.

4.7 Stability plot

Up to this point, the script accepted input in several forms, as described in section 4.4. For the stability plot, it is convenient to specify input in terms of the dimensionless number N_{sub} and N_{dh}, since the output will also be defined in N_{sub} and N_{dh}.[4] A minimal and maximal value for both dimensionless numbers, N_{sub} and N_{dh} must be specified. Also the number of points in the respective intervals are needed as input. For all points it will be computed whether it is a stable or unstable point. This will result in a plot with green and red dots for respectively stable and unstable behaviour. Using dots is easier than plotting a curve on the neutral stability boundary (since MATLAB draws the curve in the order in which the points are defined). Also the green and red dots are fairer than an interpolated line (that suggests better accuracy than is actually achieved). If better accuracy (on some part) of the plot is required, N_{sub} and N_{dh}, can be simply read off from the graph, and this can be used as input for another plot.

To be able to say something about the stability for a set of N_{sub} and N_{dh}, these numbers must first be converted to the corresponding set of h_{in} and q'''. Converting N_{sub} into the corresponding h_{in} is a straightforward task (as can be seen from the definition, equation 3.7). The conversion of the specified N_{dh} into a q''' is a bit trickier however. The function `findcond` was written for this task. It receives a target value for N_{dh} as input, and uses an initial guess for q'''. The inital guess q''' is used as input in the steady-state code to find the corresponding G. With the set q''' and G, N_{dh} is calculated using equation 3.8. Depending on this N_{dh} and the target value for N_{dh}, a new q''' is chosen. This process repeats itself until N_{dh} is found to within a specified tolerance or

[4] If the input is given in terms of h_{in} and q''', then the domain of the resulting stability plane (which is in terms of N_{sub} and N_{dh}) is not known before the calculation starts.

when a pre-set maximum number of iteration steps has passed. If the first is the case then the corresponding value of q''' is used for the stability plot, if the latter is the case, then no q''' is found and no stability calculation will be done (resulting in an empty spot in the stability plot).

The function `stabtest` has the input parameters h_{in} and q'''. For these operating conditions the eigenvalues are calculated and subsequently filtered according to a specified upper limit, as described in section 4.6. Finally it is checked whether the real parts of all the eigenvalues are smaller than the specified threshold. Theoretically this value should be equal to zero, but since there can be small errors due to round off, a value close to zero is chosen (for instance 10^{-4}).

The method as described above is computationally more expensive than the one used in Sanders (2009). This method is schematically described in figure 4.3. It starts with an initial point, if this is in the stable region the power is increased and if it happens to be in the unstable region then the power is decreased. This procedure is repeated until the neutral stability boundary is crossed. When the neutral stability boundary is found to within a certain tolerance, then N_{sub} is increased and the last value for the power is taken as the new initial guess.

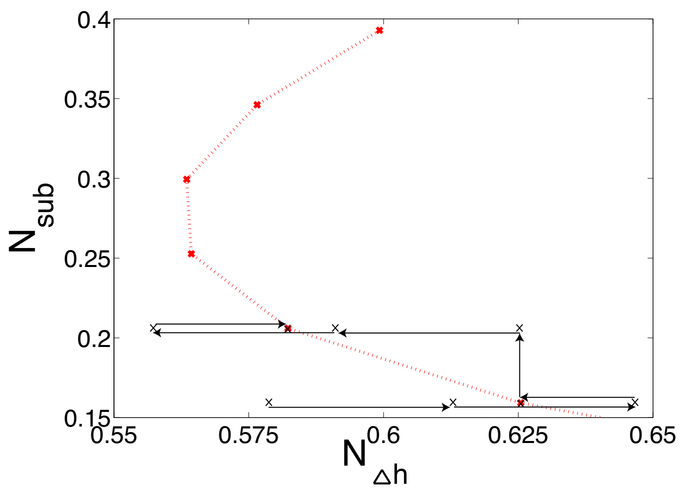


Figure 4.3: Schematic overview explaining the method used by Sanders. Figure adopted from Sanders (2009).

The drawback of this method is that it does not work when there is more than one neutral stability boundary for a single N_{sub}, since it moves up when the first is found. Therefore it was decided to work with a more robust algorithm despite its computational expense.

Chapter 5

Results

In the previous chapter the implementation of the theory into a computer code was discussed. In the current chapter the output of the computer code will be interpreted and benchmarked. First the pressure-flow characteristic of a loop is presented and discussed. After this, the variation of the steady-state variables h and p will be shown and interpreted. In the third section, the power-flow map for a natural convection loop will be benchmarked. In the last section, the stability plot for a forced system will be benchmarked.

5.1 Pressure-flow characteristic

The pressure-flow characteristic of the loop defined in appendix B has been calculated and is plotted in figure 5.1. The calculation is done by imposing a flow through the loop and calculating the difference in pressure for the input and output of the loop (this can be done without iteration using equations 2.13 and 2.16). The operational parameters were set to $\dot{q} = 1 \cdot 10^6$ W and $h_{\text{in}} = 1.558046 \cdot 10^6$ J/kg. The resolution was set to 100 cells per meter and therefore $N = 100 \cdot 10 = 10^4$ cells.

