Työterveyslaitos | Arbetshälsoinstitutet
Finnish Institute of Occupational Health

Temperature distribution in regenerators

Rauno Holopainen

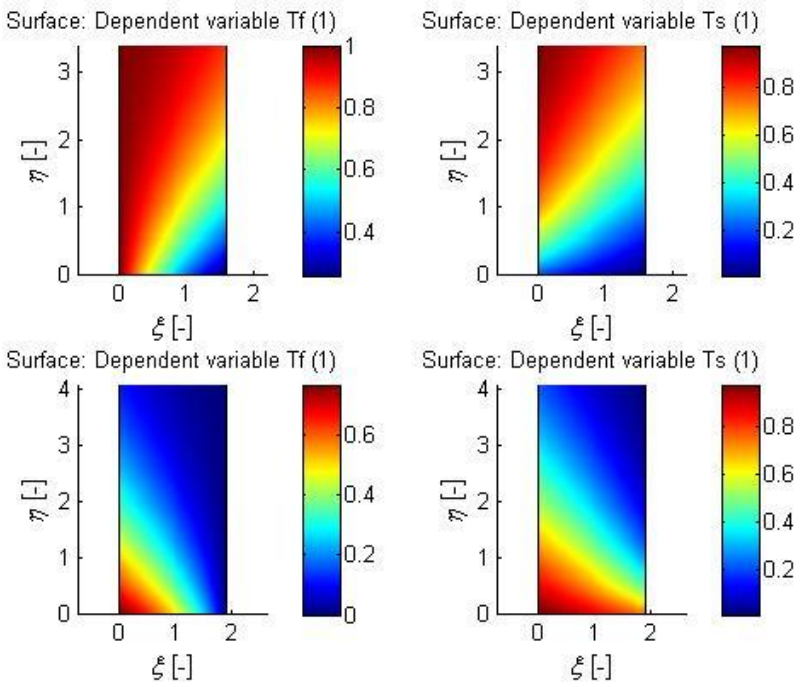
Eero-Matti Salonen

Finnish Institute of Occupational Health

Temperature distribution in regenerators

Rauno Holopainen, Finnish Institute of Occupational Health

Eero-Matti Salonen, Aalto University



Finnish Institute of Occupational Health

Helsinki 2014

Finnish Institute of Occupational Health

User-centric Indoor Environments -Theme

Topeliuksenkatu 41 a A

00250 Helsinki

www.ttl.fi

Editorial staff: Rauno Holopainen and Eero-Matti Salonen

Cover: Albert Hall Finland Ltd

ISBN 978-952-261-462-9 (nid.)

ISBN 978-952-261-430-8 (pdf)

FOREWORD

Heat recovery is a topical issue, as it affects buildings' energy efficiency. A typical heat recovery application is a ventilation system, in which intake air is heated by warm outlet air. The heat recovery of ventilation systems can be roughly divided into recuperative and regenerative heat exchangers. This report focuses on regenerative heat exchanger computational modelling, in which mass is alternately heated and cooled by changing the direction of the air flow.

As the authors carried out the necessary modelling and programming mainly during evenings and weekends, it took over five years to complete the report. The authors wish to warmly thank the Finnish Institute of Occupational Health and Aalto University for making this study possible. Further, the advice of Jorma Kinnunen MSc (Tech.), Pasi Marttila MSc (Tech.) and Jukka Tarvo MSc (Tech.) on the use of COMSOL Multiphysics with MATLAB program is gratefully acknowledged. In addition, the authors would like to warmly thank Alice Lehtinen for her advice on the use of the English language.

Helsinki in July 2014

Authors

ABSTRACT

The solution to the equations governing regenerator behaviour demand numerical methods, as analytical solutions are not available. The classic numerical method in this case seems to be finite difference method applying stepwise marching in time. This report presents an alternative approach, which has some advantages from the coding effort and accuracy perspectives. The approach applies a space-time finite element formulation, and thus no marching in time was performed. Each calculation case used three meshes of consecutively increasing densities. This made it possible to obtain an estimate of the errors in the results obtained by the finest mesh. Programming and simulation was carried out using the commercial COMSOL Multiphysics and MATLAB programs (COMSOL 4.3 with MATLAB). We provide the results of some example cases to show the accuracy and workings of the method.

The developed model can be applied to regenerators of both fixed and rotary heat storage mass. The model is a useful tool in the design and optimization of heat exchangers when, for example, studying the effect of different hot and cold period lengths on outlet air temperatures and on thermal ratios.

CONTENTS

NOMENCLATURE

Latin symbols

A	total heat transfer surface area, m^2
c	specific heat at constant pressure, $\mathrm{kJ/kg\,K}$
$\overline{e}_\mathrm{r}$	relative error estimate, $-$
$\overline{h}$	overall heat transfer coefficient, $\mathrm{W/m^2\,K}$
k	thermal conductivity, $\mathrm{W/m\,K}$
L	length of storage unit, m
m	mass, kg
$\dot{m}_\mathrm{f}$	mass rate of fluid flow, $\mathrm{kg/s}$
P	perimeter, m, period time length, s
p	pressure, $\mathrm{N/m^2}$
q	heat flux density, $\mathrm{W/m^2}$
$\mathbf{q}$	heat flux vector, $\mathrm{W/m^2}$
r	effective convergence rate, $-$
S	surface, cross-sectional area, m^2
T	nondimensional temperature, $-$
t	temperature, $\mathrm{°C}$, K
V	volume, m^3
$\mathbf{v}$	velocity vector, $\mathrm{m/s}$
v_x	axial velocity, $\mathrm{m/s}$
x	axial coordinate, m

Greek symbols

α	weight factor, –
ρ	density, $\mathrm{kg/m^3}$
β	volume expansion coefficient, $1/\mathrm{K}$
Φ	dissipation function, $1/\mathrm{s^2}$
η	nondimensional time coordinate, –
η_{REG}	thermal ratio, –
Λ	reduced length, –
λ	grid refinement ratio, –
μ	viscosity, $\mathrm{Ns/m^2}$
ξ	nondimensional position coordinate, –
Π	reduced period, –
τ	time, s

Subscripts

c	coarse
f	fluid, fine
i	inlet
m	medium
s	solid (storage) material
0	initial

Superscripts

′ hot period

″ cold period

~ time average

^ space average

1 INTRODUCTION

We considered the storage unit configurations depicted schematically in Figure 1.1 (a). The main direction of the fluid flow is in the axial direction of the unit (the x-axis direction). We assumed that the cross-sectional (plane perpendicular to the x-axis) geometry of the unit was approximately constant with respect to axial coordinate x.

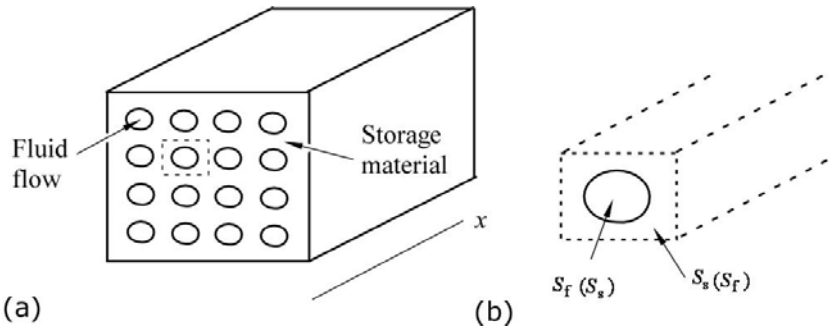


Figure 1.1. (a) Storage unit configuration. (b) Representative prism.

The round openings in Figure 1.1 (a) represent cross-sections of fluid passages or channels. A typical representative fluid channel has a cross-sectional area S_f and a hydraulic perimeter P with solid storage material (cf. Remark 2.1). The dashed rectangular shape in the figure encloses a representative portion of the cross-section, and the corresponding prismatic domain is depicted in Figure 1.1 (b). The associated cross-sectional area of the solid storage material is S_s, and the perimeter with the fluid channel is also P. We assume that the heat flux through the mantle surface of the storage medium prism domain vanishes due to symmetry reasons, or at the boundaries of the unit, due to heat insulation.

Remark 1.1. It should be noted that the cross-sectional shape of a representative prism is not rectangular in general. In the case depicted in Figure 1.1, the rectangular shape seems natural due to the double symmetry of fluid openings. With more irregular openings, the shapes of the corresponding storage domains also become more complicated. The boundaries are determined from the assumed lines through which the heat flux vanishes. However, the exact positions of these border lines are not significant, at least in the specific model considered later, due to the final summation procedure discussed in Remarks 2.1 and 2.2. □

Remark 1.2. Figure 1.1 (a) does not describe a case in which the storage material consists of, for example, plate type elements. However, we were able to create this latter case without drawing a new figure, by simply replacing the roles of the fluid and storage domains in the figure as indicated in parentheses in Figure (b). The mantle surface of the representative prism was then in contact with the fluid of another representative prism, and due to symmetry etc., the heat flux was again assumed to vanish there. □

2 BASIC EQUATIONS

We base our presentation strongly on the theory and arguments given by Schmidt and Willmott (1981). We will thus also mainly use the same notations. However, although most of the equations treated in this report are derived in Schmidt and Willmott (1981), we still want to record the governing equations and the main assumptions for simplifications, in order to make the report reasonably self-contained. In addition, the manner of derivation differs from that given in Schmidt and Willmott (1981) as we make the starting point the energy equation in its differential equation form, and proceed in a somewhat detailed manner. A rather recent text on regenerative heat transfer is in Willmott (2002), written by the second author of Schmidt and Willmott (1981). However, this latter reference contains no essential changes with respect to the theme of the present report. The notations in Willmott (2002) differ rather much from those used Schmidt and Willmott (1981). However, as mentioned above, we prefer mostly the notations of Schmidt and Willmott (1981).

Fluid. The differential energy equation for viscous fluid flow can be written as (see e.g. Rohsenow and Choi (1961), p. 170)

$$\rho_{\mathrm{f}} c_{\mathrm{f}} \frac{\mathrm{D}t_{\mathrm{f}}}{\mathrm{D}\tau} = -\mathrm{div}\mathbf{q}_{\mathrm{f}} + t_{\mathrm{f}} \beta \frac{\mathrm{D}p}{\mathrm{D}\tau} + \mu \Phi \,, \tag{2.1}$$

where τ is time, t_{f} the thermodynamic temperature of the fluid, $\mathbf{q}_{\mathrm{f}}$ the heat flux vector, p the pressure, ρ_{f} the density, c_{f} the specific heat at constant pressure, β the volume expansion coefficient, μ the viscosity, and Φ the dissipation function. Possible heat sources are not included. The derivatives in (2.1) are substantial ones:

$$\frac{\mathrm{D}}{\mathrm{D}\tau} \equiv \frac{\partial}{\partial \tau} + v_x \frac{\partial}{\partial x} + v_y \frac{\partial}{\partial y} + v_z \frac{\partial}{\partial \mathrm{z}} \,. \tag{2.2}$$

Here, x, y and z are orthogonal Cartesian coordinates and v_x, v_y and v_z the corresponding velocity components of fluid velocity vector **v**. Equation (2.1) is thus given in the Eulerian description. Of course, some additional approximations are already contained in (2.1) as, for instance, the fluid was assumed to obey the linear Stokes' viscosity law, etc. However, due to the broad use of (2.1) in fluid dynamics applications in general, this restriction was not considered here to be a part of the modelling process to be described.

In fluid flow in a regenerator we considered the last two terms in (2.1) to be small and discarded them (*Assumption 1*). This removed the coupling with the "mechanical effect" on the temperature distribution. The energy equation became

$$\rho_{\mathrm{f}} c_{\mathrm{f}} \frac{\mathrm{D}t_{\mathrm{f}}}{\mathrm{D}\tau} + \mathrm{div}\mathbf{q}_{\mathrm{f}} = 0 . \tag{2.3}$$

Normal cases in the literature on regenerators do not include the detailed fluid flow and temperature distributions in the model. Instead, certain average "channel type models" are introduced. To this end, let us consider the detail of the representative prismatic fluid channel depicted in Figure 1.1 (b) and Figure 2.1.

As a final step, we can later easily consider the whole unit as explained in Remarks 2.1 and 2.2.

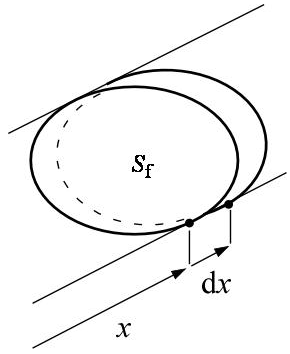


Figure 2.1. Control volume in the fluid prism domain.

We took control volume V of the channel fixed by two planes $x = \mathrm{constant}$ and $x = \mathrm{constant} + \mathrm{d}x$. We integrated both sides of (2.3) over this volume. With obvious notation, the result was

$$\int_V \rho_{\mathrm{f}} c_{\mathrm{f}} \frac{\mathrm{D}t_{\mathrm{f}}}{\mathrm{D}\tau} \mathrm{d}V + \int_V \mathrm{div}\mathbf{q}_{\mathrm{f}} \, \mathrm{d}V = 0 . \tag{2.4}$$

The most important modelling assumption was (*Assumption 2*):

$$t_{\mathrm{f}}\left(x,y,z,\tau\right)\approx t_{\mathrm{f}}\left(x,\tau\right), \tag{2.5}$$

that is, the temperature was considered to be constant on a cross-section. However, in reality, some temperature gradients must exist in the cross-sectional directions, at least in the thin fluid layers near the solid surface, for heat transfer to take place. From this, it follows that $\partial t_{\mathrm{f}}/\partial y=0$ and $\partial t_{\mathrm{f}}/\partial z=0$ and the terms $\partial t_{\mathrm{f}}/\partial\tau$ and $\partial t_{\mathrm{f}}/\partial x$ were constant over the cross-section and could be taken outside the integral. The control volume length in the axial direction in this case was infinitesimal. Further transforming the divergence term by Gauss's formula next produced the form

$$\int_V \rho_{\mathrm{f}} c_{\mathrm{f}}\,\mathrm{d}V\,\frac{\partial t_{\mathrm{f}}}{\partial\tau}+\int_V \rho_{\mathrm{f}} c_{\mathrm{f}} v_x \mathrm{d}V\,\frac{\partial t_{\mathrm{f}}}{\partial x}+\int_S q_{\mathrm{f}}\,\mathrm{d}S=0. \tag{2.6}$$

Here, $q_{\mathrm{f}}=\mathbf{n}\bullet\mathbf{q}_{\mathrm{f}}$ is the heat flux out of the control volume on control surface S and $\mathbf{n}$ is the outward unit normal vector. We next assumed that the heat flux gradient in the axial direction could be discarded (*Assumption 3*), that is, that the total net flux through the cross-sectional planes of the control volume would disappear, and flux would only take place along the perimeter surface of the channel. As the control volume was infinitesimal in the x-axis direction, (2.6) became

$$\mathrm{d}x\int_{S_{\mathrm{f}}} \rho_{\mathrm{f}} c_{\mathrm{f}}\,\mathrm{d}S_{\mathrm{f}}\,\frac{\partial t_{\mathrm{f}}}{\partial\tau}+\mathrm{d}x\int_{S_{\mathrm{f}}} \rho_{\mathrm{f}} c_{\mathrm{f}} v_x \mathrm{d}S_{\mathrm{f}}\,\frac{\partial t_{\mathrm{f}}}{\partial x}+\mathrm{d}x\int_P q_{\mathrm{f}}\,\mathrm{d}P=0, \tag{2.7}$$

or

$$\int_{S_{\mathrm{f}}} \rho_{\mathrm{f}} c_{\mathrm{f}}\,\mathrm{d}S_{\mathrm{f}}\,\frac{\partial t_{\mathrm{f}}}{\partial\tau}+\int_{S_{\mathrm{f}}} \rho_{\mathrm{f}} c_{\mathrm{f}} v_x \mathrm{d}S_{\mathrm{f}}\,\frac{\partial t_{\mathrm{f}}}{\partial x}+\int_P q_{\mathrm{f}}\,\mathrm{d}P=0, \tag{2.8}$$

where the two first integrals were over the channel cross-section and the last integral over the cross-sectional perimeter. For a fluid, it was reasonable here to assume that ρ_{f} and c_{f} were constants, or, for more generality, similar to (2.5) (*Assumption 4*):

$$\begin{aligned}\rho_{\mathrm{f}} &\approx \rho_{\mathrm{f}}\left(x,\tau\right),\\ c_{\mathrm{f}} &\approx c_{\mathrm{f}}\left(x,\tau\right).\end{aligned} \tag{2.9}$$

Thus, ρ_{f} and c_{f} could be moved outside the area integrals in (2.8) to give

$$\rho_{\mathrm{f}} c_{\mathrm{f}} \int_{S_{\mathrm{f}}} \mathrm{d}S_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + \rho_{\mathrm{f}} c_{\mathrm{f}} \int_{S_{\mathrm{f}}} v_x \mathrm{d}S_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial x} + \int_P q_{\mathrm{f}} \, \mathrm{d}P = 0 \,, \tag{2.10}$$

or with some new notation

$$\rho_{\mathrm{f}} c_{\mathrm{f}} S_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + c_{\mathrm{f}} \dot{m}_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial x} = -P \overline{q}_{\mathrm{f}} \,, \tag{2.11}$$

where (2.11)

$$\dot{m}_{\mathrm{f}} = \rho_{\mathrm{f}} \int_{S_{\mathrm{f}}} v_x \mathrm{d}S_{\mathrm{f}} \tag{2.12}$$

is the mass rate of flow,

$$\overline{q}_{\mathrm{f}} = \frac{1}{P} \int_P q_{\mathrm{f}} \, \mathrm{d}P \tag{2.13}$$

the average heat flux on the cross-sectional hydraulic perimeter, and P is the length of the perimeter.

Flux $\overline{q}_{\mathrm{f}}$ was expressed conventionally as

$$\overline{q}_{\mathrm{f}} = \overline{h} \left(t_{\mathrm{f}} - t_{\mathrm{s}} \right) , \tag{2.14}$$

where $\overline{h}$ is the overall heat transfer coefficient on the cross section and t_{s} the (average) temperature on the surface of the channel wall. The perimeter of the cross-section of a prismatic channel can be written as

$$P = A / L \,, \tag{2.15}$$

where A is the (heat transfer) surface area of the channel and L the length of the channel (or the storage unit). When (2.14) and (2.15) were substituted in (2.11) and the equation was multiplied by L, we obtained the final form

$$m_{\mathrm{f}} c_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + \dot{m}_{\mathrm{f}} c_{\mathrm{f}} L \frac{\partial t_{\mathrm{f}}}{\partial x} = \overline{h} \, A \left(t_{\mathrm{s}} - t_{\mathrm{f}} \right) , \tag{2.16}$$

where

$$m_{\mathrm{f}} = \rho_{\mathrm{f}} S_{\mathrm{f}} L \,. \tag{2.17}$$

If ρ_{f} is constant with respect to x, this is the total mass of the fluid in the channel. Equation (2.16) corresponds to equation (5.1) in Schmidt and Willmott (1981).

Remark 2.1. Equation (2.16) above was derived for a representative channel of the regenerator unit. However, if we sum both sides of all the similar equations obtained for the channels of a regenerator unit (and consider parameters ρ_{f}, c_{f}, $\bar{h}$ (and the boundary and initial conditions) to be the same for the channels), the result is again (2.16), where now $\dot{m}_{\mathrm{f}}$, m_{f}, A, S_{f} and P respectively represent the total mass rate of flow, the total fluid mass in the unit, the total convective area of the unit, the total fluid cross-sectional area, and the total hydraulic perimeter of the unit. □

Storage medium. The energy equation valid for the solid storage medium or material was obtained from (2.1) by roughly considering the medium as an incompressible ($\beta = 0$) and stationary ($v_x = 0$, $v_y = 0$, $v_z = 0$) fluid. Thus we obtained, with some approximation (see e.g. Carslaw and Jaeger (1959), p. 13)

$$\rho_{\mathrm{s}} c_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \mathrm{div}\mathbf{q}_{\mathrm{s}} = 0 \,, \tag{2.18}$$

where subscript s refers again to the solid material. Now the description with coordinates x, y and z can be considered to be the Lagrangian one as is usual in solid mechanics. The conventional constitutive relation for the heat flux vector is the Fourier law

$$\mathbf{q}_{\mathrm{s}} = -\mathbf{k}_{\mathrm{s}} \bullet \mathrm{grad} t_{\mathrm{s}} \,, \tag{2.19}$$

where $\mathbf{k}_{\mathrm{s}}$ is the thermal conductivity tensor. In this case (2.18) becomes

$$\rho_{\mathrm{s}} c_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} = \mathrm{div}\left(\mathbf{k}_{\mathrm{s}} \bullet \mathrm{grad} t_{\mathrm{s}}\right). \tag{2.20}$$

Again, we did not consider the use of the Fourier law here as a modelling approximation, as this choice is so universally accepted in heat transfer in general. For an isotropic body,

$$\mathbf{q}_{\mathrm{s}} = -k_{\mathrm{s}} \,\mathrm{grad} t_{\mathrm{s}} \,, \tag{2.21}$$

where k_s is the thermal conductivity. If the material is additionally homogeneous and k_s is constant with respect to space, the energy equation (2.18) obtains perhaps its most familiar form

$$\rho_s c_s \frac{\partial t_s}{\partial \tau} = k_s \left(\frac{\partial^2 t_s}{\partial x^2} + \frac{\partial^2 t_s}{\partial y^2} + \frac{\partial^2 t_s}{\partial z^2} \right). \tag{2.22}$$

Equation (2.22) or the more general (2.20), with suitable initial and boundary conditions, can be employed to determine the detailed temperature distribution in the storage material. Schmidt and Willmott (1981), for instance, made use of this to some extent. However, we obtained the simplest model for later use in a similar manner to that used for the fluid.

We considered a control volume of the storage medium depicted in Figure 2.2 (cf. Figure 1.1 (b) and 2.1).

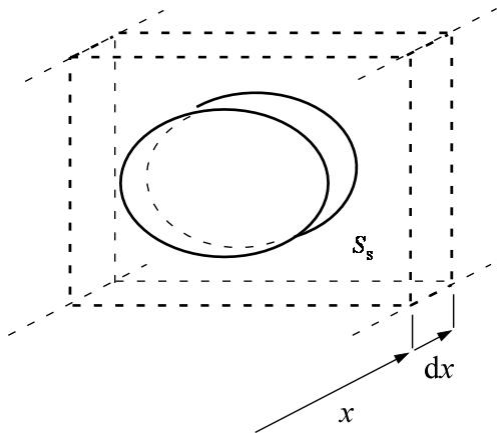


Figure 2.2. Control volume in storage material prism.

Integrating both sides of (2.18) over the control volume gives the equation

$$\int_V \rho_s c_s \frac{\partial t_s}{\partial \tau} \mathrm{d}V + \int_V \mathrm{div}\mathbf{q}_s \mathrm{d}V = 0. \tag{2.23}$$

First, assuming that (A*ssumption 1*)

$$t_{\mathrm{s}}\left(x,y,z,\tau\right)\approx t_{\mathrm{s}}\left(x,\tau\right) \tag{2.24}$$

(again, in reality, some temperature gradient in the cross-sectional directions must exist for the heat to transfer to the fluid), and transforming the divergence term by Gauss's formula produce the form

$$\int_V \rho_{\mathrm{s}} c_{\mathrm{s}} \mathrm{d}V \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \int_V q_{\mathrm{s}} \mathrm{d}S = 0\,, \tag{2.25}$$

where $q_{\mathrm{s}} = \mathbf{n}\bullet\mathbf{q}_{\mathrm{s}}$ is the heat flux out of the control volume on the control surface and **n** is the outward unit normal vector, which is opposite to that used for the fluid control volume. We further assumed that the heat flux gradient in the axial direction could be discarded (*Assumption 2*). As the control volume is infinitesimal in the x-axis direction, (2.25) became

$$\mathrm{d}x\int_{S_{\mathrm{s}}} \rho_{\mathrm{s}} c_{\mathrm{s}} \mathrm{d}S_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \mathrm{d}x\int_P q_{\mathrm{s}}\,\mathrm{d}P = 0 \tag{2.26}$$

or

$$\int_{S_{\mathrm{s}}} \rho_{\mathrm{s}} c_{\mathrm{s}} \mathrm{d}S_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \int_P q_{\mathrm{s}}\,\mathrm{d}P = 0\,. \tag{2.27}$$

Here, the first integral is over the cross-sectional area of the storage medium prism and the second integral over hydraulic perimeter P. It should be noted that application of Gauss's formula also produces, in principle, an integral over the perimeter along the mantle surface. However, this integral vanished, as we assumed, and explained in Chapter 1, that the mantle surfaces were selected so that the heat flux was zero there.

