CHAPTER 4

Mathematical Models and Numerical Solutions for Thermal Storage Processes

Contents

Abstract

This chapter presents information on mathematical models for thermal storage, covering the establishing of proper governing equations to mathematically follow the energy conservation principles for "control volumes" in a thermal storage tank when heat is charged or withdrawn; deciding the boundary condition requirements for the governing equations; and discovering the most efficient mathematical method to solve the governing equations with accuracy, so that the temperatures of fluid and solid media can be determined at any location in the tank at any time. The models address several

Thermal Energy Storage Analyses and Designs
http://dx.doi.org/10.1016/B978-0-12-805344-7.00004-3

configurations of thermal storage systems, including single fluid thermal storage, dual media (fluid and packed bed) sensible thermal storage, and dual-media PCM-based latent heat thermal storage. In the dual-media sensible thermal storage, both configurations of solid particle packed bed and integrated solid with fluid pipes passing through are considered and discussed. The transient heat transfer in a packed bed is typically assumed uniform in the radial direction in a storage tank and thus one- or two-dimensional governing equations are sufficient to describe the problems. For convenient design analysis, the models described in the following paragraphs start from a one-dimensional model which has been proven to have no sacrifice of accuracy. Models in three dimensions considering nonuniform flow are briefly introduced at the end of the chapter.

Keywords: Ideal thermal storage, Energy delivery efficiency, Thermal energy storage modeling, PCM, Enthalpy method, Method of characteristics

Nomenclature

a_f the cross-sectional area of a storage tank (m^2)
Bi Biot number ($= L_p h/k_s$)
C heat capacity (J/kg °C)
d_r nominal diameter of a single filler "particle" (rocks) (m)
$\bar{h}$ enthalpy (J/kg)
H heat transfer coefficient (W/m^2 °C)
h_{eff} effective heat transfer coefficient (W/m^2 °C)
H length or height of a storage tank (m)
H_{CR} a dimensionless parameter, Eq. (4.12)
k thermal conductivity (W/m °C)
L latent heat of fusion of storage PCM
L_p characteristic length of particles for Biot number
$\dot{m}$ mass flow rate (kg/s)
N number of tubes for HTF in a storage tank
Pr Prandtl number
R radius of the storage tank (m)
Re modified Reynolds number for porous media
S_s surface area of filler material per unit length of the storage tank (m)
t time (s)
T temperature (°C)
T_H high temperature of fluid from solar field (°C)
T_L low temperature of fluid from power plant (°C)
U fluid velocity in the axial direction in the storage tank (m/s), Eq. 4.4
V volume (m^3)
V_f volume of fluid per unit length of the storage tank (m^2)
V_s volume of filler material per unit length of the storage tank (m^2)
z location of a fluid element along the axis of the tank (m)

Greek Symbols

ε porosity of packed bed in a storage tank
η_s thermal storage efficiency

η_s dimensionless enthalpy of the encapsulated PCM
ν kinematic viscosity (m^2/s)
Π dimensionless charge or discharge time
τ_r a dimensionless parameter, see 4.9
ρ density (kg/m^3)
Θ dimensionless temperature
ϑ nodes of the numerical grid

Subscript

c energy charge process
d energy discharge process
f thermal fluid
final equilibrium temperature after each charge or discharge
ref a required reference value
s filler material (rocks), the primary thermal storage material
z location along the axis of the tank

Superscript

* dimensionless values

The most important goal of investigation and studies of thermal energy storage is to design the size of tanks for thermal storage of a certain energy demand over a desired period of time, and also to predict and understand the variation of fluid temperatures discharged from a storage tank. To accomplish the goal, we must rely on a fundamental study of the energy flow and the heat transfer between the heat transfer fluid and solid materials in a packed bed, if dual-media thermal storage is adopted.

The content of this section is a presentation of the mathematical modeling of such systems, including:

(1) Establishing proper governing equations to mathematically follow the energy conservation principles for "control volumes" in a thermal storage tank when heat is charged or withdrawn.
(2) Deciding the requirement of boundary conditions for the governing equations.
(3) Finding out the most efficient mathematical method to solve the governing equations with accuracy so that the temperatures of the fluid and solid media can be determined at any location in the tank at any time.

The models address several configurations of the thermal storage systems presented in Chapter 2, including single fluid thermal storage, dual-media (fluid and packed bed) sensible thermal storage, and dual-media phase

change material (PCM) based latent heat thermal storage. In the dual-media sensible thermal storage, both configurations of solid particle packed bed and integrated solid with fluid pipes passing through are considered and discussed.

The transient heat transfer in a packed bed is typically assumed uniform in the radial direction in a storage tank and thus one- or two-dimensional governing equations are sufficient to describe the problems. For convenient design analysis, the models described in the following subsections start from a one-dimensional model which has been proven to have no sacrifice of accuracy. Models in three dimensions considering nonuniform flow are briefly introduced at the end of the chapter.

4.1 IDEAL THERMAL STORAGE USING LIQUID ALONE

In an ideal thermal storage system, high temperature heat transfer fluid (HTF) is stored, and when it is withdrawn there should be no temperature degradation. Such a system requires that there be no heat loss and no heat transfer when the HTF is stored in or withdrawn from a tank.

Contingent upon the thermal insulation being perfectly maintained, the two-tank HTF storage system in Fig. 2.1 can operate like an ideal thermal energy storage system. It has been discussed before that a two-tank storage system can be replaced by a single-tank storage system, as shown in Fig. 2.2, in which a stratification of fluid (hot on top of cold), or a thermocline mechanism, must be maintained. However, even if the thermocline is maintained, the heat conduction between hot fluid and cold fluid may cause a temperature drop in the hot fluid, which will not allow for ideal thermal storage performance. A modification proposed by the current authors [1] uses a thermal insulation baffle in the single tank, which separates the hot fluid from the cold fluid, as shown in Fig. 4.1. In this system, if the floating thermal insulation baffle prevents heat conduction from the hot fluid to the cold fluid, an ideal thermal storage performance can be achieved. With regard to both cost reduction and energy storage performance, the single tank with floating thermal insulation baffle is the ideal thermal storage system considered in this chapter.

Whereas physically an ideal thermal storage system has the clearly identifiable features previously detailed, mathematically it should be described as a system that has an energy storage efficiency of 1.0. With this in mind, the following definition of thermal energy delivery efficiency is adopted for thermal energy storage systems:

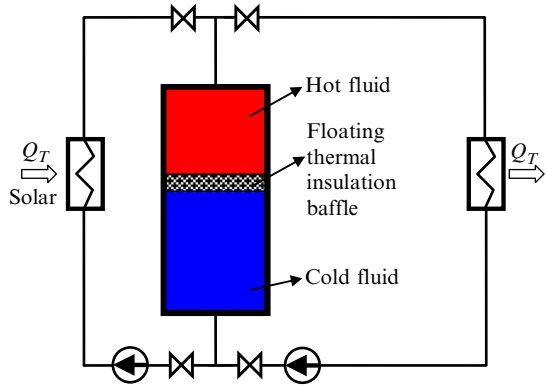


Fig. 4.1 Schematic illustration of a single tank ideal thermal storage system.

$$\eta = \frac{\int_0^{t_{ref,\,discharge}} \left[T_{f\,(z=H,t)} - T_L\right] dt}{(T_H - T_L) \cdot t_{ref,\,discharge}} \tag{4.1}$$

where z is the vertical coordinate of the tank and H is the height of the tank. In adopting this definition, we assume that the average heat capacity, Cp, and the mass flow rates of the HTF for the charging and discharging processes are the same. The integration on the numerator of Eq. (4.1) is the energy discharge in an actual process, and the value in the denominator is the ideal energy discharge.

For the ideal thermal storage system, it is assumed that the temperature of the hot fluid in a charging process is kept at constant T_H; and in the discharging process the discharged fluid maintains a constant temperature of T_H as well. After releasing heat in a heat exchanger, the fluid returns to the bottom of the storage tank at a constant temperature, T_L. To substitute these conditions from the ideal thermal storage system into Eq. (4.1), the fluid temperature $T_{f\,(z=H,t)}$ during the discharge process in a time period of 0 to $t_{ref,\,discharge}$ should be equal to the high temperature, T_H. This will make the energy delivery efficiency equal to $\eta = 1.0$ for the ideal thermal storage system.

In a real thermal energy storage system, such as the systems shown in Fig. 2.3, it is easy to understand that when cold fluid is pumped into the tank from the bottom, it will extract heat from the solid thermal storage material and be warmed up when it flows out of the tank. However, after a certain time, the cold fluid going into the tank may not be heated up sufficiently

before it flows out from the top of the tank. Unfortunately, this temperature degradation is inevitable due to the heat transfer between the solid thermal storage material and the HTF, even if initially the solid thermal storage material is fully charged, or its temperature is exactly equal to T_H.

Considering the need for an HTF in a power plant, it is always important that, during the required operational period of time $t_{ref,discharge}$, the temperature of the HTF have minimal or no degradation from the temperature at which the fluid is stored. To meet this requirement in an actual thermocline storage system, one needs to first store a sufficient amount of energy (more than the ideal amount) in the tank. This requires a storage tank having a sufficiently large thermal energy storage capacity as well as a sufficiently long charge time that allows heat to be charged to the tank. Giving this requirement as a mathematical expression, it is:

$$\left\{\left[\rho_s C_s(1-\varepsilon)+\rho_f C_f \varepsilon\right] V_{real}\right\} > \left[\left(\rho_f C_f\right) V_{ideal}\right] \tag{4.2}$$

In engineering reality, one needs to know specifically how large the real thermal storage volume, V_{real}, is and how long a charging time is needed, if the assumed operation time period of a power plant is $t_{ref,discharge}$. This must be addressed through mathematical analysis.

In the following section, the modeling of the heat transfer and energy transport between the solar thermal storage material and the HTF is described. The goal of the modeling analysis is to predict the size of the storage tank and the period of time required to charge the tank for a given subsequent period of heat discharge from the system, within which minimal or no temperature degradation must be maintained.

4.2 ONE-DIMENSIONAL MODEL FOR DUAL-MEDIA PACKED-BED SENSIBLE THERMAL STORAGE

In Chapter 2, the various types of thermal storage systems were discussed. The idea of a thermocline thermal storage system, with a medium and an HTF, is to deposit the thermal energy from the hotter temperature HTF. The stored thermal energy is retrieved by passing a cooler HTF through the storage system. The heat transfer involved is the convection within the HTF and conduction in the storage medium. This is in fact a conjugate heat transfer problem. The convection and conduction are to be solved simultaneously. They are coupled through the interface via temperature and heat flux matching. The fluid flow, on the other hand, has to satisfy

no slip and no penetration conditions along the interface. Furthermore, boundary conditions on the outer physical boundary have to be enforced, depending on the situation.

The numerical solution of the mathematical model for the conjugate heat transfer can be quite time consuming. To reduce the computational time, simplifications are made. Furthermore, the simplified models allow thorough parametric studies. Since the computational times are short, this can be used as a fast and accurate design tool. In this section, a one-dimensional model is presented for a thermocline thermal storage system with packed bed for sensible energy storage. The heat transfer between the packed bed and HTF is modeled by Newton's cooling law.

Fig. 4.2 shows the schematic diagram of the one-dimensional sensible energy storage model. A differential control volume of thickness dz is selected in the packed bed. For convenience in analysis, the positive direction of coordinate z is set to be in the direction of the fluid flow. In the energy-charging process, hot fluid flows into the tank from the top, and thus $z=0$ is at the top of the tank. During the heat-discharging process, cold fluid flows into the tank from the bottom to extract heat from the solid material, and this makes $z=0$ at the bottom of the tank.

The following assumptions are made to reasonably simplify the analysis of the heat transfer between the HTF and the solid packing material:

(1) There is a uniform radial distribution of the fluid flow and filler material throughout the storage tank. This allows the model to be one dimensional, only in the z direction.

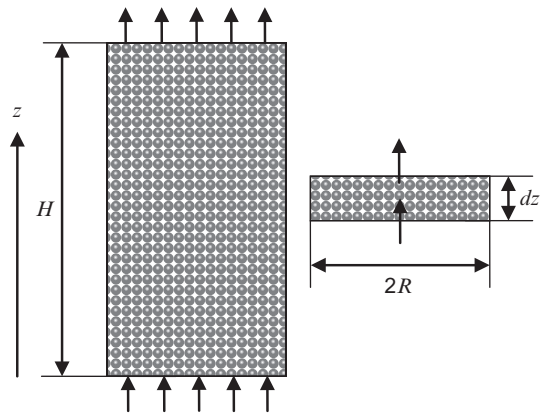


Fig. 4.2 Schematic of a packed-bed thermal storage system and a control volume for analysis.

(2) The particles of filler material have only point contact and therefore heat conduction between filler material is negligible.

(3) The heat conduction in the axial direction in the fluid is negligible compared to the convective heat transfer.

(4) The lumped heat capacitance method is applied to the transient heat conduction in the filler material (particles of size of 0.1–5.0 cm in nominal diameter). When this method is inadequate, due to the large size of the solid filler material, a *modified lumped capacitance method* will be used, which introduces a modified heat transfer coefficient for the convection heat transfer between the fluid and the solid filler material.

(5) There is no heat loss from the storage tank to the surroundings. This assumption applies to both the processes of energy charge and discharge, as well as the resting time between a charge and a discharge.

The assumption (3) is valid when the Peclet number (=RePr) in the HTF is sufficiently large, which is satisfied for most thermal energy storage applications [2]. Assumption (4) is valid when the Biot number $\left(=hL_p/k_s\right)$ for the thermal storage material is sufficiently small [3]. If the Biot number is large, a correction to the heat transfer accounting for the effects of an internal temperature gradient in the filler material will be considered. Heat loss from a thermal storage tank is inevitable and should also be considered [4]. However, from the design point of view, one needs to first decide the dimensions of the storage tank in order to find the heat loss. To compensate for the heat loss from the tank, a larger volume heat storage tank and a longer heat charge period may be adopted. A simple way of refining this design is to increase both the heat charge time and tank size with a factor that is equal to the ratio of heat loss versus the projected heat delivery. To focus on the main issues, the current work determines the dimensions of a storage tank without considering heat loss. The assumption of no heat loss to the surroundings also provides a basis for using the results from a heat charge process as the initial condition of the following discharge process, and vice versa. By using the end results of one process as the initial conditions of the following process, multiple cyclic energy charges and discharges in the actual operation can be simulated relatively easily.

Based upon these modeling assumptions (1), the cross-sectional area of the tank seen by the fluid flow is assumed constant at all locations along the axis of the tank, which gives:

$$a_f = \varepsilon \pi R^2 \tag{4.3}$$

The thermal energy balance of the fluid in the control volume dz is:

$$\rho_f \varepsilon \pi R^2 U(\hbar_z - \hbar_{z+dz}) + hS_s\left(T_s - T_f\right)dz = \rho_f C_f \varepsilon \pi R^2 dz \frac{\partial T_f}{\partial t} \tag{4.4}$$

where the parameter S_s denotes the heat transfer surface area between the filler material and the HTF per unit length of the tank; U is the actual fluid velocity in the packed bed:

$$U = \frac{\dot{m}}{\rho_f a_f} \tag{4.5}$$

The heat transfer coefficient h in Eq. (4.4) is for the convection between the HTF and the packing material. It can be different depending on the flow, packing condition of thermal storage material, fluid properties, and the interaction between packed material and HTF, such as in the schematic shown in Fig. 2.5. Detailed discussions of S_s and h are presented following the modeling work.

Using the definition of enthalpy change and a Taylor's series expansion, $\hbar_{z+dz} - \hbar_z = C_f\left(\partial T_f/\partial z\right)dz$, the energy balance equation for the HTF becomes:

$$\frac{hS_s}{\rho_f C_f \varepsilon \pi R^2}\left(T_s - T_f\right) = \frac{\partial T_f}{\partial t} + U\frac{\partial T_f}{\partial z} \tag{4.6}$$

We introduce the following dimensionless variables:

$$\theta_f = \left(T_f - T_L\right)/(T_H - T_L) \tag{4.7a}$$

$$\theta_s = (T_s - T_L)/(T_H - T_L) \tag{4.7b}$$

$$z^* = z/H \tag{4.7c}$$

$$t^* = t/(H/U) \tag{4.7d}$$

The dimensionless governing equation for the HTF is finally reduced to:

$$\frac{\partial \theta_f}{\partial t^*} + \frac{\partial \theta_f}{\partial z^*} = \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.8}$$

where

$$\tau_r = \frac{U}{H}\frac{\rho_f C_f \varepsilon \pi R^2}{hS_s} = \frac{C_f \dot{m}}{HhS_s} \tag{4.9}$$

The boundary condition for Eq. (4.8) is from the fluid inlet temperature, while the initial condition is the temperature distribution in a tank before a charge or a discharge starts.

For the energy balance of the filler material in a control volume dz as shown in Fig. 4.2, it is understood that the filler material delivers or takes heat to or from the passing fluid at the cost of a change in the internal energy of the filler. The energy balance equation is:

$$hS_s\left(T_s - T_f\right)dz = -\rho_s C_s(1-\varepsilon)\pi R^2 dz\frac{\partial T_s}{\partial t} \tag{4.10}$$

By substituting in the dimensionless variables given in Eq. (4.7a)-(4.7d), the preceding governing equation for filler material is reduced to:

$$\frac{\partial \theta_s}{\partial t^*} = -\frac{H_{CR}}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.11}$$

where

$$H_{CR} = \frac{\rho_f C_f \varepsilon}{\rho_s C_s(1-\varepsilon)} \tag{4.12}$$

The boundary condition for Eq. (4.11) is the fluid inlet temperature (varies with time) and also the solid temperature at the inlet point (directly obtained from Eq. 4.11 for the inlet point alone).

In the energy charge and discharge processes, the filler material and HTF will have a temperature difference at any local location. Once the fluid comes to rest upon the completion of a charge or discharge process, the fluid will equilibrate with the local filler material to reach the same temperature, T_{final}. The energy balance of this situation at a local location is:

$$\varepsilon\rho_f C_f T_{f-initial} + (1-\varepsilon)\rho_s C_s T_{s-initial} = \varepsilon\rho_f C_f T_{final} + (1-\varepsilon)\rho_s C_s T_{final} \tag{4.13}$$

Here, the initial temperatures of primary thermal storage material and HTF are from the results of their respective charge or discharge processes. The final temperatures of the storage material and the fluid are the same after their thermal equilibrium is reached.

According to the assumption of no heat loss from the storage tank, it can be seen that the equilibrium temperature at the end of one process (charge or discharge) will necessarily be the initial condition of the next process in the cycle. This connects the discharge and charge processes so that overall periodic results can be obtained.

The initial temperatures of filler material and fluid in the storage tank should be known. Also, the inlet fluid temperature is known as a basic boundary condition, with which the filler temperature at inlet location $z=0$ can be easily solved mathematically from Eq. (4.11).

A joint solution of Eqs. (4.8) and (4.10) simultaneously is needed in order to find the temperature of both fluid and solid at a time and location.

4.2.1 Energy Delivery Efficiency

With the solution of the governing equations for filler material and HTF, the discharged fluid temperature from a storage tank can be obtained. With the required heat discharge period being given as $t_{ref,discharge}$, an energy delivery effectiveness can be obtained from Eq. (4.1) as discussed before. For convenience of expression, the dimensionless form of the required time period of energy discharge is defined as:

$$\Pi_d = \frac{t_{ref,\,discharge}}{H/U} \tag{4.14}$$

Similarly, a dimensionless form of the time period of energy charge is defined as:

$$\Pi_c = \frac{t_{charge}}{H/U} \tag{4.15}$$

Substituting the dimensionless energy discharge period Π_d into Eq. (4.1), we obtain:

$$\eta = \frac{1}{\Pi_d}\int_0^{\Pi_d} \theta_{f\,(z^*=1,\,t^*)}\,dt^* \tag{4.16}$$

The energy discharge efficiency will obviously be affected by how much energy is charged into the storage tank. Therefore, it should be noted that η is essentially the function of the following four parameters—Π_c/Π_d, Π_d, τ_r, and H_{CR}. By specifying the dimensionless time period of the discharge process and the mass flow rate, the dimensionless time period of the energy charge can be determined to achieve the objective value of η, which is always desired to approach as close as possible to 1.0.

As has been discussed, a longer energy charging time than energy discharging time is needed in order to achieve an energy delivery effectiveness of approximately 1.0 in a packed-bed system. In addition, the energy storage capacity of a packed-bed tank must be larger than that of an ideal thermal storage tank, as expressed in Eq. (4.2).

4.2.2 Heat Transfer Area S_s and Heat Transfer Coefficient h in Different Types of Storage Systems

As has been already discussed, the governing equations for the temperatures and energy exchange between the primary thermal storage material and the HTF are generally the same for all the thermal storage systems as schematically shown in Fig. 2.5. However, the heat transfer coefficients and the heat transfer area between the primary thermal storage material and HTF for different types of storage systems can be significantly different.

The heat transfer area between rocks and fluid per unit length of tank was denoted as S_s. Therefore, the unit of S_s is in meters. For spherical filler materials, S_s is obtained through the following steps:

(1) The volume of filler material in a unit length Δz of tank is given as $\pi R^2 \Delta z(1-\varepsilon)$.

(2) One sphere of rock has a volume of $V_{sphere} = 4\pi r^3/3$, and therefore, in the length of Δz in the tank, the number of rocks is $\pi R^2 \Delta z(1-\varepsilon)/V_{sphere}$. The total surface area of rocks is then determined to be $\pi R^2 \Delta z(1-\varepsilon) \times 4\pi r^2/V_{sphere}$, which becomes $3\pi R^2(1-\varepsilon)\Delta z/r$ after V_{sphere} is substituted in.

(3) Finally, the heat transfer area of rocks per unit length of tank is:

$$S_s = 3\pi R^2(1-\varepsilon)/r \tag{4.17}$$

The preceding discussion considers the actual volume (assuming ε is known) for solid "spherical particles" in a packed volume. Depending on the packing scheme, the void fraction ε in a packed bed with spheres of a fixed diameter may range from 0.26 to 0.476 [5]. The loosest packaging of spherical rocks in a volume is given by the case where each sphere (of diameter $2r$) is packed into a cube with side lengths of $2r$. The densest packing of spheres causes a void fraction of 0.26, which is due to Kepler's conjecture [6]. Nevertheless, if the packed bed void fraction ε is known, Eq. (4.17) should be used for finding S_s.

The heat transfer coefficient h (W/m^2°C) between the primary thermal storage material (porous media) and HTF can be found in Ref. [7].:

$$h = 0.191 \frac{\dot{m} C_f}{\varepsilon \pi R^2} \mathrm{Re}^{-0.278} \mathrm{Pr}^{-2/3} \tag{4.18}$$

where Re is Reynolds number (equal to $4Gr_{char}/\mu_f$) for porous media, as defined by Nellis [7]. The mass flux of fluid through the porous bed is G (equal to $\dot{m}/(\varepsilon \pi R^2)$), and r_{char} is defined as the characteristic radius of the

filler material [7], which is equal to $0.25\varepsilon d_r/(1-\varepsilon)$ for a spherical solid filler. Here, d_r is the nominal diameter of a rock, if it is not perfectly spherical.

When the packed particles have irregular shape, such as that of rocks, the convective heat transfer coefficient between the particles and the fluid is a parameter that is difficult to predict accurately. More studies of the convective heat transfer coefficient between packed bed solid materials and fluid are needed for application of thermal energy storage.

4.2.3 Conduction Effects in the Solid Particles or Integrated Solid Region

The model and equations in Sections 4.2.1 and 4.2.2 use the lumped capacitance method to determine the heat transfer inside the filler material. This method actually ignores the resistance to heat conduction inside the filler material. This will result in the calculated energy going into, or coming out from, a filler material being higher than that in the actual physical process. It is known [3] that when the Biot number of the heat transfer of a particle is larger than 0.1, the lumped capacitance assumption will result in increased inaccuracy. In order to correct the lumped capacitance approximation for a spherical "particle" in fluid, Bradshaw et al. [8] and Jefferson [9] proposed to correct the convective heat transfer coefficient between the solid spherical "particle" and the fluid. The modified heat transfer coefficient is then used in the equations for the transient temperature in the particle from the standard lumped capacitance method.

In thermal energy storage systems, there are different types of packed beds of different geometries, as shown in Fig. 2.5. In fact, thermal storage materials could be plates, cylindrical rods immersed in fluid, or a solid core with circular channels inside for HTF. The following sections will present four types of thermal storage packed beds (Fig. 4.3). We use the methodology developed by Bradshaw et al. [8]. The formulas of the effective heat transfer coefficient (Biot number) will be presented for the four types of thermal storage packed beds. Comparison to analytical results will verify their usefulness. Hausen [10] also proposed a similar concept of using corrected heat transfer coefficients in a lumped capacitance method. The equations of corrections to the heat transfer coefficient were also summarized in the book by Schmidt and Willmott [11], which are reexamined in this work. Other works on this issue include those by Razelos and Lazaridis [12], Hughes et al. [13], and Mumma and Marvin [14]. The concluding equations in these references will be compared and evaluated with the results from the current study.

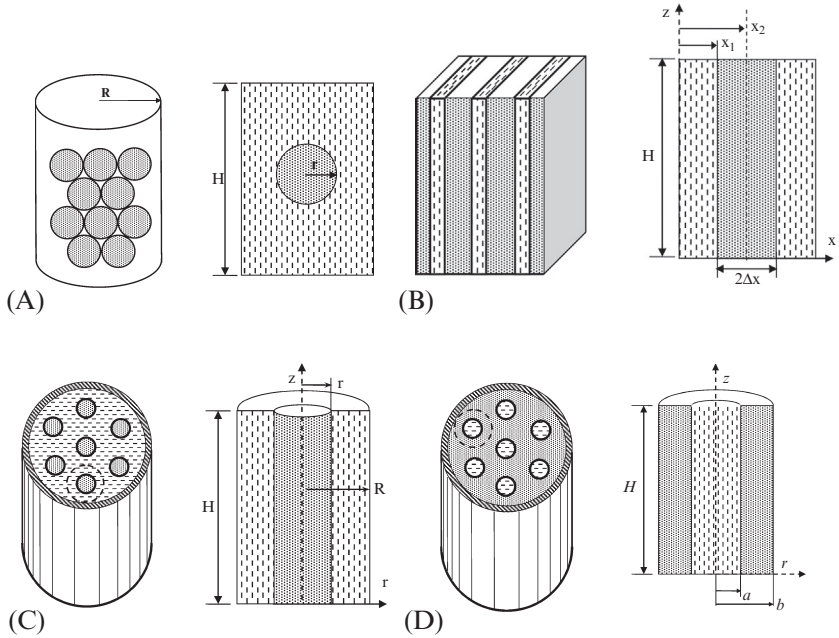


Fig. 4.3 Four typical solid-fluid structural combinations of thermal storage systems (▨ solid thermal storage material; ▤ HTF). (A) Packed solid spherical particles with HTF passing around. (B) Packed solid flat plates with HTF passing through the channels in a tank. (C) Packed solid cylinders with HTF passing along in a tank. (D) Solid thermal storage material with HTF tubes passing through in a tank.

The case of a solid sphere is briefly presented here. The readers are referred to the paper by Xu et al. [4] for detailed derivations of all the cases. For thermal storage in a solid material of general geometry, the one-dimensional transient heat transfer governing equations in the fluid and solid are given by Eqs. (4.6) and (4.10). These are rearranged and shown here in dimensional form:

$$\rho_f C_f V_f \left(\frac{\partial T_f}{\partial t} + U \frac{\partial T_f}{\partial z} \right) = h S_s \left(T_s - T_f \right) \tag{4.19}$$

$$\rho_s C_s V_s \frac{\partial T_s}{\partial t} = -h S_s \left(T_s - T_f \right) \tag{4.20}$$

where V_f is the volume of the fluid per unit length of the thermal storage tank; S_s is the surface area of the solid material in the same control volume per unit length. Eq. (4.19) simulates the heat transfer in the fluid, which has a heat source due to the convection heat transfer between the filler materials

and fluid. Eq. (4.20) simulates the heat transfer in the filler material, which has a heat sink term, which is negative but has the exact value as the heat source term in Eq. (4.19). Note that the lumped capacitance assumption for the thermal storage material is assumed to be valid in Eqs. (4.19) and (4.20). The validity of the lumped capacitance assumption is determined by the Biot number, $Bi = hL_p/k_s$, typically less than 0.1, where $L_p = V_s/S_s$.

When the lumped capacitance assumption is not valid, the heat conduction within the solid material has to be included in the formulation, which yields

$$\rho_f C_f V_f \left(\frac{\partial T_f}{\partial t} + U \frac{\partial T_f}{\partial z} \right) = \int_{S_s} -k_s \frac{\partial T_s}{\partial n} dS \tag{4.21}$$

$$\rho_s C_s V_s \frac{\partial T_s}{\partial t} = k_s \nabla^2 T_s \tag{4.22}$$

The following analysis is to approximate the internal heat conduction effect in the thermal storage material, given on the right-hand side of Eq. (4.22), in the form of Newton's cooling relationship. Specifically, this should result in the volume-integrated right-hand side of Eq. (4.22) being expressed as

$$\int_{V_s} k_s \nabla^2 T_s dV = h_{eff} S_s \left(T_s - T_f \right) \tag{4.23}$$

Substituting Eq. (4.23) into Eqs. (4.21) and (4.22), we can obtain the lumped capacitance-type of formulation similar to Eqs. (4.19) and (4.20), except the heat transfer coefficient is an effective heat transfer coefficient h_{eff}. In fact, the effective heat transfer coefficient h_{eff} is a corrected value from the heat transfer coefficient h. It is necessary to point out that the effective heat transfer coefficient h_{eff} will strongly depend on the geometry of the solid filler materials. The following analysis takes spherical filler material as an example to elucidate the methodology. Following the method, we will present similar solutions for the other three geometries of thermal storage materials shown in Fig. 4.3.