Δp is the pressure drop that is needed for realizing the corresponding flow in figure 5.1, it can be provided by a pump. If Δp equals zero, then no additional pressure drop is needed, which means that the flow can be realized by natural convection alone. If Δp is positive, then an addional pressure drop is needed for the corrseponding flow. A negative Δp means that a pump is needed that works in the negative z direction to slow down the flow. The general trend of this graph is as expected: for higher flow rates, higher pressure drops are needed.

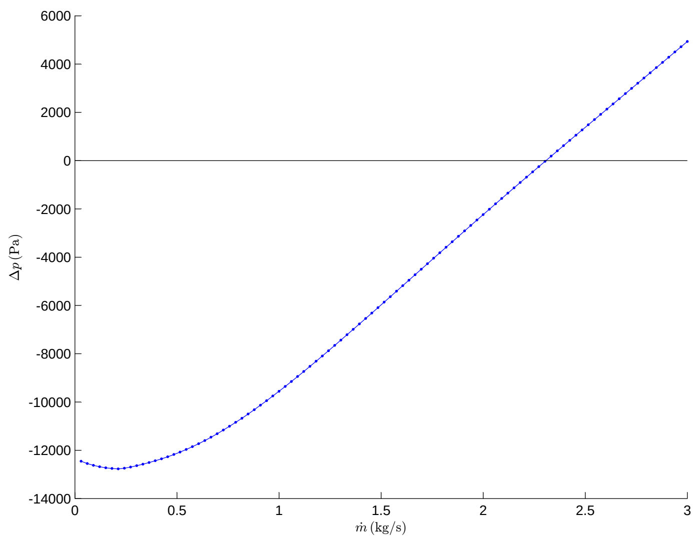


Figure 5.1: Plot of Δp as a function of $\dot{m}$ for the setup in appendix B. Other variables are $\dot{q} = 1 \cdot 10^6$ W, $h_{\text{in}} = 1.558046 \cdot 10^6$ J/kg and `resolution` was set to 100 cells per meter.

5.2 Steady-state calculations

The variation of the pressure and enthalpy as a function of z have been calculated and are shown in figure 5.2. Again the loop defined in appendix B was used. The operational parameters of the calculation were, again, set to $\dot{q} = 1 \cdot 10^6$ W and $h_{\text{in}} = 1.558046 \cdot 10^6$ J/kg, as was done in the previous section. This time, however, the calculations are restricted to a natural convection loop, which means that the pressure drop Δp over the loop is zero.

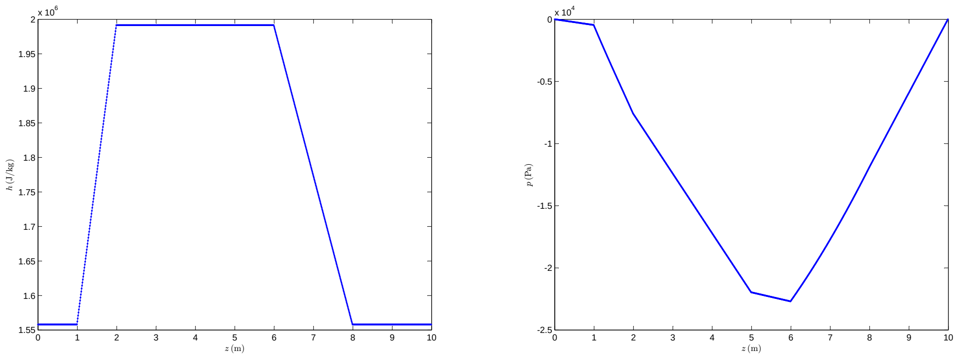


Figure 5.2: Variation of the enthalpy h (left) and the pressure p (right) as a function of the axial distance z for the setup in appendix B.

As can be seen from the plot of the enthalpy h in figure 5.2, the enthalpy is constant for certain intervals of z ($[0, 1]$, $[2, 6]$ and $[8, 10]$). These intervals correspond to the parts of the loop outside the core and the cooler. Because there is no heat transport with the walls of the system, the enthalpy can neither decrease or increase (therefore also ρ will be constant at those intervals). The interval of z where h increases corresponds to the core, the interval where h decreases corresponds to the cooler. As can be seen from figure 5.2, the magnitude of the slope of h is larger for the core than for the cooler. The absolute value of the enthalpy change for the core and cooler are always equal. This means that the slope fully depends on the length of the core and the cooler.

The pressure profile is also given in figure 5.2. The system pressure of $25 \cdot 10^6$ Pa was neglected for convenience. As can be seen from the pressure profile, the input pressure is indeed equal to the output pressure of the loop, which is needed for natural convection. The first small part of the pressure profile $[0, 1]$, shows a small decrease in pressure. This interval corresponds to the lower horizontal part of the tube, the pressure drop over this interval is due to friction. The next interval, $[1, 5]$, is the riser section. The pressure p shows the most dramatic decrease in this section, since the gravity force has the same direction as the friction force in the riser section. The interval $[5, 6]$ is the upper horizontal part of the loop. As for the first interval of the loop, gravity does not affect the pressure p, friction is responsible for the change in pressure in this interval. The reason that the slope is larger than in the first interval is that the density ρ is lower now (due to the increased enthalpy h). The last interval, $[6, 10]$, is the downcomer. It makes up for the lost pressure, as can be seen in figure 5.2. The friction force is directed in the negative z direction, gravity, however, is now helping the flow.