For the solid material, ρ_{s} and c_{s} may be taken as constants, or for more generality, we may assume the dependence (*Assumption 3*)

$$\begin{aligned} \rho_{\mathrm{s}} &\approx \rho_{\mathrm{s}}\left(x\right), \\ c_{\mathrm{s}} &\approx c_{\mathrm{s}}\left(x\right). \end{aligned} \tag{2.28}$$

The explicit dependence on position would indicate an inhomogeneous material in the axial direction. Possible dependence on time did not seem to be of practical importance here. Thus, ρ_{s} and c_{s} were taken outside the area integral in (2.27), to give

$$\rho_s c_s S_s \frac{\partial t_s}{\partial \tau} + P\bar{q}_s = 0\,, \tag{2.29}$$

where S_s is the cross-sectional area of the storage medium prism and

$$\bar{q}_s = \frac{1}{P}\int_P q_s \,\mathrm{d}P \tag{2.30}$$

the average heat flux on the hydraulic perimeter.

We employed the conventional relation

$$\bar{q}_s = \bar{h}\left(t_s - t_f\right). \tag{2.31}$$

It may be noted here that due to the difference in the positive sign definitions, $q_s = -q_f$ and $\bar{q}_s = -\bar{q}_f$. Therefore, formula (2.15):

$$P = A / L\,, \tag{2.32}$$

was still valid. Substituting (2.31) and (2.32) in (2.29) and multiplying the equation with *L* produced the final equation

$$m_s c_s \frac{\partial t_s}{\partial \tau} = \bar{h}\,A\left(t_f - t_s\right), \tag{2.33}$$

where

$$m_s = \rho_s S_s L\,. \tag{2.34}$$

If ρ_s is constant with respect to x, this is the total mass of the storage material in the prism. Equation (2.33) corresponds to equation (5.2) in Schmidt and Willmott (1981).

Remark 2.2. Here we can make a similar observation to that in Remark 2.1. Equation (2.33) above was derived for a representative storage medium prism of the regenerator unit. However, if we sum both sides of all the similar equations obtained for the prisms of a regenerator unit (and consider parameters ρ_s, c_s, $\bar{h}$ (and the boundary and initial conditions) to be the same for the prism), the result is again that of (2.33), only now m_s, A, S_s and P represent the total storage medium mass in the unit, the total convective area of the unit, the total

storage medium cross-sectional area, and the total hydraulic perimeter of the unit, respectively. □

Initial and boundary conditions. Let us review governing equations (2.16) and (2.33):

$$m_{\mathrm{f}} c_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + \dot{m}_{\mathrm{f}} c_{\mathrm{f}} L \frac{\partial t_{\mathrm{f}}}{\partial x} + \bar{h} A t_{\mathrm{f}} - \bar{h} A t_{\mathrm{s}} = 0 , \tag{2.35}$$

and

$$m_{\mathrm{s}} c_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \bar{h} A t_{\mathrm{s}} - \bar{h} A t_{\mathrm{f}} = 0 . \tag{2.36}$$

The problem is determining the unknown functions $t_{\mathrm{f}} = t_{\mathrm{f}}(x,\tau)$ and $t_{\mathrm{s}} = t_{\mathrm{s}}(x,\tau)$. The corresponding space domain is $0 \le x \le L$, when the origin of the x-axis is put on the "left boundary" of the unit. We can obtain the necessary initial and boundary conditions by simple physical reasoning by, for instance, imagining a suitable laboratory experiment with the unit. At $\tau = 0$ (when the phenomenon to be studied began) we assumed the temperature distribution in the storage medium, as known by, for instance, temperature measurements: $t_{\mathrm{s}}(x,0)$ to be given. In the most obvious case this could simply be a constant value. We then started to feed some given fluid with a given mass flow rate $\dot{m}_{\mathrm{f}}(\tau)$ into the unit at the left boundary $x = 0$ ($v_x > 0$) or the right boundary $x = L$ ($v_x < 0$), at a given inlet temperature of $t_{\mathrm{fi}}(\tau)$.

The complicated short initial phase during which the unit was only partially filled with the given fluid was not properly modelled by equations (2.35) and (2.36) (The unit could, for example, have been partly filled by some liquid and partly by some gas.). However, we neglected this initial *filling period* and assumed that in practice it would take no time. Thus we considered $\dot{m}_{\mathrm{f}}(\tau)$ and $t_{\mathrm{fi}}(\tau)$ as given for all $\tau > 0$ quantities. Although the value of $\dot{m}_{\mathrm{f}}$ is given at a boundary, it does not determine a boundary condition in the normal sense of the word, since it appears as a parameter in the field equation (2.35). On the other hand, the given value t_{fi} mathematically determines a proper boundary condition for the unknown fluid temperature field.

The initial and boundary conditions used naturally depended on the application. Here we restricted our main study to counterflow regenerators. We employed a numerical approach — the finite element method — to solve the relevant equations. This discrete formulation differed from that employed

by Schmidt and Willmott (1981) and Willmott (2002) (the finite difference method). Because of this, we first considered the single blow case (named SBC in the Appendices) in some detail. This provided the opportunity to obtain some indication of the accuracy achievable with our formulation, and finally served to solve the regenerator problem case (named REG in the Appendices).

3 SINGLE BLOW CASE

3.1 Description of problem

In the singe blow case, the mathematical solution domain in the x-direction was $0 \le x \le L$ and in the τ-direction, $0 \le \tau \le P$. Thus the solution domain was a rectangle, as shown in Figure 3.1. In fact, many of the details presented in this chapter will appear later in similar forms in Chapter 4, which deals with the counterflow regenerator case. For example, several consecutive problems with rectangular solution domains such as that in Figure 3.1 appear.

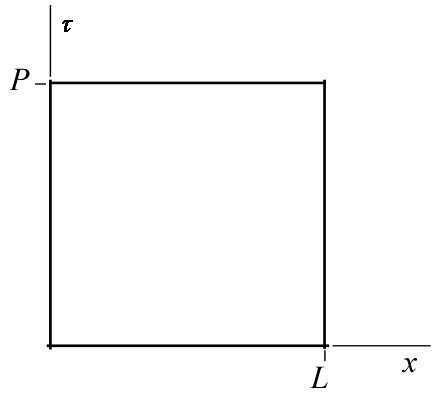


Figure 3.1. Solution domain.

The storage material initially had a constant (in position) temperature t_0. The inflowing fluid had a constant (in time) temperature t_{fi}. Thus, the initial and boundary conditions were, respectively,

$$t_{\mathrm{s}}\left(x,0\right) = t_0\,, \tag{3.1}$$

$$t_{\mathrm{f}}\left(0,\tau\right) = t_{\mathrm{fi}}\,. \tag{3.2}$$

To proceed, we transformed the equations, as in Schmidt and Willmott (1981), into nondimensional forms. The following nondimensional variables were introduced:

nondimensional length

$$\Lambda \equiv \frac{\bar{h}A}{\dot{m}_{\mathrm{f}}\, c_{\mathrm{f}}}, \tag{3.3}$$

nondimensional position coordinate

$$\xi \equiv \Lambda \frac{x}{L}, \tag{3.4}$$

nondimensional time length

$$\Pi = \frac{\bar{h}AP}{m_{\mathrm{s}} c_{\mathrm{s}}}, \tag{3.5}$$

nondimensional time coordinate

$$\eta = \Pi \frac{\tau}{P}, \tag{3.6}$$

nondimensional fluid temperature

$$T_{\mathrm{f}}(\xi,\eta) \equiv \frac{t_{\mathrm{f}}(x,\tau) - t_0}{t_{\mathrm{fi}} - t_0}, \tag{3.7}$$

nondimensional solid temperature

$$T_{\mathrm{s}}(\xi,\eta) \equiv \frac{t_{\mathrm{s}}(x,\tau) - t_0}{t_{\mathrm{fi}} - t_0}. \tag{3.8}$$

Remark 3.1. Schmidt and Willmott (1981) define nondimensional time by

$$\eta \equiv \Pi \frac{1}{P}\left(\tau - x / v_x\right), \tag{3.9}$$

where v_x is the axial velocity of the fluid. This definition is found finally to remove the analogue of the term $m_f c_f \partial t_f / \partial \tau$ in (2.35) to appear in the resulting equation (see Appendix A). However, as discussed at the end of Chapter 2, the filling period, which took the time $\Delta\tau = L / v_x$ is not properly described by the present equations. Furthermore, if we apply formula (3.9) instead of (3.6), and set the initial nondimensional time at $\eta = 0$, the corresponding dimensional time becomes $\tau = x / v_x$. As the initial condition in (3.1) was given at $\tau = 0$, there is a discrepancy. Thus we applied formula (3.6) and discarded the term $m_f c_f \partial t_f / \partial \tau$. It should be noted that this term was also discarded in Section 2.3 in (Schmidt and Willmott 1981), due to its small size. □

The new unknowns T_f and T_s are functions of the coordinates ξ and η: $T_f = T_f(\xi,\eta)$ and $T_s = T_s(\xi,\eta)$. By performing some manipulations, and using definitions (3.3) to (3.8), we arrived at the field equations (the details of the derivation are given in Appendix A):

$$R_f \equiv \frac{\partial T_f}{\partial \xi} - T_s + T_f = 0, \tag{3.10}$$

$$R_s \equiv \frac{\partial T_s}{\partial \eta} - T_f + T_s = 0. \tag{3.11}$$

Notations R refer to field equation residuals. By writing the equations in this way, the presentation of the weak forms used in the finite element formulations which follow becomes transparent. The calculation domain is now a rectangle with the measures Λ and Π (Figure 3.2).

The initial and boundary conditions become, respectively,

$$T_s(\xi,0) = 0, \tag{3.12}$$

$$T_f(0,\eta) = 1. \tag{3.13}$$

The problem has an analytical solution for T_f and T_s. By making slight changes to the expressions given in Larsen (1967), we obtained

$$T_f(\xi,\eta) = 1 - e^{-\eta} \int_0^{\xi} e^{-\alpha} I_0\left[2\sqrt{\eta\alpha}\right] d\alpha, \tag{3.14}$$

$$T_s(\xi,\eta) = e^{-\xi} \int_0^{\eta} e^{-\alpha} I_0\left[2\sqrt{\xi\alpha}\right] d\alpha, \tag{3.15}$$

where I_0 is the zero order modified Bessel function of the first kind. Pointwise values can be evaluated from these expressions using numerical integration by the Mathematica (2014) program. We also obtained an analytical solution directly from field equation (3.11) at $\xi = 0$, where $T_\mathrm{f} = 1$. This gave

$$T_\mathrm{s}(0,\eta) = 1 - \mathrm{e}^{-\eta}. \tag{3.16}$$

Schmidt and Willmott (1981) apply this as an additional boundary condition. However, here we employed only (3.16) to check the correctness of expression (3.15) at $\xi = 0$.

For further purposes, we finally defined two average temperature quantities; one with respect to time and one with respect to space. These are indicated with the tilde and hat overbar notations, respectively. Thus,

$$\tilde{t}(x) = \frac{1}{P}\int_0^P t(x,\tau)\,\mathrm{d}\tau\,, \quad \tilde{T}(\xi) = \frac{1}{\Pi}\int_0^\Pi T(\xi,\eta)\,\mathrm{d}\eta \tag{3.17}$$

and

$$\hat{t}(\tau) = \frac{1}{L}\int_0^L t(x,\tau)\,\mathrm{d}x\,, \quad \hat{T}(\eta) = \frac{1}{\Lambda}\int_0^\Lambda T(\xi,\eta)\,\mathrm{d}\xi\,. \tag{3.18}$$

The definitions can be applied equally for the fluid and the solid, that is, equipped with the subscripts f or s. The average quantities defined above are needed later especially at the boundary lines of the domain.

3.2 Solution method

The conventional approach in time dependent problems is to march the solution in a stepwise manner forward in time. For instance, the solution approach in Schmidt and Willmott (1981) and Willmott (2002) proceeds in this way, using the finite difference method. However, the capacity of present computers makes it possible to solve time-dependent problems alternatively (at least with only one space dimension) as pure boundary value problems. This is the approach applied here. We may note that field equations (3.10) and (3.11), as well as "boundary" conditions (3.12) and (3.13), are of quite a similar nature, and it is thus perhaps natural to treat ξ and η on an equal basis. The new solution domain is covered with a typical uniform mesh, as shown in Figure 3.2. We used rectangular bilinear four-node elements.

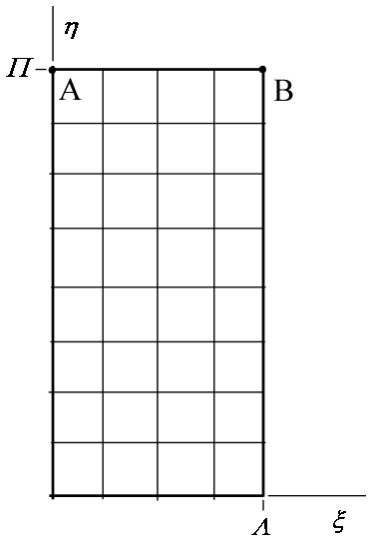


Figure 3.2. 4×8 *finite element mesh.*

The finite element method is based on integral type presentations with the method of weighted residuals. We do not enter deeply into the theory of the finite element method, as the basic ideas are well-documented by, for example, Zienkiewicz and Taylor (1989). The most common versions of the method of weighted residuals are the Galerkin method and the least-squares method. We experimented with two Galerkin-type versions (Formulations I and II) and with a least-squares version (Formulation III). The weak forms corresponding to these formulations are listed below.

Formulation I. The weak form was

$$\int_{\Omega}\left(\delta T_{\mathrm{f}} R_{\mathrm{f}} + \delta T_{\mathrm{s}} R_{\mathrm{s}}\right)\mathrm{d}\Omega = 0\,. \tag{3.19}$$

Formulation II. The weak form was

$$\int_{\Omega}\left(\delta T_{\mathrm{s}} R_{\mathrm{f}} + \delta T_{\mathrm{f}} R_{\mathrm{s}}\right)\mathrm{d}\Omega = 0\,. \tag{3.20}$$

Formulation III. The weak form was

$$\int_{\Omega}\left[\left(\frac{\partial\delta T_{\mathrm{f}}}{\partial\xi}-\delta T_{\mathrm{s}}+\delta T_{\mathrm{f}}\right)\alpha_{\mathrm{f}}R_{\mathrm{f}}+\left(\frac{\partial\delta T_{\mathrm{s}}}{\partial\eta}-\delta T_{\mathrm{f}}+\delta T_{\mathrm{s}}\right)\alpha_{\mathrm{s}}R_{\mathrm{s}}\right]\mathrm{d}\Omega=0\,. \tag{3.21}$$

The integrals are over domain Ω in Figure 3.2. The Galerkin method is normally defined as a method in which the same basis functions used in the approximations of the unknowns are employed as weighting functions. This can also be interpreted to mean that the weighting functions (before discretization) are variations of the unknown functions. The two weighting function options are seen in Formulations I and II. The least squares version was based on the functional

$$\Pi\left(T_{\mathrm{f}},T_{\mathrm{s}}\right)=\frac{1}{2}\int_{\Omega}\left[\alpha_{\mathrm{f}}\left(R_{\mathrm{f}}\right)^{2}+\alpha_{\mathrm{s}}\left(R_{\mathrm{s}}\right)^{2}\right]\mathrm{d}\Omega\,. \tag{3.22}$$

Quantities α_{f} and α_{s} are the given weight factors. As the field equations are of a similar nature, we simply employed $\alpha_{\mathrm{f}}=\alpha_{\mathrm{s}}=1$. Demanding the functional to have a stationary value produces the weak form (3.21).

In all the numerical solutions in this report, we more or less follow the recommendations of references by Roache (2009) and Sinclair et al. (2006) in order to obtain confidence in the results. We used three meshes with increasing densities in every problem case. The meshes were called coarse, medium and fine, and we employ the corresponding subscripts c, m and f, respectively, when referring to the appropriate values. If h_{c} is the typical mesh size of a coarse mesh, the corresponding mesh sizes for the medium and the fine meshes are $h_{\mathrm{m}}=h_{\mathrm{c}}/\lambda$ and $h_{\mathrm{f}}=h_{\mathrm{m}}/\lambda^{2}$, respectively, where λ is the grid refinement factor in Roache (2009) or the scale factor in Sinclair et al. (2006). We always employed the value $\lambda=2$.

The effective convergence rate r for quantity f was obtained from

$$r=\ln\left[\frac{f_{\mathrm{m}}-f_{\mathrm{c}}}{f_{\mathrm{f}}-f_{\mathrm{m}}}\right]/\ln\lambda\,. \tag{3.23}$$

The concept of the Grid Convergence Index (GCI) is discussed at length in Roache (2009). Referring to formula (5.6.1) in Roache (2009):

$$\mathrm{GCI[fine\ grid]}=F_{\mathrm{s}}\frac{\left|e_{\mathrm{fm}}\right|}{\lambda^{r}-1}\,. \tag{3.24}$$

This gives an estimation of the order of the error for a result obtained by the finest mesh. Here

$$e_{\mathrm{fm}} = f_{\mathrm{f}} - f_{\mathrm{m}} \tag{3.25}$$

and if we consider the relative error, we can replace e_{fm} in (3.24) by

$$e_{\mathrm{fmr}} = \frac{f_{\mathrm{f}} - f_{\mathrm{m}}}{f_{\mathrm{f}}}. \tag{3.26}$$

Multiplier F_{s} is the factor of safety and we assumed that $F_{\mathrm{s}} = 1.25$. Further, we call the grid convergence index here simply the error estimate and denote it by $\overline{e}$, or, if a relative error, $\overline{e}_{\mathrm{r}}$. Thus the error estimate expressions were

$$\overline{e} = 1.25 \frac{|e_{\mathrm{fm}}|}{2^r - 1} \tag{3.27}$$

and

$$\overline{e}_{\mathrm{r}} = 1.25 \frac{|e_{\mathrm{fmr}}|}{2^r - 1}. \tag{3.28}$$

Let us define the actual error as

$$e_{\mathrm{an}} = f_{\mathrm{a}} - f_{\mathrm{n}}, \tag{3.29}$$

where f_{a} is the actual analytical exact value and f_{n} the approximate numerical value. Here n can mean c, m or f. Furthermore, the actual relative error is defined as

$$e_{\mathrm{anr}} = \frac{f_{\mathrm{a}} - f_{\mathrm{n}}}{f_{\mathrm{a}}} = \frac{e_{\mathrm{an}}}{f_{\mathrm{a}}}. \tag{3.30}$$

In general we do not know the exact value of f_{a} and thus cannot evaluate either e_{an} or e_{anr}.

In the calculations to follow, we consider the results obtained by the fine mesh as the most accurate, and hopefully $f_{\mathrm{f}} \approx f_{\mathrm{a}}$ (see Remark 3.2). We can always calculate errors e_{fm} and e_{fmr}, and error estimates $\overline{e}$ and $\overline{e}_{\mathrm{r}}$.

The error estimates are designed so that normally the exact error with the fine mesh e_{af} is bounded by

$$e_{\mathrm{af}} < \overline{e} \tag{3.31}$$

or by

$$e_{\mathrm{afr}} < \overline{e}_{\mathrm{r}} \,. \tag{3.32}$$

Thus, some confidence in the numerical results can be gained. However, the two following applications are such that analytical solutions exist. Thus we were also able to calculate the actual errors e_{af} and e_{afr} and see whether inequalities (3.31) and (3.32) are satisfied. Further, we were able to draw some conclusions about the behaviour of the three formulations.

Remark 3.2. One can usually obtain more accurate values than f_{f} by applying the Richardson extrapolation, discussed thoroughly by Roache (2009). However, this manipulation was not considered necessary in the applications of this report. □

3.3 Results

3.3.1 First application

We consider Example 2.1 in Schmidt and Willmott (1981) as the first application. In this, the dimensionless problem measures were

$$\Lambda = 1.847 \,, \quad \Pi = 3.78 \,. \tag{3.33}$$

Coarse, medium and fine meshes consisted of 4×8, 8×16 and 16×32 elements, respectively. The coarse mesh is shown in Figure 3.2.

We compare the finite element solution with the four “exact” results obtained by using expressions (3.14) and (3.15). The value of the solid temperature at point A (Figure 3.2) was

$$T_{\mathrm{s}}\left(0, \Pi\right) = 0.977177 \,, \tag{3.34}$$

the values of fluid and solid material temperatures at point B (Figure 3.2) were

$$T_{\mathrm{f}}\left(\Lambda,\Pi\right)=0.853587\,,\quad T_{\mathrm{s}}\left(\Lambda,\Pi\right)=0.727\,046 \tag{3.35}$$

and the average solid material temperature on line AB (see (3.18)) was

$$\hat{T}_{\mathrm{s}}\left(\Pi\right)\equiv\frac{1}{\Lambda}\int_{0}^{\Lambda}T_{\mathrm{s}}\left(\xi,\Pi\right)\mathrm{d}\xi=0.863840\,. \tag{3.36}$$

This last type of quantity is of importance in the regenerator problems, and will be considered later. As only pointwise solid temperature values were available, result (3.36) was obtained by employing Simpson's numerical integration from 17 equidistant temperature value data. The same value (3.36) was already obtained with nine data points, so the number should, in practice, be exact.

The calculation results are collected in the following four tables 3.1, 3.2, 3.3 and 3.4.

Table 3.1. Results for T_{s} at A.

	Formulation I	Formulation II	Formulation III
e_{acr}	0.039%	0.670%	0.320%
e_{amr}	0.011%	0.317%	0.184%
e_{afr}	0.003%	0.155%	0.102%
r	1.863	1.126	0.615
$\overline{e}_{\mathrm{r}}$	0.004%	0.172%	0.205%

Table 3.2. Results for T_f at B.

	Formulation I	Formulation II	Formulation III
e_{acr}	0.190%	0.190%	-0.305%
e_{amr}	0.048%	0.048%	-0.132%
e_{afr}	0.012%	0.012%	-0.061%
r	2.005	2.004	1.291
$\overline{e}_r$	0.015%	0.015%	0.061%

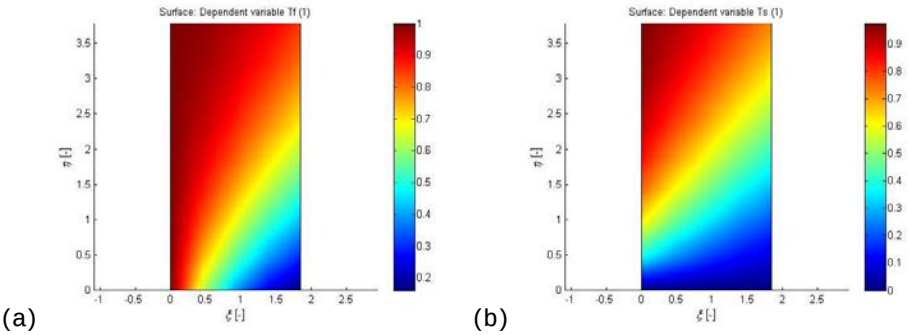
Table 3.3. Results for T_s at B.

	Formulation I	Formulation II	Formulation III
e_{acr}	0.194%	0.202%	-0.226%
e_{amr}	0.048%	0.048%	-0.081%
e_{afr}	0.012%	0.012%	-0.033%
r	1.999	2.084	1.579
$\overline{e}_r$	0.015%	0.014%	0.030%

Table 3.4. Results for $\hat{T}_s(\Pi)$.