For the purpose of extracting the effective heat transfer coefficient as explained previously, it is sufficient to consider a simple case where the fluid temperature does not vary with time and therefore the heat transfer mainly causes the fluid temperature to change in the z direction only. This simplification allows us to focus on the heat transfer between the fluid and filler

materials. Therefore, the energy balance equations from the lumped capacitance method for the fluid and thermal storage material are

$$\rho_f C_f V_f U \frac{\partial T_f}{\partial z} = h_{eff}\, S_s \left(T_s - T_f\right) \tag{4.24}$$

$$\rho_s C_s V_s U \frac{\partial T_s}{\partial t} = -h_{eff}\, S_s \left(T_s - T_f\right) \tag{4.25}$$

The initial conditions are

$$t = 0, \quad T_f = T_s = 0 \tag{4.26}$$

The boundary conditions are

$$z = 0,\ T_f = TI(t) \tag{4.27}$$

where $TI(t)$ is the fluid inlet temperature. We further assume that the solid material attains the final equilibrium temperature, M (independent of z), at the end of the charging process, in time τ. The fluid temperature at $z = L$ is $TO(t)$. Define the finite Laplace transform as

$$\Lambda_f(z, p) = \int_0^\tau T_f(z, t) e^{-pt} dt \tag{4.28}$$

and

$$\Lambda_s(z, p) = \int_0^\tau T_s(z, t) e^{-pt} dt \tag{4.29}$$

Applying these finite Laplace transforms to Eqs. (4.24) and (4.25) and eliminating $\Lambda_s(z, p)$, we obtain

$$\frac{d\Lambda_f}{dz} + \frac{\beta p}{p + \gamma}\Lambda_f + \frac{\beta M_e^{-p\tau}}{p + \gamma} = 0 \tag{4.30}$$

where M is the equilibrium temperature (constant) between the solid and the HTF,

$$\beta = \frac{hs_s}{\rho_f\, C_f \varepsilon\, U} \tag{4.31}$$

$$\gamma = \frac{hs_s}{\rho_s\, C_s} \tag{4.32}$$

The transformed boundary condition, Eq. (4.27), is

$$z=0,\ \ f(p)=\int_0^{\tau} TI(t)e^{-pt}dt \tag{4.33}$$

The solution to Eq. (4.30) subject to boundary conditions as in Eq. (4.33) is

$$\Lambda_f(z,p)=-\frac{M}{p}e^{-p\tau}+\left[f(p)+\frac{M}{p}e^{-p\tau}\right]e^{-\frac{\beta pz}{p+\gamma}} \tag{4.34}$$

It is interesting to examine the following formula for weighted average time with weighting function $g(t)$, which was introduced by Bradshaw et al. [8]:

$$\langle t\rangle=\frac{\int_0^{\tau} tg(t)dt}{\int_0^{\tau} g(t)dt}=\lim_{p\to 0}\frac{\int_0^{\tau} tg(t)e^{-pt}dt}{\int_0^{\tau} g(t)e^{-pt}dt}=-\lim_{p\to 0}\frac{\frac{d}{dp}\int_0^{\tau} g(t)e^{-pt}dt}{\int_0^{\tau} g(t)e^{-pt}dt} \tag{4.35}$$

The weighted average time can therefore be calculated by the following equation:

$$\langle t\rangle=-\lim_{p\to 0}\frac{\frac{d}{dp}\Gamma(p)}{\Gamma(p)} \tag{4.36}$$

where $\Gamma(p)$ is the finite Laplace transform of function $g(t)$.

We will now form a function, $\Delta\Lambda_f(p)=\Lambda_f(0,p)-\Lambda_f(L,p)$, which is the difference of the finite Laplace transform of the fluid temperature at the inlet and outlet. Substituting the function $\Delta\Lambda_f(p)$ as $\Gamma(p)$ into Eq. (4.36), the weighted average time formula for the lumped capacitance case is obtained as

$$\langle t\rangle=\frac{\rho_s C_s}{h_{eff}S_s}+\tau+\frac{\rho_s C_s}{2U\rho_f C_f\varepsilon}-\frac{f(0)}{M} \tag{4.37}$$

This is the functional form for the weighted average time.

We now switch our attention to the conduction effects within the solid. The heat transfer within the solid body is modeled by the heat diffusion equation in the appropriate coordinate system. Following Bradshaw et al.'s [8] method, we obtain the analytical formulas for the weighted average time. By comparing these formulas to Eq. (4.37), we can extract the effective heat transfer coefficient h_{eff}. Therefore, the governing equations are

$$\rho_f C_f V_f U \frac{\partial T_f}{\partial z} = -k_s S_s \left(\frac{\partial T_s}{\partial r}\right)_{r=r_s} \tag{4.38}$$

$$\rho_s C_s \frac{\partial T_s}{\partial z} = k_s \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial T_s}{\partial r}\right) \tag{4.39}$$

The initial and boundary conditions for T_f remain the same as in Eqs. (4.26) and (4.27). The initial and boundary conditions for T_s are:

$$T_s(r, 0) = 0 \tag{4.40}$$

$$r = 0,\ \frac{\partial T_s}{\partial r} = 0 \tag{4.41}$$

$$r = r_s,\ k_s \frac{\partial T_s}{\partial r} = h\left(T_f - T_s\right) \tag{4.42}$$

Taking the finite Laplace transform, we have

$$\rho_f C_f V_f U \frac{d\Lambda_f}{dz} + k_s S_s \left(\frac{\partial \Lambda_s}{\partial r}\right)_{r=r_s} = 0 \tag{4.43}$$

$$\rho_s C_s (M e^{-p\tau} + p\Lambda_s) = k_s \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial T_s}{\partial r}\right) \tag{4.44}$$

$$r = 0,\ \frac{\partial \Lambda_s}{\partial r} = 0 \tag{4.45}$$

$$r = r_s,\ k_s \frac{\partial \Lambda_s}{\partial r} = h\left(\Lambda_f - \Lambda_s\right) \tag{4.46}$$

The transformed temperature of the sphere can be solved as

$$\Lambda_s(r, p) = A(r, p) + B(r, p)\,\Lambda_f(z, p) \tag{4.47}$$

where

$$A(r,p) = -\frac{e^{-p\tau}M}{p} + \frac{\sin h\left[r\sqrt{\dfrac{p\,S_s}{V_f k_s(1-\varepsilon)}}\right]\left[\dfrac{e^{-p\tau}Mk_s}{p\,r_s} + \dfrac{e^{-p\tau}Mr_s\left(-\dfrac{k_s}{r_s^2}+\dfrac{h}{r_s}\right)}{p}\right]}{r\left[\dfrac{k_s\sqrt{\dfrac{p\,S_s}{V_f k_s(1-\varepsilon)}}\cos h\left[r_s\sqrt{\dfrac{p\,S_s}{V_f k_s(1-\varepsilon)}}\right]}{r_s} + \left(-\dfrac{k_s}{r_s^2}+\dfrac{h}{r_s}\right)\sin h\left[r_s\sqrt{\dfrac{p\,S_s}{V_f k_s(1-\varepsilon)}}\right]\right]} \tag{4.48}$$

$$B(r,p) = \frac{\sinh\left[r\sqrt{\frac{pS_s}{V_f k_s(1-\varepsilon)}}\right] h}{r\left[\frac{k_s\sqrt{\frac{pS_s}{V_f k_s(1-\varepsilon)}}\cosh\left[r_s\sqrt{\frac{pS_s}{V_f k_s(1-\varepsilon)}}\right]}{r_s} + \left(-\frac{k_s}{r_s^2}+\frac{h}{r_s}\right)\sinh\left[r_s\sqrt{\frac{pS_s}{V_f k_s(1-\varepsilon)}}\right]\right]} \tag{4.49}$$

Equation (4.43) can then be solved for $\Lambda_f(r,p)$,

$$\Lambda_f(z,p) = \frac{A(r_s,p)}{B(r_s,p)}\left(e^{\frac{(B(r_s,p)-1)hS_s}{\rho_f C_f V_f U}z} - 1\right) + f(0)e^{\frac{(B(r_s,p)-1)hS_s}{\rho_f C_f V_f U}z} \tag{4.50}$$

We now follow the same procedure as in the lumped capacitance case by defining $\Delta\Lambda_f(p) = \Lambda_f(0,p) - \Lambda_f(L,p)$. The weighted average time is obtained for the solid sphere, which has internal resistance

$$\begin{aligned}\langle t\rangle &= \frac{\rho_s C_s}{h\,S_s} + \frac{r_s\rho_s C_s}{5\,k_s\,S_s} + \tau + \frac{\rho_s C_s}{2U\rho_f C_f \varepsilon} - \frac{f(0)}{M} \\ &= \frac{\rho_s C_s}{h_{eff}\,S_s} + \tau + \frac{\rho_s C_s}{2U\rho_f C_f \varepsilon} - \frac{f(0)}{M}\end{aligned} \tag{4.51}$$

In Eq. (4.51) we use the two terms $\rho_s C_s/h\,S_s$ and $r_s\rho_s C_s/5\,k_s\,S_s$ as one term and introduce an effective heat transfer coefficient h_{eff}. This makes the structure of Eq. (4.51) similar to the equation from the lumped capacitance case given by Eq. (4.37). Therefore, for the sphere solid thermal storage material, the relationship of the effective heat transfer coefficient h_{eff} and the actual heat transfer coefficient h is obtained:

$$h_{eff-sp} = \frac{1}{\frac{1}{h} + \frac{r_s}{5k_s}} \tag{4.52}$$

When h_{eff-sp} is used to replace h in the lumped capacitance method, we expect the solution will approach the results from a precise analytical solution. The demonstration of this is given in Section 4.4.1. The current result for the effective heat transfer coefficient of a sphere matches the solution published by Jefferson [9]. For three other cases that have not been discussed by Bradshaw et al. [8] and Jefferson [9], the same methodology can be applied. The resulting effective heat transfer coefficient formulae are tabulated in Table 4.1.

Table 4.1 The effective heat transfer coefficients of solid thermal storage materials of different structures

	Characteristic length for lumped capacitance Biot number	Effective heat transfer coefficient h_{eff}	Effective lumped capacitance Biot number Bi_{eff}	Lumped capacitance Biot number Bi_{LC}
Sphere	$\frac{R}{3}$	$\frac{1}{\frac{1}{h}+\frac{R}{5k_r}}$	$\frac{Bi_{LC}}{1+\frac{3}{5}Bi_{LC}}$	$\frac{h}{k_r}\frac{R}{3}$
Plate	$\Delta x = x_2 - x_1$ See Fig. 4.3B for definition of x_1, x_2	$\frac{1}{\frac{1}{h}+\frac{(x_2-x_1)}{3k_r}}$	$\frac{Bi_{LC}}{1+\frac{1}{3}Bi_{LC}}$	$\frac{h}{k_r}\Delta x$
Cylinder	$\frac{R}{2}$	$\frac{1}{\frac{1}{h}+\frac{R}{4k_r}}$	$\frac{Bi_{LC}}{1+\frac{1}{2}Bi_{LC}}$	$\frac{h}{k_r}\frac{R}{2}$
Tube with fluid inside and outside insulated	$\frac{b^2-a^2}{2a}$ See Fig. 4.3D for definition of a, b; $\eta = b/a$	$\frac{1}{\frac{1}{h}+\frac{1}{k_r}\frac{a^3(4b^2-a^2)+ab^4(4Ln[b/a]-3)}{4(b^2-a^2)^2}}$	$\frac{Bi_{LC}}{1+Bi_{LC}\frac{\eta^4[4Ln(\eta)-3]+4\eta^2-1}{2(\eta^2-1)^3}}$	$\frac{h}{k_r}\frac{b^2-a^2}{2a}$

The effectiveness of the effective heat transfer coefficient can be examined by considering the heat conduction problem of a sphere subject to convective heat cooling/heating. Analytical solutions are available in textbooks [15,16]. For all the discussed solid bodies of different shapes in this study, the equations for dimensionless temperatures and heat storage/discharge are listed in Table 4.2. The definitions of the general *Bi* used in analytical solutions for different cases are also given in Table 4.2.

Fig. 4.4 shows the dimensionless heat absorbed/released in a solid body versus time determined from the lumped capacitance method, corrected lumped capacitance method, and analytical method for a sphere. At a small Biot number, 0.1, the curves of the dimensionless energy storage from all three methods agree very well, which verifies that the lumped capacitance method is valid at small Biot numbers. At a Biot number of 1.0, the corrected lumped capacitance method agrees with the analytical solution very well, while the curve from the lumped capacitance method has a significant discrepancy with the analytical solution. When the Biot number is 10.0 or 100, the discrepancy between the results from the corrected lumped capacitance method and the analytical method increases slightly, but is still acceptable. However, the lumped capacitance method predicts a very different value of energy storage and is thus unacceptable.

For other cases of fluid-solid configuration in Table 4.2, comparison of the energy storage in the solid material based on computation from the lumped capacitance method, corrected lumped capacitance method, and accurate analytical solution was detailed in the work by Xu et al. [4]. As shown in Fig. 4.4, it has been proven that the corrected lumped capacitance method can predict the energy storage with almost the same accuracy as that of the analytical method, while the original lumped capacitance method cannot predict with acceptable accuracy.

4.2.4 Numerical Solution for the One-Dimensional Model for Dual-Media Sensible Thermal Storage

A number of analyses and solutions to the heat transfer governing equations of a working fluid flowing through a filler packed bed have been presented in the past (Schumann [17]; Shitzer & Levy [18]; McMahan [19]; Beasley and Clark [20]; Zarty & Juddaimi [21]). As the pioneering work, Schumann [17] presented a set of equations governing the energy conservation of fluid flow through porous media. Schumann's equations have been widely adopted in the analysis of thermocline heat storage utilizing solid filler

Table 4.2 Dimensionless temperature and energy in a solid body from analytical solution [15]

Solid body	Solution
Sphere	$\theta_{sp}(r^*, t^*) = \sum_{m=1}^{\infty} \left[\frac{4\left(\sin\left(\overline{\beta}_m\right) - \overline{\beta}_m \cos\left(\overline{\beta}_m\right)\right)}{2\overline{\beta}_m - \sin\left(\overline{\beta}_m\right)} \right] \frac{\sin\left(\overline{\beta}_m r^*\right)}{\overline{\beta}_m r^*} e^{-\overline{\beta}_m^2 t^*}$
	$1 - \overline{\beta}_m \cdot \cot\left(\overline{\beta}_m\right) = Bi;\ Bi = hR/k_r$
	$Q_{sp}^*(t^*) = 1 - e^{-\overline{\beta}_m^2 t^*} \sum_{m=1}^{\infty} \left[\frac{4\left[\sin\left(\overline{\beta}_m\right) - \overline{\beta}_m \cos\left(\overline{\beta}_m\right)\right]}{2\overline{\beta}_m - \sin\left(2\overline{\beta}_m\right)} \right] \frac{4\pi\left(\sin\left(\overline{\beta}_m\right) - \overline{\beta}_m \cos\left(\overline{\beta}_m\right)\right)}{\overline{\beta}_m^3}$
Plate	$\theta_p(x^*, t^*) = \sum_{m=1}^{\infty} \frac{2\left(\overline{\beta}_m^2 + Bi^2\right)}{\overline{\beta}_m^2 + Bi^2 + Bi} \frac{\sin\left(\overline{\beta}_m\right)}{\overline{\beta}_m} \cos\left[\overline{\beta}_m(1 - x^*)\right] e^{-\overline{\beta}_m^2 t^*}$
	$\overline{\beta}_m \cdot \tan\left(\overline{\beta}_m\right) = Bi;\ Bi = h\Delta x/k_r$
	$Q_p^*(t^*) = 1 - e^{-\overline{\beta}_m^2 t^*} \sum_{m=1}^{\infty} \left[\frac{2\left(\overline{\beta}_m^2 + Bi^2\right)}{\overline{\beta}_m^2 + Bi^2 + Bi} \left(\frac{\sin\left(\overline{\beta}_m\right)}{\overline{\beta}_m} \right)^2 \right]$
Cylinder	$\theta_{cy-1}(r^*, t^*) = \sum_{m=1}^{\infty} \left[\frac{2\overline{\beta}_m^2}{J_0^2\left(2\overline{\beta}_m\right)\left(Bi^2 + \overline{\beta}_m^2\right)} \frac{J_1\left(2\overline{\beta}_m\right)}{2\overline{\beta}_m} \right] J_0\left(2r^*\overline{\beta}_m\right) e^{-4\overline{\beta}_m^2 t^*}$
	$2\overline{\beta}_m \cdot J_1\left(2\overline{\beta}_m R\right) = Bi \cdot J_0\left(2\overline{\beta}_m R\right);\ Bi = hR/k_r$
	$Q_{cy-1}^*(t^*) = \int_0^1 2(1 - \theta_2) r^* dr^* = 2\left(1 - \sum_{m=1}^{\infty} \left[\frac{2\overline{\beta}_m^2}{J_0^2\left(2\overline{\beta}_m\right)\left(Bi^2 + \overline{\beta}_m^2\right)} \left(\frac{J_1\left(2\overline{\beta}_m\right)}{2\overline{\beta}_m} \right)^2 \right] e^{-4\overline{\beta}_m^2 t^*} \right)$

Tube	$\theta_{cy-2}(r^*, t^*) = \sum_{m=1}^{\infty} e^{-\overline{\beta}_m^2 t^*} \left[\frac{\pi^2}{2} \frac{\overline{\beta}_m^2 G(\overline{\beta}_m)}{G(\overline{\beta}_m) - \left(Bi^2 + \overline{\beta}_m^2\right) J_1^2(\eta \overline{\beta}_m)} \right] H(\overline{\beta}_m) R(r^*)$ $R(r^*) = J_1(\overline{\beta}_m \eta) Y_0(\overline{\beta}_m r^*) - J_0(\overline{\beta}_m r^*) Y_1(\overline{\beta}_m \eta);\ Bi = ha/k_r$ $H(\overline{\beta}_m) = \int_1^{\eta} r^* R(r^*) dr^*$ $G(\overline{\beta}_m) = \left(\overline{\beta}_m J_1(\overline{\beta}_m) + Bi J_0(\overline{\beta}_m)\right)^2$ where the eigenvalues $\overline{\beta}_m$ are determined from the following equation: $\left[\overline{\beta}_m J_1(\overline{\beta}_m) + Bi J_0(\overline{\beta}_m)\right] Y_1(\overline{\beta}_m \eta) = \left[\overline{\beta}_m Y_1(\overline{\beta}_m) + Bi Y_0(\overline{\beta}_m)\right] J_1(\overline{\beta}_m \eta)$ $Q^*_{cy-2}(t^*) = \frac{2}{(\eta^2 - 1)^2} \int_1^{\eta} (1 - \theta_{cy-2}) r^* dr^* = 1 - \frac{1}{(\eta^2 - 1)^2} \sum_{m=1}^{\infty} e^{-\overline{\beta}_m^2 t^*} \left[\frac{\pi^2 \overline{\beta}_m^2 G(\overline{\beta}_m)}{G(\overline{\beta}_m) - \left(Bi^2 + \overline{\beta}_m^2\right) J_1^2(\eta \overline{\beta}_m)} \right] H^2(\overline{\beta}_m)$

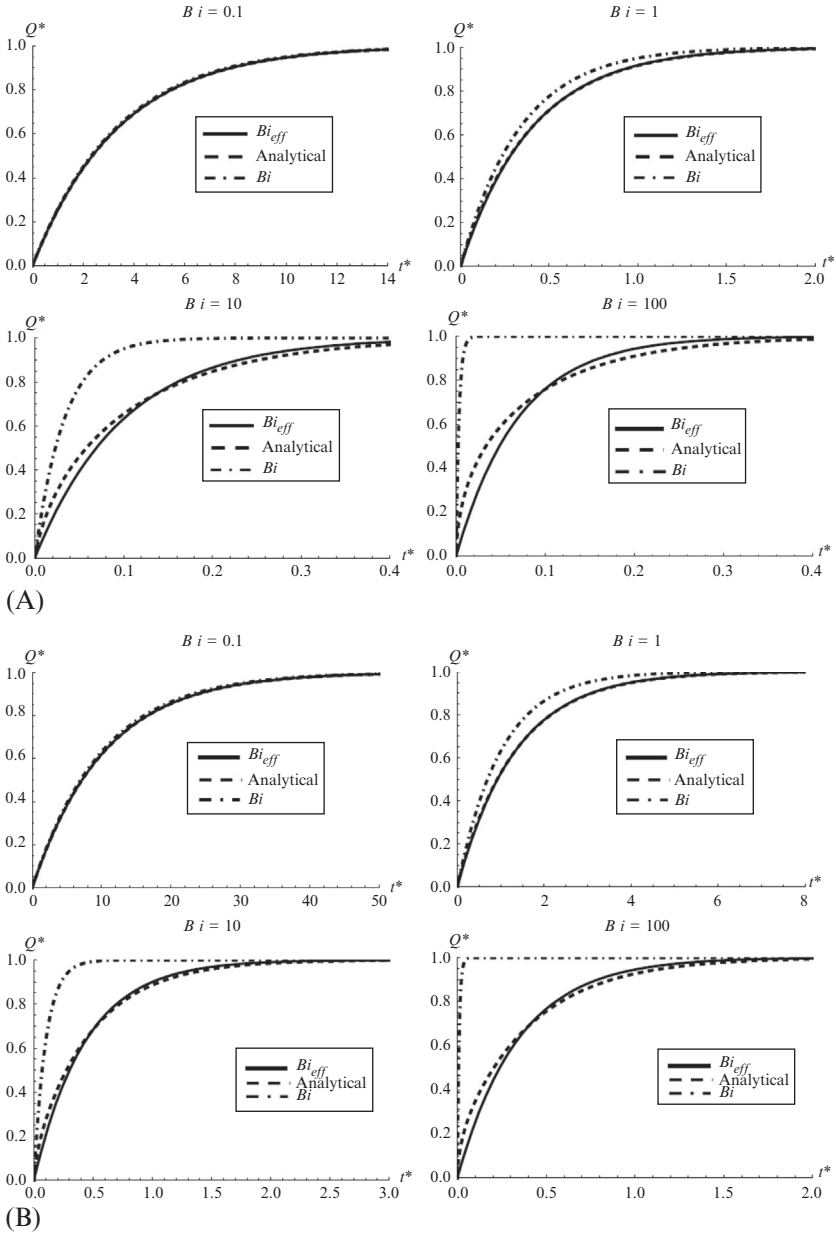


Fig. 4.4 Comparison of results from analytical method (Q^* given in Table 4.2), lumped capacitance method (with Bi used and $Q^* = 1 - e^{-Bi\,t^*}$), and corrected lumped capacitance method (with Bi_{eff} used and $Q^* = 1 - e^{-Bi_{eff}\,t^*}$) at different Biot numbers. The definition of dimensionless time is $t^* = \dfrac{k_s}{\rho_s C_s} \dfrac{t}{(V_s/S_s)^2}$. (A) Spheres; (B) plate;

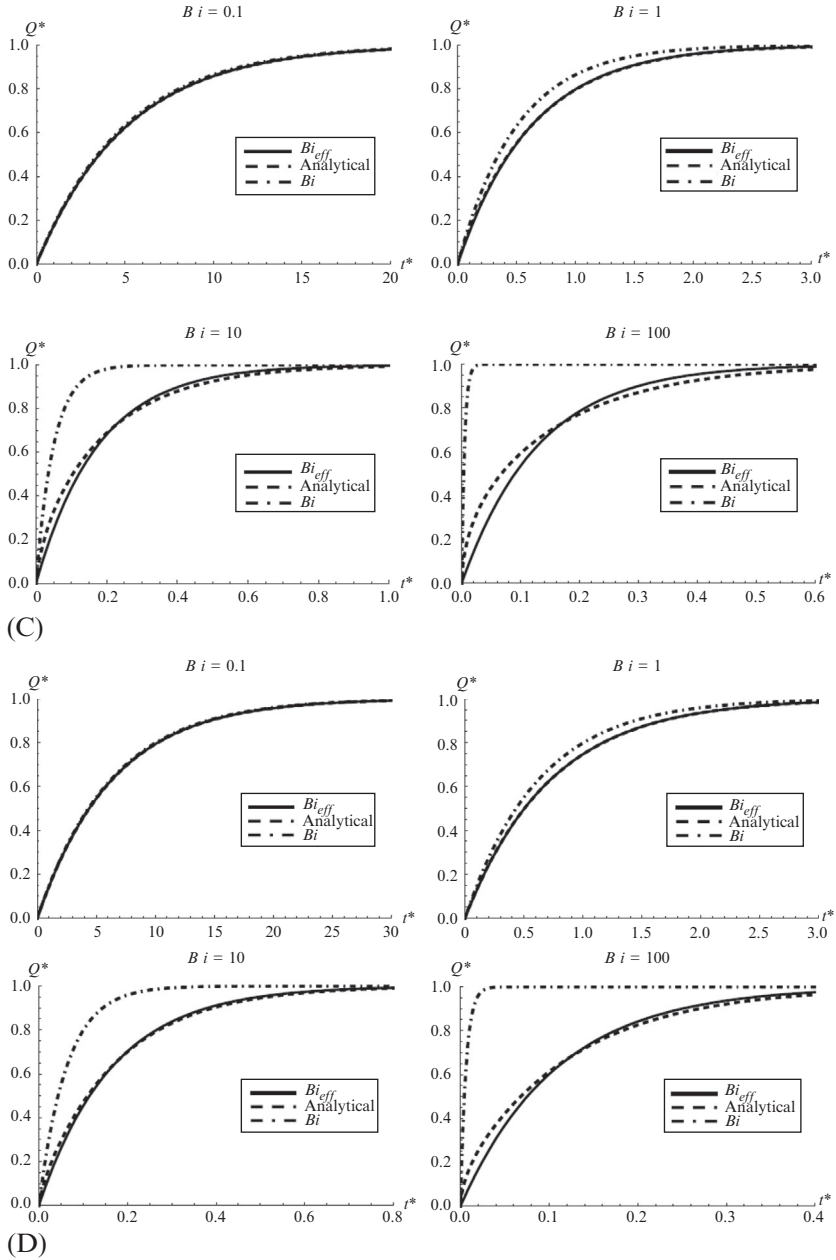


Fig. 4.4, Cont'd (C) cylinders; and (D) tube with $\eta = 1.5$.

material inside a tank. His analysis and solutions were for the special case where there is a fixed fluid temperature at the inlet to the storage system. In most solar thermal storage applications, this may not be the actual situation. To overcome this limitation, Shitzer and Levy [18] employed Duhamel's theorem on the basis of Schumann's solution to consider a transient inlet fluid temperature to the storage system. The analyses of Schumann and Shitzer and Levy, however, still carry with them some limitations. Their method does not consider a nonuniform initial temperature distribution. For a heat storage system, particularly in a solar thermal power plant, heat charge and discharge are cycled daily. The initial temperature field of a heat charge process is dictated by the most recently completed heat discharge process, and vice versa. Therefore, nonuniform and nonlinear temperature distribution is typical for both charge and discharge processes. To consider a nonuniform initial temperature distribution and varying fluid temperature at the inlet in a heat storage system, numerical methods have been deployed by researchers in the past.

To avoid the long mathematical analysis necessary in analytical solutions, numerical methods used to solve the Schumann equations were discussed in the literature by McMahan [19,22] and Pacheco et al. [23], and demonstrated in the TRNSYS software developed by Kolb and Hassani [24]. Based on the regular finite-difference method, McMahan gave both explicit and implicit discretized equations for the Schumann equations. Whereas the explicit solution method had serious solution stability issues, the implicit solution method encountered an additional computational overhead, thus requiring a dramatic amount of computation time. The solution for the complete power plant with thermocline storage provided by the TRNSYS model in the work of Kolb and Hassani [24] cites the short time step requirement for the differential equations of the thermocline as one major source of computer time consumption. To overcome the problems encountered in the explicit and implicit method, McMahan et al. [22] also proposed an infinite-NTU method. This model, however, is limited to the case in which the heat transfer of the fluid compared to the heat storage in fluid is extremely large.

The present study has approached the governing equations using a different numerical method [25]. The governing equations have been reduced to dimensionless forms, which allow for a universal application of the solution. The dimensionless hyperbolic type equations are solved numerically by the method of characteristics. This numerical method overcomes the

numerical difficulties encountered in McMahan's work—explicit, implicit, and the restriction on infinite-NTU method (McMahan [19] and McMahan et al. [22]). The current model gives a direct solution to the discretized equations (with no iterative computation needed) and completely eliminates any computational overhead. A grid-independent solution is obtained at a small number of nodes. The method of characteristics and the present numerical solution has proven to be a fast, efficient, and accurate algorithm for the Schumann equations.

The nondimensional energy balance equations for the HTF and rocks can be solved numerically along the characteristics [26–28]. Eq. (4.8) can be reduced along the characteristic $t^* = z^*$ so that:

$$\frac{D\theta_f}{Dt^*} = \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.53}$$

Separating and integrating along the characteristic, the equation becomes:

$$\int d\theta_f = \int \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.54}$$

Similarly, Eq. (4.11) for the energy balance of rocks is reposed along the characteristic $z^* =$ constant so that:

$$\frac{d\theta_s}{dt^*} = -\frac{H_{CR}}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.55}$$

The solution for Eq. (4.55) is very similar to that for Eq. (4.53) but with the additional factor of H_{CR}. The term H_{CR} is simply a ratio of fluid heat capacitance to rock heat capacitance. Therefore, the equation for the solution of θ_s will react with a dampened speed when compared to θ_f, as the filler material must have the capacity to store the energy being delivered to it, or vice versa. Finally, separating and integrating along the characteristic for Eq. (4.55) results in:

$$\int d\theta_s = -\int \frac{H_{CR}}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.56}$$

There are now two characteristic equations bound to intersections of time and space. A discretized grid of points, laid over the time-space

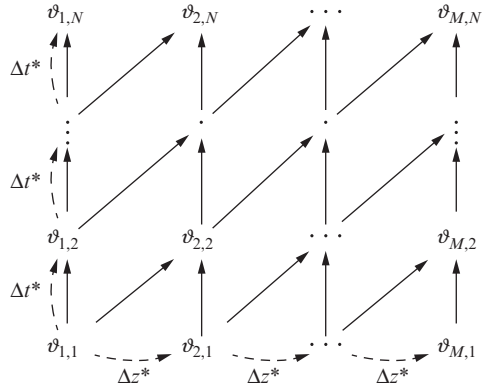


Fig. 4.5 Diagram of the solution matrix arising from the method of characteristics.

dimensions, will have nodes at these intersecting points. A diagram of these points in a matrix is shown in Fig. 4.5. In space, there are $i=1, 2, \ldots, M$ nodes broken up into step sizes of Δz^* to span all of z^*. Similarly, in time, there are $j=1, 2, \ldots, N$ nodes broken up into time-steps of Δt^* to span all of t^*. Looking at a grid of the ϑ nodes, a clear picture of the solution can arise. To demonstrate a calculation of the solution we can look at a specific point in time, along z^*, where there are two points, $\vartheta_{1,1}$ and $\vartheta_{2,1}$. These two points are the starting points of their respective characteristic waves described by Eqs. (4.53) and (4.56). After the time Δt^* there is a third point $\vartheta_{2,2}$ which has been reached by both wave equations. Therefore, Eq. (4.54) can be integrated numerically as:

$$\int_{\vartheta_{1,1}}^{\vartheta_{2,2}} d\theta_f = \int_{\vartheta_{1,1}}^{\vartheta_{2,2}} \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.57}$$

The numerical integration of the right-hand side is performed via the trapezoidal rule and the solution is:

$$\theta_{f_{2,2}} - \theta_{f_{1,1}} = \frac{1}{\tau_r}\left(\frac{\theta_{s_{2,2}} + \theta_{s_{1,1}}}{2} - \frac{\theta_{f_{2,2}} + \theta_{f_{1,1}}}{2}\right)\Delta t^* \tag{4.58}$$

where $\theta_{f_{1,1}}$ is the value of θ_f at $\vartheta_{1,1}$, and $\theta_{f_{2,2}}$ is the value of θ_f at $\vartheta_{2,2}$, and similarly so for θ_r.