The mass flow $\dot{m}$ has also been calculated for this setup and operational parameters, yielding $\dot{m} = 2.30734\,\text{kg/s}$. Note that this agrees with the flow for which the pressure drop is zero in figure 5.1.

The time needed for calculating and plotting is, in this case, approximately 0.7 seconds. It was noted that when choosing a finer grid, the increase in calculation time was linear with the increase of the number of cells.

5.3 Steady-state benchmark

For the loop defined in appendix B a benchmark was performed. A power-flow map made by an in-house code on COMSOL was used as the benchmark. 34 points were used as input. The inlet enthalpy h_{in} equals $1.558046 \cdot 10^6$ J/kg. Figure 5.3 shows that there is an excellent match between the benchmark and the data obtained form the currently developed code. This match means that output from the steady-state code is reliable (at least under conditions used in this calculation).

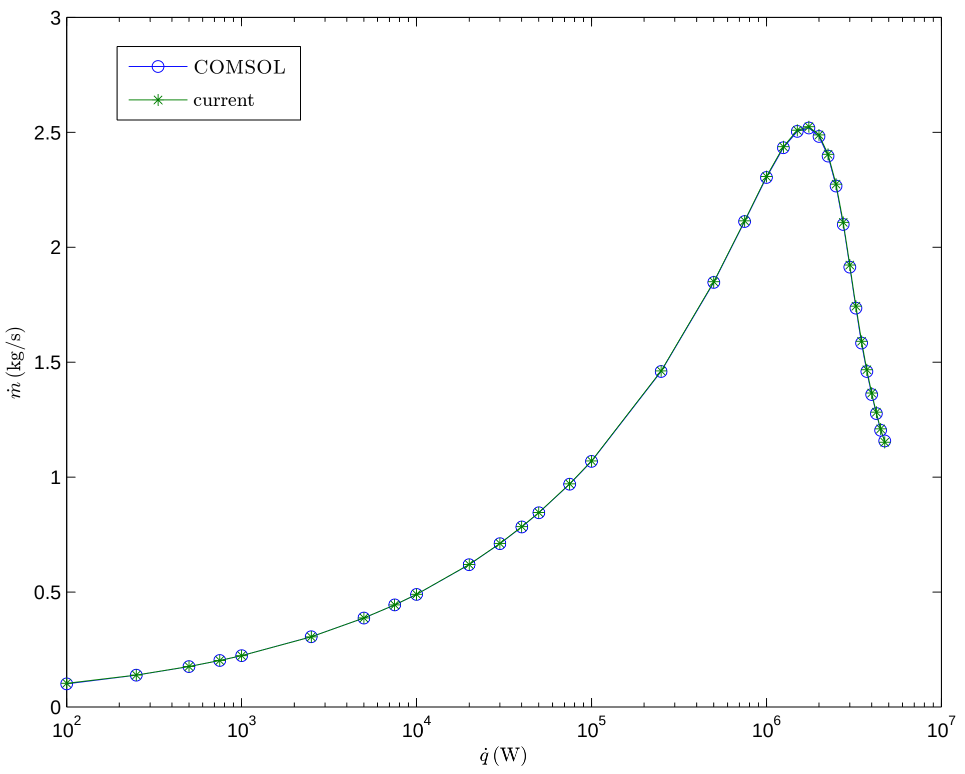


Figure 5.3: Plot of steady-state values for the mass flow $\dot{m}$ as a function of the power $\dot{q}$. The setup is described in appendix B. Data from COMSOL and from calculation. A resolution of 100 cells per meter was used for the calculation.

5.4 Stability benchmark

The code for creating a stability plot has been benchmarked on the setup described in Ambrosini and Sharabi (2008). The setup is a vertical flow channel with a constant external pressure drop, a variable h_{in} and also a variable power. For convenience, the parameters of the setup are also given in appendix C. Before making the stability plot it was first verified whether the profile of steady-state variables as given in the article could be replicated by the code up to a satisfactory degree of accuracy.

Defining the setup in the function `geometry` was a straightforward task. The pressure drop over the flow channel was estimated to be $\Delta p = 0.14\,\text{MPa}$. The number of cells was also set to the value used in the article $N = 48$. This was done by setting the `resolution` to

$$\texttt{resolution} = \frac{N}{L_{\text{C}}} = \frac{48}{4.2672} = 11.2486 \text{ cells per meter.} \tag{5.1}$$

These settings yielded $\dot{m} = 0.0555778\,\text{kg/s}$, which is close to the $\dot{m} = 0.055\,\text{kg/s}$ as stated in the article. Also the pressure, density and velocity were calculated and plotted as functions of z. In these graphs the corresponding data from the article was also plotted to be able to make a better comparison.

As can be seen in figure 5.4 the density over the setup as given by Ambrosini deviates quite a bit from the calculated values (especially in the beginnig and the middle part of the setup). As can be seen in the zoomed-in graph in figure 5.4, even the density at the inlet of the heated section does not match. The reason for this difference was somewhat unexpected since Ambrosini also uses data from NIST. The difference could be caused by one or all of the following conversions. First of all, the inlet temperature, $T_{\text{in}} = 280\,^{\circ}\text{C}$, was converted into the inlet enthalpy $h_{\text{in}} = 1.2305 \cdot 10^6$ J/kg. Second, this h_{in} was used to estimate the corresponding ρ with the cubic splines that were discussed in section 4.2. Third, the conversion of the temperature into a density, as was done by Ambrosini. A work-around could be to match the density to the density of Ambrosini and calculate the corresponding temperature and enthalpy. This, however, has not been done in this work.