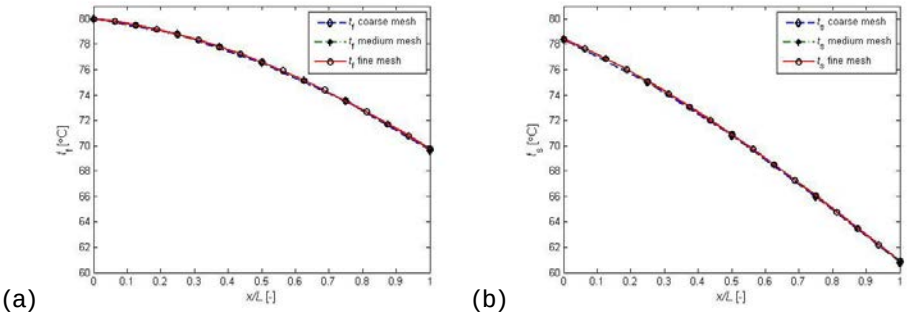
	Formulation I	Formulation II	Formulation III
e_{acr}	0.215%	0.267%	0.267%
e_{amr}	0.054%	0.067%	0.004%
e_{afr}	0.014%	0.017%	0.001%
r	1.990	2.003	6.693
$\overline{e}_r$	0.017%	0.021%	0.000%

Temperature distributions in Ω by Formulation I are shown in Figure 3.3. Dimensional temperature distributions at $\tau = P$ are shown in Figure 3.4.



(a) (b)

Figure 3.3. (a) Nondimensional fluid T_f and (b) solid material T_s temperature distributions with a 4×8 coarse mesh.



(a) (b)

Figure 3.4. (a) Dimensional fluid t_f and (b) solid material t_s temperature at $\tau = P$ with coarse, medium and fine meshes. Fluid inflow temperature $t_f = 80\,°C$ and initial solid temperature $t_s = 10\,°C$.

3.3.2 Second application

Here the dimensionless problem measures of the first application were reversed:

$$\Lambda = 3.78\,, \quad \Pi = 1.847\,. \tag{3.37}$$

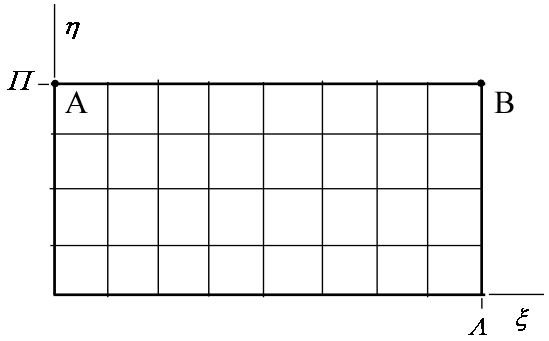


Figure 3.5. 8×4 *finite element mesh.*

Coarse, medium and fine meshes consisted of 8×4, 16×8 and 32×16 elements, respectively. The coarse mesh is shown in Figure 3.5.

We compared the finite element solution again with the four "exact" results obtained using expressions (3.14) and (3.15). The value of the solid temperature at point A (Figure 3.3) was

$$T_{\mathrm{s}}\left(0,\Pi\right)=0.842\,290\,, \tag{3.38}$$

the values of fluid and solid material temperatures at point B (Figure 3.3) were

$$T_{\mathrm{f}}\left(\Lambda,\Pi\right)=0.272\,954\,,\quad T_{\mathrm{s}}\left(\Lambda,\Pi\right)=0.146\,413 \tag{3.39}$$

and the average solid material temperature on line AB (see (3.18)) was

$$\hat{T}_{\mathrm{s}}\left(\Pi\right)\equiv\frac{1}{\Lambda}\int_0^{\Lambda}T_{\mathrm{s}}\left(\xi,\Pi\right)\mathrm{d}\xi=0.422\,093\,. \tag{3.40}$$

The calculation results are presented in the following four tables 3.5, 3.6, 3.7 and 3.8.

Table 3.5. Results for T_s at A.

	Formulation I	Formulation II	Formulation III
e_{acr}	0.285%	3.078%	-0.202%
e_{amr}	0.075%	1.419%	-0.011%
e_{afr}	0.019%	0.695%	0.018%
r	1.908	1.196	2.712
$\overline{e}_r$	0.025%	0.706%	0.007%

Table 3.6. Results for T_f at B.

	Formulation I	Formulation II	Formulation III
e_{acr}	-0.516%	-0.539%	0.601%
e_{amr}	-0.129%	-0.129%	0.216%
e_{afr}	-0.032%	-0.032%	0.087%
r	1.999	2.084	1.580
$\overline{e}_r$	0.040%	0.037%	0.081%

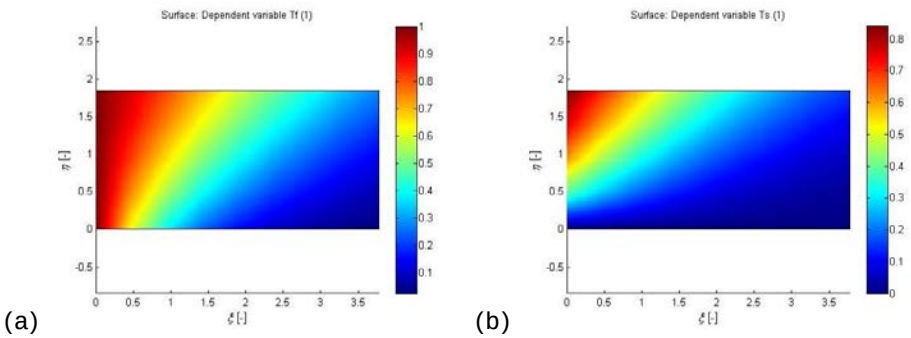
Table 3.7. Results for T_s at B.

	Formulation I	Formulation II	Formulation III
e_{acr}	-1.110%	-1.110%	1.777%
e_{amr}	-0.277%	-0.277%	0.769%
e_{afr}	-0.070%	-0.070%	0.357%
r	2.005	2.004	1.291
$\overline{e}_r$	0.086%	0.086%	0.357%

Table 3.8. Results for $\hat{T}_{\mathrm{s}}\left(\Pi\right)$.

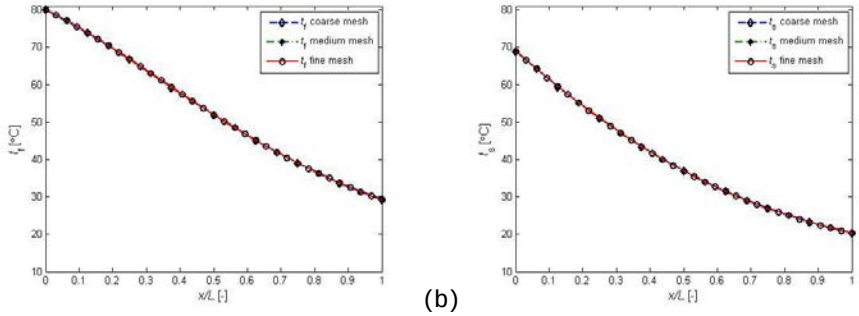
	Formulation I	Formulation II	Formulation III
e_{acr}	-0.159%	0.042%	-0.131%
e_{amr}	-0.040%	0.008%	-0.032%
e_{afr}	-0.010%	0.002%	-0.008%
r	1.994	2.405	2.050
$\overline{e}_{\mathrm{r}}$	0.013%	0.002%	0.010%

Temperature distributions in Ω by Formulation I are shown in Figure 3.6. Dimensional temperature distributions at $\tau = P$ are shown in Figure 3.7.



(a) (b)

Figure 3.6. (a) Nondimensional fluid T_{f} *and (b) solid material* T_{s} *temperature distributions with a* 8×4 *coarse mesh.*



(a) (b)

Figure 3.7. (a) Dimensional fluid t_{f} *and (b) solid material* t_{s} *temperature at* $\tau = P$ *with coarse, medium and fine meshes. Fluid inflow temperature* $t_{\mathrm{f}} = 80\,^{\circ}\mathrm{C}$ *and initial solid temperature* $t_{\mathrm{s}} = 10\,^{\circ}\mathrm{C}$*.*

3.4 Discussion

The errors by all the three formulations must be considered very small with the fine mesh. The magnitude of the maximum relative (percentage) error was under 0.7%. In most cases, the errors were much smaller. Formulations I and II seem to be somewhat more consistent than formulation III with respect to the value of the effective convergence rate, which is in general close to 2, indicating second order methods. Compared to the results of Formulations I and II, Formulation I seems to give somewhat smaller errors. To continue with only one formulation, the choice was thus to use Formulation I, for which the magnitude of the maximum relative (percentage) error was below 0.07%. It is interesting to see, considering the values of e_{afr} and $\overline{e}_{\mathrm{r}}$ in the tables, that inequality (3.32) here was always satisfied by Formulation I. This created confidence in later results, in which actual errors could not be directly calculated, as exact solutions were not known.

4 REGENERATOR PROBLEM

4.1 Description of problem

In a counterflow regenerator, *hot and cold periods* follow each other cyclically. Equations (2.35) and (2.36) can be applied for both periods. However, suitable notations must be used to discern between the periods. Following Schmidt and Willmott (1981), we used a single prime to signify the hot period, and double primes to signify the cold period. As the names indicate, inflow temperature t'_{fi} of the fluid during the hot period is higher than inflow temperature t''_{fi} of the fluid during the cold period. The periods here had fixed time lengths P' and P'' so that the length of one total cycle was $P'+P''$. We considered only the simplest case, in which $\dot{m}'_{\mathrm{f}}$, t'_{fi}, $\dot{m}''_{\mathrm{f}}$ and t''_{fi} were constants with respect to time. We tried to achieve the periodic solution by going through enough cycles beginning with arbitrary given distribution $t_{\mathrm{s}}(x,0)$.

In the terminology of Schmidt and Willmott (1981), this approach is an open method, as opposed to the closed methods. The process began with the hot period taking the flow direction in the positive x-axis direction, so that the boundary condition was given at $x=0$. The temperature distribution in the storage medium $t'_{\mathrm{s}}(x,P')$ obtained at the end of the hot period was used as the initial condition for the cold period. In reality, there exists between the hot and cold period (as well between the cold and hot period) some kind of short filling period, until the governing equations are again valid.

As already commented in Remark 3.1, we ignored this and started immediately after $\tau=P'$ the cold period analysis. Thus the boundary condition was given at $x=L$. Velocity component v''_x was negative in the general formulae of Chapter 2 and thus mass flow rate $\dot{m}''_{\mathrm{f}}$ was actually negative. However, we did not want to operate with negative rates, so $\dot{m}''_{\mathrm{f}}$ *was defined as positive*. Thus when field equation (2.35) was applied during the cold period, we had to formally use the relation $\dot{m}_{\mathrm{f}}=-\dot{m}''_{\mathrm{f}}$. The temperature distribution in the storage medium $t''_{\mathrm{s}}(x,P'')$, obtained at the end of the cold period, was used again as the new initial condition for the new hot period etc. Based on the above, we next recorded the governing problems for the hot and cold periods separately. However, we preferred to immediately employ the dimensionless formulation, as in Section 3.1. We have (see Remark 4.1)

nondimensional lengths

$$\Lambda' \equiv \frac{\bar{h}'A}{\dot{m}_{\mathrm{f}}' c_{\mathrm{f}}'}, \quad \Lambda'' \equiv \frac{\bar{h}''A}{\dot{m}_{\mathrm{f}}'' c_{\mathrm{f}}''}, \tag{4.1}$$

nondimensional position coordinates

$$\xi' \equiv \Lambda' \frac{x}{L}, \quad \xi'' \equiv \Lambda'' \frac{x}{L}, \tag{4.2}$$

nondimensional periods

$$\Pi' = \frac{\bar{h}'AP'}{m_{\mathrm{s}} c_{\mathrm{s}}'}, \quad \Pi'' = \frac{\bar{h}''AP''}{m_{\mathrm{s}} c_{\mathrm{s}}''} \tag{4.3}$$

nondimensional time coordinates

$$\eta' = \Pi' \frac{\tau}{P'}, \quad \eta'' = \Pi'' \frac{\tau}{P''}, \tag{4.4}$$

nondimensional fluid temperatures

$$T_{\mathrm{f}}'(\xi',\eta') = \frac{t_{\mathrm{f}}'(x,\tau) - t_{\mathrm{fi}}''}{t_{\mathrm{fi}}' - t_{\mathrm{fi}}''}, \quad T_{\mathrm{f}}''(\xi'',\eta'') = \frac{t_{\mathrm{f}}''(x,\tau) - t_{\mathrm{fi}}''}{t_{\mathrm{fi}}' - t_{\mathrm{fi}}''}, \tag{4.5}$$

nondimensional solid temperatures

$$T_{\mathrm{s}}'(\xi',\eta') = \frac{t_{\mathrm{s}}'(x,\tau) - t_{\mathrm{fi}}''}{t_{\mathrm{fi}}' - t_{\mathrm{fi}}''}, \quad T_{\mathrm{s}}''(\xi'',\eta'') = \frac{t_{\mathrm{s}}''(x,\tau) - t_{\mathrm{fi}}''}{t_{\mathrm{fi}}' - t_{\mathrm{fi}}''}. \tag{4.6}$$

Schmidt and Willmott (1981) and Willmott (2002) call the lengths of the solution domain in the space direction (4.1) *reduced lengths* and the corresponding measures in the time direction (4.3) *reduced periods*.

Remark 4.1. In Schmidt and Willmott (1981), coordinate ξ'' is defined to run in the opposite direction to that of ξ', that is, in the direction of the gas flow. Here, as seen in (4.2), ξ' and ξ'' were assumed to grow in the same direction. From this it follows that when we took $\dot{m}_{\mathrm{f}}''$ to be positive, sign changes appeared in (4.12). It should be noted that the dimensionless temperature definitions were

similar to those in the single blow case in Section 3.1. Only the constant temperatures t_{fi} and t_0 in the denominator were replaced by the constant temperatures t'_{fi} and t''_{fi}, respectively. Further, it should be noted that time was measured as starting from zero for each period. □

Hot period. The functions to be determined were

$$\begin{aligned} &T'_{\mathrm{f}} = T'_{\mathrm{f}}\left(\xi', \eta'\right), \\ &\qquad\qquad 0 \le \xi' \le \Lambda', \quad 0 \le \eta' \le \Pi' \\ &T'_{\mathrm{s}} = T'_{\mathrm{s}}\left(\xi', \eta'\right). \end{aligned} \tag{4.7}$$

The corresponding field equations were

$$\frac{\partial T'_{\mathrm{f}}}{\partial \xi'} - T'_{\mathrm{s}} + T'_{\mathrm{f}} = 0, \tag{4.8}$$

$$\frac{\partial T'_{\mathrm{s}}}{\partial \eta'} - T'_{\mathrm{f}} + T'_{\mathrm{s}} = 0 \tag{4.9}$$

with the boundary condition

$$T_{\mathrm{f}}{}'\left(0, \eta'\right) = 1. \tag{4.10}$$

Cold period. The functions to be determined were

$$\begin{aligned} &T''_{\mathrm{f}} = T_{\mathrm{f}}{}''\left(\xi'', \eta''\right), \\ &\qquad\qquad 0 \le \xi'' \le \Lambda'', \quad 0 \le \eta'' \le \Pi'' \\ &T''_{\mathrm{s}} = T''_{\mathrm{s}}\left(\xi'', \eta''\right). \end{aligned} \tag{4.11}$$

The corresponding field equations were (see Remark 4.1)

$$\frac{\partial T_{\mathrm{f}}{}''}{\partial \xi''} + T''_{\mathrm{s}} - T_{\mathrm{f}}{}'' = 0, \tag{4.12}$$

$$\frac{\partial T''_{\mathrm{s}}}{\partial \eta''} - T_{\mathrm{f}}{}'' + T''_{\mathrm{s}} = 0, \tag{4.13}$$

with the boundary condition

$$T_{\mathrm{f}}''\left(\Lambda'',\eta''\right)=0\,. \tag{4.14}$$

The calculations for the hot period and for the cold period were performed separately, and coupling between them become as initial conditions through the storage material temperature distribution at the ends of the periods. Thus we had

$$t_{\mathrm{s}}'\left(x,0\right)=t_{\mathrm{s}}''\left(x,P''\right), \tag{4.15}$$

$$t_{\mathrm{s}}''\left(x,0\right)=t_{\mathrm{s}}'\left(x,P'\right), \tag{4.16}$$

at the beginning of the hot and cold periods, respectively. We wanted to express these relations using dimensionless quantities. From definitions (4.6)

$$t_{\mathrm{s}}'(x,0)=\left(t_{\mathrm{fi}}'-t_{\mathrm{fi}}''\right)T_{\mathrm{s}}'(\xi',0)+t_{\mathrm{fi}}'', \tag{4.17}$$

$$t_{\mathrm{s}}''\left(x,P''\right)=\left(t_{\mathrm{fi}}'-t_{\mathrm{fi}}''\right)T_{\mathrm{s}}''(\xi'',\Pi'')+t_{\mathrm{fi}}'', \tag{4.18}$$

$$t_{\mathrm{s}}''\left(x,0\right)=\left(t_{\mathrm{fi}}'-t_{\mathrm{fi}}''\right)T_{\mathrm{s}}''(\xi'',0)+t_{\mathrm{fi}}'', \tag{4.19}$$

$$t_{\mathrm{s}}'\left(x,P'\right)=\left(t_{\mathrm{fi}}'-t_{\mathrm{fi}}''\right)T_{\mathrm{s}}'(\xi',\Pi')+t_{\mathrm{fi}}''. \tag{4.20}$$

In addition, from (4.2),

$$\frac{x}{L}=\frac{\xi'}{\Lambda'}=\frac{x}{L}=\frac{\xi''}{\Lambda''} \tag{4.21}$$

so

$$\xi'=\frac{\Lambda'}{\Lambda''}\xi''\,,\quad \xi''=\frac{\Lambda''}{\Lambda'}\xi'\,. \tag{4.22}$$

When (4.17) to (4.20) and (4.22) were applied in (4.15) and (4.16), we arrived at the rules

$$T_s'(\xi',0) = T_s''(\frac{\Lambda''}{\Lambda'}\xi', \Pi''), \tag{4.23}$$

$$T_s''(\xi'',0) = T_s'(\frac{\Lambda'}{\Lambda''}\xi'', \Pi') \tag{4.24}$$

to be used as initial conditions at the beginning of the hot and cold periods, respectively.

At the start of the first hot period, we can take, for example,

$$T_s'(\xi',0) = \frac{1}{2}(1+0) = \frac{1}{2}, \tag{4.25}$$

that is, the mean value of the dimensionless fluid inflow temperatures. The computational situation is described schematically in Figure 4.1.

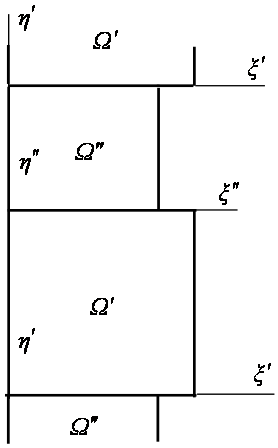


Figure 4.1. Consecutive solution domains.

4.2 Thermal ratio

We quote Schmidt and Willmott (1981, p. 112): "The effectiveness of regenerator behavior is measured in terms of the thermal ratio η_{REG}. This is defined to be the ratio of the actual heat transfer rate to the thermodynamically limited maximum obtainable heat transfer rate in a counterflow regenerator of infinite heat transfer area. Could this maximum

rate be achieved, the temperature of the gas leaving the regenerator in hot/cold period would be equal to the entrance temperature."

The definition formulae from Schmidt and Willmott (1981), with some changes in notation are

$$\eta'_{\mathrm{REG}} = \frac{t'_{\mathrm{fi}} - \tilde{t}'_{\mathrm{f}}(L)}{t'_{\mathrm{fi}} - t''_{\mathrm{fi}}} = 1 - \tilde{T}'_{\mathrm{f}}\left(\Lambda'\right), \tag{4.26}$$

$$\eta''_{\mathrm{REG}} = \frac{\tilde{t}''_{\mathrm{f}}(0) - t''_{\mathrm{fi}}}{t'_{\mathrm{fi}} - t''_{\mathrm{fi}}} = \tilde{T}''_{\mathrm{f}}(0). \tag{4.27}$$

Here, the terms with the tilde overbars are the time averages at the outlet boundaries, see (3.17). As the latter forms in (4.26) and (4.27) are perhaps not immediately obvious, a derivation is given in Appendix B.

We wanted to express the thermal ratios using the solid temperatures, as these appeared naturally at the beginning and the end of the periods. As we did not find a derivation for this in the literature, we present one derivation in Appendix C. The end results were thus

$$\eta'_{\mathrm{REG}} = \frac{\Lambda'}{\Pi'}\left[\hat{T}'_{\mathrm{s}}\left(\Pi'\right) - \hat{T}'_{\mathrm{s}}(0)\right], \tag{4.28}$$

$$\eta''_{\mathrm{REG}} = \frac{\Lambda''}{\Pi''}\left[\hat{T}''_{\mathrm{s}}(0) - \hat{T}''_{\mathrm{s}}\left(\Pi''\right)\right]. \tag{4.29}$$

4.3 Numerical solution

As discussed in Section 3.3, the calculations were performed using Formulation I. The weak form models with global definitions, geometry, mesh, field equations and boundary conditions were built with the COMSOL Multiphysics 4.3 program. The models were saved as M-files for solutions with COMSOL 4.3 using the MATLAB program (MATLAB R2012a). In addition, short MATLAB codes were written for solutions and post processing results.

4.4 Program description

The program initial data consisted of the following input values: L, P', P'', t'_{fi}, t''_{fi}, m_{s}, A, $\bar{h}'$, $\bar{h}''$, c'_{f}, c''_{f}, c'_{s}, c''_{s}, $\dot{m}'_{\mathrm{f}}$, $\dot{m}''_{\mathrm{f}}$, t'_{fi} and t''_{fi}. From these, domain sizes, Λ', Π', Λ'', Π'', for example, were evaluated.

A coarse uniform mesh is defined by giving the number of elements n_ξ and n_η in the ξ- and η- directions, respectively. The elements were of the four-node bilinear rectangular type. The medium mesh was obtained by doubling the number of elements. The fine mesh was obtained with one more doubling.

The program proceeded by solving the equations consecutively in hot and cold periods until the error limit $|\delta'| \wedge |\delta''| < 0.001$ was reached. The error was checked during each period with the previous ($\hat{T}'_{\mathrm{s,old}}(\Pi')$ and $\hat{T}''_{\mathrm{s,old}}(\Pi'')$) and the present ($\hat{T}'_{\mathrm{s,new}}(\Pi')$ and $\hat{T}''_{\mathrm{s,new}}(\Pi'')$) values, using equations

$$\delta' = \frac{\hat{T}'_{\mathrm{s,old}} - \hat{T}'_{\mathrm{s,new}}}{\hat{T}'_{\mathrm{s,old}}}, \tag{4.30}$$

$$\delta'' = \frac{\hat{T}''_{\mathrm{s,old}} - \hat{T}''_{\mathrm{s,new}}}{\hat{T}''_{\mathrm{s,old}}}. \tag{4.31}$$

The program proceeded by evaluating results with three consecutive meshes, starting with a coarse mesh. Some example results are presented in Chapter 4.5. Dimensional fluid t_{f}, solid material t_{s}, temperature at $\tau = P'$ with coarse, medium and fine meshes were calculated at fluid hot inflow temperature $t_{\mathrm{fi}} = 80\,^{\circ}\mathrm{C}$ and cold inflow temperature $t_0 = 10\,^{\circ}\mathrm{C}$. We present the number of total cycles needed for convergence, thermal ratios η'_{REG} and η''_{REG}, the effective convergence rate r, and relative error estimate $\bar{e}_{\mathrm{r}}$, as well as dimensional fluid t_{f} storage material t_{s} and temperatures at $\tau = P'$ with coarse, medium and coarse meshes. The COMSOL Multiphysics models and MATLAB codes used are given in Appendices D, E and F, respectively.

4.5 Applications

4.5.1 First application

We took a symmetrical case with

$$\Lambda' = 10\,, \quad \Pi' = 20\,, \qquad \Lambda'' = 10\,, \quad \Pi'' = 20\,. \tag{4.32}$$

Coarse, medium and fine meshes consisted of 4×8, 8×16 and 16×32 elements, respectively. The coarse mesh was similar to that in Figure 3.2. The number of total cycles needed for convergence was three for each mesh density. The behaviour of the thermal ratios is given in Table 4.1.