The integration for Eq. (4.36) along $z^* = constant$ constant is:

$$\int_{\vartheta_{2,1}}^{\vartheta_{2,2}} d\theta_s = \int_{\vartheta_{2,1}}^{\vartheta_{2,2}} -\frac{H_{CR}}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.59}$$

The numerical integration of the right-hand side is also performed via the trapezoidal rule and the solution is:

$$\theta_{s_{2,2}} - \theta_{s_{2,1}} = -\frac{H_{CR}}{\tau_r}\left(\frac{\theta_{s_{2,2}} + \theta_{s_{2,1}}}{2} - \frac{\theta_{f_{2,2}} + \theta_{f_{2,1}}}{2}\right)\Delta t^* \tag{4.60}$$

Eqs. (4.58) and (4.60) can be reposed as a system of algebraic equations for two unknowns, $\theta_{f_{2,2}}$ and $\theta_{r_{2,2}}$, while θ_f and θ_r at grid points $\vartheta_{1,1}$ and $\vartheta_{2,1}$ are known.

$$\begin{bmatrix} 1 + \dfrac{\Delta t^*}{2\tau_r} & -\dfrac{\Delta t^*}{2\tau_r} \\ -\dfrac{H_{CR}\Delta t^*}{2\tau_r} & 1 + \dfrac{H_{CR}\Delta t^*}{2\tau_r} \end{bmatrix} \begin{bmatrix} \theta_{f_{2,2}} \\ \theta_{s_{2,2}} \end{bmatrix} = \begin{bmatrix} \theta_{f_{1,1}}\left(1 - \dfrac{\Delta t^*}{2\tau_r}\right) + \theta_{r_{1,1}}\dfrac{\Delta t^*}{2\tau_r} \\ \theta_{f_{2,1}}\dfrac{H_{CR}\Delta t^*}{2\tau_r} + \theta_{r_{2,1}}\left(1 - \dfrac{H_{CR}\Delta t^*}{2\tau_r}\right) \end{bmatrix} \tag{4.61}$$

Cramer's rule [28] can be applied to obtain the solution efficiently. It is important to note that all coefficients/terms in Eq. (4.61) are independent of z^*, t^*, θ_f, and θ_s, thus they can be evaluated once for all. Therefore, the numerical computation takes a minimum of computing time, and is much more efficient than the method applied in references [19,22].

From the grid matrix in Fig. 4.5 it is seen that the temperatures of the rock and fluid at grids $\vartheta_{i,1}$ are the initial conditions. The temperatures of the fluid and rock at grid $\vartheta_{1,1}$ are the inlet conditions, which vary with time. The inlet temperature for the fluid versus time is given. The rock temperature (as a function of time) at the inlet can be easily obtained using Eq. (4.11), for which the inlet fluid temperature is known. Now, as the conditions at $\vartheta_{1,1}$, $\vartheta_{1,2}$, and $\vartheta_{2,1}$ are known, the temperatures of the rocks and fluid at $\vartheta_{2,2}$ will be easily calculated from Eq. (4.61).

Extending the preceding sample calculation to all points in the ϑ grid of time and space will give the entire matrix of solutions in time and space for both the rocks and fluid. While the march of Δz^* steps is limited to $z^* = 1$ the march of time Δt^* has no limitation.

The previous numerical integrations used the trapezoidal rule; the error of such an implementation is not straightforwardly analyzed but the formal accuracy is on the order of $O(\Delta t^* 2)$ for functions [28] such as those solved in this study.

4.2.5 Computer Code for the One-Dimensional Computation Analysis

A MATLAB computer code has been developed to accomplish the simulation and analysis for sensible thermal energy storage. The code can be used to compute cases with any of the listed configurations of solid packed bed with HTFs flowing through, as shown in Fig. 4.3.

A brief introduction to the structure of the computer code and the MATLAB code is attached as an appendix in this chapter.

4.2.6 Numerical Results for the Temperature Variation in Packed Bed Sensible Energy Storage

The first analysis of the storage system was done on a single tank configuration of a chosen geometry, using a filler and fluid with given thermodynamic properties. The advantage of having the governing equations reduced to their dimensionless form is that by finding the values of two dimensionless parameters (τ_r and H_{CR}) all the necessary information about the problem is known. The properties of the fluid and filler rocks, as well as the tank dimensions, which determines τ_r and H_{CR} for the example problem, are summarized in Table 4.3.

The numerical computation started from a discharge process assuming initial conditions of an ideally charged tank with the fluid and rocks both having the same high temperature throughout the entire tank, i.e., $\theta_f = \theta_s = 1$. After the heat discharge, the temperature distribution in the tank is taken as the initial condition of the following charge process. The discharge and charge time were each set to 4 h. The fluid mass flow rate was determined such that an empty (no filler) tank was sure to be filled by the fluid in 4 h. It was found that, with the current configuration, after five discharge and charge cycles the results of all subsequent discharge processes were identical—likewise for the charge processes. It is therefore assumed that the solution is then independent of the first initial condition. The data presented in the following portions of this section are the results from the cyclic discharge and charge processes after five cycles.

Shown in Fig. 4.6 are the temperature profiles in the tank during a discharge process, in which cold fluid enters the tank from the bottom of the tank. The location of $z^* = 0$ is at the bottom of a tank for a discharge process. The temperature profile evolves as discharging proceeds, showing the heat wave propagation and the high temperature fluid moving out of the storage tank. The fluid temperature at the exit ($z^* = 1$) of the tank gradually

Table 4.3 Dimensions and parameters of a thermocline tank [25]

ε	τ_r	H_{CR}	H	R	t
0.25	0.0152	0.3051	14.6 m	7.3 m	4 h

Fluid (Therminol VP-1) properties:

$T_H = 395°C$	$T_L = 310°C$	$\rho_f = 753.75 kg/m^3$	$C_f = 2474.5$ J/(kg K)	$k_f = 0.086$ W/(m K);	$\dot{m} = 128.74 kg/s$	$\mu_f = 1.8 \times 10^{-4}$ Pas

Filler material (granite rocks) properties:

$\rho_s = 2630 kg/m^3$	$C_s = 775$ J/(kg K)	$k_s = 2.8$ W/(m K)	$d_r = 0.04$ m

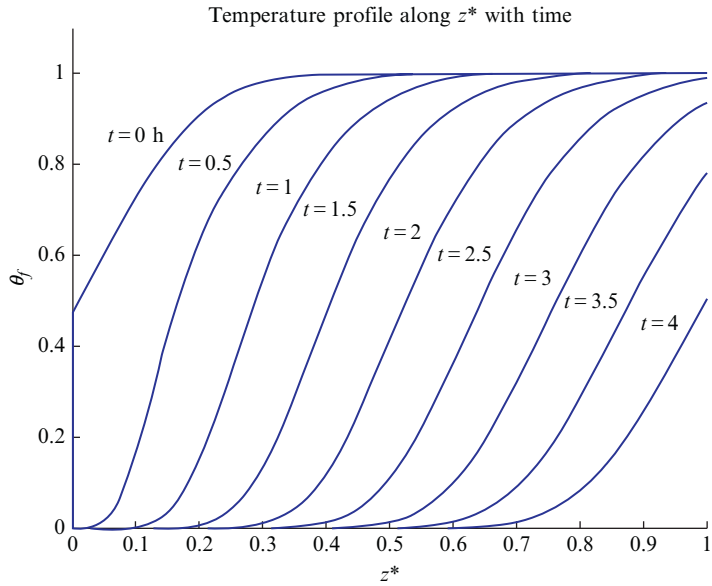


Fig. 4.6 Dimensionless fluid temperature profile in the tank for every 0.5 h.

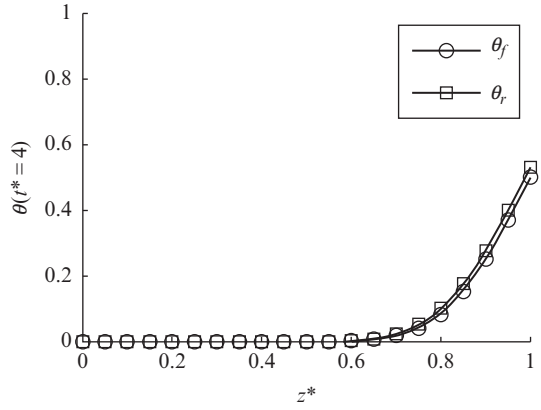


Fig. 4.7 Dimensionless temperature distribution in the tank after time $t^*=4$ of discharge (here θ_r is used to denote θ_s, as rocks are used as the storage material in the example).

decreases after 3 h of discharge. At the end of the discharge process, the temperature distribution along the tank is shown in Fig. 4.7. It is seen that the fluid and rock temperatures, θ_f and θ_s respectively (θ_s is denoted by θ_r when the filler material is rock), in the region with z^* below 0.7 are almost zero, which means that the heat in the rocks in this region has been completely extracted by the passing fluid. In the region from $z^*=0.7$ to $z^*=1.0$ the

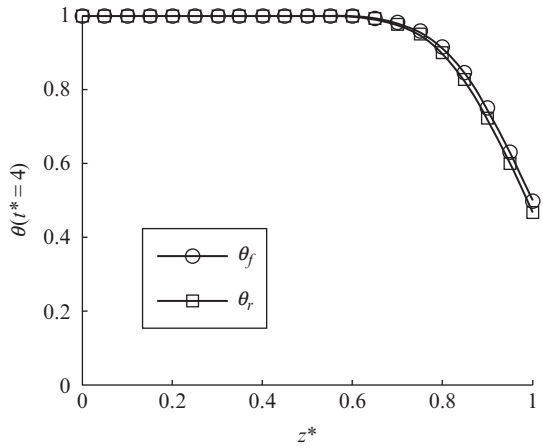


Fig. 4.8 Dimensionless temperature distribution in the tank after time $t^*=4$ of charge (here θ_r is used to denote θ_s, as rocks are used as the storage material in the example).

temperature of the fluid and rock gradually becomes higher, which indicates that some heat has remained in the tank.

A heat charge process has a similar heat wave propagation scenario. The temperature for the filler and fluid along the flow direction is shown in Fig. 4.8 after a 4 h charging process. During a charge process, fluid flows into the tank from the top, where z^* is set as zero. It is seen that for the bottom region (z^* from 0.7 to 1.0) the temperatures of the fluid and rocks decrease significantly. A slight temperature difference between the HTF and the rocks also exists in this region.

The next plots of interest are the variation of θ_f at $z^*=1$ as dimensionless time progresses for a charging or discharging process. Fig. 4.9 shows the behavior of θ_f at the outlet during both charge and discharge cycles. For the charge cycle, θ_f begins to increase when all of the initially cold fluid has been ejected from the thermocline tank. For the present thermocline tank, the fluid that first entered the tank at the start of the cycle has moved completely through the tank at $t^*=1$, which also indicates that the initially existing cold fluid of the tank has been ejected from the tank. Similarly, during the discharge cycle, after the initially existing hot fluid in the tank has been ejected, the cold fluid that first entered the tank from the bottom at the start of the cycle has moved completely through the tank at $t^*=1$. At $t^*=2.5$ or $t=2.5\,\mathrm{h}$, the fluid temperature θ_f starts to drop. This is because the energy from the rock bed has been significantly depleted and incoming cold fluid no longer can be heated to $\theta_f=1$ by the time it exits the storage tank.

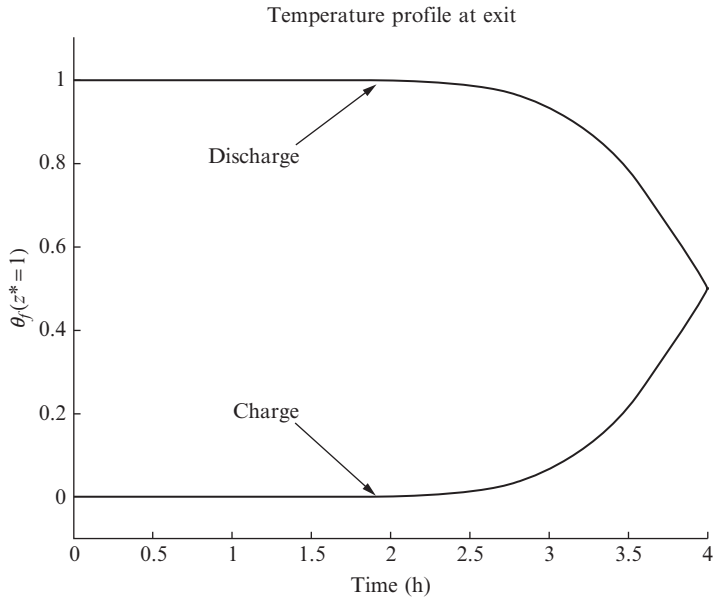


Fig. 4.9 Dimensionless temperature histories of the exit fluid at $z^*=1$ for charge and discharge processes.

The preceding numerical results agree with the expected scenario as described in Section 4.4. To validate this numerical method, analytical solutions were conducted using a Laplace transform method by the current authors [29], which were only possible for cases with a constant inlet fluid temperature and a simple initial temperature profile. Results compared in Fig. 4.10 are obtained under the same operational conditions—starting from a fully charged initial state and run for five iterations of cyclic discharge and charge processes. The fluid temperature distribution along the tank ($z^*=0$ for the bottom of the tank) from numerical results agrees with analytical results very well. This essentially proves the effectiveness and reliability of the numerical method developed in the present study.

As can be seen, the temperature distribution along z^* at the end of a charge is nonlinear. This distribution will be the initial condition for the next discharge cycle. Similarly, a discharge process will result in a nonlinear temperature distribution, which will be the initial condition for the next charge. It is evident that the analytical solutions developed by Schumann [17] could not handle this type of situation.

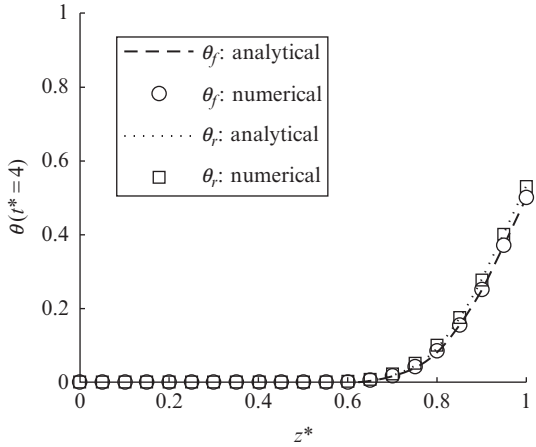


Fig. 4.10 Comparison of numerical and analytical results of the temperature distribution in the tank after time $t^*=4$ of a discharge (here θ_r is used to denote θ_s, as rocks are used as the storage material in the example).

Another special comparison was made to demonstrate the efficiency of the method of characteristics at solving the dimensionless form of the governing equations. Shown in Fig. 4.11 are the temperature profiles at $t^* = 4$ obtained by using different numbers of nodes (20, 100, and 1000) for z^*. The high level of accuracy of the current numerical method, even with only 20 nodes, demonstrates the accuracy and stability of the method with minimal computing time.

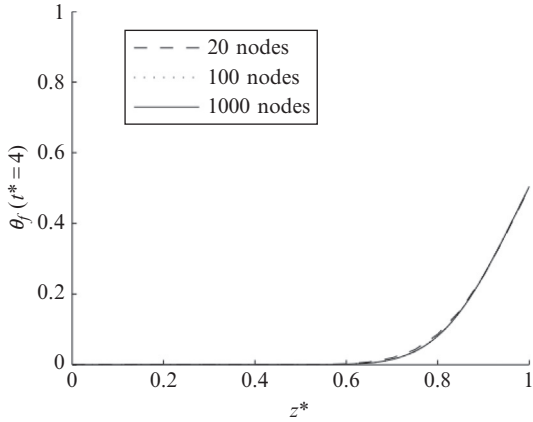


Fig. 4.11 Comparison of dimensionless temperature distributions in the tank after time $t^*=4$ of discharge for different numbers of discretized nodes.

4.3 ONE-DIMENSIONAL HEAT TRANSFER MODEL FOR ENCAPSULATED PCM PACKED BED

In this section, we considered the case of using encapsulated PCM as the storage material. Based on properties alone, the use of a PCM, combining both sensible and latent heat, allows a significantly higher energy storage density as compared to the use of sensible heat exclusively. Experimental studies of various tank filler materials confirm enhanced performance through use of a PCM, show resulting tank volume reduction by as much as a factor of 10, and suggest PCM fillers as fully viable alternatives for all thermal energy storage applications [30–33].

Efforts in thermal energy storage modeling go back as far as Schumann in 1929 [17], whose equations formed a basis for representing fluid flow through a porous packed bed thermal storage tank. Following models [18,23] expanded consideration, where most recently Van Lew et al. [25] applied the method of characteristics to produce a direct, fast, and accurate numerical solution to model thermocline interactions. A model by Regin and Solanki [34] considered a simple charge process of a tank with PCM filler for a parametric study of material properties. Following, a model by Wu et al. [35] applied an implicit finite difference method to solve the equations for the case with presence of PCM filler in the tank as a more general scenario, though results from the model featured numerous oddities and oscillations in temperature distribution profiles. To overcome the lower thermal conductivity of PCM material, Nithyanandam and Pitchumani [36,37] introduced heat transfer augmentation using thermosiphons or heat pipe. Different configurations were investigated by using computational fluid dynamics (CFD). Optimal orientation and design parameters were obtained. Archibold et al. [38] focused their attention on the fluid flow and heat transfer of the PCM within the spherical encapsulate. Recirculating vortexes were found in the upper region and therefore more intense melting occurs in this region. On the other hand, Vyshak and Jilani [39] used a modified enthalpy method to investigate the melting times for rectangular, cylindrical, and cylindrical shell storage configurations. The melting time was the least for cylindrical shell storage. They also investigated the effects of inlet temperature of the HTF. However, among these models, a comprehensive and accurate model for thermal energy storage with an encapsulated PCM filler has yet to be conceived.

The current work followed suit after the success of Van Lew et al. [25], with a much-needed expansion of analysis to an encapsulated PCM filler. An enthalpy-based version of the Schumann equations was used to allow tracking of interactions throughout the thermocline processes—a change

especially necessary in the latent region where PCM filler temperature remained constant. The new set of equations was nondimensionalized for general application. With the resulting equations being of the hyperbolic type, the method of characteristics was applied for a numerical solution. The process gave fluid temperature and PCM filler enthalpy according to the discretized grid in time and space. With the equations following a similar form to those Van Lew obtained, we too expected the method to produce a direct solution that is both accurate and efficient.

The addition of enthalpy to consideration required an equation of state to close the gap in unknowns for solution. For proper application of this equation in the governing thermocline interactions, PCM filler phase states had to be tracked closely. More importantly, to maintain accuracy as these PCM filler phase states change throughout the space, a careful tracking of PCM filler phase state interfaces had to be implemented as well. This allowed proper application of the equations to all possible orientations and conditions of the PCM filler phase state interfaces in the numerical grid of characteristics. The method of characteristics made this possible, though the extent of generality and versatility hinged on the completeness of physical cases considered in its application.

4.3.1 Mathematical Model

To obtain the governing equations for fluid and PCM filler interactions in the thermocline, we first made some necessary assumptions. The work assumed a strictly vertical fluid flow through the tank, along with a uniform fluid distribution in the radial direction, as shown in Fig. 4.12. This reduced

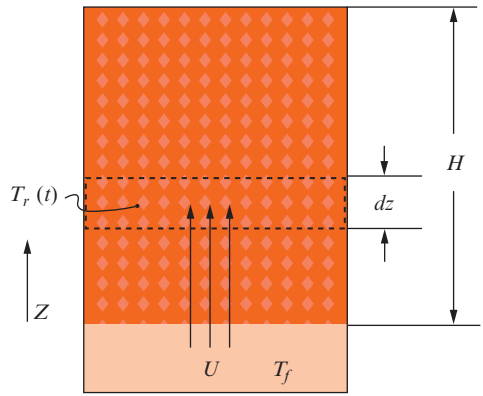


Fig. 4.12 General thermocline (pebbles or PCM capsules) for modeling.

consideration into a single spatial dimension z, which followed the direction of fluid flow. In this regard, z became a coordinate system that can be chosen, allowing identical application of the governing equations to both charge and discharge processes. Fluid thermophysical properties were assumed invariant with temperature, and thus constant. A general representation of the to-be-modeled thermocline tank can be viewed in Fig. 4.12.

With assumptions covered, we applied an energy balance to both fluid and filler for the chosen differential control volume. For the fluid, necessary terms included the enthalpy of the flow in and out of the volume, energy exchange between the fluid and PCM filler material within the volume, and the internal energy change of the fluid over the instance in time. The fluid governing equation became:

$$\rho_f C_f V_f \left(\frac{\partial T_f}{\partial t} + U \frac{\partial T_f}{\partial z} \right) = h S_s \left(T_s - T_f \right) \tag{4.62}$$

For the PCM filler, we needed only to consider the energy exchange between it and the fluid, and the change of internal energy of the PCM filler over the instance in time; see Tumilowicz et al. (2013). The PCM filler governing equation became

$$\rho_s V_s \frac{\partial \bar{h}_s}{\partial t} = -h S_s \left(T_s - T_f \right) \tag{4.63}$$

The preceding equations still retain a PCM filler temperature term, for which we applied an equation of state to relate to the enthalpy per PCM filler phase state.

$$T_s = \begin{cases} \dfrac{\bar{h}_s - \bar{h}_{s\_ref}}{C_{s\_s}} + T_{s\_ref} & \bar{h}_s < \bar{h}_{s\_melt} \\ T_{s\_melt} & \text{for } \bar{h}_{s\_melt} < \bar{h}_s < \bar{h}_{s\_melt} + L \\ \dfrac{\bar{h}_s - (\bar{h}_{s\_melt} + L)}{C_{s\_\ell}} + T_{s\_melt} & \bar{h}_{s\_melt} + L < \bar{h}_s \end{cases} \tag{4.64}$$

For this enthalpy-based one-dimensional model, the lumped capacitance assumption is applied, considering the fact that the size of encapsulated fillers is small, which ensures a small Biot number. However, if the encapsulated filler material is large, which gives a large Biot number, the internal thermal resistance becomes significant, and a modification to the lumped capacitance has to be considered. This can be done by introducing an effective convective heat transfer coefficient in Eqs. (4.62) and (4.63). Detailed derivation of the formula of the effective heat transfer coefficient has been provided by Xu et al. [4].

For generality, dimensionless analysis was applied to the governing equations by introducing the following variables. Temperatures T_H and T_L are characteristic to the working system, representing the highest temperature inlet fluid used for heating during the charge process, and the lowest temperature fluid used for cooling during the discharge process, respectively.

$$\theta_f = \frac{T_f - T_L}{T_H - T_L} \tag{4.65}$$

$$\theta_s = \frac{T_s - T_L}{T_H - T_L} \tag{4.66}$$

$$t^* = \frac{t}{H/U} \tag{4.67}$$

$$z^* = \frac{z}{H} \tag{4.68}$$

$$\eta_s = \frac{\bar{h}_s - \bar{h}_{s\_ref}}{C_{s\_s}(T_{melt} - T_L)} \tag{4.69}$$

The governing equations in dimensionless form followed. Note that the subscripted "melt" and "r_ref" variables correspond to values plugged directly into their dimensionless variable (i.e., $\theta_s(T_s = T_{s\_melt}) \rightarrow \theta_{s\_melt}$).

$$\frac{\partial \theta_f}{\partial t^*} + \frac{\partial \theta_f}{\partial z^*} = \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.70}$$

$$\frac{\partial \eta_s}{\partial t^*} = \frac{-H_{CR}}{\tau_r}\frac{1}{\theta_{s\_melt}}\left(\theta_s - \theta_f\right) \tag{4.71}$$

$$\theta_s = \begin{cases} \eta_s \theta_{s\_melt} + \theta_{s\_ref} & \eta_s < \eta_{s\_melt} \\ \theta_{s\_melt} & \text{for } \eta_{s\_melt} < \eta_s < \eta_{s\_melt} + \dfrac{1}{Stf} \\ \left(\eta_s - \left(\eta_{s\_melt} + \dfrac{1}{Stf}\right)\right)\left(\dfrac{C_{s\_s}}{C_{s\_\ell}}\right)\theta_{s\_melt} + \theta_{s\_melt} & \eta_{s\_melt} + \dfrac{1}{Stf} < \eta_s \end{cases} \tag{4.72}$$

with dimensionless parameters defined as

$$\tau_r = \frac{U}{H}\frac{\rho_f C_f V_f}{h S_s} \tag{4.73}$$

$$H_{CR} = \frac{\rho_f C_f V_f}{\rho_s C_{s\_s} V_s} \tag{4.74}$$

$$Stf = \frac{C_{s\_s}(T_{s\_melt} - T_L)}{L} \tag{4.75}$$

4.3.2 Numerical Method and Procedures of Solution

To solve the equations presented, the method of characteristics was applied. Using an equal step size in both time and space $\Delta t^* = \Delta z^*$, we chose a numerical grid featuring both diagonal characteristics $t^* = z^*$ and vertical characteristics $z^* = constant$, as shown in Fig. 4.5. Steps in time progressed for $j = 1, 2, \ldots, N$, while steps in space progressed for $i = 1, 2, \ldots, M$.

Along the diagonal characteristic $t^* = z^*$, we recognized the substantial derivative in Eq. (4.70):

$$\frac{D\theta_f}{Dt^*} = \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) \tag{4.76}$$

Here, we first saw the advantage of applying the method of characteristics to this hyperbolic system. By choosing the diagonal characteristic, the fluid equation reduced from a partial differential equation to an ordinary differential equation along this curve. Separating and integrating along the characteristic, we obtained:

$$\int d\theta_f = \int \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.77}$$

We continued with a similar process for the PCM filler energy balance. Eq. (4.71) was solved along the characteristic $z^* = constant$. Again applying separation and integrating, we obtained

$$\int d\eta_s = \int \frac{-H_{CR}}{\tau_r}\frac{1}{\theta_{s\_melt}}\left(\theta_s - \theta_f\right) dt^* \tag{4.78}$$

Clearly, the result represented a system of ordinary differential equations along their corresponding characteristics. Referring back to the grid in Fig. 4.5, we see the two characteristics intersect as they progress in time and space. We exploited this in our manner of solution throughout the nodal grid. The hyperbolic nature of the original equations passes information from node to node in a wave-like fashion. As an example, we chose two neighboring spatial nodes at time $j = 1$, $\vartheta_{1,1}$ and $\vartheta_{2,1}$, which will serve as the starting points for information propagation through their corresponding characteristics. After the passing of one time step to $j = 2$, the meeting point of the two characteristics, $\vartheta_{2,2}$, will have received information from the two starting nodes. To represent this mathematically, we applied numerical integration to the equations. Along the diagonal characteristic, Eq. (4.77) became

$$\int_{\vartheta_{1,1}}^{\vartheta_{2,2}} d\theta_f = \int_{\vartheta_{1,1}}^{\vartheta_{2,2}} \frac{1}{\tau_r}\left(\theta_s - \theta_f\right) dt^* \tag{4.79}$$

Applying the trapezoidal rule for integration of the right-hand side, the solution to Eq. (4.79) became

$$\theta_{f_{2,2}} - \theta_{f_{1,1}} = \frac{\Delta t^*}{\tau_r}\left(\frac{\theta_{s_{2,2}} + \theta_{s_{1,1}}}{2} - \frac{\theta_{f_{2,2}} + \theta_{f_{1,1}}}{2}\right) \tag{4.80}$$

Repeating the process for Eq. (4.78), we obtained

$$\int_{\vartheta_{2,1}}^{\vartheta_{2,2}} d\eta_s = \int_{\vartheta_{2,1}}^{\vartheta_{2,2}} \frac{-H_{CR}}{\tau_r}\frac{1}{\theta_{s\_melt}}\left(\theta_s - \theta_f\right) dt^* \tag{4.81}$$

Once again implementing the trapezoidal rule for integration of the right-hand side, the integral solution became:

$$\eta_{s_{2,2}} - \eta_{s_{2,1}} = -\frac{\Delta t^* H_{CR}}{\tau_r \theta_{r\_melt}}\left(\frac{\theta_{s_{2,2}} + \theta_{s_{2,1}}}{2} - \frac{\theta_{f_{2,2}} + \theta_{f_{2,1}}}{2}\right) \tag{4.82}$$

Using the equation of state Eq. (4.72) to transform the unknown PCM filler temperature value at node $\vartheta_{2,2}$ to enthalpy based on the local PCM filler phase state left the system of two equations (4.80) and (4.82) to solve for the two unknowns, $\theta_{f_{2,2}}$ and $\eta_{r_{2,2}}$. With these values obtained, we stepped once in space to the new pair of neighboring nodes, $\vartheta_{2,1}$ and $\vartheta_{3,1}$, and used them identically to obtain values at node $\vartheta_{3,2}$. This was repeated until all values were found at $j=2$. We then fully repeated the spatial sweep at $j=2$ to obtain all new values at $j=3$.

Thus, with a boundary condition provided at the inlet $i=1$, along with an initial condition at time $j=1$ in the storage tank, solutions could be swept through space, stepped in time, and repeated, until the entire grid was fully calculated. Application of the trapezoidal rule for numerical integration implies accuracy of order $O(\Delta t^{*2})$ [28]. The preceding was a mere example calculation outlining the numerical process applied. With PCM filler phase state changes added, numerous technicalities had to be considered throughout the full application. While use of the state equation and solution of the system of equations followed simply in regions of continuous phase state, PCM filler phase state interfaces and their travel throughout the grid of characteristics created a plethora of more complex calculations. The necessary numerical considerations were most generally divided into calculations of type "boundary" and "spatial." Within these, we covered all cases of heating and cooling between different types of PCM filler phase states, PCM filler phase state interface presence/positioning, and the possibilities of

overheating and undercooling. These more extreme cases were a necessary consideration with the application of the method of characteristics. In the chosen numerical grid, information could only propagate at the characteristic speed. With extreme heating or cooling conditions, an interface could be found traveling faster than the maximum allowed 45 degrees (the diagonal characteristic), at which point its position had to be suppressed to the maximum, and the additional energy transfer was accumulated in the resulting PCM filler enthalpy at the phase state interface.

At the boundary ($i = 1$), only the vertical characteristic was used for calculation, and known fluid values allowed direct calculation at this point in space for all instances in time. Analysis of PCM filler enthalpy results revealed any resulting phase state changes in the filler. Overall, the model included a total of 11 considerations for grid calculations of this type: (1) continuous solid phase state, (2) continuous melting phase state, (3) continuous liquid phase state, (4) solidus phase state interface onset with heating, (5) solidus phase state interface onset with cooling, (6) liquidus phase state interface onset with heating, (7) liquidus phase state interface onset with cooling, (8) solidus phase state interface onset with overheating, (9) solidus phase state interface onset with undercooling, (10) liquidus phase state interface onset with overheating, and (11), liquidus phase state interface onset with undercooling.