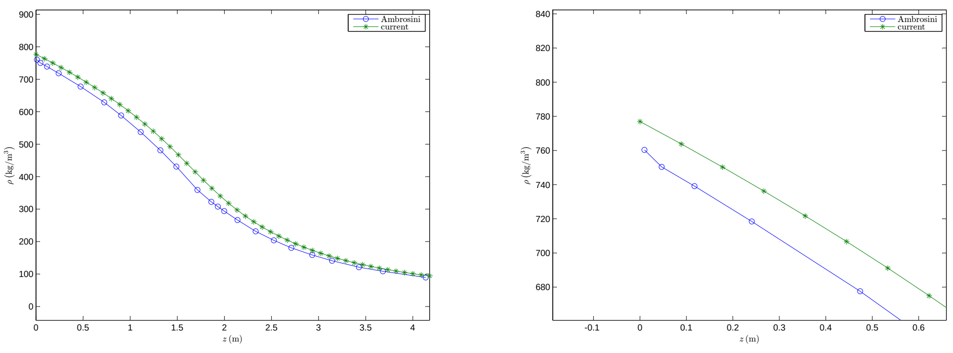


Figure 5.4: Plot of the density ρ as a function of the axial distance z for the setup in Ambrosini and Sharabi (2008). Full range of z(left) and zoomed-in on the entrance of the setup (right).

The variation of pressure as a function of the height z is shown in figure 5.5. The match seems to be better than for the density in figure 5.4. At the end of the setup, the variation between the pressure profiles suddenly increases. This can be seen more easily in the zoomed-in plot in figure 5.5. The graph of the calculated values has a sudden change in slope at the end. The reason for this change is that there is a K_{out} defined in the setup. Therefore this behaviour is actually expected. Although expected, this is not found in the pressure profile of the data from Ambrosini flattens.

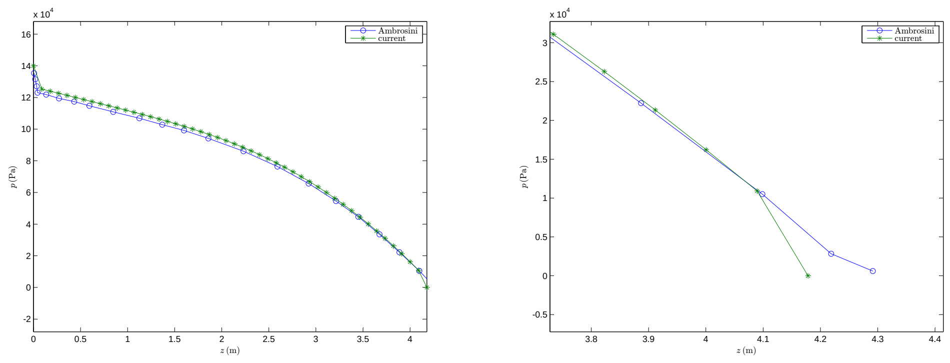


Figure 5.5: Plot of pressure p as a function of axial distance z for the setup in Ambrosini and Sharabi (2008). Full range of z(left) and zoomed-in on the end part of the setup (right).

The match between the velocity profiles is good. A zoomed-in version is included revealing no irregularities.

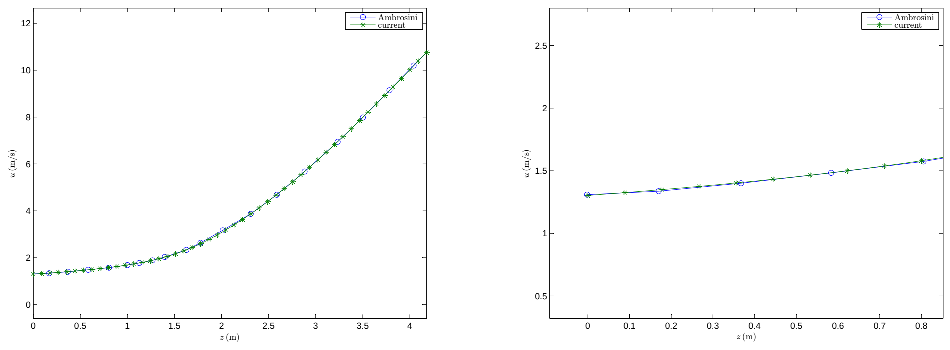


Figure 5.6: Plot of velocity u as a function of axial distance z for the setup in Ambrosini and Sharabi (2008). Full range of z(left) and zoomed-in on the first part (right).