Table 4.1. Thermal ratio η'_{REG} and η''_{REG}.

	Numerical solution in reference*)	Numerical solution (4×8)	Numerical solution (8×16)	Numerical solution (16×32)
η'_{REG}	0.494	0.49268	0.49332	0.49350
η''_{REG}	0.494	0.49268	0.49332	0.49350

*) Schmidt and Willmott (1981)

Temperature distributions in Ω are shown in Figure 4.2, and dimensional temperature distributions at $\tau = P'$ in Figure 4.3.

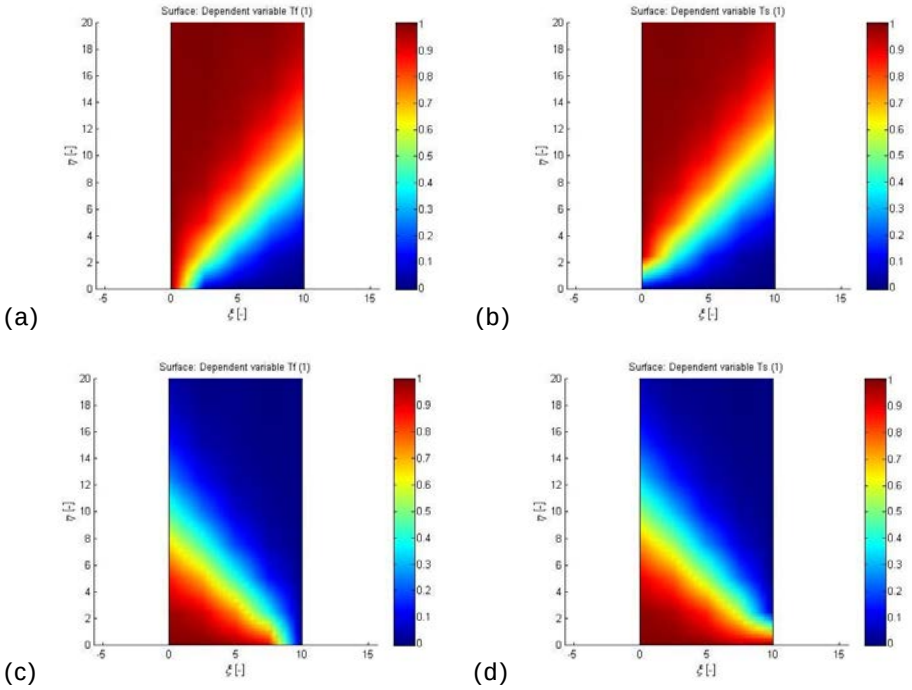


Figure 4.2. Nondimensional fluid T_f *and solid material* T_s *temperature distribution with a* 4×8 *coarse mesh during the hot ((a) and (b)), and the cold period ((c) and (d)).*

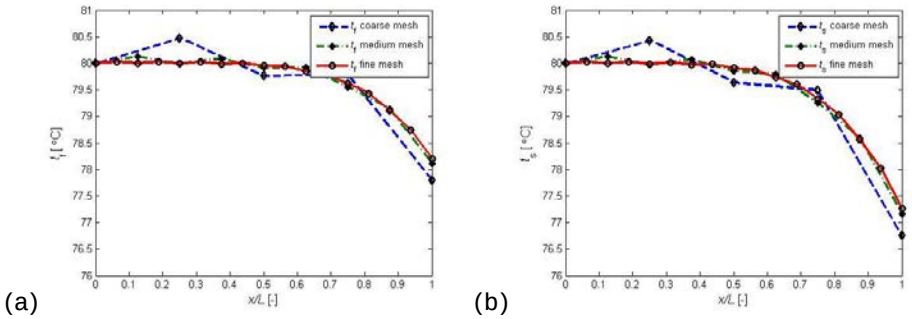


Figure 4.3. (a) Dimensional fluid t_f *and (b) solid material* t_s *temperature at* $\tau = P'$ *with coarse, medium and fine meshes. Fluid hot inflow temperature* $t'_{fi} = 80\,°C$ *and cold inflow temperature* $t''_{fi} = 10\,°C$*.*

The effective convergence rate r (3.23) and relative (percentage) error $\overline{e}_{\mathrm{r}}$ (3.28) calculated for η'_{REG} and η''_{REG} are shown in Table 4.2.

Table 4.2. Effective convergence rate for thermal ratios and relative (percentage) error estimates.

	η'_{REG}	η''_{REG}
r	1.79863	1.79862
$\overline{e}_{\mathrm{r}}$	0.01874%	0.01874%

4.5.2 Second application

We used Example 5.5 in Schmidt and Willmott (1981). The data is as follows:

$$\begin{aligned} &A = 31.40\,\mathrm{m}^2,\ m_{\mathrm{s}} = 4898.4\,\mathrm{kg},\ c_{\mathrm{s}} = 0.92\,\mathrm{kJ/kg\,K},\\ &c_{\mathrm{f}} = 1.011\,\mathrm{kJ/kg\,K},\ \dot{m}'_{\mathrm{f}} = 0.156\,\mathrm{kg/s},\ \dot{m}''_{\mathrm{f}} = 0.078\,\mathrm{kg/s},\\ &\overline{h}' = 50.23\,\mathrm{W/m^2K},\ \overline{h}'' = 25.11\,\mathrm{W/m^2K},\ P' = 3600\,\mathrm{s},\\ &P'' = 10800\,\mathrm{s}. \end{aligned} \tag{4.33}$$

From this, we obtained

$$\begin{aligned} &\Lambda' = 10.0,\quad \Pi' = 1.26,\\ &\Lambda'' = 10.0,\quad \Pi'' = 1.88. \end{aligned} \tag{4.34}$$

Coarse, medium and fine meshes consisted of 8×4, 16×8 and 32×16 elements, respectively. The number of total cycles needed for convergence was 25 with each mesh density. Table 4.3 presents the behaviour of the thermal ratio.

Table 4.3. Thermal ratio η'_{REG} and η''_{REG}.

	Numerical solution in reference*)	Numerical solution (25 × 4)	Numerical solution (50 × 8)	Numerical solution (100 × 16)
η'_{REG}	0.947	0.96111	0.94956	0.94760
η''_{REG}	0.635	0.64519	0.63746	0.63616

*) Schmidt and Willmott (1981)

Temperature distributions in Ω are shown in Figure 4.4 and dimensional temperature distributions at $\tau = P'$ in Figure 4.5.

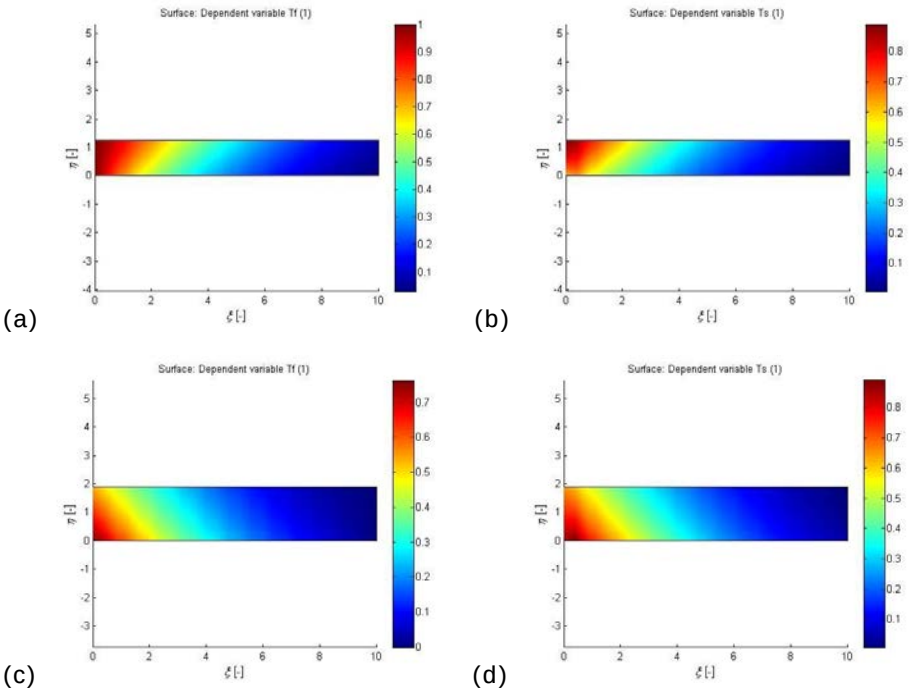
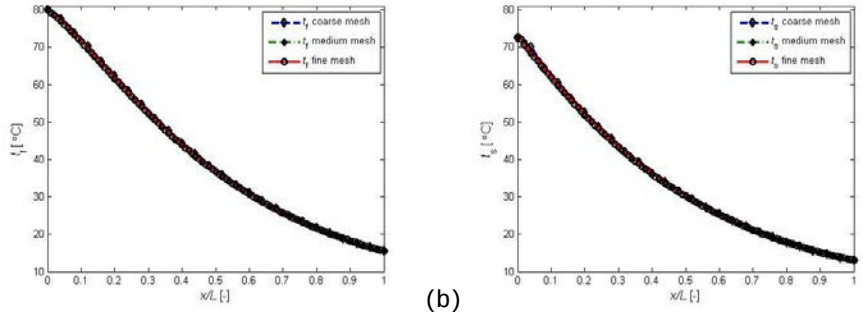


Figure 4.4. Nondimensional fluid T_{f} and solid material T_{s} temperature distribution with a 4×8 coarse mesh during the hot ((a) and (b)), and the cold period ((c) and (d)).



(a) (b)

Figure 4.5. (a) Dimensional fluid t_f and (b) solid material t_s temperature at $\tau = P'$ with coarse, medium and fine meshes. Fluid hot inflow temperature $t'_{fi} = 80\,°C$ and cold inflow temperature $t''_{fi} = 10\,°C$.

Effective convergence rate r and relative (percentage) error $\overline{e}_r$ for η'_{REG} and η''_{REG} are shown in Table 4.4.

Table 4.4. Effective convergence rate for thermal ratios and relative (percentage) error estimates.

	η'_{REG}	η''_{REG}
r	2.55493	2.57525
$\overline{e}_r$	0.05317%	0.05138%

4.5.3 Third application

We modified the first case slightly so that ratio $\Lambda'/\Lambda'' = \Pi'/\Pi'' = 0.83333$ and

$$\Lambda' = 1.6\,,\quad \Pi' = 3.4\,,$$
$$\Lambda'' = 1.92\,,\quad \Pi'' = 4.08\,. \tag{4.35}$$

Coarse, medium and fine meshes consisted of 4×8, 8×16 and 16×32 elements, respectively. The number of total cycles needed for convergence was four for each mesh density. The calculation results are presented in Table 4.5.

Table 4.5. Thermal ratio η'_{REG} and η''_{REG}.

	Numerical solution (4 × 8)	Numerical solution (8 × 16)	Numerical solution (16 × 32)
η'_{REG}	0.36210	0.36271	0.36279
η''_{REG}	0.36210	0.36271	0.36279

Temperature distributions in Ω are shown in Figure 4.6 and dimensional temperature distributions at $\tau = P'$ in Figure 4.7.

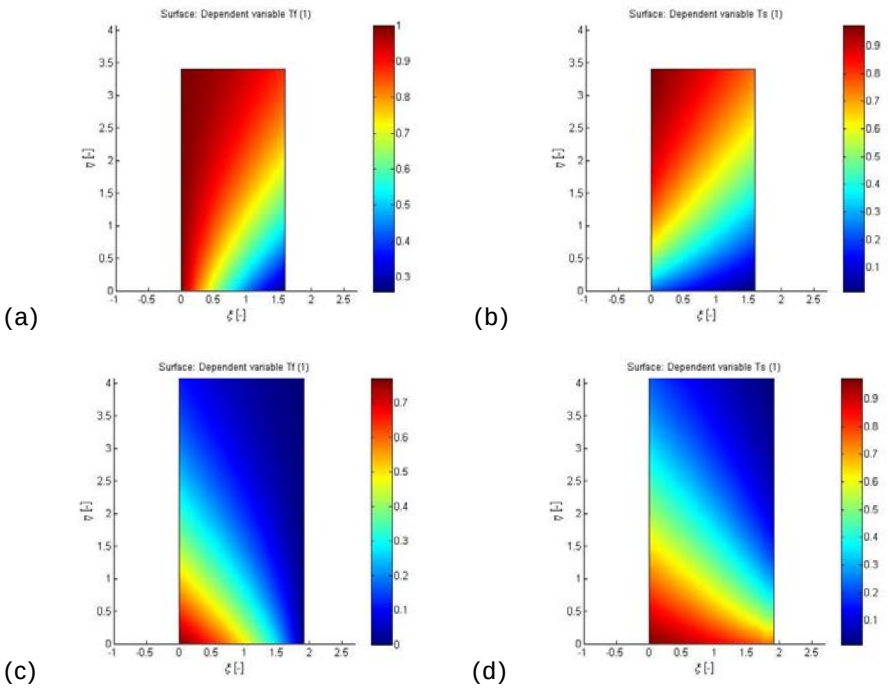
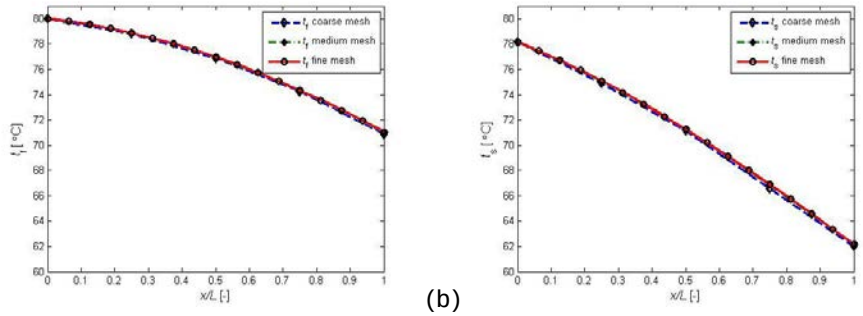


Figure 4.6. Nondimensional fluid T_{f} and solid material T_{s} temperature distribution with a 4×8 coarse mesh during the hot ((a) and (b)), and the cold period ((c) and (d)).



(a) (b)

Figure 4.7. (a) Dimensional fluid t_{f} and (b) solid material t_{s} temperature at $\tau = P'$ with coarse, medium and fine meshes. Fluid hot inflow temperature $t'_{\mathrm{fi}} = 80\,^{\circ}\mathrm{C}$ and cold inflow temperature $t''_{\mathrm{fi}} = 10\,^{\circ}\mathrm{C}$.

Effective convergence rate r and relative (percentage) error $\overline{e}_{\mathrm{r}}$ for η'_{REG} and η''_{REG} are shown in Table 4.6.

Table 4.6. Effective convergence rate for thermal ratios and relative (percentage) error estimates.

	η'_{REG}	η''_{REG}
r	2.84466	2.84835
$\overline{e}_{\mathrm{r}}$	0.00469%	0.00466%

4.6 Discussion

From a practical point of view, the accuracy of the results obtained by the finest meshes is more than adequate, considering the assumptions used in the modelling. As expected, we found that the error limit affected the number of total cycles needed for convergence and the accuracy of the solution. For example in the second case, the number of total cycles needed for convergence was 2, 15 and 25 with error limits of 0.1, 0.01 and 0.001, respectively.

According to (4.28) and (4.29), ratios Λ'/Π' and Λ''/Π'' affect thermal ratios η'_{REG} and η''_{REG}. This information can be utilized in the design of heat exchangers. In practice, e.g. in the heat recovery of a ventilation system that

transfers exhaust air to heat the supply air, the aim is to achieve a high thermal ratio during a hot period. Therefore, the ratio $\Lambda'/\Lambda'' = \Pi'/\Pi''$ is typical in a heat recovery application.

5 CONCLUSIONS

The classical regenerator problem has been numerically solved in a new way. Instead of marching in time, the hot and cold periods were solved simply as boundary value problems in rectangular domains. We applied the finite element method using commercial package COMSOL with a Galerkin formulation to solve the boundary value problems. This approach made the effort of coding rather small. The code solved a problem automatically using three consecutive meshes with increasing densities. This ensured reliable knowledge regarding the accuracy of the results. Further, a similar formulation was performed in a single blow case, the results of which can be compared directly with an analytical solution. We reported the results from three regenerator example cases in considerable detail. The finest meshes gave very good accuracy.

It should not be difficult to extend the analysis to cases in which the inflow fluid temperature depends on time. This is achieved by simply changing the boundary condition in the time direction appropriately.

REFERENCES

Carslaw HS, Jaeger JC. (1959) *Conduction of Heat in Solids*. Second edition. Clarendon Press.

Larsen FW. (1967) Rapid Calculation of Temperature in a Regenerative Heat Exchanger Having Arbitrary Initial Solid and Entering Fluid Temperatures. *Int. J. Heat Mass Transfer*, **10**, 149-168.

Mathematica (2014) www.wolfram.com/products/mathematica (22th April, 2014).

Roache PJ. (2009) *Fundamentals of Verification and Validation*. Hermosa publishers.

Rohsenow WM, Choi H. (1961) *Heat, Mass, and Momentum Transfer*. Prentice-Hall, Inc.

Schmidt FW, Willmott AJ. (1981) *Thermal Energy Storage and Regeneration*. McGraw-Hill Book Company.

Sinclair GB, Beisheim JR, Sezer S. (2006) Practical Convergence-Divergence Checks for Stresses from FEA. International ANSYS Users Conference and Exposition, 2-4 May 2006, Pittsburgh, PA, U.S.A. http://ansys.com/staticassets/ANSYS/staticassets/resourcelibrary/confpaper/2006-Int-ANSYS-Conf-28.pdf (18th April, 2014).

Willmott AJ. (2002) *Dynamics of Regenerative Heat Transfer*. Taylor & Francis.

Zienkiewicz OC, Taylor RL. (1989) *The Finite Element Method*. Fourth edition. McGraw-Hill.

APPENDIX A

Governing field equations

Nondimensional equations (3.10) and (3.11) were derived in some detail. A short derivation is given in Willmott (2002, p. 10), but for completeness we will go through the required steps.

Let us rewrite the governing dimensional field equations (2.35) and (2.36):

$$m_{\mathrm{f}} c_{\mathrm{f}} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + \dot{m}_{\mathrm{f}} c_{\mathrm{f}} L \frac{\partial t_{\mathrm{f}}}{\partial x} + \bar{h} A \left(t_{\mathrm{f}} - t_{\mathrm{s}} \right) = 0 , \tag{A.1}$$

$$m_{\mathrm{s}} c_{\mathrm{s}} \frac{\partial t_{\mathrm{s}}}{\partial \tau} + \bar{h} A \left(t_{\mathrm{s}} - t_{\mathrm{f}} \right) = 0 . \tag{A.2}$$

Some manipulation produces

$$\frac{m_{\mathrm{f}}}{\dot{m}_{\mathrm{f}} L} \frac{\partial t_{\mathrm{f}}}{\partial \tau} + \frac{\partial t_{\mathrm{f}}}{\partial x} + \frac{\bar{h} A}{\dot{m}_{\mathrm{f}} c_{\mathrm{f}} L} \left(t_{\mathrm{f}} - t_{\mathrm{s}} \right) = 0 , \tag{A.3}$$

$$\frac{\partial t_{\mathrm{s}}}{\partial \tau} + \frac{\bar{h} A}{m_{\mathrm{s}} c_{\mathrm{s}}} \left(t_{\mathrm{s}} - t_{\mathrm{f}} \right) = 0 . \tag{A.4}$$

We now introduce the dimensionless quantities

$$\xi = \Lambda \frac{x}{L} = \frac{\bar{h} A}{\dot{m}_{\mathrm{f}} c_{\mathrm{f}} L} x , \tag{A.5}$$

$$\eta = \Pi \frac{1}{P} \left(\tau - \frac{x}{v_x} \right) = \frac{\bar{h} A}{m_{\mathrm{s}} c_{\mathrm{s}}} \left(\tau - \frac{m_{\mathrm{f}} x}{\dot{m}_{\mathrm{f}} L} \right) . \tag{A.6}$$

The last formula makes use of the fact that with uniform fluid distribution, the mass flow rate is

$$\dot{m}_\mathrm{f} = \frac{m_\mathrm{f}}{L} v_x \,. \tag{A.7}$$

Chain differentiation gives

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} = \frac{\bar{h}A}{\dot{m}_\mathrm{f} c_\mathrm{f} L}\frac{\partial}{\partial \xi} - \frac{\bar{h}A}{m_\mathrm{s} c_\mathrm{s}}\frac{m_\mathrm{f}}{\dot{m}_\mathrm{f} L}\frac{\partial}{\partial \eta}\,, \tag{A.8}$$

$$\frac{\partial}{\partial \tau} = \frac{\partial \xi}{\partial \tau}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial \tau}\frac{\partial}{\partial \eta} = 0\frac{\partial}{\partial \xi} + \frac{\bar{h}A}{m_\mathrm{s} c_\mathrm{s}}\frac{\partial}{\partial \eta}\,. \tag{A.9}$$

When these are applied in (A.3) and (A.4), we obtain

$$\frac{m_\mathrm{f}}{\dot{m}_\mathrm{f} L}\frac{\bar{h}A}{m_\mathrm{s} c_\mathrm{s}}\frac{\partial t_\mathrm{f}}{\partial \eta} + \frac{\bar{h}A}{\dot{m}_\mathrm{f} c_\mathrm{f} L}\frac{\partial t_\mathrm{f}}{\partial \xi} - \frac{\bar{h}A}{m_\mathrm{s} c_\mathrm{s}}\frac{m_\mathrm{f}}{\dot{m}_\mathrm{f} L}\frac{\partial t_\mathrm{f}}{\partial \eta} + \frac{\bar{h}A}{\dot{m}_\mathrm{f} c_\mathrm{f} L}\left(t_\mathrm{f} - t_\mathrm{s}\right) = 0\,, \tag{A.10}$$

$$\frac{\bar{h}A}{m_\mathrm{s} c_\mathrm{s}}\frac{\partial t_\mathrm{s}}{\partial \eta} + \frac{\bar{h}\,A}{m_\mathrm{s} c_\mathrm{s}}\left(t_\mathrm{s} - t_\mathrm{f}\right) = 0\,, \tag{A.11}$$

or, after simplifications

$$\frac{\partial t_\mathrm{f}}{\partial \xi} + t_\mathrm{f} - t_\mathrm{s} = 0\,, \tag{A.12}$$

$$\frac{\partial t_\mathrm{s}}{\partial \eta} + t_\mathrm{s} - t_\mathrm{f} = 0\,. \tag{A.13}$$

Furthermore, definitions (3.7) and (3.8) give

$$t_\mathrm{f} = \left(t_\mathrm{fi} - t_0\right)T_\mathrm{f} + t_0\,, \quad t_\mathrm{s} = \left(t_\mathrm{fi} - t_0\right)T_\mathrm{s} + t_0\,, \tag{A.14}$$

where t_fi and t_0 are constants. Substituting these in (A.12) and (A.13) immediately provides the final field equations

$$\frac{\partial T_\mathrm{f}}{\partial \xi} - T_\mathrm{s} + T_\mathrm{f} = 0\,, \tag{A.15}$$

$$\frac{\partial T_{\mathrm{s}}}{\partial \eta} - T_{\mathrm{f}} + T_{\mathrm{s}} = 0 \,. \tag{A.16}$$

We obtained the same equations in the regenerator case with definitions (4.5) and (4.6).