To handle the discretized grids in the tank space, we worked with the system of equations resulting from the intersection of the two characteristics. Specific calculation depended on the phase state(s) present, and the phase state interface placement among the corresponding characteristic lines. To better detail the process, we considered a case where phase state interface travel during an instance in time leaves it within the same block in the discretized grid. With all values at the current time known, we first shifted the solution characteristics to solve for the phase state interface position at the new time, β_{New}, which represented a fractional positioning of the phase state interface in the grid $0 < \beta < 1$. The corresponding general solution space in the discretized grid can be viewed in Fig. 4.13.

The characteristics were discretized as before, with integration of the fluid equation from point 5 to the interface location β_{New}, and of the PCM filler enthalpy equation from point 6 to the interface location β_{New}. Application of linear interpolation (using the nearest known points) for intermediate terms 5 and 6, followed by elimination of the unknown fluid temperature at β_{New} from the system, resulted in a second-degree polynomial to solve for β_{New}. From the previously mentioned definition, $0 < \beta < 1$, selection of the proper root followed simply. With the position

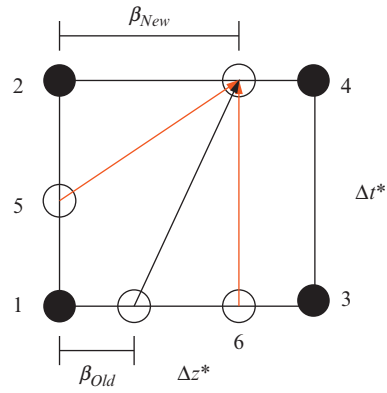


Fig. 4.13 Solution grid for first step of standard spatial interface travel.

at the new time found, the corresponding fluid value was obtained from the same characteristic equations.

With travel of the phase state interface now defined within the grid, we continued to define necessary values at the upper right point in the spatial sweep. Due to the presence of a phase state interface, the solution characteristic must have been divided accordingly. The general procedure is depicted in Fig. 4.14.

Linear interpolation was applied along the interface for values at its intersection with the diagonal characteristic. Finally, the system of equations defined by the two preceding characteristics was solved for the fluid and PCM filler temperatures at point 4.

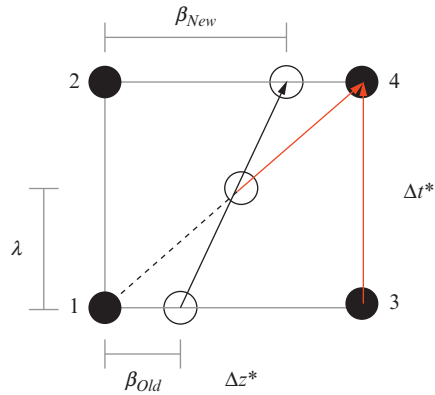


Fig. 4.14 Solution grid for second step of standard spatial interface travel.

Overall, the model included 19 considerations for grid calculations of spatial type: (1) continuous solid phase state, (2) continuous melting phase state, (3) continuous liquid phase state, (4) solidus phase state interface within grid with heating, (5) solidus phase state interface within grid with cooling, (6) liquidus phase state interface within grid with heating, (7) liquidus phase state interface within grid with cooling, (8) solidus phase state interface crossing vertical characteristic with heating, (9) solidus phase state interface crossing vertical characteristic with cooling, (10) liquidus phase state interface crossing vertical characteristic with heating, (11) liquidus phase state interface crossing vertical characteristic with cooling, (12) solidus phase state interface leaving spatial domain with heating, (13) solidus phase state interface leaving the spatial domain with cooling, (14) liquidus phase state interface leaving the spatial domain with heating, (15) liquidus phase state interface leaving the spatial domain with cooling, (16) solidus phase state interface crossing vertical characteristic with overheating, (17) solidus phase state interface crossing vertical characteristic with undercooling, (18) liquidus phase state interface crossing vertical characteristic with overheating, and (19) liquidus phase state interface crossing vertical characteristic with undercooling.

Extensive verifications were done to ensure the accuracy and robustness of the algorithm. The readers are referred to the details in the reference by Tumilowicz et al. [40]. Here, we will present a couple of examples to demonstrate the capabilities of the algorithm.

4.3.3 Examples of Results From Numerical Solution for Packed Bed of PCM Capsules

In this section, we briefly show some of the developed encapsulated PCM model's capabilities in application. We first consider a simple charge/discharge process for a thermocline utilizing this sort of material. We start with the charge process, with key numerical parameters set to

$$H_{CR} = 0.5785;\ \ \tau_r = 0.1117;\ \ \theta_{f_{inlet}} = 1;\ \ \theta_{f_o} = \theta_{s_o} = 0;\ \ \theta_{s\_melt} = 0.5$$
$$\theta_{s\_ref} = 0;\ \ \eta_{s\_melt} = 1;\ \ \frac{C_{s\_s}}{C_{s\_\ell}} = 1.1268;\ \ Stf = 0.1143;\ \ \Delta t^* = \Delta z^+ = 0.001 \tag{4.83}$$

The initially uniform temperature in the tank is heated by an HTF at a higher constant temperature. Heating occurred from the initial state to a final dimensionless time of 4.0, for which temperature profiles are displayed

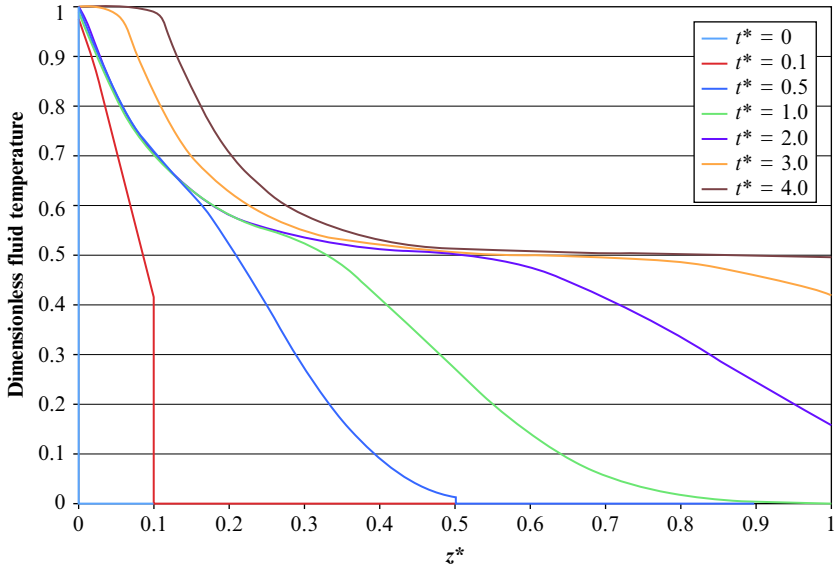


Fig. 4.15 Fluid temperature profiles at various instances of time for charge process of PCM filler.

at dimensionless times 0, 0.1, 0.5, 1.0, 2.0, 3.0, and 4.0. These can be viewed in Figs. 4.15 and 4.16, for the fluid and PCM filler, respectively.

The material behavior in the tank during the process can be observed. With the time allotted, the higher temperature HTF inflow raised the PCM filler temperature, melted it fully, and continued heating it in its liquid phase state. We see the PCM filler first started transitioning into its melting phase state, $\theta_{r\_melt} = 0.5$, soon after dimensionless time 0.1, and then finally collecting enough energy to continue to a liquid PCM filler just before dimensionless time 2.0.

Spatially, the positioning of the corresponding PCM filler phase states became quite clear. At the end of the charge, with PCM filler temperature at the outlet remaining just below the PCM filler melting temperature, heat propagation sent the solidus phase state interface near the end of the tank, leaving only a small amount of solid PCM filler near the outlet. The constant filler temperature of 0.5 throughout most of the space shows the large latent heat maintaining the PCM filler in this intermediate melting phase state. Finally, near the inlet, the temperature of the PCM filler finally rose above 0.5, signifying a liquid PCM filler, and placing the liquidus phase state interface near this inlet boundary. The interface positions as functions of time in Fig. 4.17 confirm this behavior.

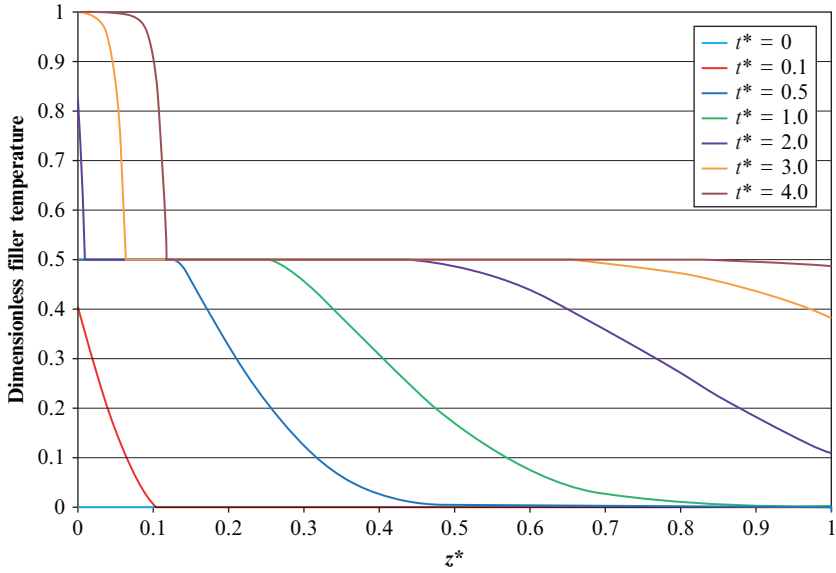


Fig. 4.16 Filler temperature profiles at various instances of time for charge process of PCM filler.

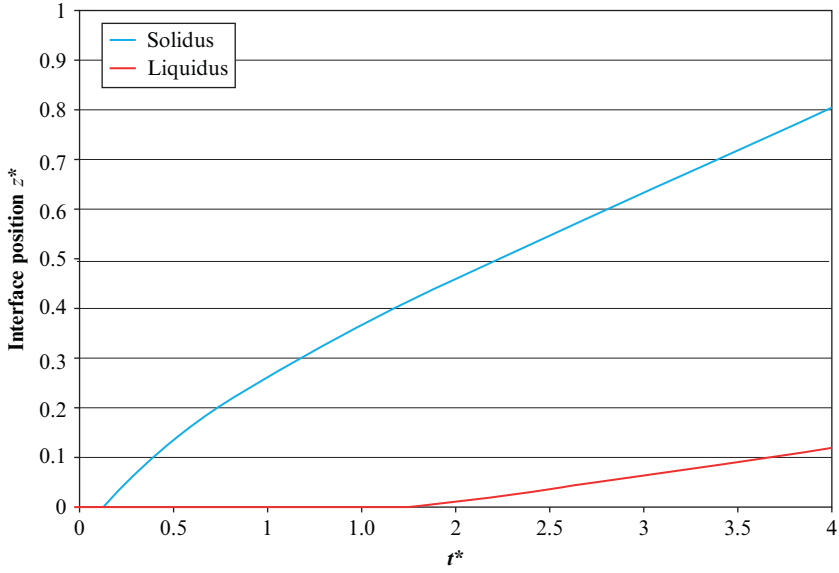


Fig. 4.17 Phase change interface positions as functions of time for standard charge process of PCM filler.

The plot confirms what is seen in the temperature distributions, in Fig. 4.16. Melting occurred relatively soon after heating began, corresponding to the start of the solidus phase state interface. With continued heating, this interface traveled through space, and is left near the end of the tank. Late in the charging cycle, enough heat had been input to produce a fully liquid PCM filler, corresponding to the beginning of a liquidus phase state interface. With only a little time left in the process, this second interface traveled only slightly into the space.

Postcharge process, we review how the tank achieves equilibrium to extract the effects of having a PCM filler. An equilibration calculation was applied to the final fluid and PCM filler temperature profiles before turning fluid flow around and discharging the stored energy. The final profiles of the tank, along with their corresponding equilibrium profile, can be viewed below in Fig. 4.18.

Reviewing the results, we see the effects of latent heat of equilibration results. With the end of charge profile featuring mostly melted PCM filler, the temperature difference between fluid and PCM filler was not enough to lift the PCM filler above its melting temperature. Thus, we saw a larger reduction in a relatively wide portion of the fluid temperature profile from equilibration, leading to a larger decrease of fluid output temperature for eventual energy production.

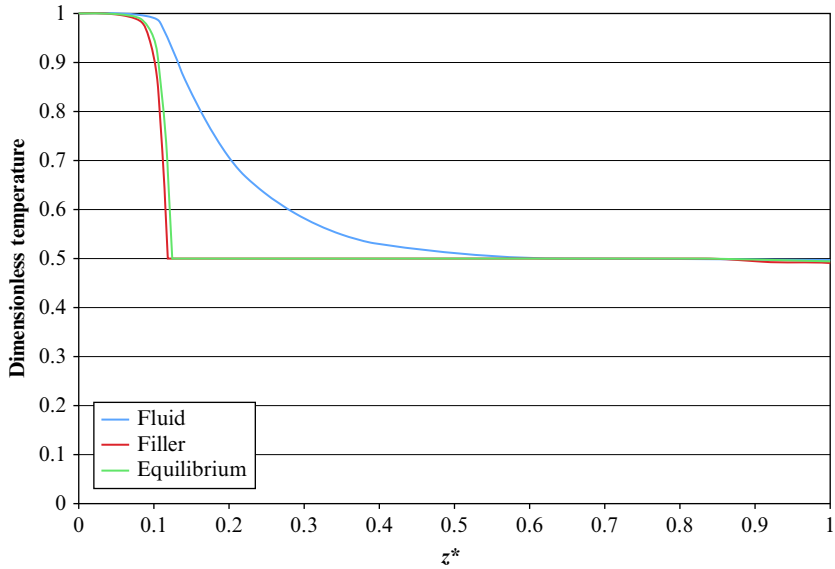


Fig. 4.18 Equilibration of final temperature profiles after charge process for PCM filler.

With equilibrium applied, the resulting temperature profile became the initial fluid and PCM filler condition for the discharge process. The tank was cooled at the inlet by a constant low fluid temperature, and key numerical parameters remained the same. Discharge was again run to a final dimensionless time of 4.0, for which the tank profiles are displayed at dimensionless time values 0, 0.1, 0.5, 1.0, 2.0, 3.0, and 4.0. Results are plotted in Figs. 4.19 and 4.20 for the fluid and PCM filler, respectively.

For this discharge process, the profiles behaved in a reverse order from the charge. Cooling brought the PCM filler temperature down, reduced the amount of liquid PCM filler, and increased the amount of solid PCM filler. At dimensionless time 1.0, cooling sent the liquidus phase state interface out of the storage tank, thus solidifying all liquid PCM filler. Heat extraction continued until the solidus phase state interface traveled past the position 0.8, leaving mostly solid PCM filler, with a comparatively small amount of melting PCM filler. The fluid temperature profiles followed as expected, with discontinuities between inlet affected fluid and initial state seen before dimensionless time 1.0, as before.

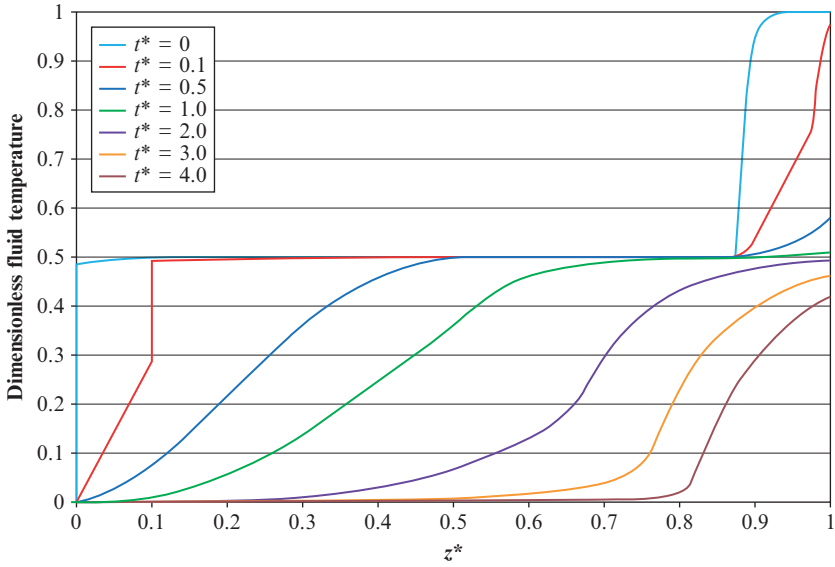


Fig. 4.19 Fluid temperature profiles at various instances of time for discharge process for PCM filler.

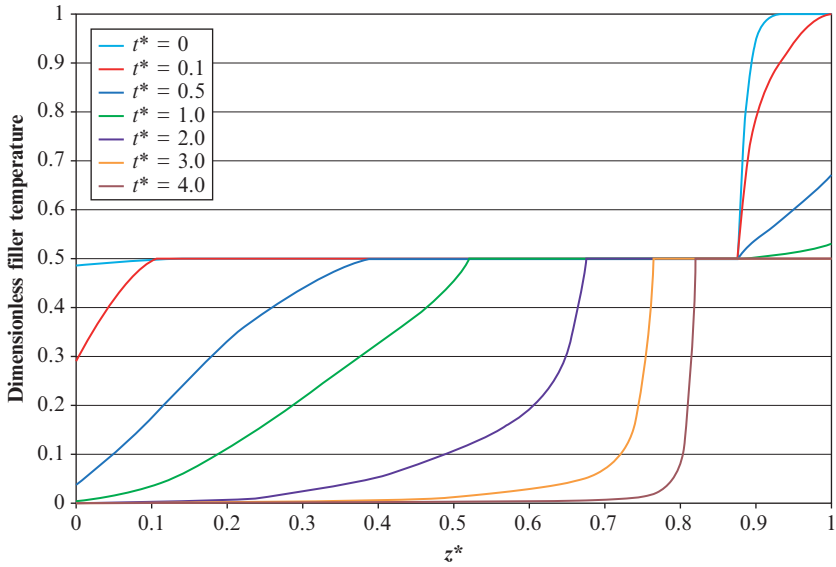


Fig. 4.20 Filler temperature profiles at various instances of time for discharge process for PCM filler.

Variable Heat Transfer Fluid Temperature at the Inlet

We showcase the versatility of the model by applying a variable boundary condition. With it, we aimed to apply heat to the PCM filler, forcing phase change(s), at which time the boundary condition was reversed, removing heat to produce an additional phase state interface reverting to the earlier phase state(s). Seeing satisfactory modeling in a complex heat transfer case such as this will confirm the general consideration implemented. The model was run as a charge process until a final dimensionless temperature equal to 5.0. Key properties were as follows:

$$\begin{gathered} H_{CR} = 0.5785;\ \ \tau_r = 0.1117;\ \ \theta_{f_{inlet}} = 1; \\ \theta_{f_o} = \theta_{s_o} = 0;\ \ \theta_{s\_melt} = 0.1;\ \ \theta_{s\_ref} = 0; \\ \eta_{s\_melt} = 1;\ \ \frac{C_{s\_s}}{C_{s\_\ell}} = 1.1268;\ \ Stf = 0.1143; \\ \Delta t^* = \Delta z^+ = 0.001 \end{gathered} \tag{4.84}$$

with inlet fluid temperature as follows.

$$\theta_{s\_inlet} = \begin{cases} 1.0 & 0 \le t^* < 1.5 \\ 0.6 & 1.5 \le t^* < 2.5 \\ 0.3 & 2.5 \le t^* < 4.0 \\ 0 & 4.0 \le t^* < 5.0 \end{cases} \quad \text{for} \tag{4.85}$$

This boundary condition represented a discontinuous series of constant fluid temperature values applied over specific amounts of time. Heating was applied quickly to produce two phase state interfaces, and then a gradually stepped cooling was used to revert to a previous phase state. Below, we trace the temperature profile evolutions throughout the various changes in boundary condition.

We first look at the temperatures within the tank at dimensionless time 1.499, which is the last instance of inlet fluid condition equal to 1.0. This can be viewed in Fig. 4.21. The profiles produced represent a simple constant inlet PCM filler temperature charge. Heating produced two phase state interfaces, leaving three separate phase states of PCM filler in the tank. Though paraffin features a large latent heat, the large temperature gradient left above the melting temperature due to the high inlet fluid temperature heated through the melting PCM phase state relatively quickly. We now continue to the distributions at dimensionless time 2.499, Fig. 4.22, which is the last instance in time with the inlet temperature of 0.6 as the boundary condition.

At this instant in time, we see an interesting response in the profiles. Due to the wide spread of PCM filler temperature after the previous heating, the new inlet condition worked to both cool and heat, depending on the inlet

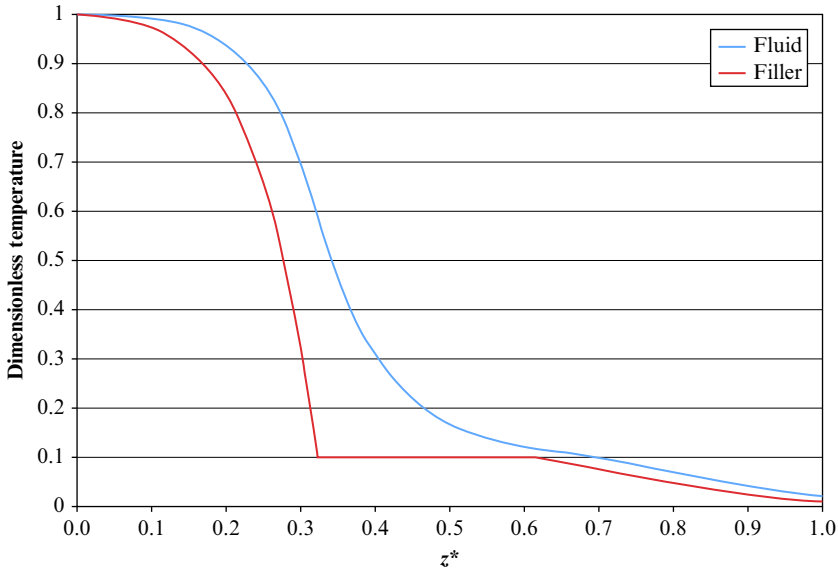


Fig. 4.21 Fluid and Filler temperature profiles at dimensionless time 1.499.

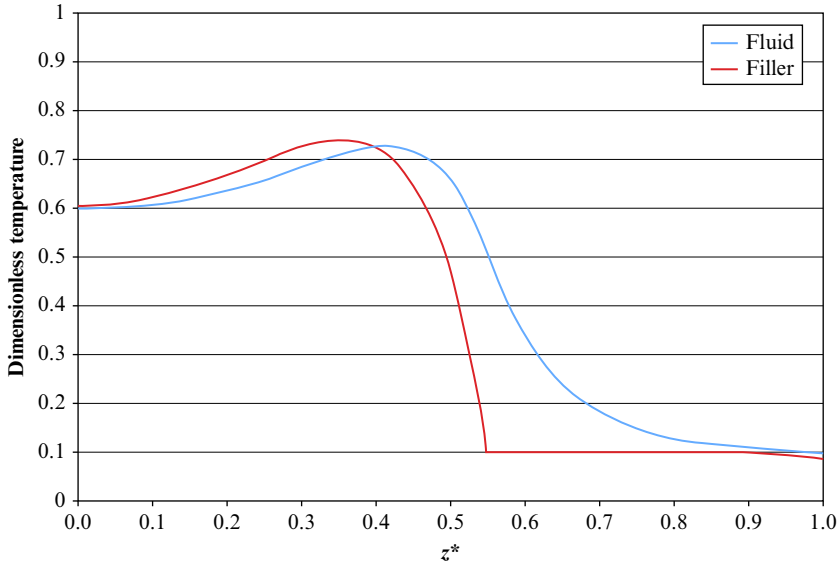


Fig. 4.22 Fluid and filler temperature profiles at dimensionless time 2.499.

fluid's relation to the temperatures within the tank at a certain position. The effect was seen throughout the entire tank, as this new condition, traveling at the characteristic speed, was able to traverse the entire space in the $\Delta t^* = 1.0$ it was applied. When interacting with the high temperatures towards the tank inlet, we saw a cooling that brings both PCM filler and fluid profiles gradually down to the inlet temperature condition. Further in space, where tank temperature values remain lower, the new condition continued to heat the tank profiles. The two heat transfer types met in a continuous parabolic fashion, representing the eventual change of liquid PCM filler being heated to liquid PCM filler experiencing cooling. Heat propagation through the tank had also facilitated the propagation of the two previously mentioned interfaces, as expected. The initial solidus phase state interface was found very close to the outlet, leaving little solid PCM filler in the tank, while the creation of more liquid PCM filler placed the liquidus phase state interface past the midway point in the tank.

Past this time, another lower fluid boundary temperature was applied up until the dimensionless time 3.999. The profiles at this time can be viewed in Fig. 4.23. As expected, this new inlet fluid temperature dropped temperature profiles even lower. All solid PCM filler has now melted, meaning the solidus phase state interface has left the space at some time during the application of this inlet condition. Conditions had also produced even more

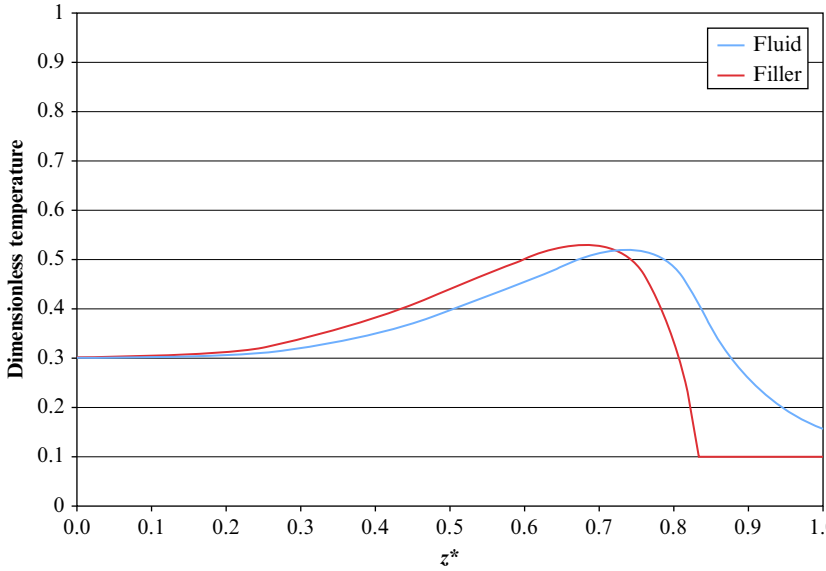


Fig. 4.23 Fluid and filler temperature profiles at dimensionless time 3.999.

liquid PCM filler, and correspondingly shifted the liquidus phase state interface further towards the outlet. Finally, the last drop in boundary temperature was applied from dimensionless time 4.0 through the end of the run. The resulting end temperature profile can be viewed in Fig. 4.24.

With this inlet fluid temperature, the system saw the lowest cooling temperature in the run. The PCM filler temperature was even further decreased, eventually returning the PCM filler at the boundary back to a melting PCM filler, and creating a second liquidus phase state interface. The original melting PCM filler was almost all fully liquefied, meaning we expected the first liquidus phase state interface to be very close to the tank outlet. This left three unique phase states in the tank from input to output—melting, liquid, and melting. To complete our understanding of the process, we compiled the travel of interfaces throughout. This can be viewed in Fig. 4.25.

The interface travel confirms what was outlined in the preceding profiles. The first phase state interface, of solidus type, was created early in the heating. Slightly past the halfway mark of the process, this phase state interface traveled far enough to leave the space entirely, taking all solid PCM filler with it. After heating through the latent region of melting PCM filler, the first liquidus phase state interface was created, bringing with it liquid PCM filler to the tank. This phase state interface traveled nearly the

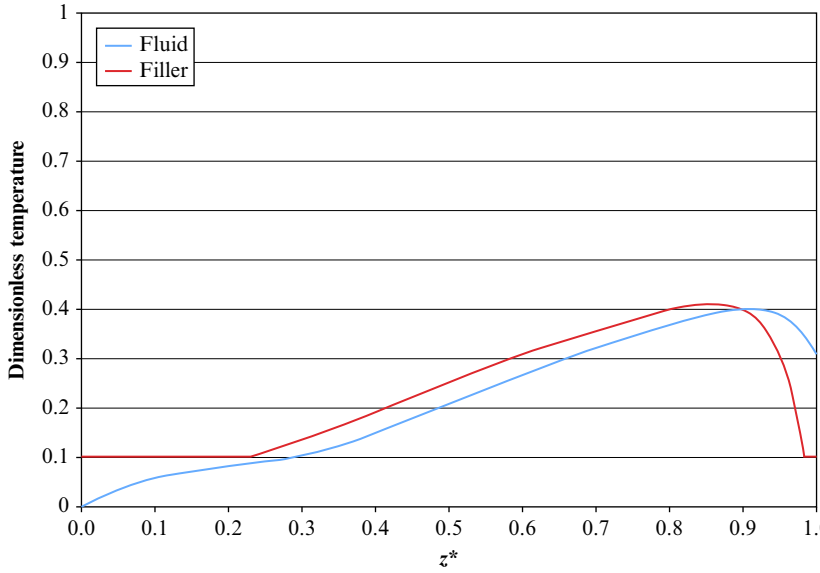


Fig. 4.24 Fluid and filler temperature profiles at dimensionless time 5.0.

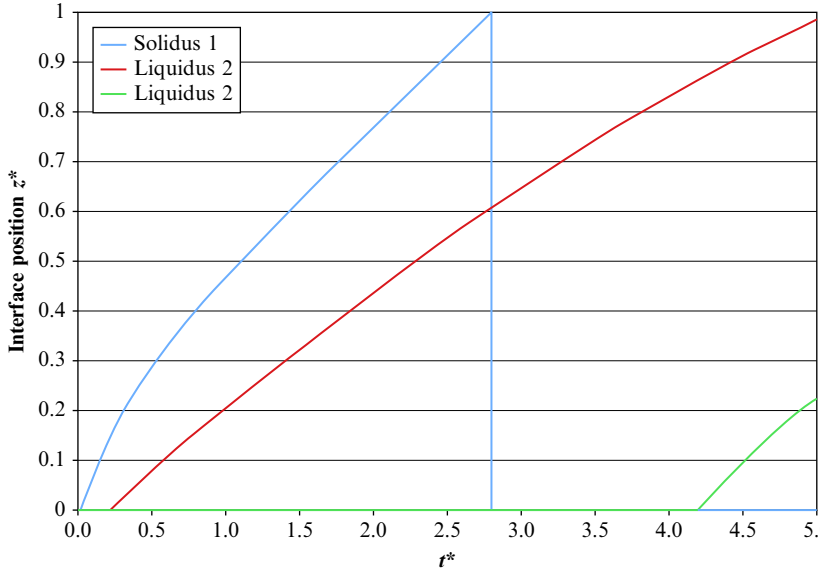


Fig. 4.25 Phase change interface positions as functions of time for variable boundary charge process.

entire space, and turned nearly all the remaining melting PCM filler into liquid PCM filler. Finally, after adequate cooling, a second liquidus phase state interface was created as the liquid PCM filler began returning to a melting PCM filler. This interface saw little travel, as this phase change occurred near the end of the run. Thus, throughout the entire case, heating and cooling created three unique phase state interfaces, all of which were tracked by the model. Interface travel was clearly nonlinear, though speed decreased to a near constant rate as the temperature difference between fluid and PCM filler driving their travel decreased with time.