Although the match was not 100%, the stability plots have been compared. To facilitate the comparison, the stability plane defined by Ambrosini's dimensionless numbers was converted into a stability plane defined by the dimensionless numbers in equations 3.7 and 3.8. The dimensionless numbers used in Ambrosini and Sharabi (2008) are

$$N_{\mathrm{SUBPC}} \equiv \frac{\beta_{\mathrm{pc}}}{C_{p,\mathrm{pc}}}\left(h_{\mathrm{pc}} - h_{\mathrm{in}}\right) \tag{5.2}$$

$$N_{\mathrm{TPC}} \equiv \frac{\beta_{\mathrm{pc}}}{C_{p,\mathrm{pc}}}\frac{\dot{q}}{GA} \tag{5.3}$$

Comparison with equations 3.7 and 3.8 reveals that the dimensionless numbers differ only up to a multiplicative constant. The familiar dimensionless numbers (equations 3.7 and 3.8) are obtained by multiplying Ambrosini's dimensionless numbers by the constant

$$\frac{C_{p,\mathrm{pc}}}{\beta_{\mathrm{pc}} h_{\mathrm{pc}}}, \tag{5.4}$$

where β_{pc} is the isobaric thermal expansion coefficient at the pseudo critical point, $C_{p,\mathrm{pc}}$ is the specific heat at constant pressure, also taken at the pseudo critical point and h_{pc} is the specific enthalpy at pseudo-critical conditions. For water the numerical values for the constants are: $\beta_{\mathrm{pc}} = 0.12849\,\frac{1}{\mathrm{K}}$, $C_{p,\mathrm{pc}} = 76444\,\frac{\mathrm{J}}{\mathrm{kg\,K}}$ and $h_{\mathrm{pc}} = 2.1529 \cdot 10^6$ J/kg.

As was mentioned in section 3.2.1, the maximal value for N_{sub} equals 1. While converting the stability plot, as described above, there were also values encountered for $N_{\mathrm{sub}} \geq 1$. These numbers have been considered to be unphysical and are therefore not included in the stability plot in figure 5.7.

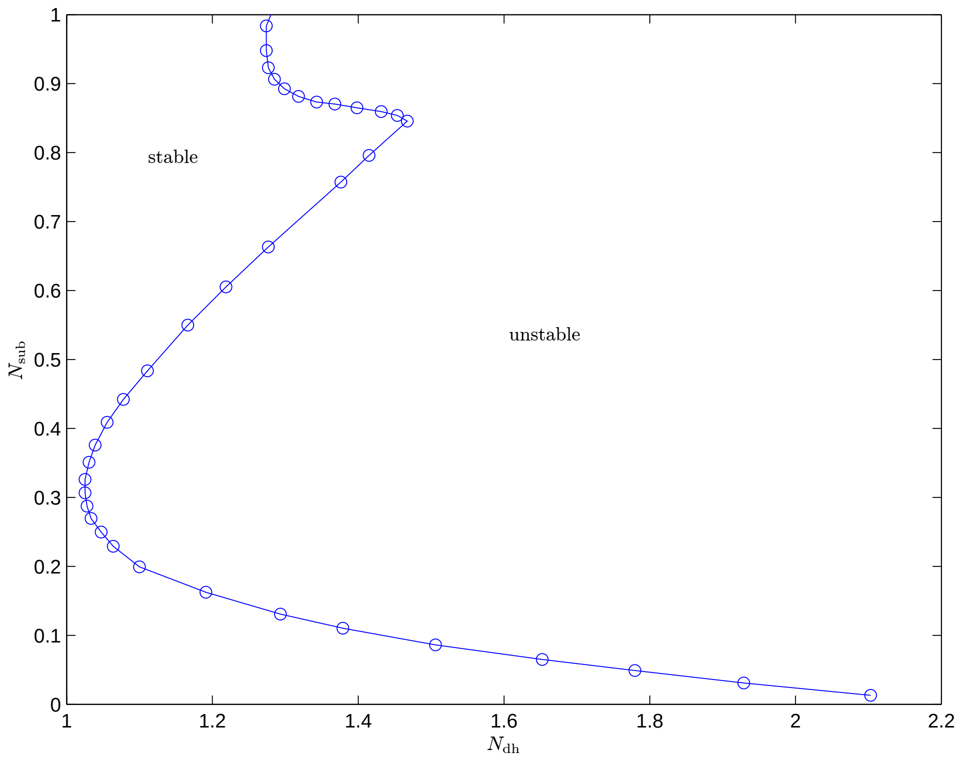


Figure 5.7: Stability plane from Ambrosini and Sharabi (2008), converted to the dimensionless numbers N_{sub} and N_{dh} to facilitate comparison. The stable and unstable regions are indicated. $N = 48$ cells.

In figure 5.8 the stability plots for two different resolutions are shown. The figures do not match with the stability plot from figure 5.7. Also, figure 5.8 shows that there is a grid dependency in the stability plots, this grid dependency disappears, however, for resolutions higher than 40 cells per meter.

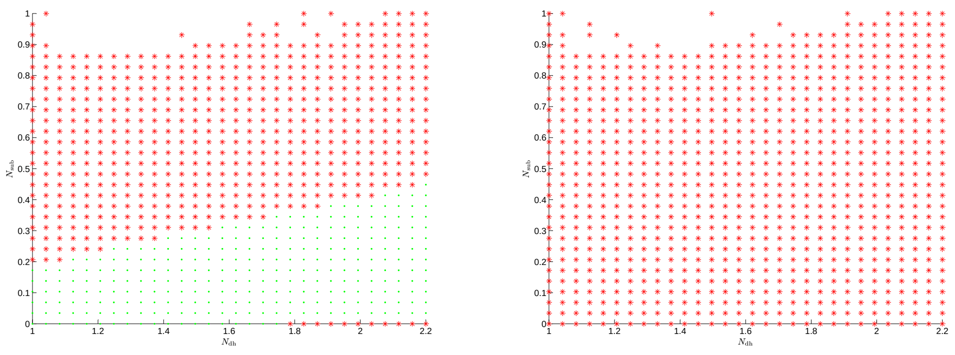


Figure 5.8: Stability plots for the setup in Ambrosini and Sharabi (2008) with a resolution of 11.2486 cells per meter (which means N = 48, see equations 5.1) (left) and resolution = 50 cells per meter (right) For this resolution, the stability plot is grid independent.