APPENDIX B

Dimensional temperatures and thermal ratios

From definition (4.5),

$$t'_{\mathrm{f}}\left(x,\tau\right)=T_{\mathrm{f}}{}'\left(\xi',\eta'\right)\left(t'_{\mathrm{fi}}-t''_{\mathrm{fi}}\right)+t''_{\mathrm{fi}}\,. \tag{B.1}$$

Integrating both sides of this over $(0,\Pi')$ gives

$$\int_0^{\Pi'} t'_{\mathrm{f}}\left(x,\tau\right)\mathrm{d}\eta'=\left(t'_{\mathrm{fi}}-t''_{\mathrm{fi}}\right)\int_0^{\Pi'} T_{\mathrm{f}}{}'\left(\xi',\eta'\right)\mathrm{d}\eta'+t''_{\mathrm{fi}}\Pi'\,. \tag{B.2}$$

From (4.4),

$$\mathrm{d}\eta'=\frac{\Pi'}{P'}\mathrm{d}\tau \tag{B.3}$$

and the left-hand side of (B.2) can be transformed to

$$\int_0^{\Pi'} t'_{\mathrm{f}}\left(x,\tau\right)\mathrm{d}\eta'=\frac{\Pi'}{P'}\int_0^{P'} t'_{\mathrm{f}}\left(x,\tau\right)\mathrm{d}\tau\,. \tag{B.4}$$

Making use of the time average definitions (3.17), (B.2) can now be written as

$$\Pi'\tilde{t}'_{\mathrm{f}}\left(x\right)=\left(t'_{\mathrm{fi}}-t''_{\mathrm{fi}}\right)\Pi'\tilde{T}_{\mathrm{f}}{}'\left(\xi'\right)+t''_{\mathrm{fi}}\Pi' \tag{B.5}$$

or

$$\tilde{t}'_{\mathrm{f}}\left(x\right)=\left(t'_{\mathrm{fi}}-t''_{\mathrm{fi}}\right)\tilde{T}_{\mathrm{f}}{}'\left(\xi'\right)+t''_{\mathrm{fi}}\,. \tag{B.6}$$

and in particular,

$$\tilde{t}'_{\mathrm{f}}\left(L\right)=\left(t'_{\mathrm{fi}}-t''_{\mathrm{fi}}\right)\tilde{T}_{\mathrm{f}}{}'\left(\Lambda'\right)+t''_{\mathrm{fi}}\,. \tag{B.7}$$

Finally, the left-hand side of (4.26) becomes

$$\eta'_{\mathrm{REG}} = \frac{t'_{\mathrm{fi}} - \tilde{t}'_{\mathrm{f}}(L)}{t'_{\mathrm{fi}} - t''_{\mathrm{fi}}} = \frac{t'_{\mathrm{fi}} - \left(t'_{\mathrm{fi}} - t''_{\mathrm{fi}}\right)\tilde{T}_{\mathrm{f}}'\left(\Lambda'\right) - t''_{\mathrm{fi}}}{t'_{\mathrm{fi}} - t''_{\mathrm{fi}}}$$

$$= \frac{t'_{\mathrm{fi}} - t''_{\mathrm{fi}} + t''_{\mathrm{fi}} - \left(t'_{\mathrm{fi}} - t''_{\mathrm{fi}}\right)\tilde{T}_{\mathrm{f}}'\left(\Lambda'\right) - t''_{\mathrm{fi}}}{t'_{\mathrm{fi}} - t''_{\mathrm{fi}}} = 1 - \tilde{T}'_{\mathrm{f}}\left(\Lambda'\right). \qquad \text{(B.8)}$$

Formula (4.27) can be similarly proven.

APPENDIX C

Nondimensional temperatures and thermal ratios

Eliminating the term $T_s - T_f$ between (3.10) and (3.11) gives the equation

$$\frac{\partial T_f}{\partial \xi} = -\frac{\partial T_s}{\partial \eta}. \tag{C.1}$$

Both sides of this are integrated over Ω:

$$\int_0^{\Pi}\left(\int_0^{\Lambda}\frac{\partial T_f}{\partial \xi}\,d\xi\right)d\eta = -\int_0^{\Lambda}\left(\int_0^{\Pi}\frac{\partial T_s}{\partial \eta}\,d\eta\right)d\xi. \tag{C.2}$$

The integration order is intentionally different on each side. Performing the inner integrations gives

$$\int_0^{\Pi}\left[T_f(\Lambda,\eta) - T_f(0,\eta)\right]d\eta = -\int_0^{\Lambda}\left[T_s(\xi,\Pi) - T_s(\xi,0)\right]d\xi. \tag{C.3}$$

Performing the outer integrations and taking definitions (3.17) and (3.18) into account gives

$$\Pi\,\tilde{T}_f(\Lambda) - \Pi\,\tilde{T}_f(0) = -\Lambda\,\hat{T}_s(\Pi) + \Lambda\,\hat{T}_s(0). \tag{C.4}$$

This equation reveals interesting relations between the average values of temperature at the domain boundaries. We will apply it to the hot and cold periods.

In the hot period, the inflow fluid temperature at $\xi' = 0$ is $T_{fi}' = 1$, thus also $\tilde{T}_f(0) = 1$. Equation (C.4) therefore first becomes

$$\Pi'\,\tilde{T}_f'(\Lambda') - \Pi' = -\Lambda'\,\hat{T}_s'(\Pi') + \Lambda'\,\hat{T}_s'(0). \tag{C.5}$$

From this, (see (4.26))

$$\eta'_{\mathrm{REG}} = 1 - \tilde{T}'_{\mathrm{f}}\left(\Lambda'\right) = \frac{\Lambda'}{\Pi'}\left[\hat{T}'_{\mathrm{s}}\left(\Pi'\right) - \hat{T}'_{\mathrm{s}}\left(0\right)\right]. \tag{C.6}$$

When applying the cold period, a change in the signs appears, because $\dot{m}''_{\mathrm{f}}$ is defined as positive. Therefore (C.4) first becomes

$$\Pi''\tilde{T}''_{\mathrm{f}}\left(\Lambda''\right) - \Pi''\tilde{T}''_{\mathrm{f}}\left(0\right) = \Lambda''\hat{T}''_{\mathrm{s}}\left(\Pi''\right) - \Lambda''\hat{T}''_{\mathrm{s}}\left(0\right). \tag{C.7}$$

In the cold period, the inflow fluid temperature at $\xi'' = \Lambda''$ is $T''_{\mathrm{fi}} = 0$, thus also $\tilde{T}''_{\mathrm{f}}\left(\Lambda''\right) = 0$. Equation (C.7) therefore becomes

$$-\Pi''\tilde{T}''_{\mathrm{f}}\left(0\right) = \Lambda''\hat{T}''_{\mathrm{s}}\left(\Pi''\right) - \Lambda''\hat{T}''_{\mathrm{s}}\left(0\right). \tag{C.8}$$

From this (see (4.27)

$$\eta''_{\mathrm{REG}} = \frac{\Lambda''}{\Pi''}\left[\hat{T}''_{\mathrm{s}}\left(0\right) - \hat{T}''_{\mathrm{s}}\left(\Pi''\right)\right]. \tag{C.9}$$

In Willmott (2002, p. 192), the corresponding thermal ratio expressions are given by using the solid nondimensional temperatures at the beginning of the periods. Due to (4.23) and (4.24), and by taking the space averages we have

$$\hat{T}''_{\mathrm{s}}\left(\Pi''\right) = \hat{T}'_{\mathrm{s}}\left(0\right), \tag{C.10}$$

$$\hat{T}'_{\mathrm{s}}\left(\Pi'\right) = \hat{T}''_{s}\left(0\right). \tag{C.11}$$

When these are employed in (C.6) and (C.9), we arrive at the formulas

$$\eta'_{\mathrm{REG}} = \frac{\Lambda'}{\Pi'}\left[\hat{T}''_{s}\left(0\right) - \hat{T}'_{\mathrm{s}}\left(0\right)\right], \tag{C.12}$$

$$\eta''_{\mathrm{REG}} = \frac{\Lambda''}{\Pi''}\left[\hat{T}''_{s}\left(0\right) - \hat{T}'_{\mathrm{s}}\left(0\right)\right]. \tag{C.13}$$

These agree with Willmott (2002), except for the signs. Willmott (2002) may contain printing errors. From the above, we find the simple relation

$$\frac{\eta'_{\mathrm{REG}}}{\eta''_{\mathrm{REG}}} = \frac{\Lambda' / \Pi'}{\Lambda'' / \Pi''}. \tag{C.14}$$

APPENDIX D

COMSOL Multiphysics Model for SBC (First application of SBC)

SBC COMSOL Multiphysics Model

Date	May 7, 2014 8:26:11 AM

Contents

1 Global Definitions

1.1 Parameters 1

Parameters

Name	Expression	Description
lamda	1.847	
phi	3.78	

2 Model 1 {mod1}

2.1 Definitions

2.1.1 Coordinate Systems

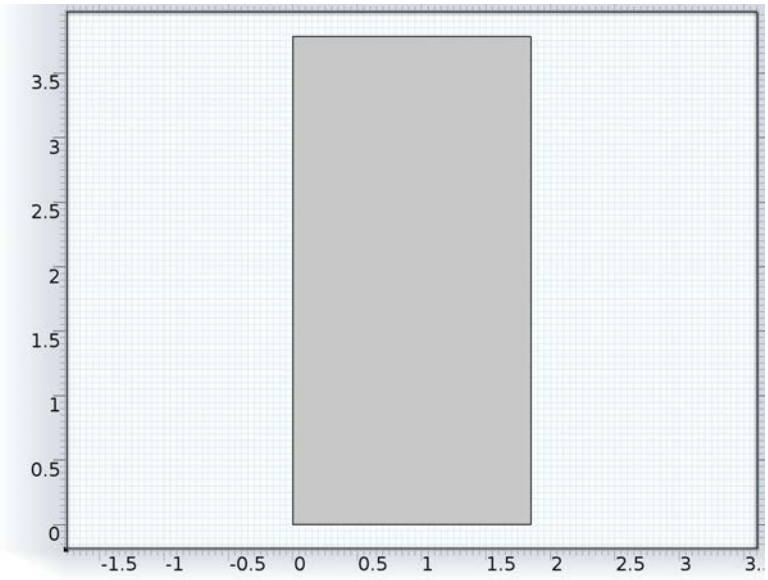
Boundary System 1

Coordinate system type	Boundary system
Identifier	sys1

Settings

Name	Value
Coordinate names	{t1, n, to}
Create first tangent direction from	Global Cartesian

2.2 Geometry 1



Geometry 1

units

Length unit	m
Angular unit	deg

Geometry statistics

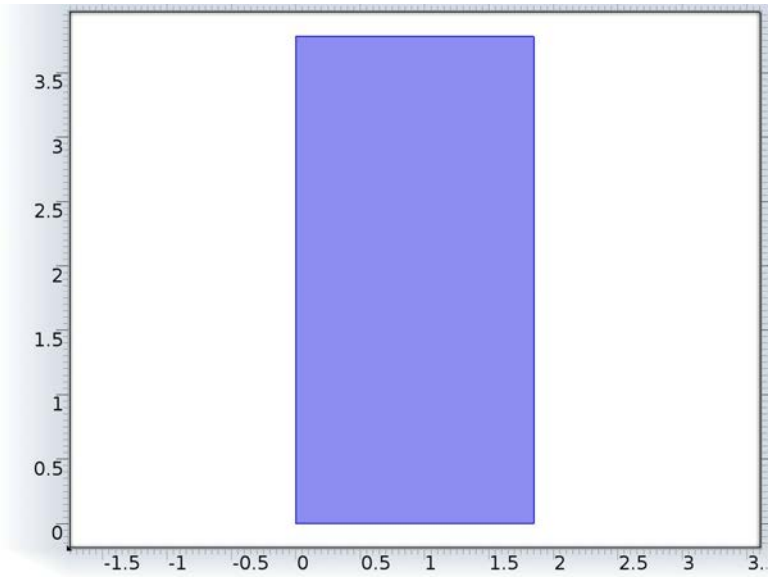
Property	Value
Space dimension	2
Number of domains	1
Number of boundaries	4

2.2.1 Rectangle 1 (r1)

Position

Name	Value
Position	{0, 0}
Width	lamda
Height	phi
Size	{lamda, phi}

2.3 Weak Form PDE {w}



Weak Form PDE

Selection

Geometric entity level	Domain
Selection	Domain 1

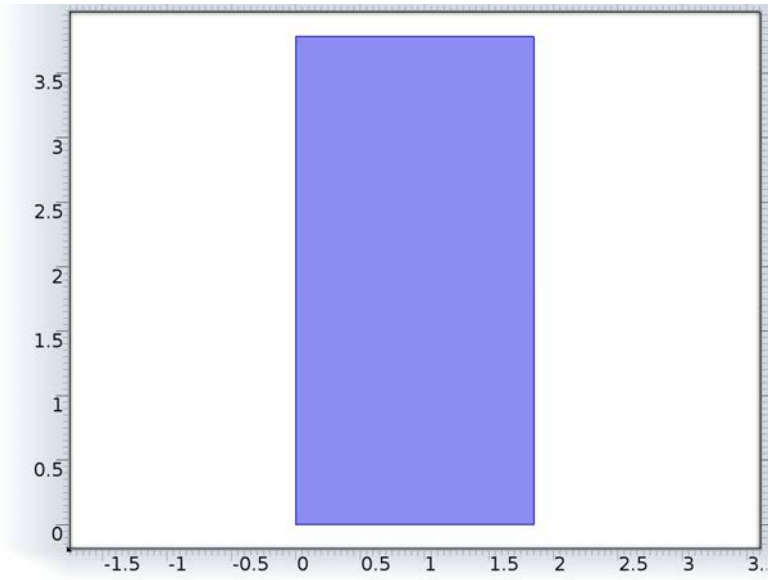
Settings

Description	Value
Element order	Linear

Used products

COMSOL Multiphysics

2.3.1 Weak Form PDE 1



Weak Form PDE 1

Selection

Geometric entity level	Domain
Selection	Domain 1

Equations

$$0 = \int_{\Omega} \text{weak}\, \partial S$$

Settings

Settings

Description	Value
Weak expressions	{test(Tf)*(Tfx - Ts + Tf), test(Ts)*(Tsy - Tf + Ts)}

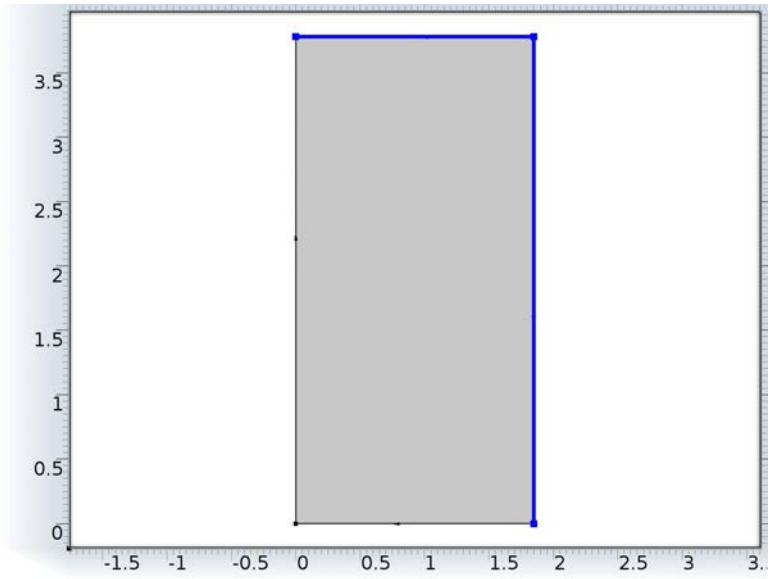
Shape functions

Name	Shape function	Unit	Description	Shape frame	Selection
Tf	Lagrange (Linear)	1	Dependent variable Tf	Material	Domain 1
Ts	Lagrange (Linear)	1	Dependent variable Ts	Material	Domain 1

Weak expressions

Weak expression	Integration frame	Selection
test(Tf)*(Tfx - Ts + Tf)	Material	Domain 1
test(Ts)*(Tsy - Tf + Ts)	Material	Domain 1

2.3.2 Zero Flux 1



Zero Flux 1

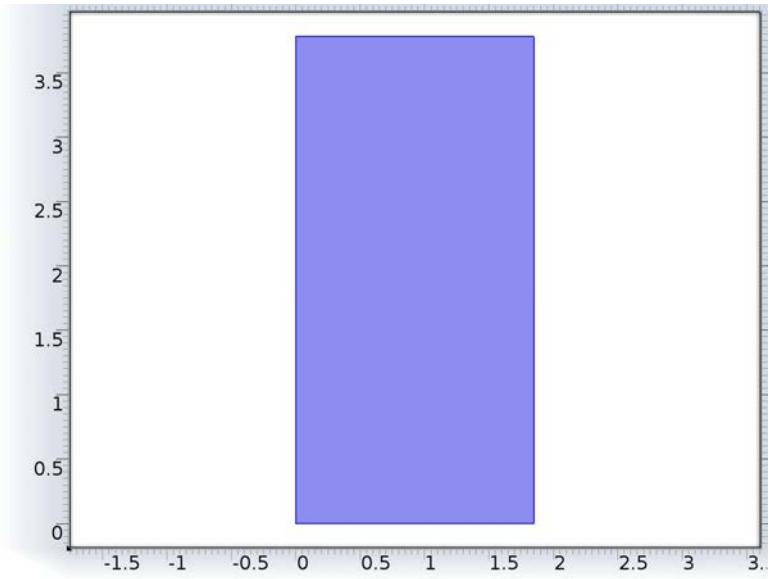
Selection

Geometric entity level	Boundary
Selection	Boundaries 3–4

Equations

$$-\mathbf{n} \cdot \text{flux} = 0$$

2.3.3 Initial Values 1

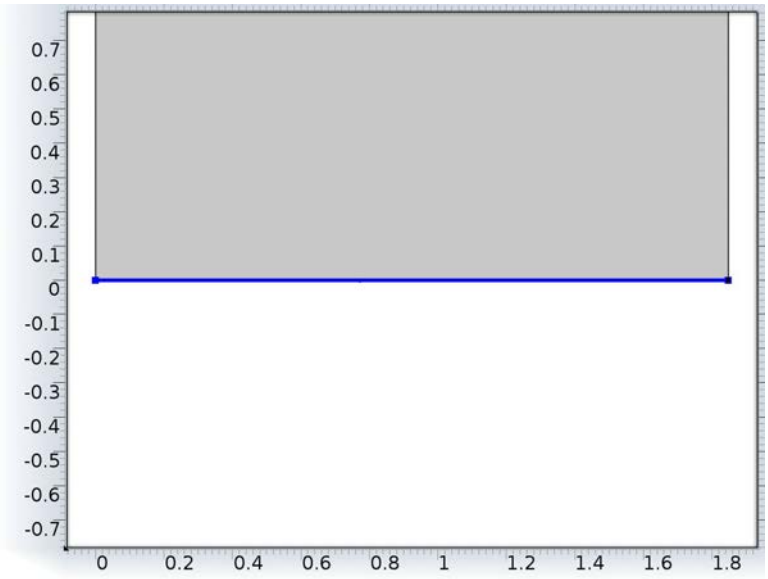


Initial Values 1

Selection

Geometric entity level	Domain
Selection	Domain 1

2.3.4 Dirichlet Boundary Condition 1



Dirichlet Boundary Condition 1

Selection

Geometric entity level	Boundary
Selection	Boundary 2

Equations

$$\mathbf{u} = r$$
$$\mathbf{u} = [Tf, Ts]^T$$
$$g_{reaction} = -\mu$$
$$\mu = [\mu_1, \mu_2]^T$$

Settings

Settings

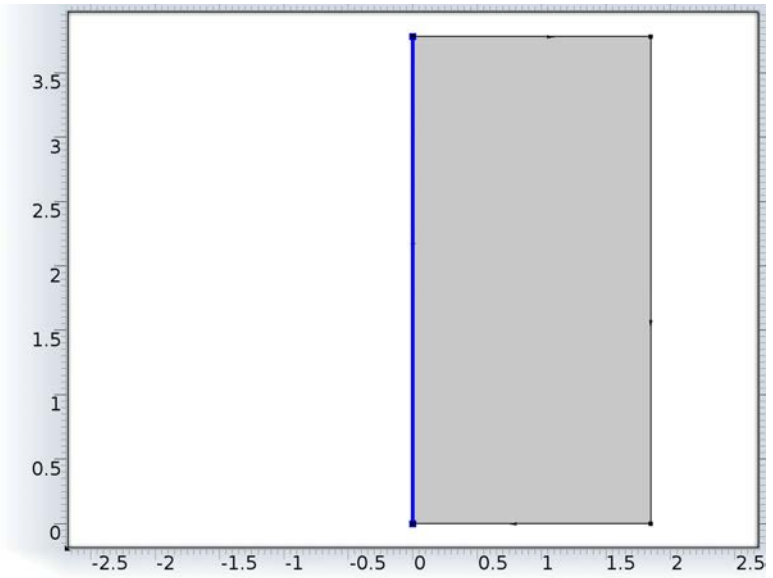
Description	Value

Description	Value
Prescribed value of Tf	0
Prescribed value of Ts	1

Constraints

Constraint	Constraint force	Shape function	Selection
-Ts	-test(Ts)	Lagrange (Linear)	Boundary 2

2.3.5 Dirichlet Boundary Condition 2



Dirichlet Boundary Condition 2

Selection

Geometric entity level	Boundary
Selection	Boundary 1

Equations

$$\mathbf{u} = r$$
$$\mathbf{u} = [Tf, Ts]^T$$
$$g_{reaction} = -\mu$$
$$\mu = [\mu_1, \mu_2]^T$$

Settings

Settings

Description	Value
Value on boundary	{1, 0}
Prescribed value of Tf	1
Prescribed value of Ts	0

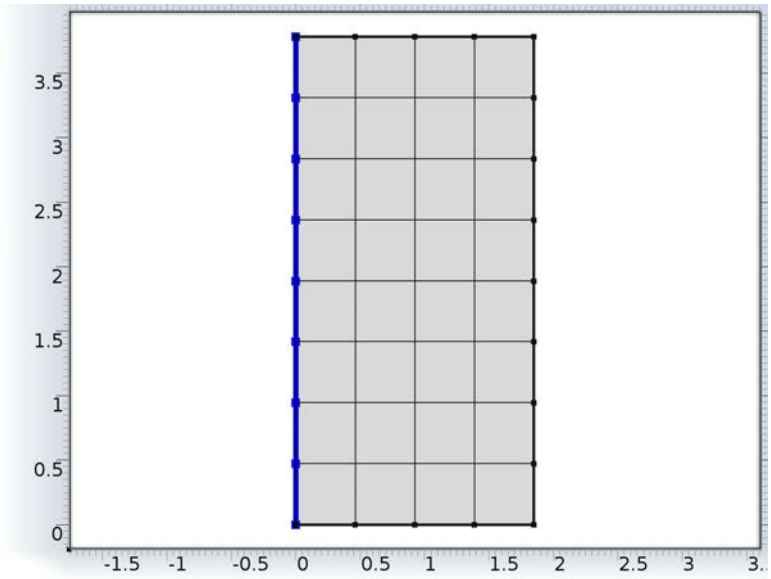
Constraints

Constraint	Constraint force	Shape function	Selection
1 - Tf	-test(Tf)	Lagrange (Linear)	Boundary 1

2.4 Mesh 1

Mesh statistics

Property	Value
Minimum element quality	0.9997
Average element quality	0.9997
Quadrilateral elements	32
Edge elements	24
Vertex elements	4



Mesh 1

2.4.1 Size (size)

Settings

Name	Value
Maximum element size	0.253
Minimum element size	0.00113
Resolution of curvature	0.3
Maximum element growth rate	1.3

3 Study 1 {std1}

3.1 Stationary

Mesh selection

Geometry	Mesh
Geometry 1 (geom1)	mesh1

Physics selection

Physics	Discretization
Weak Form PDE (w)	physics

3.2 Solver Configurations

3.2.1 Solver 1

Compile Equations: Stationary {stat} (st1)

Study and step

Name	Value
Use study	Study 1
Use study step	Stationary

Dependent Variables 1 (v1)

General

Name	Value
Defined by study step	Stationary

Initial values of variables solved for

Name	Value
Solution	Zero

Values of variables not solved for

Name	Value
Solution	Zero

mod1.Ts (mod1_Ts)

General

Name	Value
Field components	mod1.Ts

mod1.Tf (mod1_Tf)

General

Name	Value
Field components	mod1.Tf

Stationary Solver 1 (s1)