4.4 VALIDITY OF ONE-DIMENSIONAL MODELS OF SENSIBLE AND LATENT HEAT THERMAL STORAGE

4.4.1 Sensible Thermal Storage

In Section 4.2.3 we presented the derivations of four different geometries. Table 4.1 lists the formulas of the effective heat transfer coefficient h_{eff}, on the basis of the intrinsic heat transfer coefficients h, for the four fluid-solid structural combinations shown in Fig. 4.3. All four solid-fluid configurations are generally viewed as systems with an HTF flowing through a porous medium, each should have porosity, or equivalent porosity for cases in Fig. 4.3B–D, defined as the ratio of the volume of fluid over the total volume of the storage tank. The governing equations of the energy balance for the fluid and solid are given in Section 4.3.4 and Eqs. (4.19) and (4.20). The hot HTF flows downward through the packed bed during heat charging, and the cold HTF flows upward during heat discharging. The fluid inlet (at the top during charging and at the bottom during discharging) always has $z = 0$. The flow velocities are assumed to have a uniform radial distribution, which Yang and Garimella [41] have proven to be reasonable for packed beds.

In cases where the solid material is not in the form of spheres packed, as shown in Fig. 4.3A, but in a form such as those shown in Fig. 4.3B–D, equivalent porosity must be used. With equivalent porosity, the same governing equations can be used; however, an effective heat transfer coefficient h_{eff}, as given in Table 4.1, must be used to replace the original (or intrinsic) heat transfer coefficient h in Eqs. (4.19) and (4.20).

The parameter S_s in the governing equations, Eqs. (4.19) and (4.20), is the total surface area of solid thermal storage material in contact with HTF per unit length of the thermal storage tank (see Li et al. [42] for details on obtaining this parameter). As a consequence, the heat transfer and energy

balance in the solid-fluid structural combinations in Fig. 4.3B–D can all be analyzed by only considering one typical volume, which includes a typical solid and fluid region as indicated on the right side of each of the configurations in Fig. 4.3. For the case of plates, the length in the direction normal to the paper in Fig. 4.3B is chosen to be a unit length of $L = 1.0\,m$. As a result, the flow channel will have a ratio of $L/D_f > 20$, where D_f is the width of the flow channel. Table 4.4 gives the obtained S_r for the other three cases in Fig. 4.3.

Introducing the following dimensionless variables,

$$\theta_f = \left(T_f - T_L\right)/\left(T_H - T_L\right) \tag{4.86.a}$$

$$\theta_r = \left(T_{LM} - T_L\right)/\left(T_H - T_L\right) \tag{4.86.b}$$

$$z^* = z/H \tag{4.86.c}$$

$$t^* = t/(H/U) \tag{4.86.d}$$

the governing equations become

$$\frac{\partial \theta_f}{\partial t^*} + \frac{\partial \theta_f}{\partial z^*} = \frac{1}{\tau_r}\left(\theta_r - \theta_f\right) \tag{4.87}$$

$$\frac{\partial \theta_r}{\partial t^*} = -\frac{H_{CR}}{\tau_r}\left(\theta_r - \theta_f\right) \tag{4.88}$$

where

$$\tau_r = \frac{U}{H}\frac{\rho_f C_f \varepsilon \pi R^2}{h_{eff} S_{filler}} \tag{4.89}$$

and

$$H_{CR} = \frac{\rho_f C_f \varepsilon}{\rho_r C_r (1-\varepsilon)} \tag{4.90}$$

Under the assumption of no heat loss from the thermal storage tank, it is reasonable that the equilibrium temperature between the HTF and the solid

Table 4.4 The total heat transfer surface area of a solid per unit height of a typical volume as shown in Fig. 4.3B–D

	Total heat transfer surface area (m^2)	Height (m)	S_{filler} (m)
Plate	$2(1 \times H)$	H	2
Cylinder	$2\pi rH$	H	$2\pi r$
Tube	$2\pi aH$	H	$2\pi a$

filler material at the end of one charge or discharge will necessarily be the initial condition of the next discharge or charge process in the thermal storage cycle. This connects the discharge and charge processes so that results of a number of periodic charges and discharges can be obtained.

For the initial condition of fluid and filler material in any charging or discharging process, $t^* = 0$; $\theta_r = \theta_f$, which is the equilibrium state after settling down from the last process.

For the inlet condition, $z^* = 0$ and $\theta_f = 1$ for a charging process; otherwise $\theta_f = 0$ for a discharging process.

The θ_r at the inlet boundary can be directly calculated using Eq. (4.88) from the known inlet fluid temperature.

A pair consisting of an HTF and a thermal storage material has been chosen for the one-dimensional model as well as the CFD study for the purpose of comparison and verification. The high temperature of fluid in the charge process is 390°C (663.15 K) and the low temperature of fluid flowing into the tank during a discharge is 310°C (583.15 K). The HTF is HITEC molten salt [43], having properties of kinetic viscosity $\nu_f = 1.17 \times 10^{-6}\mathrm{m}^2/\mathrm{s}$, heat capacity $Cp_f = 1549.12\mathrm{J/kgK}$, density $\rho_f = 1794.07\mathrm{kg/m}^3$, and thermal conductivity k_f =0.57 W/m K. The thermal storage material is another molten salt with properties of $\rho_r = 1680\mathrm{kg/m}^3$, $C_r = 1560\mathrm{J/(kgK)}$, and $k_r = 0.61\mathrm{W/(mK)}$. The required time period for energy charge and discharge is 4 h each.

The three cases of solid-fluid structural combinations, as shown in Fig. 4.3B–D, were calculated using the method of characteristics based on the 1D model. For the convenience of defining the dimensions of the solid and liquid area, as well as the computational domain for CFD analysis, an equivalent control volume was defined. For the bundle of solid rods or the fluid tubes in Fig. 4.3C and D, we may consider that each rod or tube has a control area enclosed by a hexagon, as shown in Fig. 4.26. The outer boundary surface is thermally insulated. In the CFD analysis, the hexagon is equivalent to a circle in the same area. A comprehensive CFD analysis was performed by Li et al. [44] in which they found that the energy storage is the same in both hexagon and circle geometries.

As shown in Fig. 4.26, the equivalent diameter for the control area should satisfy the relationship

$$A_{hexagon} = A_{eq} = \pi R_{eq}^2 \tag{4.91}$$

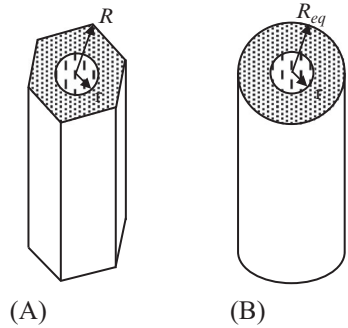


Fig. 4.26 (A) Hexagonal control volume; (B) equivalent circular control volume.

The equivalent diameters of the control area for the solid rod and the fluid tube are given in Table 4.5, which essentially defines the dimensions of flow channels and solid thermal storage materials. The height of the tank is 10 m. The related intrinsic heat transfer coefficient of laminar flow and the corrected heat transfer coefficient in the 1D model simulations are listed in Table 4.6.

The laminar flow heat transfer Nusselt numbers listed in Table 4.6 are based on heat transfer cases with constant wall heat flux [2] for fully developed internal flows. The full thermal and hydraulic development can be easily satisfied due to the very long channels. The assumption of a constant wall heat flux heat transfer is because the temperature difference between the solid materials and the fluid along the height of a storage tank is similar to that of a countercurrent flow heat exchanger—hot fluid charges into the tank from the top, where there is a high temperature, and during a discharge cold fluid flows into the tank from the bottom of the tank, where there is a low temperature. A detailed description of this variation of temperatures of fluid and solid during energy charge and discharge is given by Li et al. [45]. The Biot numbers of all cases of heat conduction in solid materials in this analysis are above 1.0.

The charging and discharging cyclic operations start with a charge to a cold tank. After several cyclic runs, the temperature distribution in a tank after a discharge becomes independent of the initial temperature distribution. This state is called a cyclic steady state, and it is the real situation in a solar power plant practical operation.

Table 4.5 Dimensions of the fluid channels and solid thermal storage structure

	Equivalent porosity ε	**Equivalent outer diameter D_{eq} (mm)**	**Diameter d (mm)**	**S_r**	**Fluid channel[a] $2x_1$ (mm)**	**Solid plate[a]** $2(x_2 - x_1)$ (mm)
Plate case	0.33	No	No	2.0	4.56	9.25
Cylinder case	0.33	43.5	35.6	0.1118	No	No
Tube case	0.33	43.5	25.0	0.0785	No	No

[a]See definition of x_1, x_2 in Fig. 4.3B.

Table 4.6 Parameters in the simulation of 1D transient model

Hydraulic diameter for Re and Nu	Nu number (constant heat flux)	Intrinsic heat transfer coefficient h (W/m^2K)	Effective heat transfer coefficient h_{eff}(W/m^2K)	τ_r	H_{CR}
Plate case ($D_h = 4x_1$)	8. 24	305	221	0.0039	0.5223
Cylinder case ($D_h = D_{eq} - d$)	5.18 (inner wall) (outer wall no heat flux)	352	99	0.0087	0.5223
Tube case ($D_h = d$)	4.36	94	64	0.0366	0.5223

The average fluid temperatures at height locations along the tank after a charging are shown in Fig. 4.27 for the number of cycles of charging/discharging as indicated by the legend. The velocity of HTF flowing into the tank for the following cases is $V_{inlet} = 1.36 \times 10^{-3}$m/s. The results become the same with no more changes in charge/discharge cycles after a certain number of cycles for all the cases. For the case of channels formed by flat plates, 11 cycles are required to reach the cyclic steady state; this number for the cylinder case and tube case is 10 and 7, respectively. Because the 1D transient model is rather convenient, it can simulate many cycles of charging/discharging in a relatively short computational time. This is the advantage of the 1D transient model over a CFD analysis, which would take a significant amount of computational time for the analysis of cyclic charges and discharges.

A comprehensive CFD study was employed to analyze the energy storage process and validate the 1D simplified transient model. The commercial software ANSYS Fluent@ 6.3 was chosen for this analysis, and GAMBIT 2.4 was used to generate the computational domain and grid system.

Computational Specifications

The flow and heat transfer in the thermal storage tank are incompressible and transient with constant properties. The fluid flow Reynolds numbers are usually in the laminar region. In the current three configurations of solid-fluid in storage tanks, the velocities of the HTF all equal $V_{inlet} = 1.36 \times 10^{-3}$m/s, according to the desired mass flow rate. Correspondingly, the Reynolds numbers are: $\mathrm{Re}_{plate} = 10.6$, $\mathrm{Re}_{tube} = 28.9$, and $\mathrm{Re}_{cylinder} = 9.2$. The hydraulic

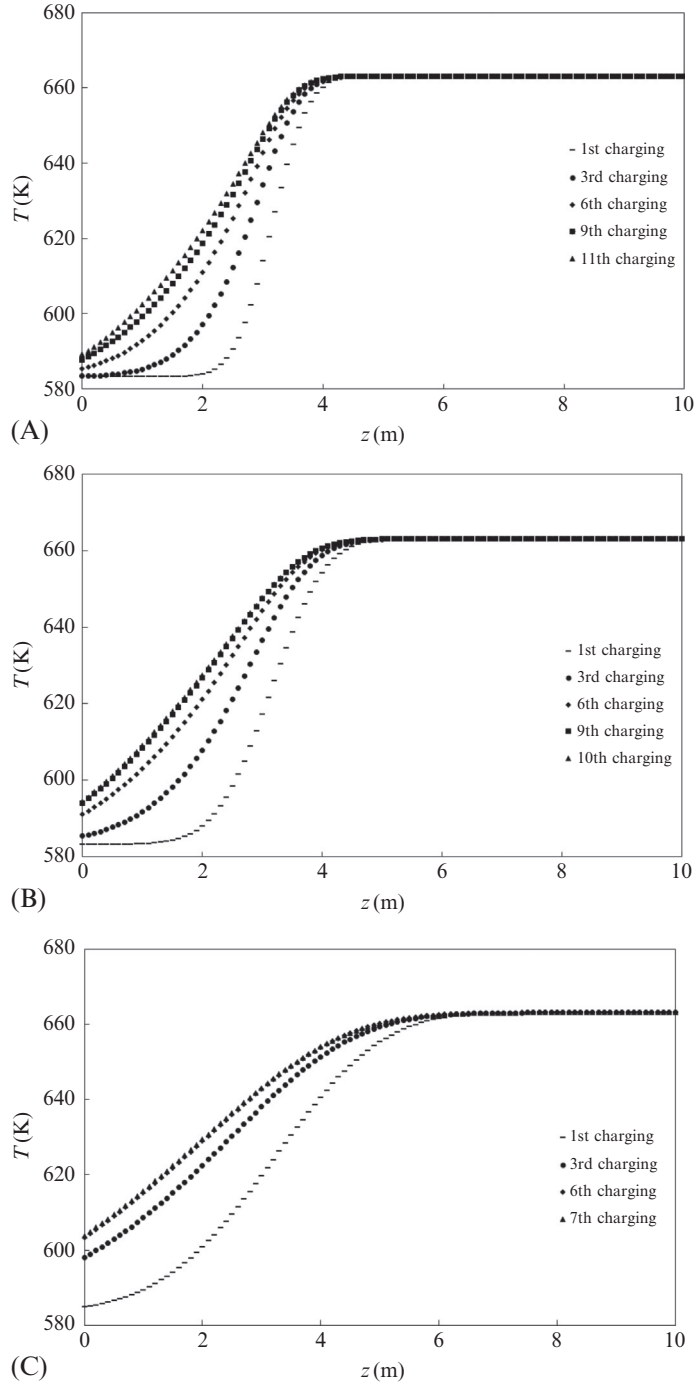


Fig. 4.27 The temperature of HTF at locations from bottom to top in a tank after heat charging processes. (A) Flat plate case, (B) cylinder case(C) tube case.

diameters defined in Table 4.6 for the flow channels were used to calculate these Reynolds numbers.

The continuity, momentum, and energy equations in differential form for laminar and incompressible flow are:

$$\nabla \cdot \vec{V} = 0 \tag{4.92}$$

$$\frac{\partial \vec{V}}{\partial t} + \left(\vec{V} \cdot \nabla\right) \vec{V} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{V} \tag{4.93}$$

$$\frac{\partial T}{\partial t} + \left(\vec{V} \cdot \nabla\right) T = \kappa \nabla^2 T \tag{4.94}$$

Because of the thermocline effect (hot fluid on top of cold fluid) in thermal storage tanks, typically there is no natural convection to consider. The computational domains and boundary conditions of the three cases are described in the following:

(1) For the solid plates and the 2D channels, as shown in Fig. 4.3B, half of the flow channel and half of the plate were included in the computational domain. At the symmetric line of the flow channel, $u_x = 0$, $\frac{\partial u_z}{\partial x} = 0$, and $\frac{\partial T}{\partial x} = 0$. At the symmetric line of the solid plate, $u_z = 0$, $u_x = 0$, and $\frac{\partial T}{\partial x} = 0$. At the solid walls of $z = 0$ and $z = H$, $u_z = 0$, $u_x = 0$, and $\frac{\partial T}{\partial z} = 0$. Full development conditions at the outflow boundary are used, as the flow channels are sufficiently long. This gives $u_x = 0$, $\frac{\partial u_z}{\partial z} = 0$, $\frac{\partial T}{\partial z} = 0$.

(2) For the solid rod with fluid flowing along the length, as shown in Fig. 4.3C, the rod and the equivalent circular area around the rod are included in the computational domain. The boundary conditions include: at the centerline of the rod, $u_z = 0$, $u_r = 0$, and $\frac{\partial T}{\partial r} = 0$. At the outer boundary, in the r direction, $u_r = 0$, $\frac{\partial u_z}{\partial r} = 0$, and $\frac{\partial T}{\partial r} = 0$. At the solid walls of $z = 0$ and $z = H$, $u_z = 0$, $u_r = 0$, and $\frac{\partial T}{\partial z} = 0$. At the outflow boundary, fully developed boundary conditions are: $u_r = 0$, $\frac{\partial u_z}{\partial z} = 0$, and $\frac{\partial T}{\partial z} = 0$.

(3) For the tubes surrounded by solid areas, as shown in Fig. 4.3D, a tube and the equivalent circular solid area around the tube are included in the computational domain. At the centerline of the tube, $u_r = 0$, $\frac{u_z}{\partial r} = 0$, and $\frac{\partial T}{\partial r} = 0$. At the outer boundary, in the r direction, $u_z = 0$, $u_r = 0$, and $\frac{\partial T}{\partial r} = 0$. At the solid walls of $z = 0$ and $z = H$, $u_z = 0$, $u_r = 0$, and $\frac{\partial T}{\partial z} = 0$. At the outflow boundary, fully developed boundary conditions are: $u_r = 0$, $\frac{\partial u_z}{\partial z} = 0$, and $\frac{\partial T}{\partial z} = 0$. The fluid and solid interfaces inside the computational domain have conjugated heat transfer, which can be typically treated in the software package ANSYS Fluent®.

Grid and Time Step Independent Study

A grid-independent study was conducted to choose a grid number that ensured high accuracy of computational results at a reasonable computational load. The flow and heat transfer fields were computed in 2D with cell numbers of 10,000, 20,000, 40,000, 60,000, and 80,000. For this study, we also assume the initial condition that the thermal storage tank is fully charged to a high temperature of 663.15 K, and the cold HTF with constant temperature 583.15 K flows into the tank with a constant velocity of $V_{inlet} = 1.36 \times 10^{-3}$m/s to extract the heat for 4 h. The temperature of the HTF at 1.0 m downstream of the inlet was examined for the transient process. It was found that the relative difference of the temperature at this location and any time instance during 4 h has a variation of no more than 10% when increasing the cell number from 10,000 to 20,000. The variation went down to 2.5% from 20,000 to 40,000, and 0.86% from 40,000 to 60,000, and 0.33% from 60,000 to 80,000. Consequently, 60,000 cells were adopted for the computational domains in all the formal computations of the study.

The time step for the transient flow field computation was set as 2 s, based on the time-step independence analysis. Time steps of 0.5, 1, 2, 4, and 5 s were tested in the computations. It was found that the relative difference of the temperature of the HTF at the location of 1 m from the inlet had a variation of 0.16% when increasing the time step from 0.5 to 1 s. The variation was 0.83% from 1 to 2 s, 1.95% from 2 to 4 s, and 3.67% from 4 to 5 s. Consequently, a small time step of 2 s was chosen for the computation of

the transient process. At each time step of computation, convergence was checked to meet a convergence criterion.

Comparison of CFD Results and the Results From 1D Modeling

The temperature of the HTF at the exit of a discharge is an important indication of energy storage efficiency. Therefore, the HTF temperature at the exit during the 4 h discharging period has been monitored for comparison between the CFD results and the 1D modeling results. The results are for the cases after a sufficient number of cyclic chargings and dischargings, thus achieving cyclic steady state. In any 4 h charge process, the entering hot HTF has a constant temperature of 663.15 K, and in any discharge the exiting cold HTF has a constant temperature of 583.15 K. The inlet flow velocity of the fluid is always set as $V_{inlet} = 1.36 \times 10^{-3}\text{m/s}$.

As shown in Fig. 4.28, the fluid temperatures in the 4 h discharge process between the results of CFD and the 1D transient model agree very well. The maximum difference between them is 0.06% for the plate case, 0.05% for the cylinder case, and 0.01% for the tube case. This comparison clearly shows that the 1D transient model, which predicted fluid temperature at the exit of the discharge process during the time period, is sufficiently accurate. Engineers can use this method for the analysis of thermal energy storage systems without using comprehensive CFD computations.

To further compare the results from the 1D transient modeling with the results from CFD computation, Fig. 4.29 shows the distribution of temperature of the HTF at locations in the storage tank along the flow direction after 4 h of heat discharge. The heat discharge results are from a typical process after many cyclic charges and discharges so that a cyclic steady state has been achieved. It is understandable that after discharge, the temperature of fluid at the lower part of the tank is low. The temperature distribution from the 1D transient model and from the CFD computation agree very well, which further verifies the accuracy of the 1D transient modeling.

In conclusion, it is evident that the 1D transient model with the corrected heat transfer coefficient is robust and highly accurate for analysis of the heat transfer and energy storage behavior for the three studied typical fluid-solid structural combinations.

Because the 1D transient model significantly simplifies analysis, and especially the computational time, it is expected to be one of the most convenient and accurate tools available for industrial engineers to analyze the behavior of thermal energy storage systems of various fluid-solid configurations.

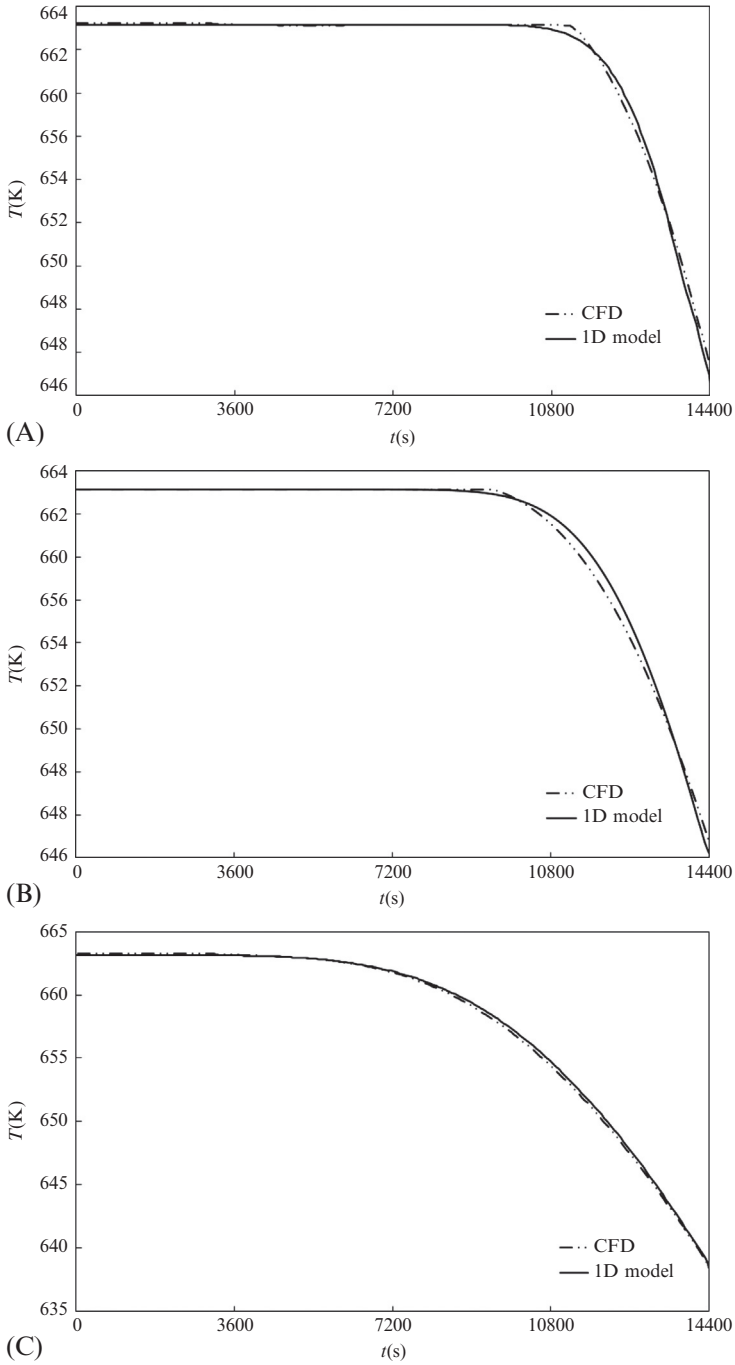


Fig. 4.28 Comparisons of results from CFD and one-dimensional model for the time variation of exit temperature of HTF. (A) Plate case; (B) cylinder case; (C) tube case.

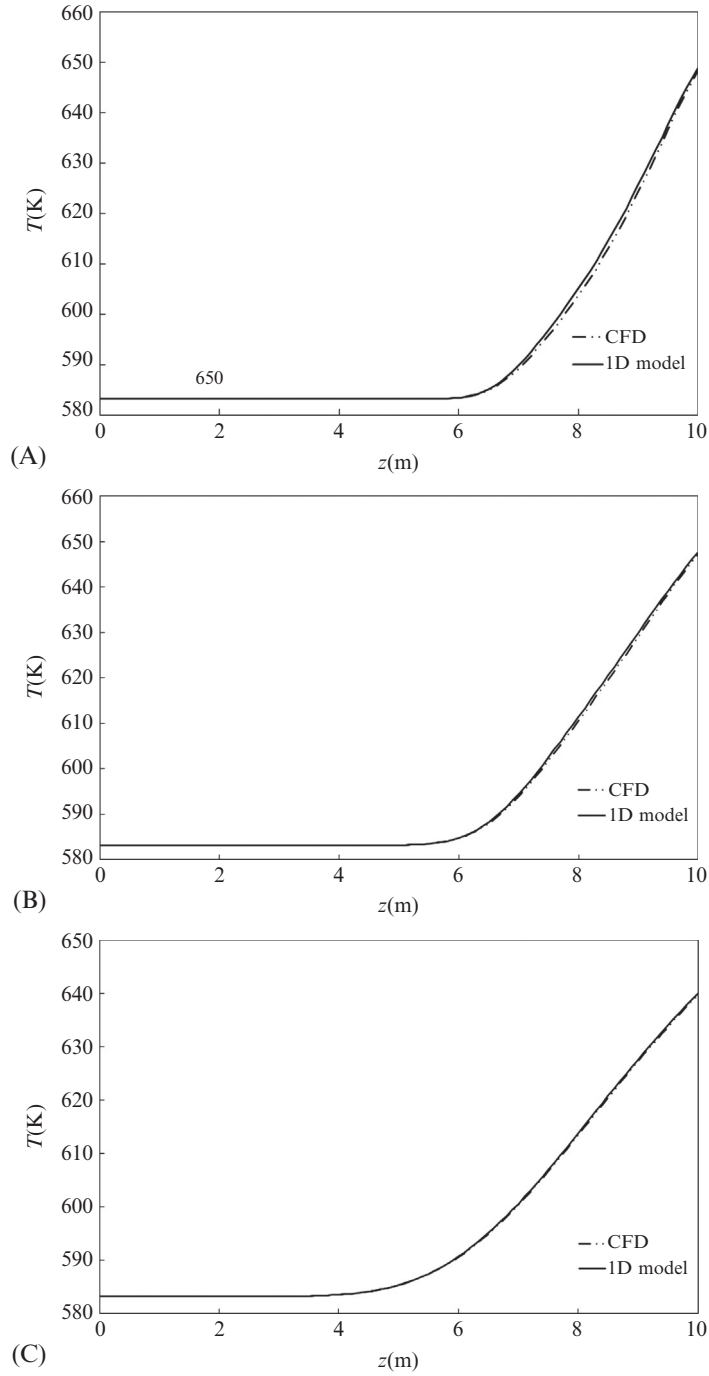


Fig. 4.29 Comparisons of results from CFD and one-dimensional model for the temperature distribution of HTF. (A) Plate case; (B) cylinder case; (C) tube case.

Finally, it is worth noting that, typically, the Schumann equations (1D transient model) for thermal energy storage ignore the axial heat conduction in both the solid and fluid, which is sufficiently accurate for regular dual-material thermal storage systems. However, if the solid and the liquid are highly conductive metals and liquid metals, respectively, which is rarely the case, the validity of the model has to be further examined.

4.4.2 Latent-Heat Thermal Storage

The 1D latent heat thermal model is developed in Section 4.3.1 and its numerical solution using the method of characteristics is presented in Section 4.3.2. We validate this numerical 1D model by comparing the current simulation results to experimental data by Nallusamy et al. [30]. In their experimental work, they used encapsulated spherical capsules of paraffin with melting temperature at 60°C as the PCM, and the HTF is water. The inlet fluid temperature was maintained at 70°C and the mass flow rate was fixed at 2 L/min. Using their experimental conditions and properties, the important parameters were estimated to be:

$$\begin{aligned} &H_{CR} = 1.008;\ \ \tau_r = 1.0269;\ \ \theta_{f_{inlet}} = 1; \\ &\theta_{f_o} = \theta_{s_o} = 0;\ \ \theta_{s\_melt} = 0.7368;\ \ \theta_{s\_ref} = 0; \\ &\eta_{s\_melt} = 1;\ \ \frac{C_{s\_s}}{C_{s\_\ell}} = 1.1268;\ \ Stf = 0.1143; \\ &\Delta t^* = \Delta z^+ = 0.001; \end{aligned} \tag{4.95}$$

Figs. 4.30 and 4.31 respectively show the temperatures of the fluid and the PCM at the middle of the tank as a function of time during a heat charging process. The results from simulation and experimental tests are in satisfactory agreement, which validates the correctness of the modeling and the computer code.

It is worth noting that the current model does not consider any heat loss from the tank due to the assumption of perfect thermal insulation. This will cause some discrepancy between the simulation and test results. Studies of the influence of heat loss on the temperatures in the thermal storage tanks have been reported by Modi and Pérez-Segarra [46]. The heat loss at the tank surface needs a certain length of time to penetrate and influence the temperature at the center of the tank. Nevertheless, the currently proposed 1D model has the capacity to incorporate the heat loss by adding a heat loss term on the right-hand side of Eq. (4.62), with little to no sacrificing of computational efficiency.