Attempts were made to alter the stability plot by manually varying the friciton. Altering the friction was of no effect on the resulting stability plane however. This is another signal that the stability plots are incorrect.

The time needed for the calculation with a resolution of 11.2486 cells per meter was slightly less than 2 minutes. The calculation time for a resolution of 50 cells per meter is approximately 30 minutes. This means that the calculation time increases strong with an increase in the number of cells.

Chapter 6

Conclusions

The steady-state code is succesfully benchmarked for a natural convection loop. The steady-state variation of enthalpy over this loop is according to expectation. The succesfull benchmark also proves that for some lower threshold for the resolution, the solution becomes grid independent. For steady-state calculations seemed to increase linear for an increased number of cells. The grid independent solution was found for calculation times of less than one second.

For the stability code, the benchmark was unsuccessful. There was a grid independency found, however. It was also found that the stability plot does not respond as expected to a change in friction. It has also been observed that the calculation time for the stability plane increases strong for an increase in the number of cells.

Further research is necessary for the stability code. In the discussion and recommendations that follow, possible reasons for the stability code not working properly are listed and subsequently possible solutions are suggested.

Chapter 7

Discussion and Recommendations

In this chapter several things that are open to improvement are listed. In some cases possible solutions are presented.

7.1 Eigenvalues

The criterion used to filter the eigenvalue, as desribed in section 4.6, is not a very nice way to solve the problem of wrong eigenvalues, because, although it effectively removes the wrong values, it also removes the actual eigenvalues that happen to be large. A solution to this problem could be to use Fortran instead of MATLAB. The `eig` function in MATLAB itself is based on Fortran packages, the Fortran packages however, seem to have better documentation than the MATLAB help. Also the possibilities of using `eigs` routine (a function comparable to `eig` for sparse matrices) to benefit from the obvious sparsity of the matrices seems promising, because it is based on algorithms that are faster.

7.2 Variable cross-section

A natural extension to the current code is allowing the diameter of the system to vary. For simplicity, this was not done in this case study. For a lot of interesting setups, however, the cross-section is not constant. When a variable-cross section is allowed, it might be more convenient to work with the mass flow $\dot{m}$ instead of the mass flux G, since the steady-state mass flow will still be constant in this case.

7.3 Errors

The reason for failing to make a correct stability plot could also be an error, instead of using the wrong approach. An error could be made in the mathematical derivation in the first few chapters of this thesis. Also the translating process from mathematics to the MATLAB language is sometimes tricky, despite their similarity, or maybe because of their similarity. This scenario has only one solution: rereading and checking everything. For checking the structure of the matrices in MATLAB, the `spy` function was used as described in appendix D.

Appendices

Appendix A

Test case for `eig`

As mentioned in section 4.6 the output from the function `eig` was different than what was expected. To get more feeling for the function and its output, a test case was considered. The test case had to be simple yet representative. For simplicity the size of the matrices was kept small. Still, the set of equations was chosen such, that it leads to a singular generalized eigenvalue problem. Consider the following set of equations

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = 3x_1, \tag{A.1}$$

$$0 = x_1 - x_2. \tag{A.2}$$

It can be easily verified that this system of equations is solved by $x_1 = x_2 = e^{3t}$. Therefore, both x_1 and x_2 have eigenvalue 3. The system can be rewritten in the form $\mathsf{A}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathsf{B}\boldsymbol{x}$, (same as in chapter 3) as follows

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{A.3}$$

Inserting

$$\boldsymbol{x} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} e^{\lambda t}, \tag{A.4}$$

and performing the time derivative yields

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \lambda \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} e^{\lambda t} = \begin{bmatrix} 3 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} e^{\lambda t}. \tag{A.5}$$

Finally, division by $e^{\lambda t}$ leads to the generalized eigenvalue problem

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \lambda \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \tag{A.6}$$

which can be rewritten as

$$\begin{bmatrix} \lambda - 3 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = 0. \tag{A.7}$$

This equation has a non-trivial solution (a solution other than $c_1 = c_2 = 0$) only when the determinant of the matrix equals zero:

$$\det \begin{bmatrix} \lambda - 3 & 0 \\ -1 & 1 \end{bmatrix} = 0. \tag{A.8}$$

This reduces to the very simple equation

$$\lambda - 3 = 0 \qquad \Rightarrow \qquad \lambda = 3. \tag{A.9}$$

This corresponds to what was expected. It should be noted that the algebraic multiplicity is 1.