General

Name	Value
Defined by study step	Stationary

Log

```
Stationary Solver 1 {s1} in Solver 1 {sol1} started at 5-Jun-
2014 14:29:25.
Linear solver
Number of degrees of freedom solved for: 90.
Nonsymmetric matrix found.
Scales for dependent variables:
mod1.Ts: 1
mod1.Tf: 0.96
Iter     Damping    Stepsize #Res #Jac #Sol
   1   1.0000000        0.97    1    1    1
Stationary Solver 1 {s1} in Solver 1 {sol1}: Solution time: 0 s
.
```

Fully Coupled 1 (fc1)

General

Name	Value
Linear solver	Direct

4 Results

4.1 Data Sets

4.1.1 Solution 1

Selection

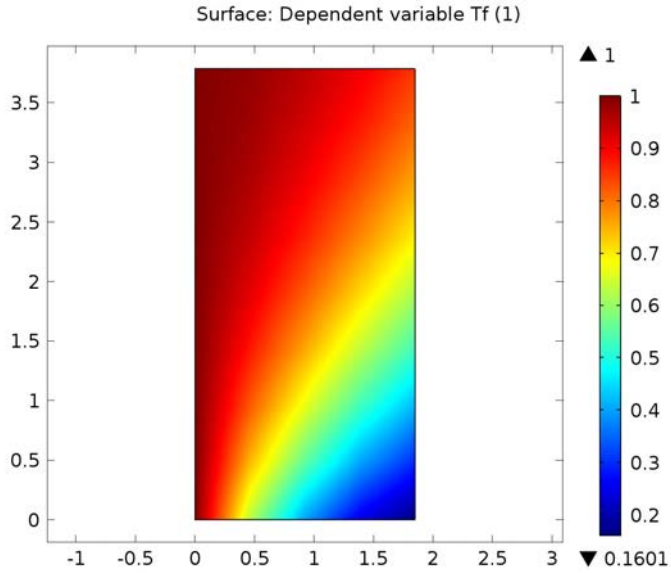
Geometric entity level	Domain
Selection	Geometry geom1

Solution

Name	Value
Solution	Solver 1
Model	Save Point Geometry 1

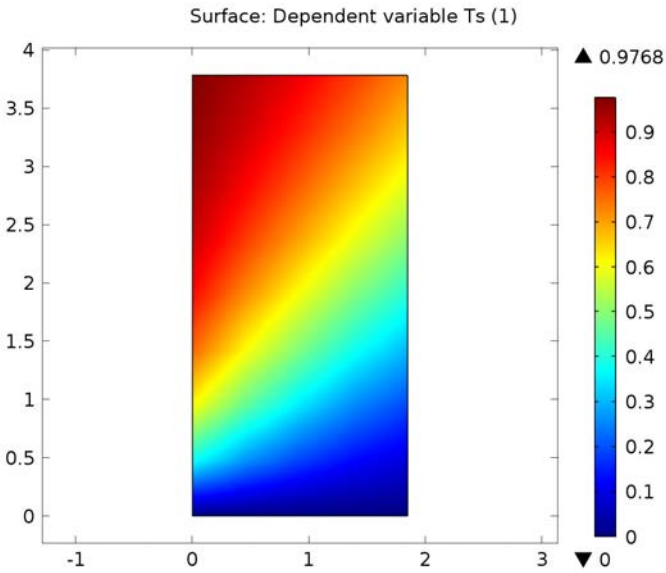
4.2 Plot Groups

4.2.1 2D Plot Group 1



Surface: Dependent variable Tf (1)

4.2.2 2D Plot Group 2



Surface: Dependent variable Ts (1)

COMSOL Multiphysics Model for REG (First application of REG, hot period)

REG COMSOL Multiphysics Model

Date	Apr 10, 2014 9:13:42 AM

Contents

1 Global Definitions

1.1 Parameters 1

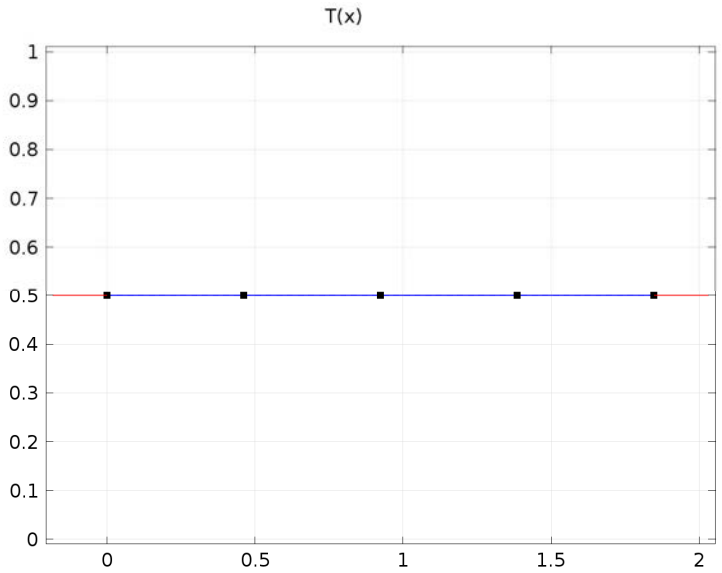
Parameters

Name	Expression	Description
lamda	10	
phi	20	
d	1	
Tci	0	
Thi	1	

1.2 Functions

1.2.1 Interpolation 1

Function name	int1
Function type	Interpolation



Interpolation 1

2 Model 1 {mod1}

2.1 Definitions

2.1.1 Coordinate Systems

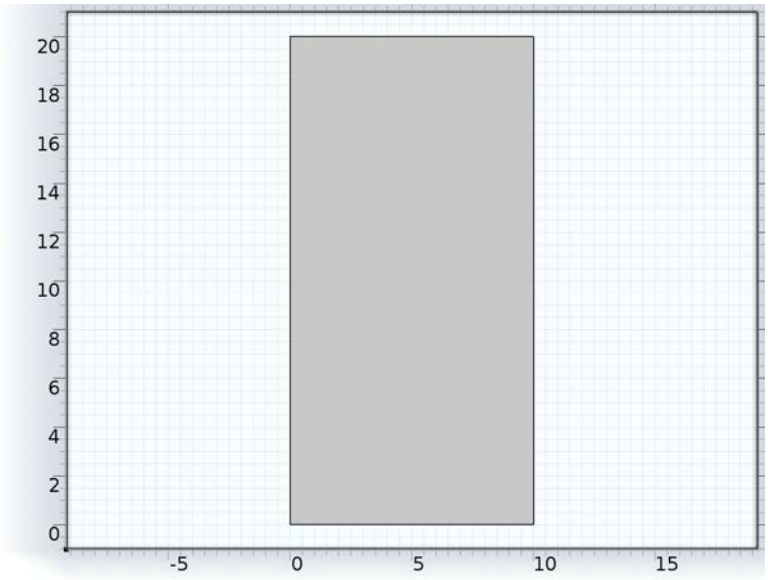
Boundary System 1

Coordinate system type	Boundary system
Identifier	sys1

Settings

Name	Value
Coordinate names	{t1, n, to}
Create first tangent direction from	Global Cartesian

2.2 Geometry 1



Geometry 1

units

Length unit	m
Angular unit	deg

Geometry statistics

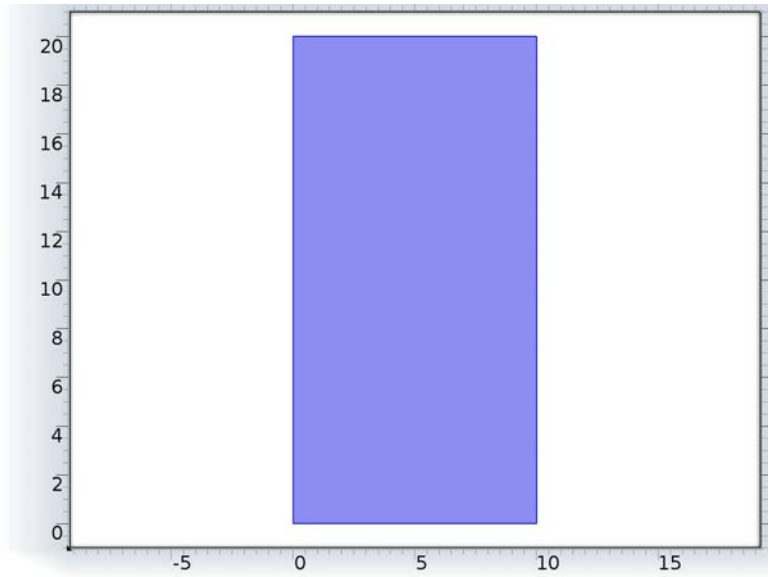
Property	Value
Space dimension	2
Number of domains	1
Number of boundaries	4

2.2.1 Rectangle 1 (r1)

Position

Name	Value
Position	{0, 0}
Width	lamda
Height	phi
Size	{lamda, phi}

2.3 Weak Form PDE {w}



Weak Form PDE

Selection

Geometric entity level	Domain
Selection	Domain 1

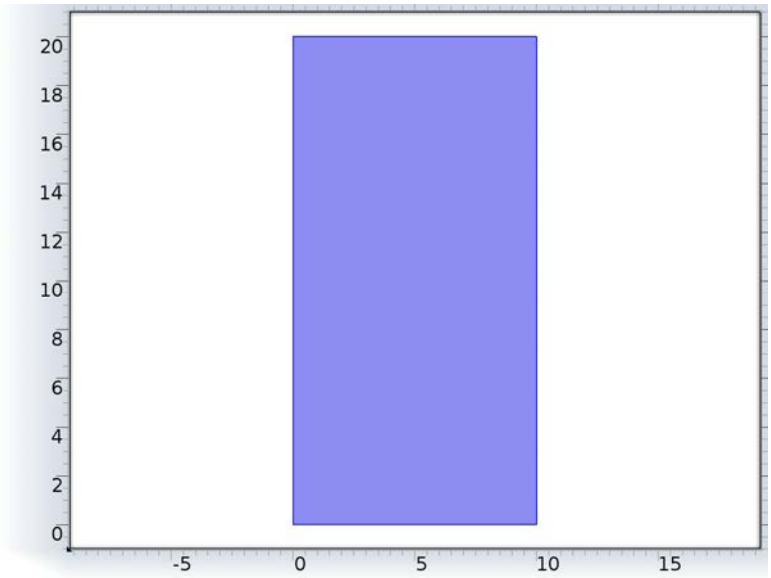
Settings

Description	Value
Element order	Linear

Used products

COMSOL Multiphysics

2.3.1 Weak Form PDE 1



Weak Form PDE 1

Selection

Geometric entity level	Domain
Selection	Domain 1

Equations

$$0 = \int_{\Omega} \text{weak}\, \partial S$$

Settings

Settings

Description	Value
Weak expressions	{test(Tf)*(Tfx - d*Ts + d*Tf), test(Ts)*(Tsy - Tf + Ts)}

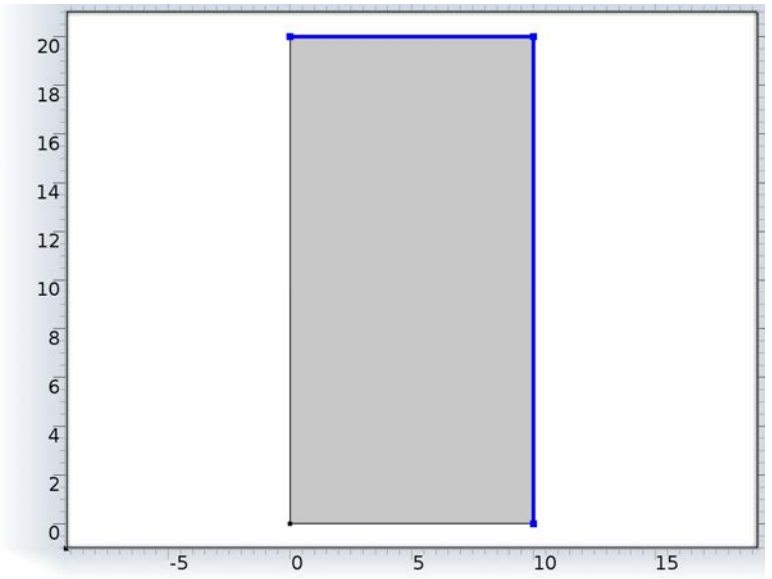
Shape functions

Name	Shape function	Unit	Description	Shape frame	Selection
Tf	Lagrange (Linear)	1	Dependent variable Tf	Material	Domain 1
Ts	Lagrange (Linear)	1	Dependent variable Ts	Material	Domain 1

Weak expressions

Weak expression	Integration frame	Selection
test(Tf)*(Tfx - d*Ts + d*Tf)	Material	Domain 1
test(Ts)*(Tsy - Tf + Ts)	Material	Domain 1

2.3.2 Zero Flux 1



Zero Flux 1

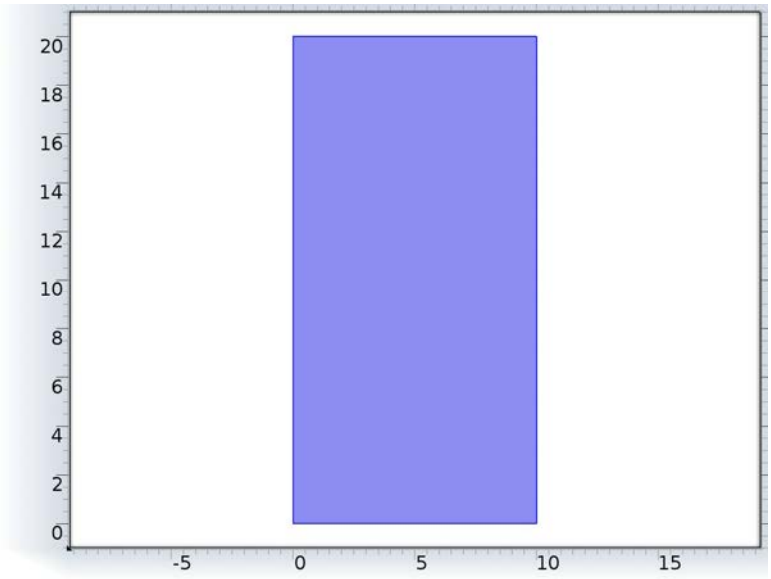
Selection

Geometric entity level	Boundary
Selection	Boundaries 3–4

Equations

$$-\mathbf{n} \cdot \text{flux} = 0$$

2.3.3 Initial Values 1

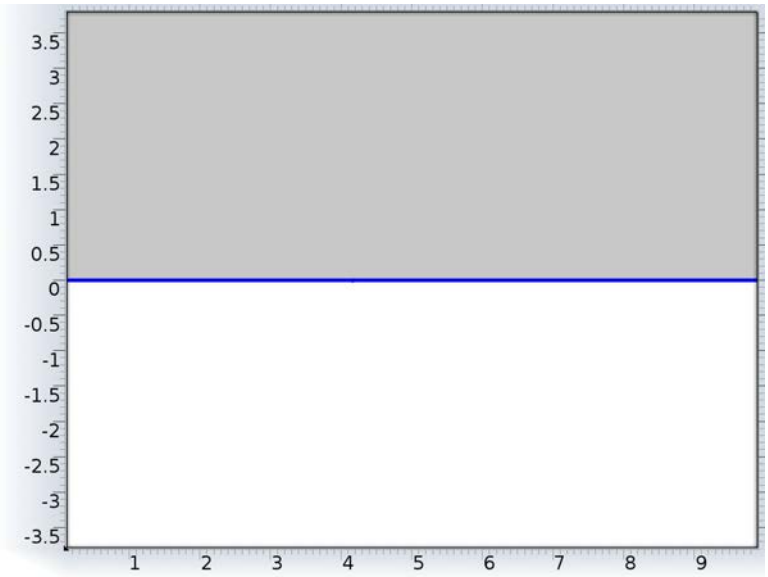


Initial Values 1

Selection

Geometric entity level	Domain
Selection	Domain 1

2.3.4 Dirichlet Boundary Condition 1



Dirichlet Boundary Condition 1

Selection

Geometric entity level	Boundary
Selection	Boundary 2

Equations

$$\mathbf{u} = r$$
$$\mathbf{u} = [Tf, Ts]^T$$
$$g_{reaction} = -\mu$$
$$\mu = [\mu_1, \mu_2]^T$$

Settings

Settings

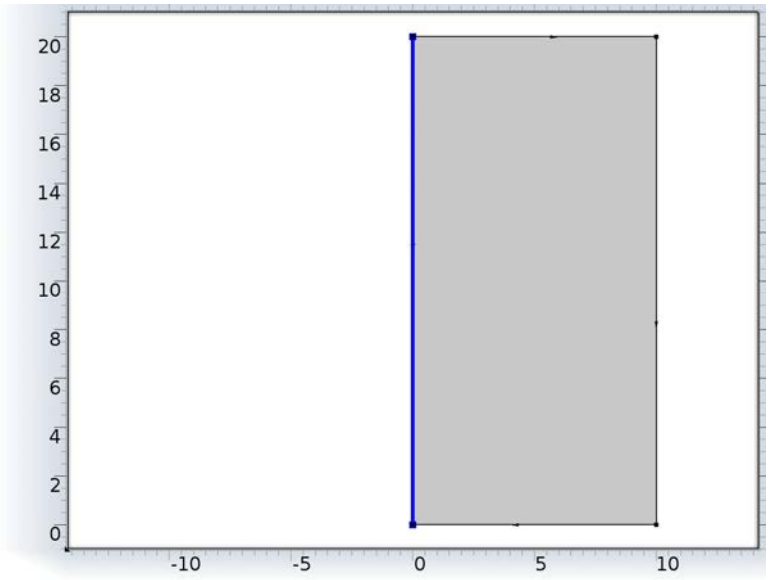
Description	Value

Description	Value
Value on boundary	{0, T(x)}
Prescribed value of Tf	0
Prescribed value of Ts	1

Constraints

Constraint	Constraint force	Shape function	Selection
T(x) - Ts	-test(Ts)	Lagrange (Linear)	Boundary 2

2.3.5 Constraint 1



Constraint 1

Selection

Geometric entity level	Boundary

Selection	Boundary 1

Equations

$$0 = R$$
$$g_{reaction} = \left(\frac{\partial R}{\partial \mathbf{u}}\right)^T \mu$$
$$\mathbf{u} = [Tf, Ts]$$
$$\mu = [\mu_1, \mu_2]^T$$

Settings

Settings

Description	Value
Bidirectional constraint, R = 0	{Thi - Tf, 0}

Variables

Name	Expression	Unit	Description	Selection
w.R_Tf	Thi - Tf	1	Bidirectional constraint, R = 0	Boundary 1
w.R_Ts	0		Bidirectional constraint, R = 0	Boundary 1

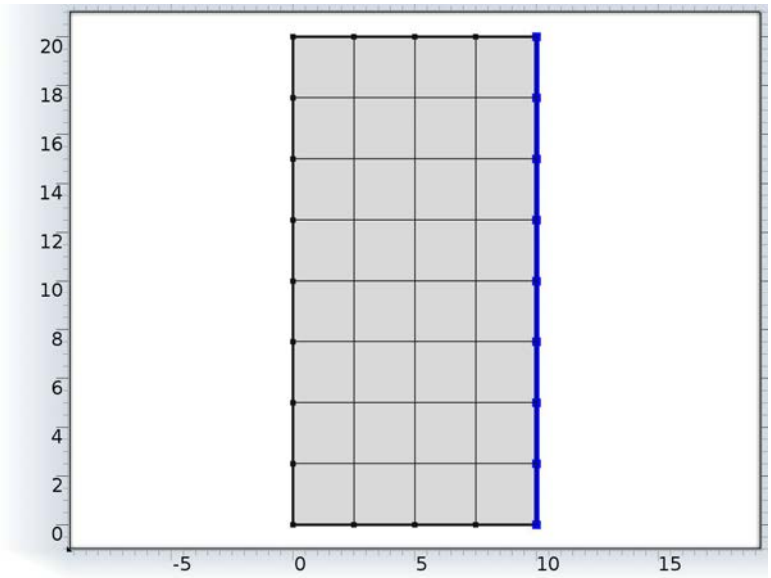
Constraints

Constraint	Constraint force	Shape function	Selection
Thi - Tf	test(Thi - Tf)	Lagrange (Linear)	Boundary 1
0	0	Lagrange (Linear)	Boundary 1

2.4 Mesh 1

Mesh statistics

Property	Value
Minimum element quality	1.0
Average element quality	1.0
Quadrilateral elements	32
Edge elements	24
Vertex elements	4



Mesh 1

2.4.1 Size (size)

Settings

Name	Value

Name	Value
Maximum element size	1.34
Minimum element size	0.0060
Resolution of curvature	0.3
Maximum element growth rate	1.3

3 Study 1 {std1}

3.1 Stationary

Mesh selection

Geometry	Mesh
Geometry 1 (geom1)	mesh1

Physics selection

Physics	Discretization
Weak Form PDE (w)	physics

3.2 Solver Configurations

3.2.1 Solver 1

Compile Equations: Stationary {stat} (st1)

Study and step

Name	Value
Use study	Study 1
Use study step	Stationary

Dependent Variables 1 (v1)

General

Name	Value
Defined by study step	Stationary

Initial values of variables solved for

Name	Value
Solution	Zero

Values of variables not solved for

Name	Value
Solution	Zero

mod1.Ts (mod1_Ts)

General

Name	Value
Field components	mod1.Ts

mod1.Tf (mod1_Tf)

General

Name	Value
Field components	mod1.Tf

Stationary Solver 1 (s1)

General

Name	Value
Defined by study step	Stationary

Log

```
Stationary Solver 1 {s1} in Solver 1 {sol1} started at 5-Jun-
2014 15:27:10.
Linear solver
Number of degrees of freedom solved for: 90.
Nonsymmetric matrix found.
Scales for dependent variables:
mod1.Ts: 1
mod1.Tf: 1
Iter     Damping    Stepsize #Res #Jac #Sol
   1   1.0000000           1    1    1    1
Stationary Solver 1 {s1} in Solver 1 {sol1}: Solution time: 0 s
.
```

Fully Coupled 1 (fc1)

General

Name	Value
Linear solver	Direct

4 Results

4.1 Data Sets

4.1.1 Solution 1

Selection

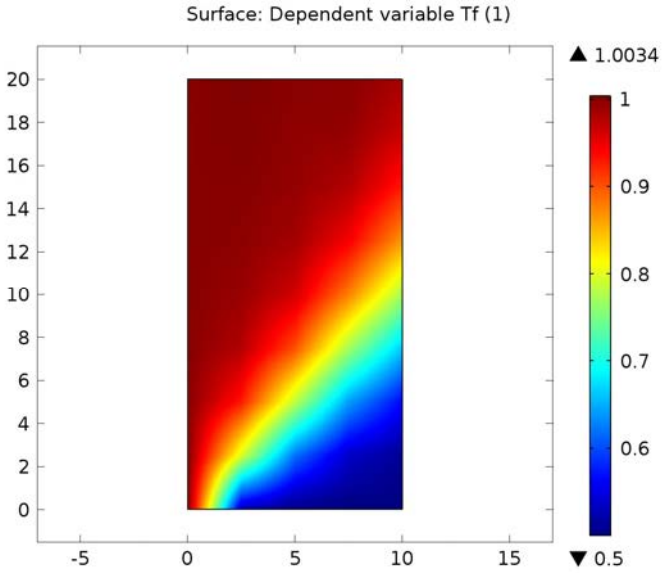
Geometric entity level	Domain
Selection	Geometry geom1

Solution

Name	Value
Solution	Solver 1
Model	Save Point Geometry 1

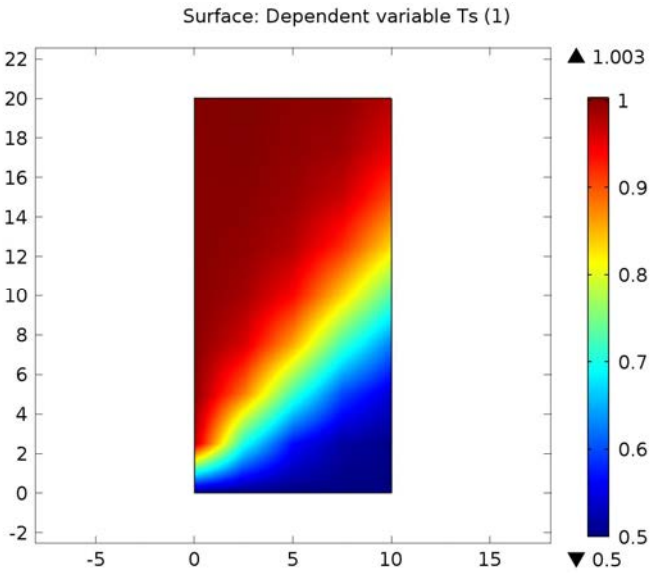
4.2 Plot Groups

4.2.1 2D Plot Group 1



Surface: Dependent variable Tf (1)