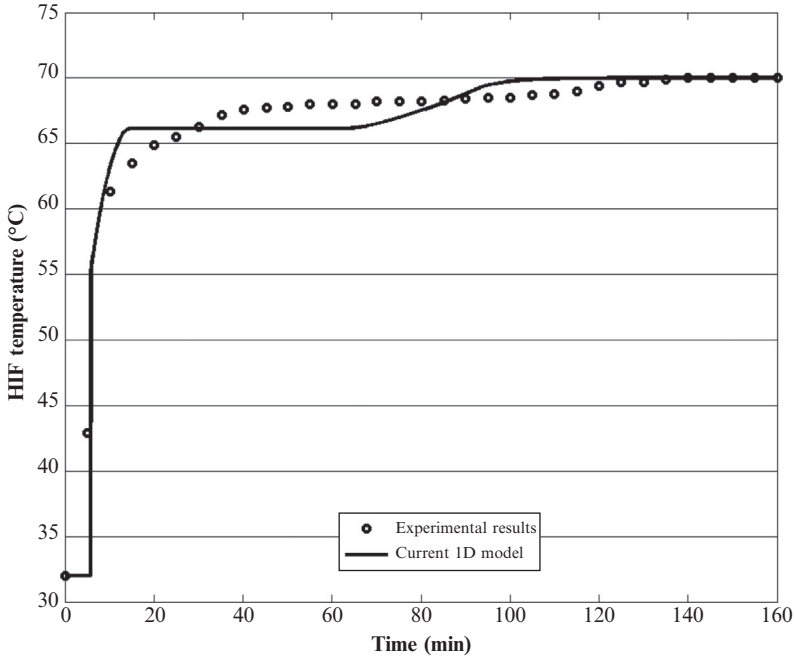


Fig. 4.30 Comparison of fluid temperature located at the middle of the tank. *(Experimental data from Nallusamy N, Sampath S, Velraj R. Experimental investigation on a combined sensible and latent heat storage system integrated with constant/varying (solar) heat sources. Renew Energy 2007;32:1206–27).*

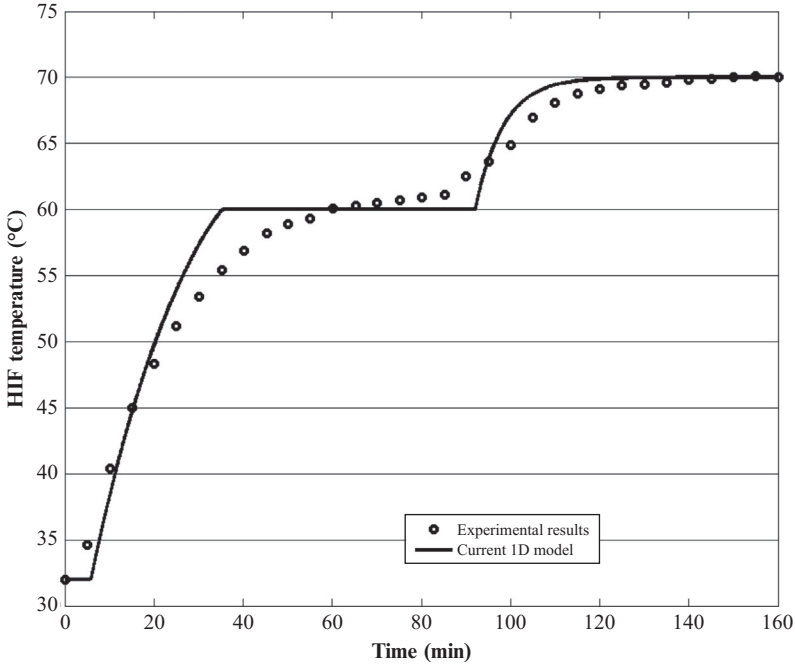


Fig. 4.31 Comparison of PCM temperature located at the middle of the tank. *(Experimental data from Nallusamy N, Sampath S, Velraj R. Experimental investigation on a combined sensible and latent heat storage system integrated with constant/varying (solar) heat sources. Renew Energy 2007;32:1206–27).*

4.5 MODELS DIRECTLY COMPUTE HEAT CONDUCTION IN SOLID AND HTF

The one-dimensional models in Sections 4.2 and 4.3 represent robust, fast, and accurate algorithms. As we have demonstrated, these models can be used for parametric studies and design purposes. When using the effective or corrected heat transfer coefficient in the one-dimensional models to overcome the limit of the lumped heat capacity method, one can deal with all the dual-media thermal storage configurations discussed in Chapter 2. Therefore, the one-dimensional models are cost-effective, convenient, and accurate for design of all types of thermal storage systems.

When detailed local temperature distributions and local heat transfer rates (between a solid material and HTF) are needed, more detailed modeling and algorithms may be employed. However, the computational efforts and time will increase significantly in the modeling if the heat conduction in fluid and solid is computed numerically.

The heat transfer phenomenon in thermal energy storage is truly a conjugate problem. The convection within the HTF and the conduction within the filler materials and container are ongoing simultaneously. These two heat transfer phenomena are coupled and affecting each other. Consequently, they should be solved simultaneously. In the one-dimensional models presented in Sections 4.2 and 4.3, we make use of the heat transfer coefficient and the effective heat transfer coefficient to couple them. This is a very effective way to simplify the conjugate heat transfer problem.

One logical extension of the one-dimensional model is to extend it to two- or three-dimensional formulations. Yang and Garimella [47] introduced an axisymmetric model with a volume-average continuum, momentum, and energy equations for the HTF and lumped capacitance energy equation for the solid filler materials. The coupling between the HTF and filler was treated using an empirical heat transfer coefficient. The equations can be numerically solved. In their work [47] the computations were conducted with the help of the commercial software FLUENT (FLUENT 6.1 Documentation). Flueckiger et al. [48] made use of this axisymmetric model and extended it to include a composite wall with the thermal insulation included in the computational domain. This allowed them to investigate the thermal ratcheting problem in a thermal storage tank.

Xu et al. [49] developed a transient axisymmetric model similar to that of Yang and Garimella. The fluid flow of the HTF is modeled as a porous medium, while the energy conservation is modeled by a convection

equation for the HTF and a heat diffusion equation for the solid fillers and the insulation layers and steel wall of the tank. The coupling between the HTF and solid filler materials was treated by use of a heat transfer coefficient, which should come from empirical correlations. At the interface between the HTF and the tank, no slip, same temperature, and energy balance were enforced. The finite volume method was implemented to solve the governing equations. The following discussion briefly introduces the previously mentioned modeling.

Considering the thermal storage system in Fig. 4.32, the packed-bed thermal storage region is viewed as a continuous, homogeneous, and isotropic porous medium. The fluid flow and heat transfer is assumed to be uniform and symmetrical about the axis, and therefore the governing equations for transport within the storage tank are two-dimensional. The flow of molten salt through the packed-bed region is mostly laminar and incompressible, and the properties of the solid fillers are constant. The fluid

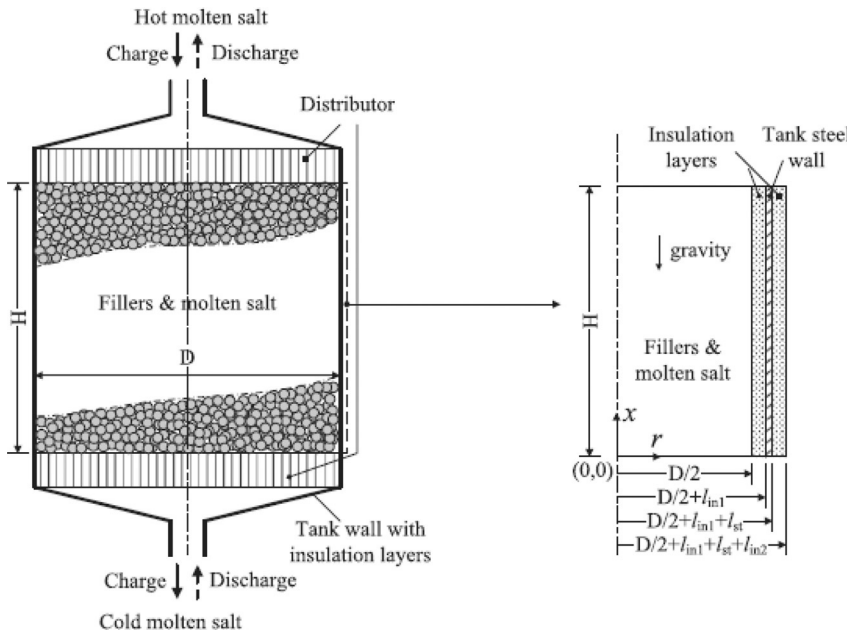


Fig. 4.32 Physical model of packed-bed thermal storage with molten salt HTF. *(Courtesy of Xu C, Wang Z, He Y, Li X, Bai F. Sensitivity analysis of the numerical study on the thermal performance of a packed-bed molten salt thermocline thermal storage system. Appl Energy 2012;92:65-75).*

flow and energy conservation governing equations are established in the following. First, the continuity equation is

$$\frac{\partial\left(\varepsilon\rho_f\right)}{\partial t}+\nabla\cdot\left(\rho_f\,\vec{u_f}\right)=0 \tag{4.96}$$

and the momentum is

$$\frac{\partial\left(\rho_f\vec{u}_f\right)}{\partial t}+\frac{1}{\varepsilon}\nabla\cdot\left(\rho_f\vec{u}_f\vec{u}_f\right)=-\varepsilon\nabla P+\varepsilon\nabla\cdot\left(\mu_f\nabla\vec{u}_f\right)+\varepsilon\rho_f \vec{g}-\varepsilon\left(\frac{\mu_f}{K}+\frac{C_F\rho_f}{\sqrt{K}}\left|\vec{u}_f\right|\right)\vec{u}_f \tag{4.97}$$

The energy equation for the HTF can be expressed as

$$\frac{\partial\left(\varepsilon\rho_f C_{Pf}T_f\right)}{\partial t}+\nabla\cdot\left(\rho_f C_{Pf}\vec{u}_f T_f\right)=\nabla\cdot\left(k_{\mathit{eff}}\nabla T_f\right)+h\left(T_s-T_f\right) \tag{4.98}$$

Here on the right-hand side, a heat source/sink term, $h(T_s-T_f)$, is introduced to model the heat transfer between the HTF and the solid filler materials. Here h is the volumetric interstitial heat transfer coefficient, which can be found in Ref. [49].

The energy equation for the heat transfer in the solid filler materials is governed by the diffusion equation with a source term:

$$\frac{\partial((1-\varepsilon)\rho_s C_{Ps}T_s)}{\partial t}=\nabla\cdot\left(k_{s,\,\mathit{eff}}\nabla T_s\right)-h\left(T_s-T_f\right) \tag{4.99}$$

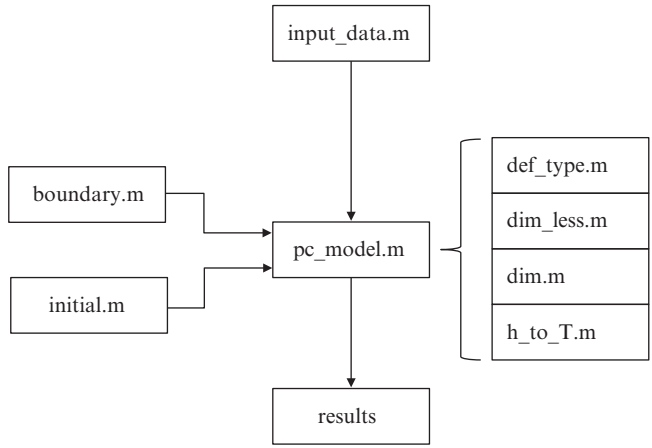
To consider the heat transfer between the packed bed and the wall of the storage tank, the governing equation is

$$\rho_w C_{Pw}\frac{\partial(T_w)}{\partial t}=\nabla\cdot\left(k_w\nabla T_w\right) \tag{4.100}$$

Appropriate initial and boundary conditions are needed to solve this set of governing equations. Typically, nonslip and nonpenetration conditions are enforced on the solid boundary. Since sink and source terms are introduced in the energy equation, there is no need of coupling conditions between the HTF and filler material. Along the interface between the HTF and tank wall, temperature continuity and conservation of energy are imposed.

To solve this set of equations, there are a number of possible numerical methods, such as finite volume, finite element, and spectral methods. For convenience, commercial numerical tools (such as ANSYS FLUENT) can be used to solve the governing equations.

APPENDIX STRUCTURE OF THE COMPUTER CODE FOR DUAL-MEDIA THERMAL STORAGE ANALYSIS



pc_model.m The main code that provides the solution to the governing equations using the method of characteristics, and also conducts postprocess for temperature equilibrium when settled down after each charging/discharging cycle in case multiple cycles of charging/discharging are computed. Results of the computation are also output from the main code.

input_data.m Provides all the required material properties, HTF mass flow rate, charging/discharging time length, size of storage tank, correlation/equation for effective heat transfer coefficient, time step, and mesh size.

boundary.m HTF inlet boundary condition.

initial.m Provides initial temperature of HTF, and initial enthalpy of storage medium (solid or PCM material at solid or liquid state).

def_type.m Definition of melting process, type = 1 is for solid part, type = 2 for melting part, type = 3 for liquid part (for sensible materials, the value of type is always set to be 1).

dim.m Converts dimensionless parameters back to dimensional parameters.

dim_less.m Converts dimensional parameters to dimensionless parameters.

h_to_T.m Equation of state to provide the conversion from enthalpy to temperature for storage media.

Boundary

```
%Phase Change Thermal Energy Storage Model
%Boundary Condition File
%Standard SI Units
```

```
function T_f=boundary(k,t,t_steps,T_f)

for i=1:t_steps+1
     %Boundary Condition for Fluid Temperature

     if rem(k,2)==0

         T_f(i,1)=(310+273.15);

     else

         T_f(i,1)=(390+273.15);

     end

end

end
```

def_type

```
%Phase Change Thermal Energy Storage Model
%Define type for each node

%Standard SI Units
function ans=def_type(h_r,h_melt,stf,k)

%Define type solid
if rem(k,2)==0
   if h_r<=h_melt
      ans=1;

%Define type as melting region
   elseif h_r>h_melt && h_r<=(h_melt+(1/stf))
       ans=2;

%Define type as liquid
   else
       ans=3;
   end
else
     if h_r<=h_melt
        ans=1;

%Define type as melting region
   elseif h_r>h_melt && h_r<(h_melt+(1/stf))
       ans=2;

%Define type as liquid
   else
       ans=3;
     end
end
end
```

dim

```
%Phase Change Thermal Energy Storage Model
%Dimensionalizing File

%Standard SI Units
function out=dim(var,num_ref,den_h,den_l,extra)

out=(var*extra*(den_h-den_l))+num_ref;

end
```

dim_less

```
%Phase Change Thermal Energy Storage Model
%Non-Dimensionalizing File

%Standard SI Units
function [out]=dim_less(var,num_ref,den_h,den_l,extra)

out=(var-num_ref)/(extra*(den_h-den_l));

end
```

h_to_T

```
%Phase Change Thermal Energy Storage Model
%Filler Enthalpy to Temperature Conversion File

%DIMENSIONLESS UNITS!!!!
function ans=h_to_T(h_r,h_r_melt,stf,T_r_melt,T_r_ref,cp_r_s,cp_r_l)

%solid
if h_r<h_r_melt
    ans=h_r*T_r_melt+T_r_ref;

%melting region
elseif h_r>=h_r_melt && h_r<(h_r_melt+(1.0/stf))
     ans=T_r_melt;

%liquid
else %h_r>(h_r_melt+(1/stf))
     ans=((h_r-h_r_melt-1.0/stf)*T_r_melt*cp_r_s/cp_r_l)+T_r_melt;
end

end
```

initial

```
%Phase Change Thermal Energy Storage Model
%Initial Condition File

%Standard SI Units
function [T_f,h_r]=initial(z,z_steps,T_f,h_r,h_r_H,h_r_L,h_r_melt)
```

```
for i=1:z_steps+1
     %Initial Condition for Fluid Temperature
     T_f(1,i)=(26.85+273.15);

     %Initial Condition for filler enthalpy
     h_r(1,i)=h_r_L;

end

end
```

input_data

```
%Phase Change Thermal Energy Storage Model
%Input File

%Standard SI Units
function
[z_o,z_f,del_z,t_o,t_f,del_t,H,A_f,A_r,eps,U,htc,Sr,T_H,T_L,T_r_ref,T_r_melt,
h_r_ref,h_r_melt,h_r_H,h_r_L,L,cp_r_s,cp_r_l,cp_f,rho_f,rho_r]=input_data(k)

%Domain and Step Sizes (DIMENSIONLESS!!!!)
z_o=0;
z_f=1.0;
del_z=1/50;

t_o=0;
if rem(k,2)==0
     t_f=0.0;
else
     t_f=19403.3;
end
del_t=1/50;

%Tank Properties
R=0.1015;                                   %Tank Radius (m)
H=0.6;                                      %Tank Height (m)
A_tank=pi*R*R;                              %Tank Cross Sectional Area (m^2)

eps=0.0407396;                              %Porosity

A_f=eps*A_tank;                             %Effective Fluid Cross Sectional
Area (m^2)
A_r=(1-eps)*A_tank;                         %Effective Filler Cross
Sectional Area (m^2)

%Fluid and Material Properties
m_dot=0.00504854;                           %Inlet mass flow rate (kg/s)

%r=0.02;                                    %Filler Radius (m)
%f_s=2.04;                                  %Surface Factor
Sr=pi*0.0094*19;

T_H=(92+273.15);                            %High Temperature (K)
```

```
T_L=(26.85+273.15);                              %Low Temperature  (K)
T_r_ref=T_L;                                     %Reference Temperature
T_r_melt=(500+273.15);                           %Filler Melting Temperature (K)

cp_r_s=994.893;                                  %Filler Solid Specific Heat
(J/kg*K)
cp_r_l=1000;                                     %Filler Liquid Specific Heat
(J/kg*K)

rho_r=2071.43;                                   %Filler Density (kg/m^3)

h_r_ref=40000;                                   %Filler Reference Enthalpy
(J/kg);
h_r_L=h_r_ref;                                   %Filler Enthalpy at T_L
(J/kg);
h_r_melt=cp_r_s*(T_r_melt-T_r_ref)+h_r_ref;      %Filler Melting Enthalpy (J/kg);
L=119000;                                        %Filler Latent Heat (J/kg);
h_r_H=cp_r_s*(T_H-T_L)+h_r_ref;                  %Filler Enthalpy at T_H for
sensible case  (J/kg);

cp_f=1005;                                       %Liquid Specific Heat (J/kg*K)

rho_f=1.2585;                                    %Fluid Density (kg/m^3)

U=m_dot/rho_f/A_f;                               %Bulk Fluid Velocity (m/s)

kf=0.026;                                        %Fluid conductivity (W/m*K)

ks=0.493;                                        %Solid conductivity (W/m.K)

%nuf=1.807*10^(-5);                              %Fluid Kinematic Viscosity
(m^2/s)

%Pr=nuf*rho_f*cp_f/kf;                           %Prandtl number for fluid

%Re=U*2*R/2/nuf;                                 %Reynolds Number for fluid

Nuselt=12.09;                                    % Nusselt number

Dtube=0.0094;

htt=Nuselt*kf/(Dtube);                           %Convective Heat Transfer Coef-
ficient (W/(m^2-K))

rii=Dtube/2;

roo=sqrt(R^2/19.0);

B=(rii^3*(4*roo^2-rii^2)+rii*roo^4*(4*log(roo/rii)-3))/(4*(roo^2-rii^2)^2);

htc=1/(1/htt+B/ks);                               %Effective heat transfer coef-
ficient

end
```