Next, this problem was solved by the function `eig`. This is easily done by first defining the matrices A and B and subsequently calling the `eig` function.

```
>> A = [1 0;0 0];
>> B = [3 0;1 -1];
>> E = eig(B,A);
```

Now, `E` has the value

$$\texttt{E} = \begin{bmatrix} \texttt{-Inf} \\ \texttt{3} \end{bmatrix}.$$

MATLAB finds the eigenvalue 3 with algebraic multiplicity 1 as was predicted. But the reason why MATLAB also returns the value `-Inf` is unclear. As stated in section 4.6 this problem was solved by filtering the values with magnitude larger than a specified limit.

Appendix B

Dimensions of test loop

The following natural convection loop has been used extensively during the coding and has been used for the calculations in sections 5.1 and 5.3. Therefore it was decided to have a separate page to define the geometry of the setup.

The size of the different parts of the setup can be read off from figure B.1. The height of the setup, for instance, is 4 m. The total length of the loop is $2 \cdot 1 + 2 \cdot 4 = 10\,\mathrm{m}$. The diameter of the circular tube is 5 cm.

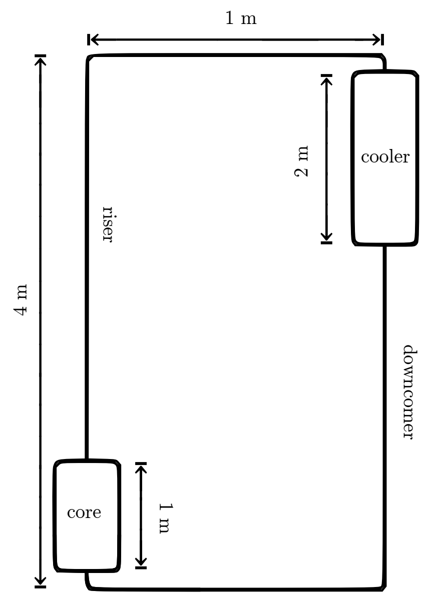


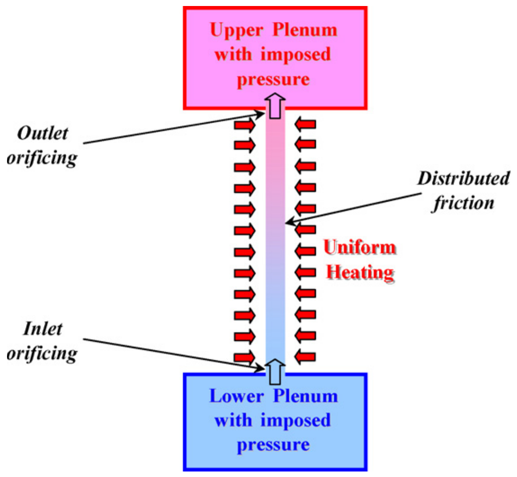
Figure B.1: Schematic diagram with dimensions of test loop. Note that the individual parts in this drawing are not proportionally scaled.

There are no local pressure coeffcients defined and the friciton is implemented by the Haaland approximation (equation 2.6) with relative roughness $\left(\frac{\epsilon}{D_H}\right.$ equal to $\left.0.0015\right)$. The fluid used is water.

Appendix C

Dimensions of test channel

The vertical flow channel from Ambrosini and Sharabi (2008) has been used in this thesis for benchmarking the stability calculation as described in section 5.4.



Upper Plenum with imposed pressure
Outlet orificing
Distributed friction
Uniform Heating
Inlet orificing
Lower Plenum with imposed pressure

Figure C.1: Schematic diagram of vertical flow channel. Figure adopted from Ambrosini and Sharabi (2008).

The parameters describing the geometry are listed below

- channel length $L_{\mathrm{C}} = 4.2672\,\mathrm{m}$
- coolant flow area $A = 5.49 \cdot 10^{-5}\,\mathrm{m}^2$
- heated perimeter $P = 32.04 \cdot 10^{-3}\,\mathrm{m}$
- hydraulic diameter $D_{\mathrm{H}} = 3.4 \cdot 10^{-3}\,\mathrm{m}$
- system pressure $p = 25 \cdot 10^{6}\,\mathrm{Pa}$
- inlet orifice pressure loss coefficient $K_{\mathrm{in}} = 20$
- outlet pressure loss coefficient $K_{\mathrm{out}} = 1$

A constant has been used to model the friction factor: $f = 0.0352$. The fluid used is water.

Appendix D

Sparsity pattern

Before the eigenvalues can be calculated, the matrices A and B from equation 3.3 need to be defined in MATLAB. Although this might not seem a very difficult task, it is a very tedious task where mistakes are easily made. These mistakes can alter the resulting eigenvalues and therefore they need to be avoided.

The built-in function `spy` can be used to check whether the sparsity patterns of the matrices `A` and `B` as defined in MATLAB are equal to those of the matrices A and B. The structure of A is written out in equation 3.2. The structure of B can be easily derived from equations 2.23, 2.28 and 2.35.[1] To use the `spy` function on a matrix, the matrix must first be converted into a sparse matrix. The sparsity pattern for `A` is plotted by executing

```
>> A = sparse(A);
>> spy(A)
```

Repeating this procedure for matrix `B` yields the figures shown below.