4.2.2 2D Plot Group 2



Surface: Dependent variable Ts (1)

APPENDIX E

COMSOL Model as MATLAB form for SBC

```
function out = model
%
% SBC_COMSOL_Multiphysics_model.m
%
% Model exported on Jan 25 2014, 17:58 by COMSOL 4.3.0.151.

import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.modelPath('C:\');

model.name('SBC_COMSOL_Multiphysics_model.mph');

model.param.set('lamda', '3.78');
model.param.set('phi', '1.847');

model.modelNode.create('mod1');

model.geom.create('geom1', 2);
model.geom('geom1').feature.create('r1', 'Rectangle');
model.geom('geom1').feature('r1').set('size', {'lamda'
'phi'});
model.geom('geom1').run;

model.physics.create('w', 'WeakFormPDE', 'geom1');
model.physics('w').field('dimensionless').field('Tf');
model.physics('w').field('dimensionless').component({'Tf'
```

```
'Ts'});
model.physics('w').feature.create('dir1',
'DirichletBoundary', 1);
model.physics('w').feature('dir1').selection.set([2]);
model.physics('w').feature.create('dir2',
'DirichletBoundary', 1);
model.physics('w').feature('dir2').selection.set([1]);


model.mesh.create('mesh1', 'geom1');
model.mesh('mesh1').feature.create('map1', 'Map');
model.mesh('mesh1').feature('map1').feature.create('dis1',
'Distribution');
model.mesh('mesh1').feature('map1').feature('dis1').selecti
on.set([2 3]);
model.mesh('mesh1').feature('map1').feature.create('dis2',
'Distribution');
model.mesh('mesh1').feature('map1').feature('dis2').selecti
on.set([1 4]);


model.view('view1').axis.set('xmin', '-
0.8489214181900024');
model.view('view1').axis.set('xmax', '4.6289215087890625');
model.view('view1').axis.set('ymin', '-
3.2946972846984863');
model.view('view1').axis.set('ymax', '5.141697406768799');


model.physics('w').prop('ShapeProperty').set('order', '1');
model.physics('w').feature('wfeq1').set('weak',
{'test(Tf)*(Tfx-Ts+Tf)'; 'test(Ts)*(Tsy-Tf+Ts)'});
model.physics('w').feature('dir1').set('useDirichletConditi
on', {'0'; '1'});
model.physics('w').feature('dir2').set('r', {'1'; '0'});
model.physics('w').feature('dir2').set('useDirichletConditi
on', {'1'; '0'});


model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem', '32');
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
```

```
melem', '16');
model.mesh('mesh1').run;

model.frame('material1').sorder(1);

model.study.create('std1');
model.study('std1').feature.create('stat', 'Stationary');

model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').attach('std1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('fc1',
'FullyCoupled');
model.sol('sol1').feature('s1').feature.remove('fcDef');

model.result.create('pg1', 'PlotGroup2D');
model.result('pg1').feature.create('surf1', 'Surface');
model.result.create('pg2', 'PlotGroup2D');
model.result('pg2').feature.create('surf1', 'Surface');

model.sol('sol1').attach('std1');
model.sol('sol1').feature('st1').name('Compile Equations:
Stationary {stat}');
model.sol('sol1').feature('st1').set('studystep', 'stat');
model.sol('sol1').feature('v1').set('control', 'stat');
model.sol('sol1').feature('s1').set('control', 'stat');
model.sol('sol1').runAll;

model.result('pg2').feature('surf1').set('expr', 'Ts');
model.result('pg2').feature('surf1').set('descr',
'Dependent variable Ts');

out = model;
```

COMSOL Model as MATLAB form for REG

```
function out = model
%
% REG_COMSOL_Multiphysics_model.m
%
% Model exported on Apr 10 2014, 09:13 by COMSOL 4.3.0.151.

import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.modelPath('C:\');

model.name('REG_COMSOL_Multiphysics_model.mph');

model.param.set('lamda', '1.847');
model.param.set('phi', '3.78');
model.param.set('d', '1');
model.param.set('Tci', '0');
model.param.set('Thi', '1');

model.modelNode.create('mod1');

model.file.create('res1');

model.func.create('int1', 'Interpolation');
model.func('int1').set('funcs', {'T' '1'});
model.func('int1').set('source', 'file');
model.func('int1').set('filename', 'C:\Tinitial.txt');

model.geom.create('geom1', 2);
model.geom('geom1').feature.create('r1', 'Rectangle');
```

```
model.geom('geom1').feature('r1').set('size', {'lamda'
'phi'});
model.geom('geom1').run;

model.physics.create('w', 'WeakFormPDE', 'geom1');
model.physics('w').field('dimensionless').field('Tf');
model.physics('w').field('dimensionless').component({'Tf'
'Ts'});
model.physics('w').feature.create('dir1',
'DirichletBoundary', 1);
model.physics('w').feature('dir1').selection.set([2]);
model.physics('w').feature.create('cons1', 'Constraint',
1);
model.physics('w').feature('cons1').selection.set([1]);
model.physics('w').feature.create('cons2', 'Constraint',
1);
model.physics('w').feature('cons2').selection.set([4]);

model.mesh.create('mesh1', 'geom1');
model.mesh('mesh1').feature.create('map1', 'Map');
model.mesh('mesh1').feature('map1').feature.create('dis1',
'Distribution');
model.mesh('mesh1').feature('map1').feature('dis1').selecti
on.set([2 3]);
model.mesh('mesh1').feature('map1').feature.create('dis2',
'Distribution');
model.mesh('mesh1').feature('map1').feature('dis2').selecti
on.set([1 4]);

model.view('view1').axis.set('xmin', '-
0.38010746240615845');
model.view('view1').axis.set('xmax', '2.227107524871826');
model.view('view1').axis.set('ymin', '-
0.1889999955892563');
model.view('view1').axis.set('ymax', '3.9689998626708984');

model.physics('w').prop('ShapeProperty').set('order', '1');
model.physics('w').feature('wfeq1').set('weak',
```

```
{'test(Tf)*(Tfx-d*Ts+d*Tf)'; 'test(Ts)*(Tsy-Tf+Ts)'});
model.physics('w').feature('dir1').set('r', {'0'; 'T(x)'});
model.physics('w').feature('dir1').set('useDirichletConditi
on', {'0'; '1'});
model.physics('w').feature('cons1').set('R', {'Thi-Tf';
'0'});
model.physics('w').feature('cons2').set('R', {'Tci-Tf';
'0'});

model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem', '4');
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem', '8');
model.mesh('mesh1').run;

model.frame('material1').sorder(1);

model.study.create('std1');
model.study('std1').feature.create('stat', 'Stationary');

model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').attach('std1');
model.sol('sol1').feature.create('st1', 'StudyStep');
model.sol('sol1').feature.create('v1', 'Variables');
model.sol('sol1').feature.create('s1', 'Stationary');
model.sol('sol1').feature('s1').feature.create('fc1',
'FullyCoupled');
model.sol('sol1').feature('s1').feature.remove('fcDef');

model.result.create('pg1', 'PlotGroup2D');
model.result('pg1').feature.create('surf1', 'Surface');
model.result.create('pg2', 'PlotGroup2D');
model.result('pg2').feature.create('surf1', 'Surface');

model.sol('sol1').attach('std1');
model.sol('sol1').feature('st1').name('Compile Equations:
```

```
Stationary {stat}');
model.sol('sol1').feature('st1').set('studystep', 'stat');
model.sol('sol1').feature('v1').set('control', 'stat');
model.sol('sol1').feature('s1').set('control', 'stat');
model.sol('sol1').runAll;

model.result('pg2').feature('surf1').set('expr', 'Ts');
model.result('pg2').feature('surf1').set('descr',
'Dependent variable Ts');

out = model;
```

APPENDIX F

MATLAB codes for SBC

```
% MAIN CODE FOR SINGLE BLOW CASE (SBC)
%
% This main code determines SBC case temperature
distribution by using COMSOL 4.3 with MATLAB program
% The main code includes four subroutines: (1) the subcode
for given data, (2) the subcode for numerical integration
with trapezoidal method, (3) the subcode for post
processing, and (4) the subcode for dimensional fluid and
solid temperatures
% The equation numbers, referred to, are the same as in the
main part of the report


clear all


format long


model = mphload('SBC_COMSOL_Multiphysics_model'); % Load
COMSOL Multiphysics SBC model (m-file)


SBCdata % Subcode for given data


model.param.set('lamda',SBC_lamda); % Set nondimensional
length, equation (3.3)
model.param.set('phi',SBC_phi); % Set nondimensional time
length, equation (3.5)


% Coarse mesh
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',SBC_coarse_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',SBC_coarse_numelem2); % Set number of elements in
```

```
time direction

SBC_numerical_integration % Subcode for numerical
integration with trapezoidal method

save SBC_Ts_coarse.txt data6 -ascii -tabs % Save solid
temperatures for post processing
save SBC_Tf_coarse.txt data7 -ascii -tabs % Save fluid
temperatures for post processing
save SBC_Tsmean_coarse.txt data8 -ascii -tabs % Save solid
mean temperature for post processing

% Medium mesh
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',SBC_medium_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',SBC_medium_numelem2); % Set number of elements in
time direction

SBC_numerical_integration % Subcode for numerical
integration with trapezoidal method

save SBC_Ts_medium.txt data6 -ascii -tabs % Save solid
temperatures for post processing
save SBC_Tf_medium.txt data7 -ascii -tabs % Save fluid
temperatures for post processing
save SBC_Tsmean_medium.txt data8 -ascii -tabs % Save solid
mean temperature for post processing

% Fine mesh
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',SBC_fine_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',SBC_fine_numelem2); % Set number of elements in time
direction

SBC_numerical_integration % Subcode for numerical
integration with trapezoidal method
```

```
save SBC_Ts_fine.txt data6 -ascii -tabs % Save solid
temperatures for post processing
save SBC_Tf_fine.txt data7 -ascii -tabs % Save fluid
temperatures for post processing
save SBC_Tsmean_fine.txt data8 -ascii -tabs % Save solid
mean temperature for post processing


SBC_post_processing % Subcode for post processing


SBC_dimensional_temperatures % Subcode for dimensional
fluid and solid temperatures
```

```
% SUBCODE FOR GIVEN DATA (SBC)
%
% This subroutine loads the given SBC values from the data
matrix

data1 = load('C:\SBCdata.txt'); % Load given SBC data

A = data1(1,1); % Total heat transfer surface area, m^2
ms = data1(2,1); % Total storage medium mass, kg
cs = data1(3,1); % Storage medium specific heat at constant
pressure, kJ/kg K
cf = data1(4,1); % Fluid specific heat at constant
pressure, kJ/kg K
mf = data1(5,1); % Total mass rate of fluid, kg/s
h = data1(6,1); % Overall heat transfer coefficient,
W/m^2 K
P = data1(7,1); % Period, s
t0 = data1(8,1); % Dimensional constant initial storage
material temperature, equation (3.1)
tfi = data1(9,1); % Dimensional constant fluid inflow
temperature, equation (3.2)

SBC_coarse_numelem1 = data1(10,1); % Number of elements in
axial direction with coarse mesh
SBC_coarse_numelem2 = data1(11,1); % Number of elements in
time direction with coarse mesh

SBC_medium_numelem1 = data1(12,1); % Number of elements in
axial direction with medium mesh
SBC_medium_numelem2 = data1(13,1); % Number of elements in
time direction with medium mesh

SBC_fine_numelem1 = data1(14,1); % Number of elements in
axial direction with fine mesh
SBC_fine_numelem2 = data1(15,1); % Number of elements in
time direction with fine mesh

SBC_lamda = h*A/(mf*cf); % Nondimensional length lamda,
equation (3.3)
```

```
SBC_phi = h*A*P/(ms*cs); % Nondimensional time length phi,
equation (3.5)
```

```
% SUBCODE FOR NUMERICAL INTEGRATION WITH TRAPEZOIDAL METHOD
(SBC)
%
% This subroutine calculates solid mean temperature on
boundary 3 with trapezoidal method


model.study('std1').run;


data1 =
mpheval(model,'Ts','edim','boundary','selection',3); % Load
solid temperatures on boundary 3
data2 =
mpheval(model,'Tf','edim','boundary','selection',3); % Load
fluid temperatures on boundary 3

ind = data1.p(1,:)'; % Nondimensional position coordinate
nodes, equation (3.4)
Tsnewh = data1.d1'; % Solid temperatures
Tfnewh = data2.d1'; % Fluid temperatures

data1h = [ind Tsnewh];
data2h = [ind Tfnewh];
data11h = sortrows(data1h); % Sorts the rows of data1h in
ascending order
data22h = sortrows(data2h); % Sorts the rows of data2h in
ascending order

ind2h = data11h(:,1); % Shorted nondimensional position
coordinate nodes
Tshnew = data11h(:,2); % Shorted solid temperatures
Tfhnew = data22h(:,2); % Shorted fluid temperatures

Tshnewm = trapz(ind2h,Tshnew)/(ind2h(end)- ind2h(1)); %
Solid mean temperature calculated with trapezoidal method,
equation (3.18)


data5 = load('C:\T.txt');
data6 = [ind2h Tshnew];
```

```
data7 = [ind2h Tfhnew];
data8 = Tshnewm;

h1 = figure;
mphplot(model,'pg1','rangenum',1) % Plot fluid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
hgsave(h1,'SBC_Tf','-v6')

h2 = figure;
mphplot(model,'pg2','rangenum',1) % Plot solid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
hgsave(h2,'SBC_Ts','-v6')
```

```
% SUBCODE FOR POST PROCESSING (SBC)
%
% This subroutine post processes the calculated SBC values

lamda = 2; % Refinement ratio or scale factor
Fs = 1.25; % Factor of safety


Ts_0 = 0.977177; % Given exact solution point Ts(0,phi),
equation (3.34)
Tf_lamda = 0.853587; % Given exact solution point
Tf(lamda,phi), equation (3.35)
Ts_lamda = 0.727046; % Given exact solution point
Ts(lamda,phi), equation (3.35)
Ts_mean = 0.863840; % The solid mean temperature on line
AB, equation (3.36)


data61 = load('C:\SBC_Ts_coarse.txt'); % Load solid
temperatures with coarse mesh
data62 = load('C:\SBC_Ts_medium.txt'); % Load solid
temperatures with medium mesh
data63 = load('C:\SBC_Ts_fine.txt'); % Load solid
temperatures with fine mesh


data71 = load('C:\SBC_Tf_coarse.txt'); % Load fluid
temperatures with coarse mesh
data72 = load('C:\SBC_Tf_medium.txt'); % Load fluid
temperatures with medium mesh
data73 = load('C:\SBC_Tf_fine.txt'); % Load fluid
temperatures with fine mesh


data81 = load('C:\SBC_Tsmean_coarse.txt'); % Load solid
mean temperature with coarse mesh
data82 = load('C:\SBC_Tsmean_medium.txt'); % Load solid
mean temperature with medium mesh
data83 = load('C:\SBC_Tsmean_fine.txt'); % Load solid mean
temperature with fine mesh


Tsc = data61(1,2); % Solid temperature with coarse mesh at
```

```
point A(0,phi)
Tsm = data62(1,2); % Solid temperature with medium mesh at
point A(0,phi)
Tsf = data63(1,2); % Solid temperature with fine mesh at
point A(0,phi)


Tsmeanc = data81; % Solid mean temperature with coarse mesh
on line AB, equation (3.18)
Tsmeanm = data82; % Solid mean temperature with medium mesh
on line AB, equation (3.18)
Tsmeanf = data83; % Solid mean temperature with fine mesh
on line AB, equation (3.18)


disp(['Results for Ts at A']);


eacr = (Ts_0 - Tsc)/Ts_0 * 100 % Actual relative
(percentage) error with coarse mesh, equation (3.30)
eamr = (Ts_0 - Tsm)/Ts_0 * 100 % Actual relative
(percentage) error with medium mesh, equation (3.30)
eafr = (Ts_0 - Tsf)/Ts_0 * 100 % Actual relative
(percentage) error with fine mesh, equation (3.30)


efm = Tsf - Tsm; % An error measure, equation (3.30)
efmr = (Tsf - Tsm)/Tsf * 100; % Relative (percentage)
error, equation (3.26)


r = log((Tsm - Tsc)/(Tsf - Tsm))/log(lamda) % Effective
convergence rate, equation (3.23)
er = Fs * abs(efmr)/(lamda^r-1) % Relative (percentage)
error estimate, equation (3.28)


disp(['Results for Tf at B']);


Tfc = data71(end,2); % Fluid temperature with coarse mesh
at point B(0,phi)
Tfm = data72(end,2); % Fluid temperature with medium mesh
at point B(0,phi)
Tff = data73(end,2); % Fluid temperature with fine mesh at
```

```
point B(0,phi)
eacr = (Tf_lamda - Tfc)/Tf_lamda * 100 % Actual relative
(percentage) error with coarse mesh, equation (3.30)
eamr = (Tf_lamda - Tfm)/Tf_lamda * 100 % Actual relative
(percentage) error with medium mesh, equation (3.30)
eafr = (Tf_lamda - Tff)/Tf_lamda * 100 % Actual relative
(percentage) error with fine mesh, equation (3.30)


efm = Tff - Tfm; % An error measure, equation (3.25)
efmr = (Tff - Tfm)/Tff * 100; % Relative (percentage)
error, equation (3.26)
r = log((Tfm - Tfc)/(Tff - Tfm))/log(lamda) % Effective
convergence rate, equation (3.23)
er = Fs * abs(efmr)/(lamda^r-1) % Relative (percentage)
error estimate, equation (3.28)


disp(['Results for Ts at B']);


Tsc = data61(end,2); % Solid temperature with coarse mesh
at point B(lamda,phi)
Tsm = data62(end,2); % Solid temperature with medium mesh
at point B(lamda,phi)
Tsf = data63(end,2); % Solid temperature with fine mesh at
point B(lamda,phi)


eacr = (Ts_lamda - Tsc)/Ts_lamda * 100 % Actual relative
(percentage) error with coarse mesh, equation (3.30)
eamr = (Ts_lamda - Tsm)/Ts_lamda * 100 % Actual relative
(percentage) error with medium mesh, equation (3.30)
eafr = (Ts_lamda - Tsf)/Ts_lamda * 100 % Actual relative
(percentage) error with fine mesh, equation (3.30)


efm = Tsf - Tsm; % An error measure, equation (3.30)
efmr = (Tsf - Tsm)/Tsf * 100; % Relative (percentage)
error, equation (3.26)
r = log((Tsm - Tsc)/(Tsf - Tsm))/log(lamda) % Effective
convergence rate, equation (3.23)
er = Fs * abs(efmr)/(lamda^r-1) % Relative (percentage)
error estimate, equation (3.28)
```

```
disp(['Results for average Ts on line AB']);


eacr = (Ts_mean - Tsmeanc)/Ts_mean * 100 % Actual relative
(percentage) error with coarse mesh, equation (3.30)
eamr = (Ts_mean - Tsmeanm)/Ts_mean * 100 % Actual relative
(percentage) error with medium mesh, equation (3.30)
eafr = (Ts_mean - Tsmeanf)/Ts_mean * 100 % Actual relative
(percentage) error with fine mesh, equation (3.30)


efm = Tsmeanf - Tsmeanm; % An error measure, equation
(3.25)
efmr = (Tsmeanf - Tsmeanm)/Tsmeanf * 100; % Relative
(percentage) error, equation (3.26)


r = log((Tsmeanm - Tsmeanc)/(Tsmeanf - Tsmeanm))/log(lamda)
% Effective convergence rate, equation (3.23)
er = Fs * abs(efmr)/(lamda^r-1) % Relative (percentage)
error estimate, equation (3.28)
```

```
% SUBCODE FOR DIMENSIONAL TEMPERATURES (SBC)
%
% This subroutine calculates dimensional temperatures

disp(['Dimensional fluid and solid temperatures']);

data1 = load('C:\SBCdata.txt'); % Load SBC data
t0 = data1(8,1); % Dimensional constant initial storage
material temperature, equation (3.1)
tfi = data1(9,1); % Dimensional constant fluid inflow
temperature, equation (3.2)

data61 = load('C:\SBC_Ts_coarse.txt'); % Load
nondimensional storage material temperatures with coarse
mesh
data62 = load('C:\SBC_Ts_medium.txt'); % Load
nondimensional storage material temperatures with medium
mesh
data63 = load('C:\SBC_Ts_fine.txt'); % Load nondimensional
storage material temperatures with fine mesh

data71 = load('C:\SBC_Tf_coarse.txt'); % Load
nondimensional fluid temperatures with coarse mesh
data72 = load('C:\SBC_Tf_medium.txt'); % Load
nondimensional fluid temperatures with medium mesh
data73 = load('C:\SBC_Tf_fine.txt'); % Load nondimensional
fluid temperatures with fine mesh

[xc] = data61(:,1)/SBC_lamda;
[xm] = data62(:,1)/SBC_lamda;
[xf] = data63(:,1)/SBC_lamda;

[tsc] = data61(:,2)*(tfi-t0)+t0; % Dimensional solid
temperatures with coarse mesh, equation (A.14)
[tsm] = data62(:,2)*(tfi-t0)+t0; % Dimensional solid
temperatures with medium mesh, equation (A.14)
[tsf] = data63(:,2)*(tfi-t0)+t0; % Dimensional solid
temperatures with fine mesh, equation (A.14)
```

```
[tfc] = data71(:,2)*(tfi-t0)+t0; % Dimensional fluid
temperatures with coarse mesh, equation (A.14)
[tfm] = data72(:,2)*(tfi-t0)+t0; % Dimensional fluid
temperatures with medium mesh, equation (A.14)
[tff] = data73(:,2)*(tfi-t0)+t0; % Dimensional fluid
temperatures with fine mesh, equation (A.14)


data81 = [xc tfc];
data82 = [xm tfm];
data83 = [xf tff];


data91 = [xc tsc];
data92 = [xm tsm];
data93 = [xf tsf];


subplot(1,2,1)
plot(xc,tfc,'--d',xm,tfm,'-.*',xf,tff,'-
ro','LineWidth',2,'MarkerEdgeColor','k') % Plot dimensional
fluid temperatures
xlabel('{\itx/L} [-]', 'FontSize', 12);
ylabel('{\itt}_f [\circ{C}]', 'FontSize', 12);
legend('tf with coarse mesh','tf with medium mesh','tf with
fine mesh')
axis([0 1 10 81]);

subplot(1,2,2)
plot(xc,tsc,'--d',xm,tsm,'-.*',xf,tsf,'-
ro','LineWidth',2,'MarkerEdgeColor','k') % Plot dimensional
solid temperatures
xlabel('{\itx/L} [-]', 'FontSize', 12);
ylabel('{\itt}_s [\circ{C}]', 'FontSize', 12);
legend('ts with coarse mesh','ts with medium mesh','ts with
fine mesh')
axis([0 1 10 81]);


save SBC_tf_coarse_dimensional_temperatures.txt data81 -
ascii -tabs % Save dimensional fluid temperatures with
coarse mesh
```

```
save SBC_tf_medium_dimensional_temperatures.txt data82 -ascii -tabs % Save dimensional fluid temperatures with medium mesh
save SBC_tf_fine_dimensional_temperatures.txt data83 -ascii -tabs % Save dimensional fluid temperatures with fine mesh for post processing

save SBC_ts_coarse_dimensional_temperatures.txt data91 -ascii -tabs % Save dimensional solid temperatures with coarse mesh
save SBC_ts_medium_dimensional_temperatures.txt data92 -ascii -tabs % Save dimensional solid temperatures with medium mesh
save SBC_ts_fine_dimensional_temperatures.txt data93 -ascii -tabs % Save dimensional solid temperatures with fine mesh
```