pc_model

```
%Phase Change Thermal Energy Storage 1D Model
%Main Function

clc
clear all

for k=1:100% k is used to define the index of cyclic operation

% Enter 'start_type=0' for first run, 'start_type=1' to compute cycles with
% previous ending distributions of temperatures
if k==1
   start_type=0;
end

if k>1
   start_type=1;
end

% Call the subfunction of input_data.m to provide all the required
% parameters for the main function
[z_o,z_f,del_z,t_o,t_f,del_t,H,A_f,A_r,eps,U,htc,Sr,T_H,T_L,T_r_ref,...
T_r_melt,h_r_ref,h_r_melt,h_r_H,h_r_L,L,cp_r_s,cp_r_l,cp_f,rho_f,rho_r]...
=input_data(k);

% Define the dimensionless parameter of tau_r
tau_r=U/H*rho_f*cp_f*A_f/htc/Sr;

% Define the dimensionless paramter of H_CR
H_CR=rho_f*cp_f*eps/rho_r/cp_r_s/(1-eps);

t_steps=round((t_f-t_o)/del_t);    % Total number of time steps
t=(t_o:del_t:t_f);                 % Define t matrix


z_steps=round((z_f-z_o)/del_z);    % Total number of spatial grids
z=(z_o:del_z:z_f);                 % Define z matrix


% Initialization of HTF temp., filler temp. and enthalpy
T_f=zeros(t_steps+1,z_steps+1);
T_r=zeros(t_steps+1,z_steps+1);
h_r=zeros(t_steps+1,z_steps+1);


% Initialization of the filler melting process matrix
type=zeros(t_steps+1,z_steps+1);


% Interfaces Initializations
s=0;                               % Total number of solid interfaces
```

```
l=0;                                % Total number of liquid interfaces
T_f_int=0;                          % Fluid temperature at solid interface
T_f_liq=0;                          % Fluid temperature at liquid interface
h_r_int=0;                          % Filler enthalpy at solid interface
h_r_liq=0;                          % Filler enthalpy at liquid interface
T_r_int=0;                          % Filler temperature at solid interface
T_r_liq=0;                          % Filler temperature at liquid interface

% Index matrix for cases that HTF is traveling too fast
toofast=0;

% Total number of computational errors
error=0;

% Initial value of partial fraction for solid-liquid interface in each grid
beta_new=0;

% Initial value of index matrix for cases that the interface is traveling
% with the same speed with the HTF
test_count=0;

% Initial Condition
if start_type==0 % Initial Run
    cycle_type=1;                   % Equilibrium

    % call the function initial.m
    [T_f,h_r]=initial(z,z_steps,T_f,h_r,h_r_H,h_r_L,h_r_melt);
    s_ct=0;                         % Initialization of the number of solid
interfaces
    l_ct=0;                         % Initialization of the number of liquid
interface

% Cycle computation with results from previous run as initial condition
elseif start_type==1

    % Load the saved data from the previous cycle
    load(['cycle_',num2str(k-1),'.mat'])

    if cycle_type==0
        cycle_type=1;               % Equilibrium
```

```
    else
        cycle_type=1;              % Equilibrium
    end

    T_f(1,:)=T_f_last;             % Initial value of HTF temperature at new
time step
    h_r(1,:)=h_r_last;             % Initial value of filler enthalpy at new
time step
    beta_new=beta_last; % Initial value of Beta at new time step

% Initial value of HTF temp. on the solid and liquid interface at new time
% step
    T_f_int=T_f_int_last;
    T_f_liq=T_f_liq_last;

% Initial value of Filler enthalpy on the solid and liquid interface at new
% time step
    h_r_int=h_r_int_last;
    h_r_liq=h_r_liq_last;

% Initial value of Filler temp. on the solid and liquid interface at new
% time step
    T_r_int=T_r_int_last;
    T_r_liq=T_r_liq_last;

% Initial value of the number of solid and liquid interfaces at new time
% step
    s_ct=s_ct_last;
    l_ct=l_ct_last;
end

% Call the function boundary.m
T_f=boundary(k,t,t_steps,T_f);

% Call the function dim_less.m to calculate the dimensionless HTF
% temperature and the enthalpy of storage medium
for i=1:z_steps+1
    T_f(1,i)=dim_less(T_f(1,i),T_L,T_H,T_L,1);
    h_r(1,i)=dim_less(h_r(1,i),h_r_ref,T_r_melt,T_L,cp_r_s);
end
```

```
%Boundary Condition for Fluid Temperature
for i=2:t_steps+1
    T_f(i,1)=dim_less(T_f(i,1),T_L,T_H,T_L,1);
end

% Calculate the dimensionless melting enthalpy
h_r_melt=dim_less(h_r_melt,h_r_ref,T_r_melt,T_L,cp_r_s);

% Calculate the Stefan number
stf=cp_r_s*(T_r_melt-T_L)/L;

% Calculate the dimensionless reference temperature
T_r_ref=dim_less(T_r_ref,T_L,T_H,T_L,1);

% Calculate the dimensionless melting temperature
T_r_melt=dim_less(T_r_melt,T_L,T_H,T_L,1);

% Create T_r matrix corresponding to h_r based on the equation of state
% definedfunction h_to_T.m
for i=1:z_steps+1
    T_r(1,i)=h_to_T(h_r(1,i),h_r_melt,stf,T_r_melt,T_r_ref,cp_r_s,cp_r_l);
end

% Send the initial value of the number of solid and liquid interface to
% s matrix in the current time step
s=s_ct;
l=l_ct;

%Time iterations for the boundary conditions
for j=1:t_steps
    %Starting from below melting (sensible part)
    if h_r(j,1)<h_r_melt
        type(j,1)=1;
        h_r(j+1,1)=((2*h_r(j,1))-(H_CR*del_t/tau_r)*(h_r(j,1)+(2*T_r_ref...
                    /T_r_melt)-((T_f(j+1,1)+T_f(j,1))/T_r_melt)))...
                    /(2+(H_CR*del_t/tau_r));
        T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,cp_r_s,
cp_r_l);
```

```
        %Check if reached melting (charging process)
        if h_r(j+1,1)>h_r_melt
            disp('new solidus interface 1')
            j
            s=s+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f(j,1));
            b=del_t*H_CR/tau_r/T_r_melt/2*(2*T_f(j,1)-T_r_melt-T_r(j,1));
            c=-1*(h_r_melt-h_r(j,1));

            % Calculate the partial fraction of solid interface at the
            % boundary time iteration
            if a==0
                alpha(s)=-c/b;
            else
                alpha(s)=(-b+sqrt(b^2-4*a*c))/2/a;
            end
            j_melt=j;
            T_f_int(s)=T_f(j,1)+alpha(s)*(T_f(j+1,1)-T_f(j+1,1));
            h_r(j+1,1)=h_r_melt-(1-alpha(s))*del_t*H_CR/tau_r*(1-...
                       ((T_f(j+1,1)+T_f_int(s))/2/T_r_melt));
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);

        end

        %Check if breached into liquid region (charging process)
        if h_r(j+1,1)>(h_r_melt+(1/stf))
            disp('new liquidus interface 1')
            l=l+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f_int(s));
            b=del_t*H_CR/tau_r/T_r_melt*(T_f_int(s)-T_r_melt);
            c=-1/stf;

            % Calculate the partial fraction of liquid interface at the
            % boundary time iteration
            if a==0
                gamma(l)=-c/b;
            else
                gamma(l)=(-b+sqrt(b^2-4*a*c))/2/a;
            end
```

```
            j_liq=j;
            T_f_liq(l)=T_f_int(s)+gamma(l)*(T_f(j+1,1)-T_f_int(s));
            h_r(j+1,1)=(2*h_r_melt+2/stf-(H_CR*del_t/tau_r*(1-gamma(l)...
                       -alpha(s)))*((2-(h_r_melt+(1/stf))*(cp_r_s/cp_r_l))...
                       -((T_f(j+1,1)+T_f_liq(l))/T_r_melt)))/(2+(H_CR*...
                       del_t/tau_r*(1-gamma(l)-alpha(s))*cp_r_s/cp_r_l));
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);
        end

    %Start in Melting Region
    elseif h_r(j,1)>=h_r_melt && h_r(j,1)<(h_r_melt+(1/stf))
        type(j,1)=2;
        h_r(j+1,1)=h_r(j,1)-del_t*H_CR/tau_r*(1-((T_f(j+1,1)+T_f(j,1))...
                   /2/T_r_melt));
        T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,cp_r_s,
cp_r_l);

        %Check whether breach into liquid region (charging process)
        if h_r(j+1,1)>(h_r_melt+(1/stf))
            disp('new liquidus interface 2')
            j
            l=l+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f(j,1));
            b=del_t*H_CR/tau_r/T_r_melt*(T_f(j,1)-T_r_melt);
            c=-1*(h_r_melt+1/stf-h_r(j,1));

            % Calculate the partial fraction of liquid interface at the
            % boundary time iteration
            if a==0
                gamma(l)=-c/b;
            else
                gamma(l)=(-b+sqrt(b^2-4*a*c))/2/a;
            end

            T_f_liq(l)=T_f(j,1)+gamma(l)*(T_f(j+1,1)-T_f(j,1));
            h_r(j+1,1)=(2*h_r_melt+2/stf-(H_CR*del_t/tau_r*(1-gamma(l)))...
                       *((2-(h_r_melt+(1/stf))*(cp_r_s/cp_r_l))-((T_f(j
+1,1)...
                       +T_f_liq(l))/T_r_melt)))/(2+(H_CR*del_t/tau_r*...
                       (1-gamma(l))*cp_r_s/cp_r_l));
```

```
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);
        end

        %Check if back to fully solidified (discharging process)
        if h_r(j+1,1)<h_r_melt
            disp('new solidus interface 2')
            s=s+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f(j,1));
            b=del_t*H_CR/tau_r/T_r_melt/2*(2*T_f(j,1)-T_r_melt-T_r(j,1));
            c=-1*(h_r_melt-h_r(j,1));

            % Calculate the partial fraction of solid interface at the
            % boundary time iteration
            if a==0
                alpha(s)=-c/b;
            else
                alpha(s)=(-b+sqrt(b^2-4*a*c))/2/a;
            end

            T_f_int(s)=T_f(j,1)+alpha(s)*(T_f(j+1,1)-T_f(j,1));
            h_r(j+1,1)=((2*h_r_melt)-((1-alpha(s))*H_CR*del_t/tau_r...
                        /T_r_melt)*(T_r_ref+T_r_melt-T_f(j+1,1)-T_f_int
(s)))...
                        /(2+((1-alpha(s))*H_CR*del_t/tau_r));
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);
        end

    %Start in Liquid Region h_r(j,1)>(h_r_melt+(1/stf))
    else
        type(j,1)=3;
        h_r(j+1,1)=(2.0*h_r(j,1)-(H_CR*del_t/tau_r)*...
                    ((1.0-(h_r_melt+(1.0/stf))*(cp_r_s/cp_r_l))+((T_r(j,1)...
                    -T_f(j+1,1)-T_f(j,1))/T_r_melt)))/(2.0+(H_CR*del_t/...
                    tau_r*cp_r_s/cp_r_l));
        T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,cp_r_s,
cp_r_l);

        %Check if return to melting region (discharging process)
        if h_r(j+1,1)<(h_r_melt+(1.0/stf))
```

```
            disp('new liquidus interface 3')
            l=l+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f(j,1));
            b=del_t*H_CR/tau_r/T_r_melt/2*(2*T_f(j,1)-T_r_melt-T_r(j,1));
            c=-(h_r_melt+1/stf-h_r(j,1));

            % Calculate the partial fraction of liquid interface at the
            % boundary time iteration
            if a==0
                gamma(l)=-c/b;
            else
                gamma(l)=(-b+sqrt(b^2-4*a*c))/2/a;
            end
            T_f_liq(l)=T_f(j,1)+gamma(l)*(T_f(j+1,1)-T_f(j,1));
            h_r(j+1,1)=h_r_melt+1/stf-(H_CR*del_t/tau_r*(1-gamma(l)))*...
                       (1.0-((T_f(j+1,1)+T_f_liq(l))/2/T_r_melt));
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);
        end

        %Check if fully solidified (discharging process)
        if h_r(j+1,1)<h_r_melt
            disp('new solidus interface 3')
            s=s+1;
            a=del_t*H_CR/tau_r/T_r_melt/2*(T_f(j+1,1)-T_f_liq);
            b=del_t*H_CR/tau_r/T_r_melt/2*(T_f_liq-T_r_melt);
            c=1/stf;

            % Calculate the partial fraction of solid interface at the
            % boundary time iteration
            if a==0
                alpha(s)=-c/b;
            else
                alpha(s)=(-b+sqrt(b^2-4*a*c))/2/a;
            end
            T_f_int(s)=T_f_liq+alpha(s)*(T_f(j+1,1)-T_f_liq(l));
            h_r(j+1,1)=(2*h_r_melt-(H_CR*del_t/tau_r*(1-gamma(l)-alpha(s)))
*...
                       (1+((T_r_ref+T_r_melt-T_f(j+1,1)-T_f_int(s))/
T_r_melt)))/...
                       (2+(H_CR*del_t/tau_r*(1-gamma(l)-alpha(s))));
```

```
            T_r(j+1,1)=h_to_T(h_r(j+1,1),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);
        end

    end

end

%Define type for remainder of points known
type(t_steps+1,1)=def_type(h_r(t_steps+1,1),h_r_melt,stf,k);
for i=2:z_steps+1
    type(1,i)=def_type(h_r(1,i),h_r_melt,stf,k);
end

% Define the position matrix of solid and liquid interface
pos_int=zeros(s+l,t_steps+1);
track=0;

%'Spatial Sweep' Time Iteration
for j=1:t_steps

    %Reassign Necessary values
    s=s_ct;
    l=l_ct;
    beta_old=beta_new;
    T_f_int_old=T_f_int;
    T_f_liq_old=T_f_liq;
    h_r_int_old=h_r_int;
    h_r_liq_old=h_r_liq;
    T_r_int_old=T_r_int;
    T_r_liq_old=T_r_liq;

    % Display the current time step
    j

    for i=1:z_steps
        if type(j,i)==1
            if type(j,i+1)==1
                % Display('in 1/1'), for solid sensible part;
```

```
            LHS=[(1+del_t/tau_r/2), (-del_t*T_r_melt/2/tau_r);...
                (-del_t*H_CR/2/tau_r/T_r_melt),(1+del_t*H_CR/2/tau_r)];
            RHS=[((1-del_t/tau_r/2)*T_f(j,i)+(del_t/2/tau_r*T_r(j,i))+...
                (del_t/2/tau_r*T_r_ref));((1-del_t*H_CR/tau_r/2)*...
                h_r(j,i+1)+(del_t*H_CR/2/tau_r/T_r_melt*T_f(j,i+1))-...
                (del_t*H_CR/tau_r/T_r_melt*T_r_ref))];
            matrix=LHS\RHS;
            T_f(j+1,i+1)=matrix(1);
            h_r(j+1,i+1)=matrix(2);
            T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

        elseif type(j,i+1)==2
            disp('in 1/2'); % Solid interface created
            a=(del_t/tau_r*(T_r(j+1,i)-T_r(j,i)+T_f(j,i)-T_f(j+1,i)))+...
              (del_t/tau_r/(1-beta_old(s+l))*(T_r(j,i+1)+...
              T_f_int_old(s)-T_r_int_old(s)-T_f(j,i+1)))+(2*T_r_melt...
              /H_CR/(1-beta_old(s+l))*(h_r_int_old(s)-h_r(j,i+1)));
            b=(2/(1-beta_old(s+l))*(T_r(j,i+1)-T_r_int_old(s)+...
              T_f_int_old(s)-T_f(j,i+1)))+4*tau_r*T_r_melt/del_t...
              /H_CR/(1-beta_old(s+l))*(h_r_int_old(s)-h_r(j,i+1))...
              +2*(T_f(j+1,i)-T_f(j,i))+del_t/tau_r*(T_r(j,i+1)...
              +T_f(j+1,i)-T_f(j,i+1)-T_r(j+1,i))+del_t/tau_r...
              /(1-beta_old(s+l))*(T_r_int_old(s)+T_f(j,i+1)-T_r(j,i+1)...
              -T_f_int_old(s))+2*T_r_melt/H_CR*(h_r_melt-h_r(j,i+1)-...
              ((h_r_int_old(s)-h_r(j,i+1))/(1-beta_old(s+l))));
            c=4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1)-...
              ((h_r_int_old(s)-h_r(j,i+1))/(1-beta_old(s+l))))+2*
(T_r_melt...
              +T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i))+2/(1-beta_old(s+l)))...
              *(T_r_int_old(s)+T_f(j,i+1)-T_r(j,i+1)-T_f_int_old(s));

             % Calculate the partial fraction beta_new to track the solid
             % interface
              if a==0
                    disp('a=0')
                    beta_new(s+l)= -c/b;
              else
                    poly_ans=roots([a b c]);
                    beta_new(s+l)= poly_ans(2);
              end
```

```
                % Calculate the temp. of HTF and filler, enthalpy of filler
                % at the solid interface
                T_f_int(s)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)-((1-beta_new(s
+l))...
                          /(1-beta_old(s+l)))*(T_r(j,i+1)-T_r_int_old(s)...
                          +T_f_int_old(s)-T_f(j,i+1))+2*tau_r*T_r_melt...
                          /H_CR/del_t*(h_r_melt-h_r(j,i+1)-((1-beta_new(s
+l))...
                          /(1-beta_old(s+l)))*(h_r_int_old(s)-h_r(j,i+1)));
                h_r_int(s)=h_r_melt;
                T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);

                % Track and update the solid interface position
                pos_int(s+l,j)=del_z*(i-1)+beta_old(s+l)*del_z;
                if j==t_steps
                    track=track+1;
                    last_i(track)=i;
                end

                % Check if new solid liquid interface cross into the
                % next spatial grid
                if beta_new(s+l)>1.0

                    % Check whether in last spatial step or not
                    if i==z_steps
                       disp('crossed spacial boundary')
                       a= del_t/2/tau_r/beta_old(s+l)*(T_f(j,i)-T_f_int_old
(s))+...
                          del_t/2/tau_r/beta_old(s+l)*(T_r_melt-T_r(j,i));
                       b= del_t/2/tau_r*(T_r(j,i+1)-T_f(j,i+1)+T_f(j,i)-...
                          T_r(j,i))+del_t/2/tau_r/beta_old(s+l)*...
                          (T_f_int_old(s)-T_f(j,i)+T_r(j,i)-T_r_melt)+...
                          (T_f_int_old(s)-T_f(j,i))/beta_old(s+l);
                       c= T_r_melt/H_CR*(h_r_melt-h_r(j,i+1))+T_r_melt...
                          +T_r(j,i+1)-T_f(j,i+1)-T_f(j,i)+(T_f(j,i)-...
                          T_f_int_old(s))/beta_old(s+l);
                       d= 2*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1));
```

```
                        disp('zeta calc')
                        poly_ans_z=roots([a b c d]);
                        for root_find=1:1:3
                            if (poly_ans_z(root_find)>0 && poly_ans_z
(root_find)<1.0)
                                eta=poly_ans_z(root_find);
                                beta_new(s+1)=-1;
                                T_f_int(s)=-1;
                                h_r_int(s)=-1;
                            end
                        end

                        % Calculate HTF temperature at the solid interface
                        T_f_eta=2*tau_r*T_r_melt/H_CR/del_t/eta*...
                                (h_r_melt-h_r(j,i+1)+eta*H_CR*del_t/2/tau_r...
                                /T_r_melt*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)));

                        LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt/tau_r/2;...
                             (-(1-eta)*del_t*H_CR/2/tau_r/T_r_melt), (1+(1-
eta)*H_CR*del_t/tau_r/2)];
                        RHS=[T_f(j,i)+del_t/2/tau_r*(T_r_ref-T_f(j,i)+T_r(j,
i));...
                             h_r_melt-(1-eta)*H_CR*del_t/2/tau_r/T_r_melt*
(T_r_ref+T_r_melt-T_f_eta)];
                        matrix=LHS\RHS;
                        T_f(j+1,i+1)=matrix(1);
                        h_r(j+1,i+1)=matrix(2);
                        T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                        % Calculate the position of solid interface
                        pos_int(s+1,j+1)=del_z*i;

                    else
                        a= 2*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f(j,i+1))+...
                           4*tau_r*T_r_melt/H_CR/del_t*(h_r(j,i+1)-h_r(j,i
+2))+...
                           2/beta_old(s+1)*(T_f(j,i)-T_f_int_old(s))+...
                           del_t/tau_r/beta_old(s+1)*(T_r(j,i)-T_r_int_old
(s))+...
```

```
                del_t/tau_r*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f
(j,i+1))+...
                2*T_r_melt/H_CR*(h_r(j,i+1)-h_r(j,i+2))+...
                del_t/tau_r/beta_old(s+1)*(T_f_int_old(s)-T_f
(j,i));
            b= 2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i))+...
                4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1))-
...
                del_t/tau_r*(T_r(j,i)-T_r(j,i+1)+T_f(j,i+1)-T_f(j,
i))+...
                2*T_r_melt/H_CR*(h_r_melt-h_r(j,i+1));

            % Check for singulatiry in calculation
            if (abs(a)<1.0e-10) & (abs(b)<1.0e-10)
                disp('coefficients are zero')
                beta_new(s+1)=0;
            else
                beta_new(s+1)=-b/a;
            end

            % Special Calculation for Beta=0
            if beta_new(s+1)<=0
                beta_new(s+1)=0;
                test_count=test_count+1;
                test(test_count,:)=[(T_f(j,i)+del_t/2/tau_r...
                    *(T_r_melt+T_r(j,i)-T_f(j,i)))/(1+del_t/2/
tau_r),...
                    (h_r_melt-h_r(j,i+1)+H_CR*del_t/2/tau_r/
T_r_melt...
                    *(T_r_melt+T_r(j,i+1)-T_f(j,i+1)))...
                    /(H_CR*del_t/2/tau_r/T_r_melt),s+1,j];

                % Calculate the temp. of HTF and filler, enthalpy
of filler
                % at the solid interface
                T_f_int(s)=test(test_count,1);
                T_f(j+1,i+1)=T_f_int(s);
                h_r(j+1,i+1)=h_r_melt;
                T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

            else
```

```
                        % Check to make sure HTF is not going faster
                        % than the HTF characteristics
                        if beta_new(s+l)>beta_old(s+l)
                            beta_new(s+l)=beta_old(s+l);
                            disp('going too fast');
                            LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt/tau_r/
2;...
                                (-del_t*H_CR/2/tau_r/T_r_melt),(1
+H_CR*del_t/tau_r/2)];
                            RHS=[T_f_int_old(s)+del_t/2/tau_r*(T_r_ref
+T_r_int_old(s)...
                                -T_f_int_old(s));h_r(j,i+1)+beta_new(s+l)*
(h_r(j,i+2)...
                                -h_r(j,i+1))-H_CR*del_t/2/tau_r/
T_r_melt*...
                                (T_r_ref+T_r(j,i+1)+beta_new(s+l)*(T_r(j,i
+2)...
                                -T_r(j,i+1))-T_f(j,i+1)-beta_new(s+l)*(T_f
(j,i+2)-T_f(j,i+1)))];
                            matrix=LHS\RHS;

                            % Calculate the temp. of HTF and filler,
enthalpy of filler
                            % at the solid interface
                            T_f_int(s)=matrix(1);
                            h_r_int(s)=matrix(2);
                            T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                            toofast(s+l)=toofast(s+l)+1;
                        else

                            % Calculate the temp. of HTF and filler,
enthalpy of filler
                            % at the solid interface
                            T_f_int(s)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)...
                                      +beta_new(s+l)*(T_r(j,i+2)-...
                                      T_r(j,i+1)-T_f(j,i+2)+...
                                      T_f(j,i+1))+2*tau_r*T_r_melt/...
                                      H_CR/del_t*(h_r_melt-h_r(j,i+1)...
                                      -beta_new(s+l)*(h_r(j,i+2)-h_r(j,i
+1)));
                            h_r_int(s)=h_r_melt;
                            T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end
```

```
                            % Calculate New Values
                            eta=(1-beta_old(s+l))/(beta_new(s+l)-beta_old(s+l)
+1);
                            T_f_eta=T_f_int_old(s)+eta*(T_f_int(s)-T_f_int_old
(s));
                            T_r_eta=T_r_int_old(s)+eta*(T_r_int(s)-T_r_int_old
(s));
                            h_r_eta=h_r_int_old(s)+eta*(h_r_int(s)-h_r_int_old
(s));

                            LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt/tau_r/
2;...
                                (-(1-eta)*del_t*H_CR/2/tau_r/T_r_melt), (1+(1-
eta)*H_CR*del_t/tau_r/2)];
                            RHS=[T_f(j,i)+del_t/2/tau_r*(T_r_ref-T_f(j,i)+T_r
(j,i));...
                                h_r_eta-(1-eta)*H_CR*del_t/2/tau_r/T_r_melt*
(T_r_ref+T_r_eta-T_f_eta)];
                            matrix=LHS\RHS;

                            % Update the temp. and enthalpy of HTF and filler in
                            % the next time step and the following grid (j+1,i
+1)
                            T_f(j+1,i+1)=matrix(1);
                            h_r(j+1,i+1)=matrix(2);
                            T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end

                        if j==t_steps
                            last_i(track)=i+1;
                        end
                    end

                else % For cases that beta_new < 1.0
                    % Calculate New Values
                    eta=beta_old(s+l)/(1-beta_new(s+l)+beta_old(s+l));
                    T_f_eta=T_f_int_old(s)+eta*(T_f_int(s)-T_f_int_old(s));
                    T_r_eta=T_r_int_old(s)+eta*(T_r_int(s)-T_r_int_old(s));
                    h_r_eta=h_r_int_old(s)+eta*(h_r_int(s)-h_r_int_old(s));
```

```
                    % Update the temp. and enthalpy of HTF and filler in
                    % the next time step and the following grid (j+1,i+1)
                    T_f(j+1,i+1)=(T_f_eta*(1-(1-eta)*del_t/2/tau_r)+(1-
eta)...
                                *del_t/2/tau_r*(T_r_melt+T_r_eta))/...
                                (1+(1-eta)*del_t/2/tau_r);
                    h_r(j+1,i+1)=h_r(j,i+1)-H_CR*del_t/2/tau_r/T_r_melt*...
                                (T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i
+1));
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

                end
                %s=s-1;

            else %type(j,i+1)==3
                disp('Heat Transfer is occuring too fast...try a smaller time
step to better resolve');
                error=error+1;
            end

        elseif type(j,i)==2 % Cases for discharging process
            if type(j,i+1)==1
                disp('in 2/1');
                a=(del_t/tau_r*(T_r(j+1,i)-T_r(j,i)+T_f(j,i)-T_f(j+1,i)))+...
                  (del_t/tau_r/(1-beta_old(s+1))*(T_r(j,i+1)+T_f_int_old
(s)...
                  -T_r_int_old(s)-T_f(j,i+1)))+(2*T_r_melt/H_CR/(1-beta_old(s
+1)))...
                  *(h_r_int_old(s)-h_r(j,i+1)));
                b=(2/(1-beta_old(s+1))*(T_r(j,i+1)-T_r_int_old(s)+T_f_int_old
(s)...
                  -T_f(j,i+1)))+4*tau_r*T_r_melt/del_t/H_CR/(1-beta_old(s
+1)))...
                  *(h_r_int_old(s)-h_r(j,i+1))+2*(T_f(j+1,i)-T_f(j,i)))...
                  +del_t/tau_r*(T_r(j,i+1)+T_f(j+1,i)-T_f(j,i+1)-T_r(j+1,i))
+...
                  del_t/tau_r/(1-beta_old(s+1))*(T_r_int_old(s)+T_f(j,i+1)...
                  -T_r(j,i+1)-T_f_int_old(s))+2*T_r_melt/H_CR*(h_r_melt...
                  -h_r(j,i+1)-((h_r_int_old(s)-h_r(j,i+1))/(1-beta_old(s
+1))));
                c=4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1)-...
                  ((h_r_int_old(s)-h_r(j,i+1))/(1-beta_old(s+1))))+...
```

```
                  2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i))+...
                  2/(1-beta_old(s+l))*(T_r_int_old(s)+T_f(j,i+1)-T_r(j,i+1)-
T_f_int_old(s));

               % Calculate the partial fraction beta_new to track the
               % solid interface for discharging process
                if a==0
                    disp('a=0')
                    beta_new(s+l)= -c/b;
                else
                    poly_ans=roots([a b c]);
                    beta_new(s+l)= poly_ans(2);
                end

                % Calculate the temp. of HTF and filler, enthalpy of filler
                % at the solid interface for discharging process
                T_f_int(s)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)-((1-beta_new(s
+l))...
                           /(1-beta_old(s+l)))*(T_r(j,i+1)-T_r_int_old(s)...
                           +T_f_int_old(s)-T_f(j,i+1))+2*tau_r*T_r_melt...
                           /H_CR/del_t*(h_r_melt-h_r(j,i+1)-((1-beta_new(s
+l))...
                           /(1-beta_old(s+l)))*(h_r_int_old(s)-h_r(j,i+1)));
                h_r_int(s)=h_r_melt;

                T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);

                % Track and update the position of solid interface
                pos_int(s+l,j)=del_z*(i-1)+beta_old(s+l)*del_z;
                if j==t_steps
                    track=track+1;
                    last_i(track)=i;
                end

                %Check if new Beta passes into the nex grid
                if beta_new(s+l)>1.0
                   %Recalculate Beta New
                    disp('in new z');
```

```
                    %Check whether in last spatial grid or not
                    if i==z_steps
                        disp('crossed spacial boundary')
                        a= del_t/2/tau_r/beta_old(s+l)*(T_f(j,i)-T_f_int_old
(s))+...
                            del_t/2/tau_r/beta_old(s+l)*(T_r_melt-T_r(j,i));
                        b= del_t/2/tau_r*(T_r(j,i+1)-T_f(j,i+1)+T_f(j,i)-T_r
(j,i))+...
                            del_t/2/tau_r/beta_old(s+l)*(T_f_int_old(s)-...
                            T_f(j,i)+T_r(j,i)-T_r_melt)+...
                            (T_f_int_old(s)-T_f(j,i))/beta_old(s+l);
                        c= T_r_melt/H_CR*(h_r_melt-h_r(j,i+1))+...
                            T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i)+...
                            (T_f(j,i)-T_f_int_old(s))/beta_old(s+l);
                        d= 2*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1));

                        disp('zeta calc')
                        poly_ans_z=roots([a b c d]);
                        for root_find=1:1:3
                            if (poly_ans_z(root_find)>0 & poly_ans_z
(root_find)<1.0)
                                eta=poly_ans_z(root_find);
                                beta_new(s+l)=-1;
                                T_f_int(s)=-1;
                                h_r_int(s)=-1;
                            end
                        end

                        T_f_eta=2*tau_r*T_r_melt/H_CR/del_t/eta*...
                                (h_r_melt-h_r(j,i+1)+eta*H_CR*del_t/2/...
                                tau_r/T_r_melt*(T_r_melt+T_r(j,i+1)-T_f(j,i
+1)));
                        T_f(j+1,i+1)=(T_f(j,i)+del_t/2/tau_r*(T_r_melt+...
                                T_r(j,i)-T_f(j,i)))/(1+del_t/2/tau_r);
                        h_r(j+1,i+1)=h_r_melt-(1-eta)*H_CR*del_t/2/tau_r/
T_r_melt*...
                                (2*T_r_melt-T_f(j+1,i+1)-T_f_eta);
                        T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                        pos_int(s+l,j+1)=del_z*i;

                    else
```

```
                    a=2*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f(j,i+1))+...
                        4*tau_r*T_r_melt/H_CR/del_t*(h_r(j,i+1)-h_r(j,i
+2))+...
                        2/beta_old(s+l)*(T_f(j,i)-T_f_int_old(s))+...
                        del_t/tau_r/beta_old(s+l)*(T_r(j,i)-T_r_int_old
(s))+...
                        del_t/tau_r*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f
(j,i+1))+...
                        2*T_r_melt/H_CR*(h_r(j,i+1)-h_r(j,i+2))+...
                        del_t/tau_r/beta_old(s+l)*(T_f_int_old(s)-T_f
(j,i));
                    b=2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i))+...
                        4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,i+1))-
...
                        del_t/tau_r*(T_r(j,i)-T_r(j,i+1)+T_f(j,i+1)-T_f(j,
i))+...
                        2*T_r_melt/H_CR*(h_r_melt-h_r(j,i+1));

                    %Check for singulatiry in calculation
                    if (abs(a)<1.0e-10) & (abs(b)<1.0e-10)
                        disp('coefficients are zero')
                        beta_new(s+1)=0;
                    else
                        beta_new(s+l)=-b/a;
                    end

                    if beta_new(s+l)<=0
                        %Special Calculation for Beta=0
                        beta_new(s+l)=0;
                        test_count=test_count+1;
                        test(test_count,:)=[(T_f(j,i)+del_t/2/tau_r*...
                            (T_r_melt+T_r(j,i)-T_f(j,i)))/(1+del_t/2/
tau_r),...
                            (h_r_melt-h_r(j,i+1)+H_CR*del_t/2/tau_r/
T_r_melt...
                            *(T_r_melt+T_r(j,i+1)-T_f(j,i+1)))...

                            /(H_CR*del_t/2/tau_r/T_r_melt),s+l,j];

                        T_f_int(s)=test(test_count,1);
                        T_f(j+1,i+1)=T_f_int(s);
                        h_r(j+1,i+1)=h_r_melt;
```

```
                        T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                    else
                        %Check to make sure not going faster than
                        %the HTF characteristic
                        if beta_new(s+l)>beta_old(s+l)
                            beta_new(s+l)=beta_old(s+l);
                            disp('going too fast');
                            LHS=[(1+del_t/tau_r/2), 0;...
                                (-del_t*H_CR/2/tau_r/T_r_melt), 1];
                            RHS=[T_f_int_old(s)+del_t/2/tau_r*(T_r_melt...
                                +T_r_int_old(s)-T_f_int_old(s));...
                                h_r(j,i+1)+beta_new(s+l)*(h_r(j,i+2)...
                                -h_r(j,i+1))-H_CR*del_t/2/tau_r/
T_r_melt*...
                                (T_r_melt+T_r(j,i+1)+beta_new(s+l)...
                                *(T_r(j,i+2)-T_r(j,i+1))-T_f(j,i+1)-...
                                beta_new(s+l)*(T_f(j,i+2)-T_f(j,i+1)))];
                            matrix=LHS\RHS;
                            T_f_int(s)=matrix(1);
                            h_r_int(s)=matrix(2);
                            T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                            toofast(s+l)=toofast(s+l)+1;
                        else
                            % Calculate the cases for beta_new<=beta_old
                            T_f_int(s)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)
+beta_new(s+l)*...
                                    (T_r(j,i+2)-T_r(j,i+1)-T_f(j,i
+2)...
                                    +T_f(j,i+1))+2*tau_r*T_r_melt/H_CR/
del_t*...
                                    (h_r_melt-h_r(j,i+1)-beta_new(s+l)*
(h_r(j,i+2)-h_r(j,i+1)));
                            h_r_int(s)=h_r_melt;
                            T_r_int(s)=h_to_T(h_r_int(s),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                        end
```

```
                            %Calculate New Values
                            eta=(1-beta_old(s+l))/(beta_new(s+l)-beta_old(s+l)
+1);
                            T_f_eta=T_f_int_old(s)+eta*(T_f_int(s)-T_f_int_old
(s));
                            T_r_eta=T_r_int_old(s)+eta*(T_r_int(s)-T_r_int_old
(s));
                            h_r_eta=h_r_int_old(s)+eta*(h_r_int(s)-h_r_int_old
(s));