[1]In deriving the structure of B it should be remembered that the first N rows correspond to the mass equation (equation 2.23), the rows $N+1$ to $2N$ correspond to the energy equation (equation 2.28) and finally the momentum equation (equation 2.35) fills up rows $2N+1$ to $3N$. The same order was used in defining A as given in equation 3.2

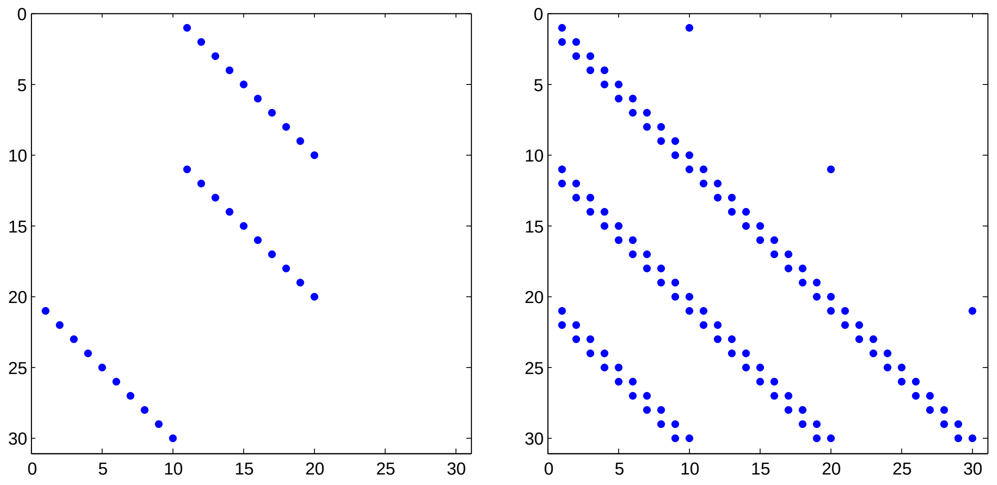


Figure D.1: Sparsity patterns of `A` (left) and `B` (right) as obtained by the `spy` function. Number of cells equals 10 and therefore both `A` and `B` are 30-by-30 matrices.

A small number of cells was chosen to be able to distinguish more clearly the individual dots (i.e. the individual non-zero elements in the matrices). The structure of the matrices in figure D.1 is the same as that of A and B as derived in this thesis. This assures that at least the structure of the matrices is correctly definded in MATLAB.

Bibliography

W. Ambrosini and M. Sharabi. Dimensionless parameters in stability analysis of heated channels with fluids at supercritical pressures. *Nuclear Engineering and Design*, 238(8):1917 – 1929, 2008.

W.E. Boyce and R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley & Sons, New York, USA, eighth edition, 2004.

D.D.B. van Bragt. *Analytical Modeling of Boiling Water Dynamics.* PhD thesis, TU Delft, Faculty of Applied Sciences, Delft, The Netherlands, 1998.

K. Fischer, T. Schulenberg, and E. Laurien. Design of a supercritical water-cooled reactor with a three-pass core arrangement. *Nuclear Engineering and Design*, 239:800–812, 2009.

Generation IV International Forum. *A Technology Roadmap for Generation IV Nuclear Energy Systems*, 2002.

T. Ortega Gómez. *Stability Analysis of the High Performance Light Water Reactor.* PhD thesis, Forschungszentrum Karlsruhe, Karlsruhe, Germany, 2008.

S.E. Haaland. Simple and explicit formulas for the friction factor in turbulent pipe flow. *Journal of Fluids Engineering*, 105(1):89–90, 1983.

J. Hofmeister, C. Waata, J. Starflinger, T. Schulenberg, and E. Laurien. Fuel assembly design study for a reactor with supercritical water. *Nuclear Engineering and Design*, 237(14):1513 – 1521, 2007.

L.P.B.M. Janssen and M.M.C.G. Warmoeskerken. *Transport Phenomena Data Companion.* VSSD, Delft, The Netherlands, third edition, 2006.

C.P. Marcel, M. Rohde, V.P. Masson, and T.H.J.J. van der Hagen. Fluid-to-fluid modeling of supercritical water loops for stability analysis. *International Journal of Heat and Mass Transfer*, 52(21-22):5046 – 5054, 2009.

C.B. Moler and G.W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, 1973.

M.J. Moran and H.N. Shapiro. *Fundamentals of Engineering Thermodynamics.* John Wiley & Sons, Chichester, UK, fifth edition, 2006.

S.V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Publishing Corporation, Washington, D.C., USA, 1980.

M.B. Sanders. Thermo-hydraulic stability analysis of the high performance light water reactor and a scaled experimental facility. Master's thesis, TU Delft, Faculty of Applied Sciences, Delft, The Netherlands, 2009.

D. Squarer, T. Schulenberg, D. Struwe, Y. Oka, D. Bittermann, N. Aksan, C. Maraczy, R. Kyrki-Rajamäki, A. Souyri, and P. Dumaz. High performance light water reactor. *Nuclear Engineering and Design*, 221(1-3):167 – 180, 2003. Mid-Term Symposium on Shared-Cost and Concerted Actions.

J. Stewart. *Calculus - Early Transcendentals.* Thomson Brooks/Cole, sixth edition, 2008.

MATLAB Function Reference. The MathWorks.

N.E. Todreas and M.S. Kazimi. *Nuclear Systems I - Thermal Hydraulic Fundamentals.* Taylor & Francis, New York, USA, 1990.