MATLAB codes for REG

```
% MAIN CODE FOR REGENERATOR PROBLEM CASE (REG)

% This main code determines REG problem case temperature
distribution by using COMSOL 4.3 with MATLAB program
% The main code includes seven subroutines: (1) the subcode
for given data, (2) the subcode for initial coordinate
nodes, (3) the subcode for while-loop, (4) the subcode for
hot period, (5) the subcode for_cold period, (6) the
subcode for post processing, and (7) the subcode for
dimensional fluid and solid temperatures
% The equation numbers, referred to, are the same as in the
main part of the report

clear all

format long

model = mphload('REG_COMSOL_Multiphysics_model'); % Load
COMSOL Multiphysics REG model (m-file)

REGdata % Subcode for given data

% Coarse mesh
disp(['coarse mesh']);
e=1;
data1 = load('C:\Tinitial.txt'); % Load initial values of
interpolation function for boundary condition
save T.txt data1 -ascii -tabs % Save initial values for
boundary condition
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',REG_coarse_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',REG_coarse_numelem2); % Set number of elements in
time direction
```

```
REG_initial_coordinate_nodes % Subcode for initial
coordinate nodes

REG_while_loop % Subcode for while-loop

data2 = load('C:\etaREG.txt'); % Load etaREG.txt file
data21 = [REG_lamdah REG_lamdac REG_phih REG_phic delta
etaREGh etaREGc Number_of_iteration]; % Lamdah, lamdac,
phih, phic, delta, etaREGh, etaREGc and number of total
cycles needed for convergence

save REGcoarse.txt data21 -ascii -tabs % Save REG values
for post processing
save REG_Ts_coarse.txt data61h -ascii -tabs % Save
nondimensional storage material temperatures for post
processing
save REG_Tf_coarse.txt data81h -ascii -tabs % Save
nondimensional fluid temperatures for post processing

% Medium_mesh
disp(['medium mesh']);
e=1;
data1 = load('C:\Tinitial.txt'); % Load initial values of
interpolation function for boundary condition
save T.txt data1 -ascii -tabs % Save initial values for
boundary condition
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',REG_medium_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',REG_medium_numelem2); % Set number of elements in
time direction

REG_initial_coordinate_nodes % Subcode for initial
coordinate nodes

REG_while_loop % Subcode for while-loop

data3 = load('C:\etaREG.txt'); % Load etaREG.txt file
```

```
data31 = [REG_lamdah REG_lamdac REG_phih REG_phic delta
etaREGh etaREGc Number_of_iteration]; % Lamdah, lamdac,
phih, phic, delta, etaREGh, etaREGc and number of total
cycles needed for convergence


save REGmedium.txt data31 -ascii -tabs % Save REG values
for post processing
save REG_Ts_medium.txt data61h -ascii -tabs % Save
nondimensional storage material temperatures for post
processing
save REG_Tf_medium.txt data81h -ascii -tabs % Save
nondimensional fluid temperatures for post processing


% Fine mesh
disp(['fine mesh']);
e=1;
data1 = load('C:\Tinitial.txt'); % Load initial values of
interpolation function for boundary condition
save T.txt data1 -ascii -tabs % Save initial values for
boundary condition
model.mesh('mesh1').feature('map1').feature('dis1').set('nu
melem',REG_fine_numelem1); % Set number of elements in
axial direction
model.mesh('mesh1').feature('map1').feature('dis2').set('nu
melem',REG_fine_numelem2); % Set number of elements in time
direction


REG_initial_coordinate_nodes % Subcode for initial
coordinate nodes


REG_while_loop % Subcode for while-loop


data4 = load('C:\etaREG.txt'); % Load etaREG.txt file
data41 = [REG_lamdah REG_lamdac REG_phih REG_phic delta
etaREGh etaREGc Number_of_iteration]; % Lamdah, lamdac,
phih, phic, delta, etaREGh, etaREGc and number of total
cycles needed for convergence


save REGfine.txt data41 -ascii -tabs % Save REG values for
post processing
```

```
save REG_Ts_fine.txt data61h -ascii -tabs % Save
nondimensional storage material temperatures for post
processing
save REG_Tf_fine.txt data81h -ascii -tabs % Save
nondimensional fluid temperatures for post processing


REG_post_processing % Subcode for post processing


REG_dimensional_temperatures % Subcode for dimensional
fluid and solid temperatures
```

```
% SUBCODE FOR GIVEN DATA (REG)
%
% This subroutine loads the given REG values from the data
matrix

data5 = load('C:\REGdata.txt'); % Load given REG data

A = data5(1,1); % Total heat transfer surface area, m^2
ms = data5(2,1); % Total storage medium mass, kg
cs = data5(3,1); % Medium specific heat at constant
pressure, kJ/kg K
cf = data5(4,1); % Fluid specific heat at constant
pressure, kJ/kg K
mfh = data5(5,1); % Total mass rate of fluid in hot period,
kg/s
mfc = data5(6,1); % Total mass rate of fluid in cold
period, kg/s
hh = data5(7,1); % Overall heat transfer coefficient in hot
period, W/m^2 K
hc = data5(8,1); % Overall heat transfer coefficient in
cold period, W/m^2 K
Ph = data5(9,1); % Hot period, s
Pc = data5(10,1); % Cold period, s
delta = data5(11,1); % Given stopping criteria

Tshnewmo = data5(12,1); % Initial value for Tshnewmo
temperature for equation (4.28)
Tscnewmo = data5(13,1); % Initial value for Tscnewmo
temperature for equation (4.29)

REG_coarse_numelem1 = data5(14,1); % Number of elements in
axial direction with coarse mesh
REG_coarse_numelem2 = data5(15,1); % Number of elements in
time direction with coarse mesh

REG_medium_numelem1 = data5(16,1); % Number of elements in
axial direction with medium mesh
REG_medium_numelem2 = data5(17,1); % Number of elements in
time direction with medium mesh
```

```
REG_fine_numelem1 = data5(18,1); % Number of elements in
axial direction with fine mesh
REG_fine_numelem2 = data5(19,1); % Number of elements in
time direction with fine mesh


REG_lamdah = hh*A/(mfh*cf); % Nondimensional length lamda
for hot period, equation (4.1)
REG_lamdac = hc*A/(mfc*cf); % Nondimensional length lamda
for cold period, equation (4.1)


REG_phih = hh*A*Ph/(ms*cs); % Nondimensional time length
phi for hot period, equation (4.3)
REG_phic = hc*A*Pc/(ms*cs); % Nondimensional time length
phi for cold period, equation (4.3)


tfih = data5(24,1); % Dimensional constant fluid inflow
temperature during hot period, equations (4.5) with one
dash
tfic = data5(25,1); % Dimensional constant fluid inflow
temperature during cold period, equations (4.5) with double
dash


Ratioh = REG_lamdah/REG_phih; % Ratio of nondimensional
length lamda to nondimensional time length phi during hot
period
Ratioc = REG_lamdac/REG_phic; % Ratio of nondimensional
length lamda and nondimensional time length phi during cold
period
```

```
% SUBCODE FOR INITIAL COORDINATE NODES FOR COLD PERIOD
(REG)
%
% This subroutine determines initial coordinate nodes

model.param.set('lamda',REG_lamdah); % Set nondimensional
length lamda for hot period, equation (4.1)
model.param.set('phi',REG_phih); % Set nondimensional time
length phi for cold period, equation (4.3)
model.param.set('d','-1'); % Set parameter for sign of Tf,
equations (4.8) and (4.12)

model.physics('w').feature('cons1').active(false); %
Boundary condition for cold period, equation (4.10)
model.physics('w').feature('cons2').active(true); %
Boundary condition for cold period, equation (4.14)
model.study('std1').run;


data6 =
mpheval(model,'Ts','edim','boundary','selection',3); % Load
solid temperatures on boundary 3
data7 =
mpheval(model,'Ts','edim','boundary','selection',2); % Load
solid temperatures on boundary 2
data8 =
mpheval(model,'Tf','edim','boundary','selection',3); % Load
fluid temperatures on boundary 3

indc = data6.p(1,:)'; % Initial nondimensional coordinate
nodes, equation (4.2)
```

```
% SUBCODE FOR WHILE-LOOP (REG)
%
% This subroutine calculates regenerator temperature
distribution in hot and cold periods and stops calculation,
when the given error limit (stopping criteria) is reached

Number_of_iteration = 0; % Initial iteration number


tic % Record elapsed time
while (e>delta)

  REG_hot % Subcode for hot period
  subplot(2,2,1)
  mphplot(model,'pg1','rangenum',1) % Plot fluid
temperatures
  xlabel('{\it\Lambda} [-]', 'FontSize', 12);
  ylabel('{\it\Pi} [-]', 'FontSize', 12);
  subplot(2,2,2)
  mphplot(model,'pg2','rangenum',1) % Plot solid
temperatures
  xlabel('{\it\Lambda} [-]', 'FontSize', 12);
  ylabel('{\it\Pi} [-]', 'FontSize', 12);

  REG_cold % Subcode for cold period
  subplot(2,2,3)
  mphplot(model,'pg1','rangenum',1) % Plot fluid
temperatures
  xlabel('{\it\Lambda} [-]', 'FontSize', 12);
  ylabel('{\it\Pi} [-]', 'FontSize', 12);
  subplot(2,2,4)
  mphplot(model,'pg2','rangenum',1) % Plot solid
temperatures
  xlabel('{\it\Lambda} [-]', 'FontSize', 12);
  ylabel('{\it\Pi} [-]', 'FontSize', 12);

  if (eh<delta) && (ec<delta)
    e>delta % Stopping criteria
```

```
    e = sum(eh + ec)/2; % Mean value of stopping criteria
(mean value of hot and cold periods)
  else
    e<delta;
  end


  Number_of_iteration = Number_of_iteration + 1; %
Calculates the number of total cycles needed for
convergence

end
toc % Elapsed time


etaREGh = REG_lamdah/REG_phih*(Tshnewm - Tsholdm) % Thermal
ratio for hot period, equation (4.28)
etaREGc = REG_lamdac/REG_phic*(Tscoldm - Tscnewm) % Thermal
ratio for cold period, equation (4.29)

Number_of_iteration % Number of total cycles needed for
convergence
```

```
% SUBCODE FOR HOT PERIOD (REG)

% This subroutine calculates regenerator solid mean
temperature on boundary 3 in hot period

disp(['hot period']);

model.func('int1').set('filename','C:\T.txt'); % Set solid
nodal temperatures for boundary condition, equation (4.10)
model.param.set('lamda',REG_lamdah); % Set nondimensional
length lamda for hot period, equation (4.1)
model.param.set('phi',REG_phih); % Set nondimensional time
length phi for hot period, equation (4.3)
model.param.set('d','1'); % Set parameter for sign of Tf,
equations (4.8) and (4.12)

model.physics('w').feature('cons1').active(true); %
Boundary condition for hot period, equation (4.10)
model.physics('w').feature('cons2').active(false); %
Boundary condition for hot period, equation (4.14)
model.study('std1').run;

data6 =
mpheval(model,'Ts','edim','boundary','selection',3); % Load
solid temperatures on boundary 3
data7 =
mpheval(model,'Ts','edim','boundary','selection',2); % Load
solid temperatures after previous cold period on boundary 2
data8 =
mpheval(model,'Tf','edim','boundary','selection',3); % Load
fluid temperatures on boundary 3

indh = data6.p(1,:)' % Nondimensional position coordinate
nodes, equation (4.2)
Tsnewh = data6.d1' % Solid temperatures
Tsoldh = data7.d1' % Solid temperatures after previous cold
period
Tfnewh = data8.d1'; % Fluid temperatures
```

```
data6h = [indh Tsnewh];
data7h = [indh Tsoldh];
data8h = [indh Tfnewh];

data61h = sortrows(data6h); % Sorts the rows of data6h in
ascending order
data71h = sortrows(data7h); % Sorts the rows of data7h in
ascending order
data81h = sortrows(data8h); % Sorts the rows of data8h in
ascending order

ind2h = data61h(:,1); % Sorted nondimensional position
coordinate nodes
Tshnew = data61h(:,2); % Sorted solid temperatures
Tshold = data71h(:,2); % Sorted solid temperatures after
cold period
Tfhnew = data81h(:,2); % Sorted fluid temperatures

Tshnewm = trapz(ind2h,Tshnew)/(ind2h(end)- ind2h(1)); %
Solid mean temperature calculated with trapezoidal method,
equation (3.18)
Tsholdm = trapz(ind2h,Tshold)/(ind2h(end)- ind2h(1)); %
Solid mean temperature after previous cold period
calculated with trapezoidal method, equation (3.18)

eh = abs((Tshnewmo - Tshnewm)/Tshnewmo) % Stopping criteria

Tshnewmo = Tshnewm; % Save solid mean temperature for next
hot period

Ts1 = Tsnewh(1);
Ts2 = Tsnewh(2);
Tsnewh (1:2,1) = [Ts2 Ts1];
data9h = load('C:\T.txt'); % Load interpolation function
T.txt file

data91h = [indc Tsnewh];
data10h = [indc Tfnewh];
```

```
save T.txt data91h -ascii -tabs % Save solid temperatures
for boundary condition of next cold period
```

```
% SUBCODE FOR COLD PERIOD (REG)

% This subroutine calculates regenerator solid mean
temperature on boundary 3 in cold period

disp(['cold period']);

model.func('int1').set('filename','C:\T.txt'); % Set solid
nodal temperatures for boundary condition, equation (4.14)
model.param.set('lamda',REG_lamdac); % Set nondimensional
length lamda for hot period, equation (4.1)
model.param.set('phi',REG_phic); % Set nondimensional time
length phi for cold period, equation (4.3)
model.param.set('d','-1'); % Set parameter for sign of Tf,
equations (4.8) and (4.12)

model.physics('w').feature('cons1').active(false); %
Boundary condition for cold period, equation (4.10)
model.physics('w').feature('cons2').active(true); %
Boundary condition for cold period, equation (4.14)
model.study('std1').run;

data6 =
mpheval(model,'Ts','edim','boundary','selection',3); % Load
solid temperatures on boundary 3
data7 =
mpheval(model,'Ts','edim','boundary','selection',2); % Load
solid temperatures after previous hot period on boundary 2
data8 =
mpheval(model,'Tf','edim','boundary','selection',3); % Load
fluid temperatures on boundary 3

indc = data6.p(1,:)' % Nondimensional position coordinate
nodes, equation (4.2)
Tsnewc = data6.d1' % Solid temperatures
Tsoldc = data7.d1' % Solid temperatures after previous hot
period
Tfnewc = data8.d1'; % Fluid temperatures
```

```
data6c = [indc Tsnewc];
data7c = [indc Tsoldc];
data8c = [indc Tfnewc];
data61c = sortrows(data6c); % Sorts the rows of data6c in
ascending order
data71c = sortrows(data7c); % Sorts the rows of data7c in
ascending order
data81c = sortrows(data8c); % Sorts the rows of data8c in
ascending order

ind2c = data61c(:,1); % Sorted nondimensional position
coordinate nodes
Tscnew = data61c(:,2); % Sorted solid temperatures
Tscold = data71c(:,2); % Sorted solid temperatures after
hot period
Tfcnew = data81c(:,2); % Sorted fluid temperatures

Tscnewm = trapz(ind2c,Tscnew)/(ind2c(end)- ind2c(1)); %
Solid mean temperature calculated with trapezoidal method,
equation (3.18)
Tscoldm = trapz(ind2c,Tscold)/(ind2c(end)- ind2c(1)); %
Solid mean temperature after previous hot period calculated
with trapezoidal method, equation (3.18)

ec = abs((Tscnewmo - Tscnewm)/Tscnewmo) % Stopping criteria

Tscnewmo = Tscnewm; % Save solid mean temperature for next
cold period

Ts1 = Tsnewc(1);
Ts2 = Tsnewc(2);
Tsnewc (1:2,1) = [Ts2 Ts1];
data9c = load('C:\T.txt'); % Load interpolation function
T.txt file

data91c = [indh Tsnewc];
data10c = [indh Tfnewc];
```

```
save T.txt data91c -ascii -tabs % Save solid temperatures
for boundary condition of next hot period
```

```
% SUBCODE FOR POST PROCESSING (REG)
%
% This subroutine post processes the calculated REG values

lamda = 2; % Refinement ratio or scale factor
Fs = 1.25; % Factor of safety


data21 = load('C:\REGcoarse.txt'); % Load REG values with
coarse mesh
data31 = load('C:\REGmedium.txt'); % Load REG values with
medium mesh
data41 = load('C:\REGfine.txt'); % Load REG values with
fine mesh


etaREGhcoarse = data21(6); % Thermal ratio with coarse mesh
after hot period, equation (4.26)
etaREGccoarse = data21(7); % Thermal ratio with coarse mesh
after cold period, equation (4.27)


etaREGhmedium = data31(6); % Thermal ratio with medium mesh
after hot period, equation (4.26)
etaREGcmedium = data31(7); % Thermal ratio with medium mesh
after cold period, equation (4.27)


etaREGhfine = data41(6); % Thermal ratio with fine mesh
after hot period, equation (4.26)
etaREGcfine = data41(7); % Thermal ratio with fine mesh
after cold period, equation (4.27)


rh = log((etaREGhmedium - etaREGhcoarse)/(etaREGhfine -
etaREGhmedium))/log(lamda) % Effective convergence rate for
hot period, equation (3.23)
rc = log((etaREGcmedium - etaREGccoarse)/(etaREGcfine -
etaREGcmedium))/log(lamda) % Effective convergence rate for
cold period, equation (3.23)


efmrh = (etaREGhfine - etaREGhmedium)/etaREGhfine * 100; %
Relative (percentage) error for hot period, equation (3.26)
```

```
efmrc = (etaREGcfine - etaREGcmedium)/etaREGcfine * 100; %
Relative (percentage) error for cold period, equation
(3.26)

erh = Fs * abs(efmrh)/(lamda^rh-1) % Relative (percentage)
error for hot period, equation (3.28)
erc = Fs * abs(efmrc)/(lamda^rc-1) % Relative (percentage)
error for cold period, equation (3.28)

REG_hot % Subcode for hot period
h1 = figure;
mphplot(model,'pg1','rangenum',1) % Plot fluid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
axis([-1 2.7 0 4.08]);
hgsave(h1,'REG_Tf_hot','-v6')
h2 = figure;
mphplot(model,'pg2','rangenum',1) % Plot solid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
axis([-1 2.7 0 4.08]);
hgsave(h2,'REG_Ts_hot','-v6')

REG_cold % Subcode for cold period
h3 = figure;
mphplot(model,'pg1','rangenum',1) % Plot fluid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
axis([-1 2.7 0 4.08]);
hgsave(h3,'REG_Tf_cold','-v6')
h4 = figure;
mphplot(model,'pg2','rangenum',1) % Plot solid temperatures
xlabel('{\it\xi} [-]', 'FontSize', 12);
ylabel('{\it\eta} [-]', 'FontSize', 12);
axis([-1 2.7 0 4.08]);
hgsave(h4,'REG_Ts_cold','-v6')
```

```
% SUBCODE FOR DIMENSIONAL TEMPERATURES (REG)
%
% This subroutine calculates dimensional temperatures

disp(['Dimensional fluid and solid temperatures']);

tfih = data5(24,1); % Dimensional constant fluid inflow
temperature during hot period, equations (4.5) with one
dash
tfic = data5(25,1); % Dimensional constant fluid inflow
temperature during cold period, equations (4.5) with double
dash

data11 = load('C:\REG_Tf_coarse.txt'); % Load
nondimensional fluid temperatures with coarse mesh
data12 = load('C:\REG_Tf_medium.txt'); % Load
nondimensional fluid temperatures with medium mesh
data13 = load('C:\REG_Tf_fine.txt'); % Load nondimensional
fluid temperatures with fine mesh

data21 = load('C:\REG_Ts_coarse.txt'); % Load
nondimensional solid temperatures with coarse mesh
data22 = load('C:\REG_Ts_medium.txt'); % Load
nondimensional solid temperatures with medium mesh
data23 = load('C:\REG_Ts_fine.txt'); % Load nondimensional
solid temperatures with fine mesh

[xc] = data11(:,1)/REG_lamdah;
[xm] = data12(:,1)/REG_lamdah;
[xf] = data13(:,1)/REG_lamdah;

[tfc] = data11(:,2)*(tfih-tfic)+tfic; % Dimensional fluid
temperatures with coarse mesh, equations (4.5)
[tfm] = data12(:,2)*(tfih-tfic)+tfic; % Dimensional fluid
temperatures with medium mesh, equations (4.5)
[tff] = data13(:,2)*(tfih-tfic)+tfic; % Dimensional fluid
temperatures with fine mesh, equations (4.5)
```

```
[tsc] = data21(:,2)*(tfih-tfic)+tfic; % Dimensional solid
temperatures with coarse mesh, equations (4.6)
[tsm] = data22(:,2)*(tfih-tfic)+tfic; % Dimensional solid
temperatures with medium mesh, equations (4.6)
[tsf] = data23(:,2)*(tfih-tfic)+tfic; % Dimensional solid
temperatures with fine mesh, equations (4.6)


data14 = [xc tfc];
data15 = [xm tfm];
data16 = [xf tff];


data24 = [xc tsc];
data25 = [xm tsm];
data26 = [xf tsf];


subplot(1,2,1)
plot(xc,tfc,'--d',xm,tfm,'-.*',xf,tff,'-
ro','LineWidth',2.3,'MarkerEdgeColor','k') % Plot
dimensional fluid temperatures
xlabel('{\itx/L} [-]', 'FontSize', 12);
ylabel('{\itt}_f [ \circ{C}]', 'FontSize', 12);
legend('tf with coarse mesh','tf with medium mesh','tf with
fine mesh')
axis([0 1 60 81]);

subplot(1,2,2)
plot(xc,tsc,'--d',xm,tsm,'-.*',xf,tsf,'-
ro','LineWidth',2.3,'MarkerEdgeColor','k') % Plot
dimensional solid temperatures
xlabel('{\itx/L} [-]', 'FontSize', 12);
ylabel('{\itt}_s [ \circ{C}]', 'FontSize', 12);
legend('ts with coarse mesh','ts with medium mesh','ts with
fine mesh')
axis([0 1 60 81]);


save REG_tf_coarse_dimensional_temperatures.txt data14 -
ascii -tabs % Save dimensional fluid temperatures with
coarse mesh
```

```
save REG_tf_medium_dimensional_temperatures.txt data15 -
ascii -tabs % Save dimensional fluid temperatures with
medium mesh
save REG_tf_fine_dimensional_temperatures.txt data16 -ascii
-tabs % Save dimensional fluid temperatures with fine mesh


save REG_ts_coarse_dimensional_temperatures.txt data24 -
ascii -tabs % Save dimensional solid temperatures with
coarse mesh
save REG_ts_medium_dimensional_temperatures.txt data25 -
ascii -tabs % Save dimensional solid temperatures with
medium mesh
save REG_ts_fine_dimensional_temperatures.txt data26 -ascii
-tabs % Save dimensional solid temperatures with fine mesh
```

Heat recovery is a topical issue, as it affects buildings' energy efficiency. Heat recovery equipment is used to improve energy efficiency in various industrial processes and HVAC systems. A typical application of the heat recovery is a ventilation system, in which warm exhaust air heats the supply air. This report focuses on the regenerative heat exchanger theory and computational modelling, in which mass is alternately heated and cooled by changing the direction of air flow.

This report presents an unusual approach to solving the regenerator temperature distribution: instead of using stepwise marching in time, we employ the finite element method using space-time elements and the Galerkin method. We also present it as a pure boundary value problem. Programming and simulation uses the commercial COMSOL Multiphysics and MATLAB programs (COMSOL 4.3 with MATLAB). The COMSOL Multiphysics models and MATLAB codes, which are used in the report applications, are given. The developed model can be used as a tool in the design and optimization of heat exchangers.

Työterveyslaitos
Arbetshälsoinstitutet
Finnish Institute of Occupational Health

Topeliuksenkatu 41 a A

FI-00250 Helsinki

www.ttl.fi

ISBN 978-952-261-462-9 (nid.)
ISBN 978-952-261-430-8 (PDF)

Työterveyslaitos | Arbetshälsoinstitutet Finnish Institute of Occupational Health

Aalto University