                            % Update the values at (j=1,i=1)
                            T_f(j+1,i+1)=(T_f(j,i)+del_t/2/tau_r*(T_r_melt...
                                        +T_r(j,i)-T_f(j,i)))/(1+del_t/2/
tau_r);
                            h_r(j+1,i+1)=h_r_eta-(1-eta)*H_CR*del_t/2/tau_r/
T_r_melt*...
                                        (T_r_melt+T_r_eta-T_f(j+1,i+1)-
T_f_eta);
                            T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end

                        if j==t_steps
                            last_i(track)=i+1;
                        end
                    end

                else
                    % Calculate the cases for beta_new<1.0
                    eta=beta_old(s+l)/(1-beta_new(s+l)+beta_old(s+l));
                    T_f_eta=T_f_int_old(s)+eta*(T_f_int(s)-T_f_int_old(s));
                    T_r_eta=T_r_int_old(s)+eta*(T_r_int(s)-T_r_int_old(s));
                    h_r_eta=h_r_int_old(s)+eta*(h_r_int(s)-h_r_int_old(s));

                    LHS=[(1+(1-eta)*del_t/tau_r/2),-(1-eta)*del_t*T_r_melt/
tau_r/2;...
                         (-del_t*H_CR/2/tau_r/T_r_melt),(1+H_CR*del_t/tau_r/
2)];
                    RHS=[((1-(1-eta)*del_t/tau_r/2)*T_f_eta+(1-eta)*del_t/2/
tau_r*(T_r_eta+T_r_ref));...
                        (h_r(j,i+1)-(del_t*H_CR/2/tau_r/T_r_melt)*(T_r_ref
+T_r(j,i+1)-T_f(j,i+1)))];
                    matrix=LHS\RHS;
                    T_f(j+1,i+1)=matrix(1);
```

```
                    h_r(j+1,i+1)=matrix(2);
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
                end
                %s=s-1;

            elseif type(j,i+1)==2
                %disp('in 2/2');
                LHS=[(1+del_t/tau_r/2), 0;...
                     (-del_t*H_CR/2/tau_r/T_r_melt), 1];
                RHS=[((1-del_t/tau_r/2)*T_f(j,i)+(del_t/tau_r*T_r_melt));...
                     (h_r(j,i+1)-(del_t*H_CR/2/tau_r/T_r_melt)*(2*T_r_melt...
                     -T_f(j,i+1)))];
                matrix=LHS\RHS;
                T_f(j+1,i+1)=matrix(1);
                h_r(j+1,i+1)=matrix(2);
                T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

            else %type(j,i+1)==3
                disp('in 2/3');

                disp('original beta');
                a=(del_t/tau_r*(T_r(j+1,i)-T_r(j,i)+T_f(j,i)-T_f(j+1,i)))+...
                  (del_t/tau_r/(1-beta_old(s+l))*(T_r(j,i+1)+T_f_liq_old
(l)...
                  -T_r_liq_old(l)-T_f(j,i+1)))+(2*T_r_melt/H_CR/(1-beta_old
(s+l)))...
                  *(h_r_liq_old(l)-h_r(j,i+1)));
                b=(2/(1-beta_old(s+l))*(T_r(j,i+1)-T_r_liq_old(l)+T_f_liq_old
(l)-T_f(j,i+1)))+...
                  4*tau_r*T_r_melt/del_t/H_CR/(1-beta_old(s+l))*(h_r_liq_old
(l)-h_r(j,i+1))+...
                  2*(T_f(j+1,i)-T_f(j,i))+del_t/tau_r*(T_r(j,i+1)+T_f(j+1,i)-
T_f(j,i+1)-T_r(j+1,i))+...
                  del_t/tau_r/(1-beta_old(s+l))*(T_r_liq_old(l)+T_f(j,i+1)-
T_r(j,i+1)-T_f_liq_old(l))+...
                  2*T_r_melt/H_CR*(h_r_melt+1.0/stf-h_r(j,i+1)-((h_r_liq_old
(l)...
                  -h_r(j,i+1))/(1-beta_old(s+l))));
                c=4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r(j,i+1)-
...
```

```
                  ((h_r_liq_old(l)-h_r(j,i+1))/(1-beta_old(s+l))))+...
                  2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i))+...
                  2/(1-beta_old(s+l))*(T_r_liq_old(l)+T_f(j,i+1)-T_r(j,i+1)-
T_f_liq_old(l));
                if a==0
                    disp('a=0')
                    beta_new(s+l)=-c/b;
                else
                    poly_ans=roots([a b c]);
                    beta_new(s+l)=poly_ans(2);
                end
                T_f_liq(l)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)-((1-beta_new(s
+l))...
                           /(1-beta_old(s+l)))*(T_r(j,i+1)-T_r_liq_old(l)...
                           +T_f_liq_old(l)-T_f(j,i+1))+2*tau_r*T_r_melt/H_CR/
del_t*...
                           (h_r_melt+1.0/stf-h_r(j,i+1)-((1-beta_new(s+l))...
                           /(1-beta_old(s+l)))*(h_r_liq_old(l)-h_r(j,i+1)));
                h_r_liq(l)=h_r_melt+1.0/stf;
                T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);

                pos_int(s+l,j)=del_z*(i-1)+beta_old(s+l)*del_z;

                if j==t_steps
                    track=track+1;
                    last_i(track)=i;
                end

                %Check if new Beta cross into the next grid
                if beta_new(s+l)>1.0
                    disp('beta > 1')
                    %Check whether in last spacial step or not
                    if i==z_steps
                        disp('crossed spacial boundary')
                        a= del_t/2/tau_r/beta_old(s+l)*(T_f(j,i)-T_f_liq_old
(l))+...
                           del_t/2/tau_r/beta_old(s+l)*(T_r_melt-T_r(j,i));
                        b= del_t/2/tau_r*(T_r(j,i+1)-T_f(j,i+1)+T_f(j,i)...
                           -T_r(j,i))+del_t/2/tau_r/beta_old(s+l)*(T_f_liq_
old(l)-...
```

```
                        T_f(j,i)+T_r(j,i)-T_r_melt)+(T_f_liq_old(l)-...
                        T_f(j,i))/beta_old(s+l);
                    c= T_r_melt/H_CR*(h_r_melt+1.0/stf-h_r(j,i+1))+...
                        T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i)+...
                        (T_f(j,i)-T_f_liq_old(l))/beta_old(s+l);
                    d= 2*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r
(j,i+1));

                    disp('zeta calc')
                    poly_ans_z=roots([a b c d]);
                    for root_find=1:1:3
                        if (poly_ans_z(root_find)>0 & poly_ans_z
(root_find)<1.0)
                            eta=poly_ans_z(root_find);
                            beta_new(s+l)=-1;
                            T_f_liq(l)=-1;
                            h_r_liq(l)=-1;
                        end
                    end

                    T_f_eta=2*tau_r*T_r_melt/H_CR/del_t/eta*...
                            (h_r_melt+1.0/stf-h_r(j,i+1)+eta*...
                            H_CR*del_t/2/tau_r/T_r_melt*(T_r_melt+...
                            T_r(j,i+1)-T_f(j,i+1)));
                    T_f(j+1,i+1)=(T_f(j,i)+del_t/2/tau_r*(T_r_melt+...
                                T_r(j,i)-T_f(j,i)))/(1+del_t/2/tau_r);
                    h_r(j+1,i+1)=h_r_melt+1.0/stf-(1-eta)*H_CR*del_t...
                                /2/tau_r/T_r_melt*(2*T_r_melt-T_f(j+1,
i+1)-T_f_eta);
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                    pos_int(s+l,j+1)=del_z*i;

                else
                    disp('not last block');
                    a=2*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f(j,i+1))+...
                        4*tau_r*T_r_melt/H_CR/del_t*(h_r(j,i+1)-h_r(j,
i+2))+...
                        2/beta_old(s+l)*(T_f(j,i)-T_f_liq_old(l))+...
                        del_t/tau_r/beta_old(s+l)*(T_r(j,i)-T_r_liq_old
(l))+...
```

```
                    del_t/tau_r*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f
(j,i+1))+...
                    2*T_r_melt/H_CR*(h_r(j,i+1)-h_r(j,i+2))+...
                    del_t/tau_r/beta_old(s+l)*(T_f_liq_old(l)-T_f
(j,i));
                b=2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i))+...
                    4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1/stf-h_r
(j,i+1))-...
                    del_t/tau_r*(T_r(j,i)-T_r(j,i+1)+T_f(j,i+1)-T_f
(j,i))+...
                    2*T_r_melt/H_CR*(h_r_melt+1/stf-h_r(j,i+1));

                %Check for singulatiry in calculation
                if (abs(a)<1.0e-10) & (abs(b)<1.0e-10)
                    disp('coefficients are zero')
                    beta_new(s+1)=0;
                else
                    beta_new(s+l)=-b/a;
                end

                %Special Calculation for Beta=0
                if beta_new(s+l)<=0
                    %disp('beta=0')
                    beta_new(s+l)=1e-4;
                    test_count=test_count+1;
                    test(test_count,:)=[(T_f(j,i)+del_t/2/tau_r*...
                        (T_r_melt+T_r(j,i)-T_f(j,i)))/(1+del_t/2/
tau_r),...
                        (h_r_melt+1/stf-h_r(j,i+1)+H_CR*del_t/2/tau_r/
T_r_melt...
                        *(T_r_melt+T_r(j,i+1)-T_f(j,i+1)))/
(H_CR*del_t/2/tau_r/T_r_melt),...
                        s+l,j];
                    T_f_liq(l)=test(test_count,1);
                    T_f(j+1,i+1)=T_f_liq(l);
                    h_r(j+1,i+1)=h_r_melt+1/stf;
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                else
```

```
                        %Check to make sure not going faster than
                        %the HTF characteristic
                        if beta_new(s+l)>beta_old(s+l)
                            beta_new(s+l)=beta_old(s+l);
                            disp('going too fast');
                            LHS=[(1+del_t/tau_r/2), 0;...
                                (-del_t*H_CR/2/tau_r/T_r_melt), 1];
                            RHS=[T_f_liq_old(l)+del_t/2/tau_r*(T_r_melt
+T_r_liq_old(l)-T_f_liq_old(l));...
                                 h_r(j,i+1)+beta_new(s+l)*(h_r(j,i+2)-h_r
(j,i+1))-H_CR*del_t/2/tau_r/T_r_melt*...
                                (T_r_melt+T_r(j,i+1)+beta_new(s+l)*(T_r
(j,i+2)-T_r(j,i+1))-T_f(j,i+1)-beta_new(s+l)*(T_f(j,i+2)-T_f(j,i+1)))];
                            matrix=LHS\RHS;
                            T_f_liq(l)=matrix(1);
                            h_r_liq(l)=matrix(2);
                            T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                            toofast(s+l)=toofast(s+l)+1;
                        else
                            T_f_liq(l)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)
+beta_new(s+l)*...
                                (T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f
(j,i+1))+...
                                2*tau_r*T_r_melt/H_CR/del_t*...
                                (h_r_melt+1.0/stf-h_r(j,i+1)-beta_new(s+l)
*(h_r(j,i+2)-h_r(j,i+1)));
                            h_r_liq(l)=h_r_melt+1.0/stf;
                            T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end

                        %Calculate New Values
                        eta=(1-beta_old(s+l))/(beta_new(s+l)-beta_old(s+l)
+1);
                        T_f_eta=T_f_liq_old(l)+eta*(T_f_liq(l)-T_f_liq_old
(l));
                        T_r_eta=T_r_liq_old(l)+eta*(T_r_liq(l)-T_r_liq_old
(l));
                        h_r_eta=h_r_liq_old(l)+eta*(h_r_liq(l)-h_r_liq_old
(l));

                        T_f(j+1,i+1)=(T_f(j,i)+del_t/2/tau_r*(T_r_melt+T_r
(j,i)-T_f(j,i)))/...
```

```
                                        (1+del_t/2/tau_r);
                            h_r(j+1,i+1)=h_r_eta-(1-eta)*H_CR*del_t/2/tau_r/
T_r_melt*...
                                         (T_r_melt+T_r_eta-T_f(j+1,i+1)-
T_f_eta);
                            T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end

                        if j==t_steps
                            last_i(track)=i+1;
                        end
                    end

                else
                    eta=beta_old(s+l)/(1-beta_new(s+l)+beta_old(s+l));
                    T_f_eta=T_f_liq_old(l)+eta*(T_f_liq(l)-T_f_liq_old(l));
                    T_r_eta=T_r_liq_old(l)+eta*(T_r_liq(l)-T_r_liq_old(l));
                    h_r_eta=h_r_liq_old(l)+eta*(h_r_liq(l)-h_r_liq_old(l));

                    LHS=[(1+(1-eta)*del_t/tau_r/2), -cp_r_s/cp_r_l*(1-eta)
*del_t*T_r_melt/tau_r/2;...
                        (-del_t*H_CR/2/tau_r/T_r_melt), (1+H_CR*del_t/tau_r/
2*cp_r_s/cp_r_l)];
                    RHS=[((1-(1-eta)*del_t/tau_r/2)*T_f_eta+(1-eta)*del_t/2/
tau_r*(T_r_melt+T_r_eta-cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1/stf)));...
                        (h_r(j,i+1)-(del_t*H_CR/2/tau_r/T_r_melt)*(T_r_melt
+T_r(j,i+1)-T_f(j,i+1)-cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1/stf)))];
                    matrix=LHS\RHS;
                    T_f(j+1,i+1)=matrix(1);
                    h_r(j+1,i+1)=matrix(2);
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
                end
                %l=l-1;
            end

        else %type(j,i)==3

            % This is a special case that we want to avoid by choosing the
            % proper time step
            if type(j,i+1)==1
```

```
                disp('Heat transfer is too fast...try a smaller step size');
                error=error+1;

            elseif type(j,i+1)==2
                disp('in 3/2'); % Liquid interface created

                a=(del_t/tau_r*(T_r(j+1,i)-T_r(j,i)+T_f(j,i)-T_f(j+1,i)))+...
                  (del_t/tau_r/(1-beta_old(s+l))*(T_r(j,i+1)+T_f_liq_old(l)-
...
                  T_r_liq_old(l)-T_f(j,i+1)))+(2*T_r_melt/H_CR/(1-beta_old
(s+l))...
                  *(h_r_liq_old(l)-h_r(j,i+1)));
                b=(2/(1-beta_old(s+l))*(T_r(j,i+1)-T_r_liq_old(l)+...
                  T_f_liq_old(l)-T_f(j,i+1)))+4*tau_r*T_r_melt/del_t/...
                  H_CR/(1-beta_old(s+l))*(h_r_liq_old(l)-h_r(j,i+1))+...
                  2*(T_f(j+1,i)-T_f(j,i))+del_t/tau_r*(T_r(j,i+1)+...
                  T_f(j+1,i)-T_f(j,i+1)-T_r(j+1,i))+del_t/tau_r/(1-...
                  beta_old(s+l))*(T_r_liq_old(l)+T_f(j,i+1)-T_r(j,i+1)...
                  -T_f_liq_old(l))+2*T_r_melt/H_CR*(h_r_melt+1.0/stf-...
                  h_r(j,i+1)-((h_r_liq_old(l)-h_r(j,i+1))/(1-beta_old
(s+l))));
                c=4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r(j,i+1)...
                  -((h_r_liq_old(l)-h_r(j,i+1))/(1-beta_old(s+l))))+2*(...
                  T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i))+2/(1-beta_old
(s+l))...
                  *(T_r_liq_old(l)+T_f(j,i+1)-T_r(j,i+1)-T_f_liq_old(l));

                % Calculate the partial fraction beta_new to track the
                % liquid interface
                if a==0
                    disp('a=0')
                    beta_new(s+l)=-c/b;
                else
                    poly_ans=roots([a b c]);
                    beta_new(s+l)=poly_ans(2);
                end

                % Calculate the temp. of HTF and filler, enthalpy of filler
                % at the liquid interface
                T_f_liq(l)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)-((1-beta_new(s+l))/
(1-beta_old(s+l)))*...
```

```
                        (T_r(j,i+1)-T_r_liq_old(l)+T_f_liq_old(l)-T_f(j,i
+1))+...
                        2*tau_r*T_r_melt/H_CR/del_t*...
                        (h_r_melt+1.0/stf-h_r(j,i+1)-((1-beta_new(s+l))...
                        /(1-beta_old(s+l)))*(h_r_liq_old(l)-h_r(j,i+1)));
                h_r_liq(l)=h_r_melt+1.0/stf;
                T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,T_r_melt,T_r_ref,
cp_r_s,cp_r_l);

                % Track and update the position of liquid interface
                pos_int(s+l,j)=del_z*(i-1)+beta_old(s+l)*del_z;
                if j==t_steps
                    track=track+1;
                    last_i(track)=i;
                end

                %Check if new Beta cross into the next grid
                if beta_new(s+l)>1.0
                    %Recalculate Beta New
                    disp('in new z');

                    %Check whether in last spatial grid
                    if i==z_steps
                        disp('crossed spatial boundary')
                        a= del_t/2/tau_r/beta_old(s+l)*(T_f(j,i)-T_f_liq_old
(l))+...
                            del_t/2/tau_r/beta_old(s+l)*(T_r_melt-T_r(j,i));
                        b= del_t/2/tau_r*(T_r(j,i+1)-T_f(j,i+1)+T_f(j,i)-T_r
(j,i))+...
                            del_t/2/tau_r/beta_old(s+l)*(T_f_liq_old(l)-T_f
(j,i)+T_r(j,i)-T_r_melt)+...
                            (T_f_liq_old(l)-T_f(j,i))/beta_old(s+l);
                        c= T_r_melt/H_CR*(h_r_melt+1.0/stf-h_r(j,i+1))+...
                            T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i)+(T_f(j,i)-
T_f_liq_old(l))/beta_old(s+l);
                        d= 2*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r
(j,i+1));

                        disp('zeta calc')
                        poly_ans_z=roots([a b c d]);
                        for root_find=1:1:3
```

```
                        if (poly_ans_z(root_find)>0 & poly_ans_z
(root_find)<1.0)
                            eta=poly_ans_z(root_find);
                            beta_new(s+l)=-1;
                            T_f_liq(l)=-1;
                            h_r_liq(l)=-1;
                        end
                    end

                    T_f_eta=2*tau_r*T_r_melt/H_CR/del_t/eta*...
                            (h_r_melt+1.0/stf-h_r(j,i+1)+eta*H_CR*del_t...
                            /2/tau_r/T_r_melt*(T_r_melt+T_r(j,i+1)-T_f(j,i
+1)));

                    LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt/tau_r...
                        /2*cp_r_s/cp_r_l;(-(1-eta)*del_t*H_CR/2/tau_r...
                        /T_r_melt),(1+(1-eta)*H_CR*del_t/tau_r/2*cp_r_s/
cp_r_l)];
                    RHS=[T_f(j,i)+del_t/2/tau_r*(T_r(j,i)-T_f(j,i)+...
                        T_r_melt-cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1.0/
stf));...
                        h_r_melt+1.0/stf-(1-eta)*H_CR*del_t/2/tau_r/
T_r_melt...
                        *(2*T_r_melt-T_f_eta-T_r_melt*cp_r_s/cp_r_l*
(h_r_melt+1.0/stf))];
                    matrix=LHS\RHS;
                    T_f(j+1,i+1)=matrix(1);
                    h_r(j+1,i+1)=matrix(2);
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

                    pos_int(s+l,j+1)=del_z*i;

                else
                    a=2*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f(j,i+1))+...
                        4*tau_r*T_r_melt/H_CR/del_t*(h_r(j,i+1)-h_r(j,i
+2))+...
                        2/beta_old(s+l)*(T_f(j,i)-T_f_liq_old(l))+...
                        del_t/tau_r/beta_old(s+l)*(T_r(j,i)-T_r_liq_old
(l))+...
                        del_t/tau_r*(T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)+T_f
(j,i+1))+...
```

```
                2*T_r_melt/H_CR*(h_r(j,i+1)-h_r(j,i+2))+...
                del_t/tau_r/beta_old(s+l)*(T_f_liq_old(l)-T_f
(j,i));
            b=2*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j,i))+...
                4*tau_r*T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r
(j,i+1))-...
                del_t/tau_r*(T_r(j,i)-T_r(j,i+1)+T_f(j,i+1)-T_f
(j,i))+...
                2*T_r_melt/H_CR*(h_r_melt+1.0/stf-h_r(j,i+1));

            %Check for singulatiry in calculation
            if (abs(a)<1.0e-10) & (abs(b)<1.0e-10)
                disp('coefficients are zero')
                beta_new(s+1)=0;
            else
                beta_new(s+l)=-b/a;
                disp('got here')
            end

            %Special Calculation for Beta=0
            if beta_new(s+l)<=0
                beta_new(s+l)=1e-4;
                test_count=test_count+1;
                test(test_count,:)=[(T_f(j,i)+del_t/2/tau_r*...
                    (T_r_melt+T_r(j,i)-T_f(j,i)))/(1+del_t/2/
tau_r),...
                    (h_r_melt+1/stf-h_r(j,i+1)+H_CR*del_t/2/
tau_r...
                    /T_r_melt*(T_r_melt+T_r(j,i+1)-T_f(j,i+1)))...
                    /(H_CR*del_t/2/tau_r/T_r_melt),s+l,j];

                T_f_liq(l)=test(test_count,1);
                T_f(j+1,i+1)=T_f_liq(l);
                h_r(j+1,i+1)=h_r_melt+1/stf;

                T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);

            else
```

```
                        %Check to make sure not going faster than the
                        %HTF characteristic
                        if beta_new(s+l)>beta_old(s+l)
                            beta_new(s+l)=beta_old(s+l);
                            disp('going too fast')

                            LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt...
                                /2/tau_r*cp_r_s/cp_r_l;...
                                (-del_t*H_CR/2/tau_r/T_r_melt),...
                                (1+H_CR*del_t/2/tau_r*cp_r_s/cp_r_l)];
                            RHS=[T_f_liq_old(l)+del_t/2/tau_r*(...
                                T_r_melt+T_r_liq_old(l)-T_f_liq_old(l)...
                                -cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1.0/
stf));...
                                h_r_(j,i+1)+beta_new(s+l)*(h_r(j,i+2)...

                                -h_r(j,i+1))-H_CR*del_t/2/tau_r/
T_r_melt*...
                                (T_r_melt+T_r(j,i+1)+beta_new(s+l)...
                                *(T_r(j,i+2)-T_r(j,i+1))-T_f(j,i+1)...
                                -beta_new(s+l)*(T_f(j,i+2)-T_f(j,i+1))...
                                -cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1.0/
stf))];
                            matrix=LHS\RHS;
                            T_f_liq(l)=matrix(1);
                            h_r_liq(l)=matrix(2);
                            T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                            toofast(s+l)=toofast(s+l)+1;

                        else
                            T_f_liq(l)=T_r_melt+T_r(j,i+1)-T_f(j,i+1)
+beta_new(s+l)*...
                                       (T_r(j,i+2)-T_r(j,i+1)-T_f(j,i+2)
+T_f(j,i+1))+...
                                       2*tau_r*T_r_melt/H_CR/del_t*...
                                       (h_r_melt+1.0/stf-h_r(j,i+1)-beta_
new(s+l)...
                                       *(h_r(j,i+2)-h_r(j,i+1)));
                            h_r_liq(l)=h_r_melt+1.0/stf;
                            T_r_liq(l)=h_to_T(h_r_liq(l),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end
```

```
                            %Calculate New Values
                            eta=(1-beta_old(s+l))/(beta_new(s+l)-beta_old(s+l)
+1);
                            T_f_eta=T_f_liq_old(l)+eta*(T_f_liq(l)-T_f_liq_old
(l));
                            T_r_eta=T_r_liq_old(l)+eta*(T_r_liq(l)-T_r_liq_old
(l));
                            h_r_eta=h_r_liq_old(l)+eta*(h_r_liq(l)-h_r_liq_old
(l));

                            LHS=[(1+del_t/tau_r/2), -del_t*T_r_melt/tau_r/
2*cp_r_s/cp_r_l;...
                                (-(1-eta)*del_t*H_CR/2/tau_r/T_r_melt),...
                                (1+(1-eta)*H_CR*del_t/tau_r/2*cp_r_s/cp_r_l)];
                            RHS=[T_f(j,i)+del_t/2/tau_r*(T_r(j,i)-T_f(j,i)...
                                +T_r_melt-cp_r_s/cp_r_l*T_r_melt*(h_r_melt
+1.0/stf));...
                                h_r_eta-(1-eta)*H_CR*del_t/2/tau_r/
T_r_melt*...
                                (T_r_melt+T_r_eta-T_f_eta-T_r_melt*cp_r_s/
cp_r_l*(h_r_melt+1.0/stf))];
                            matrix=LHS\RHS;
                            T_f(j+1,i+1)=matrix(1);
                            h_r(j+1,i+1)=matrix(2);

                            T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,
T_r_melt,T_r_ref,cp_r_s,cp_r_l);
                        end

                        if j==t_steps
                            last_i(track)=i+1;
                        end
                    end

                else
                    eta=beta_old(s+l)/(1-beta_new(s+l)+beta_old(s+l));
                    T_f_eta=T_f_liq_old(l)+eta*(T_f_liq(l)-T_f_liq_old(l));
                    T_r_eta=T_r_liq_old(l)+eta*(T_r_liq(l)-T_r_liq_old(l));
                    h_r_eta=h_r_liq_old(l)+eta*(h_r_liq(l)-h_r_liq_old(l));
```

```
                    T_f(j+1,i+1)=(T_f_eta*(1-(1-eta)*del_t/2/tau_r)+
(1-eta)...
                                *del_t/2/tau_r*(T_r_melt+T_r_eta))/(1+(1-
eta)*del_t/2/tau_r);
                    h_r(j+1,i+1)=h_r(j,i+1)-H_CR*del_t/2/tau_r/T_r_melt*...
                                (T_r_melt+T_r(j,i+1)-T_f(j,i+1)-T_f(j+1,i
+1));
                    T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

                end
                %l=l-1;

            else %type(j,i+1)==3
                %disp('in 3/3');

                LHS=[(1+del_t/tau_r/2), (-del_t*T_r_melt/2/tau_r*cp_r_s/
cp_r_l);...
                     (-del_t*H_CR/2/tau_r/T_r_melt),(1+del_t*H_CR/2/
tau_r*cp_r_s/cp_r_l)];
                RHS=[((1-del_t/tau_r/2)*T_f(j,i)+(del_t/2/tau_r*...
                     (T_r(j,i)+T_r_melt-T_r_melt*cp_r_s/cp_r_l*(h_r_melt+1.0/
stf))));...
                     (h_r(j,i+1)-(del_t*H_CR/2/tau_r/T_r_melt)*...
                     (T_r(j,i+1)+T_r_melt-T_f(j,i+1)-T_r_melt*cp_r_s/cp_r_l*
(h_r_melt+1.0/stf)))];
                matrix=LHS\RHS;
                T_f(j+1,i+1)=matrix(1);
                h_r(j+1,i+1)=matrix(2);
                T_r(j+1,i+1)=h_to_T(h_r(j+1,i+1),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
            end

        end

        %Define Type for new calculation
        type(j+1,i+1)=def_type(h_r(j+1,i+1),h_r_melt,stf,k);
    end

    %Test if Beta calculation is necessary before
    %proceeding to next step in time
    if type(j,1)==1
        %Onset of Solidous Interface at boundary (heat being added to filler)
```

```
if type(j+1,1)==2
   disp('beta 1/2')
   s_ct=s_ct+1;
   %Initial Beta Calculation
   a=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,2)-T_r(j,1)...
     -T_f(j,2)+T_f(j,1)-T_f(j+1,1)+T_f_int_old(s_ct)+T_r(j+1,1)...
     -T_r_melt)-2*del_t/tau_r*(h_r(j,2)-h_r(j,1)));
   b=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,1)-T_f(j,1)+...
     T_f(j+1,1)-T_r(j+1,1))+2*H_CR*del_t/tau_r/T_r_melt*(T_r(j,2)...
     -T_r(j,1)-T_f(j,2)+T_f(j,1)+T_f(j+1,1)-T_f_int_old(s_ct))...
     -4*(h_r(j,2)-h_r(j,1))+2*del_t/tau_r*(h_r_melt-h_r(j,1)));
   c=2*H_CR*del_t/tau_r/T_r_melt*(T_r_melt+T_r(j,1)-T_f(j,1)...
     -T_f(j+1,1))+4*(h_r_melt-h_r(j,1));
   if a==0
       disp('a=0')
       beta_new(s_ct+l_ct)=-c/b;
   else
       poly_ans=roots([a b c]);
       beta_new(s_ct+l_ct)=poly_ans(2);
   end

   %Check to make sure interface is not travelling faster than
   %the HTF characteristic
   if beta_new(s_ct+l_ct)>(1-alpha(s_ct))
        disp('interface travelling too fast boundary');
        toofast(3)=1;
        beta_new(s_ct+l_ct)=(1-alpha(s_ct));

        LHS=[(1+del_t*(1-alpha(s_ct))/tau_r/2), 0;...
            (-del_t*H_CR/2/tau_r/T_r_melt), 1];
        RHS=[alpha(s_ct)*T_f(j+1,1)+(1-alpha(s_ct))*T_f(j,1)...
            +del_t*(1-alpha(s_ct))/2/tau_r*(2*T_r_melt-...
            alpha(s_ct)*T_f(j+1,1)-(1-alpha(s_ct))*T_f(j,1));...
            alpha(s_ct)*h_r(j,1)+(1-alpha(s_ct))*h_r(j,2)-H_CR*...
            del_t/2/tau_r/T_r_melt*(T_r_melt+alpha(s_ct)*T_r(j,1)...
            +(1-alpha(s_ct))*T_r(j,2)-alpha(s_ct)*T_f(j,1)...
            -(1-alpha(s_ct))*T_f(j,2))];
        matrix=LHS\RHS;
        T_f_int(s_ct)=matrix(1);
        h_r_int(s_ct)=matrix(2);
```

```
                T_r_int(s_ct)=h_to_T(h_r_int(s_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);


            else
                 T_f_int(s_ct)=T_r_melt+T_r(j,1)-T_f(j,1)+...
                             beta_new(s_ct+l_ct)*(T_r(j,2)+T_f(j,1)-T_r(j,1)-
T_f(j,2))+...
                             2*tau_r*T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,1)...
                             -beta_new(s_ct+l_ct)*(h_r(j,2)-h_r(j,1)));
                 h_r_int(s_ct)=h_r_melt;
                 T_r_int(s_ct)=h_to_T(h_r_int(s_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
            end


        elseif type(j+1,1)==3
             disp('Heat Transfer is occuring too fast...try a smaller step size
to better resolve');
        end


    elseif type(j,1)==2


        if type(j+1,1)==3   %Onset of Liquidoous Interface at boundary (heat
being added to filler)
             disp('beta 2/3')
             l_ct=l_ct+1;
            %Initial Beta Calculation
            a=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,2)-T_r(j,1)-...
              T_f(j,2)+T_f(j,1)-T_f(j+1,1)+T_f_liq_old(l_ct)+T_r(j+1,1)...
              -T_r_melt)-2*del_t/tau_r*(h_r(j,2)-h_r(j,1)));
            b=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,1)-T_f(j,1)...
              +T_f(j+1,1)-T_r(j+1,1))+2*H_CR*del_t/tau_r/T_r_melt*...
              (T_r(j,2)-T_r(j,1)-T_f(j,2)+T_f(j,1)+T_f(j+1,1)-...
              T_f_liq_old(l_ct))-4*(h_r(j,2)-h_r(j,1))+2*del_t/tau_r...
              *(h_r_melt+1.0/stf-h_r(j,1)));
            c=2*H_CR*del_t/tau_r/T_r_melt*(T_r_melt+T_r(j,1)-T_f(j,1)-...
              T_f(j+1,1))+4*(h_r_melt+1.0/stf-h_r(j,1));
            if a==0
                disp('a=0')
                beta_new(s_ct+l_ct)=-c/b;
            else
```

```
            poly_ans=roots([a b c]);
            beta_new(s_ct+l_ct)=poly_ans(2);

        end

        %Check to make sure interface is not travelling faster than
        %the HTF characteristic
        if beta_new(s_ct+l_ct)>(1-gamma(l_ct))
            disp('interface travelling too fast');
            toofast(4)=1;
            beta_new(s_ct+l_ct)=(1-gamma(l_ct));

            LHS=[(1+del_t*(1-gamma(l_ct))/tau_r/2), -del_t*T_r_melt...
                *(1-gamma(l_ct))/2/tau_r*cp_r_s/cp_r_l;(-del_t*H_CR...
                /2/tau_r/T_r_melt),1+H_CR*del_t/2/tau_r*cp_r_s/cp_r_l];
            RHS=[gamma(l_ct)*T_f(j+1,1)+(1-gamma(l_ct))*T_f(j,1)+del_t...
                *(1-gamma(l_ct))/2/tau_r*(2*T_r_melt-cp_r_s/cp_r_l...
                *T_r_melt*(h_r_melt+1.0/stf)-gamma(l_ct)*T_f(j+1,1)...
                -(1-gamma(l_ct))*T_f(j,1));gamma(l_ct)*h_r(j,1)...
                +(1-gamma(l_ct))*h_r(j,2)-H_CR*del_t/2/tau_r/T_r_melt*...
                (T_r_melt-cp_r_s/cp_r_l*T_r_melt*(h_r_melt+1.0/stf)...
                +gamma(l_ct)*T_r(j,1)+(1-gamma(l_ct))*T_r(j,2)...
                -gamma(l_ct)*T_f(j,1)-(1-gamma(l_ct))*T_f(j,2))];
            matrix=LHS\RHS;
            T_f_liq(l_ct)=matrix(1);
            h_r_liq(l_ct)=matrix(2);
            T_r_liq(l_ct)=h_to_T(h_r_liq(l_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

        else
            T_f_liq(l_ct)=T_r_melt+T_r(j,1)-T_f(j,1)+beta_new(s_ct
+l_ct)...
                          *(T_r(j,2)+T_f(j,1)-T_r(j,1)-T_f(j,2))
+2*tau_r...
                          *T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r
(j,1)...
                          -beta_new(s_ct+l_ct)*(h_r(j,2)-h_r(j,1)));
            h_r_liq(l_ct)=h_r_melt+1.0/stf;
            T_r_liq(l_ct)=h_to_T(h_r_liq(l_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

        end
```

```
        elseif type(j+1,1)==1  %Onset of Solidous Interface at boundary (heat
being removed from filler)
            disp('beta 2/1')
            s_ct=s_ct+1;
            %Beta Calculation
            a=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,2)-T_r(j,1)-...
               T_f(j,2)+T_f(j,1)-T_f(j+1,1)+T_f_int_old(s_ct)+T_r(j+1,1)-...
               T_r_melt)-2*del_t/tau_r*(h_r(j,2)-h_r(j,1)));
            b=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,1)-T_f(j,1)+...
               T_f(j+1,1)-T_r(j+1,1))+2*H_CR*del_t/tau_r/T_r_melt*(T_r
(j,2)...
               -T_r(j,1)-T_f(j,2)+T_f(j,1)+T_f(j+1,1)-T_f_int_old(s_ct))-...
               4*(h_r(j,2)-h_r(j,1))+2*del_t/tau_r*(h_r_melt-h_r(j,1)));
            c= 2*H_CR*del_t/tau_r/T_r_melt*(T_r_melt+T_r(j,1)-T_f(j,1)-...
               T_f(j+1,1))+4*(h_r_melt-h_r(j,1));
            if a==0
               disp('a=0')
               beta_new(s_ct+l_ct)=-c/b;
            else
               poly_ans=roots([a b c]);
               beta_new(s_ct+l_ct)=poly_ans(2);

            end

            %Check to make sure interface is not travelling faster than
            %the HTF characteristic
            if beta_new(s_ct+l_ct)>(1-alpha(s_ct))
                disp('interface travelling too fast');
                beta_new(s_ct+l_ct)=(1-alpha(s_ct));

                LHS=[(1+del_t*(1-alpha(s_ct))/tau_r/2), -del_t*T_r_melt...
                    *(1-alpha(s_ct))/2/tau_r;(-del_t*H_CR/2/tau_r/
T_r_melt)...
                    ,1+H_CR*del_t/2/tau_r];
                RHS=[alpha(s_ct)*T_f(j+1,1)+(1-alpha(s_ct))*T_f(j,1)...
                    +del_t*(1-alpha(s_ct))/2/tau_r*(T_r_melt+T_r_ref...
                    -alpha(s_ct)*T_f(j+1,1)-(1-alpha(s_ct))*T_f(j,1));...
                    alpha(s_ct)*h_r(j,1)+(1-alpha(s_ct))*h_r(j,2)-H_CR*...
                    del_t/2/tau_r/T_r_melt*(T_r_ref+alpha(s_ct)*T_r(j,1)...
```

```
                    +(1-alpha(s_ct))*T_r(j,2)-alpha(s_ct)*T_f(j,1)-...
                    (1-alpha(s_ct))*T_f(j,2))];
                matrix=LHS\RHS;
                T_f_int(s_ct)=matrix(1);
                h_r_int(s_ct)=matrix(2);
                T_r_int(s_ct)=h_to_T(h_r_int(s_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

            else
                T_f_int(s_ct)=T_r_melt+T_r(j,1)-T_f(j,1)+beta_new(s_ct
+l_ct)...
                              *(T_r(j,2)+T_f(j,1)-T_r(j,1)-T_f(j,2))
+2*tau_r...
                              *T_r_melt/H_CR/del_t*(h_r_melt-h_r(j,1)...
                              -beta_new(s_ct+l_ct)*(h_r(j,2)-h_r(j,1)));
                h_r_int(s_ct)=h_r_melt;
                T_r_int(s_ct)=h_to_T(h_r_int(s_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
            end

        end

    else %type(j,1)==3

%Onset of Liquidoous Interface at boundary (heat being removed from filler)
        if type(j+1,1)==2
           disp('beta 3/2')
           l_ct=l_ct+1;

           %Initial Beta Calculation
           a=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,2)-T_r(j,1)...
             -T_f(j,2)+T_f(j,1)-T_f(j+1,1)+T_f_liq_old(l_ct)+T_r(j+1,1)-
T_r_melt)-...
             2*del_t/tau_r*(h_r(j,2)-h_r(j,1)));
           b=(H_CR*del_t*del_t/tau_r/tau_r/T_r_melt*(T_r(j,1)-T_f(j,1)...
             +T_f(j+1,1)-T_r(j+1,1))+2*H_CR*del_t/tau_r/T_r_melt*(T_r(j,2)-
T_r(j,1)...
             -T_f(j,2)+T_f(j,1)+T_f(j+1,1)-T_f_liq_old(l_ct))-4*(h_r(j,2)-h_r
(j,1))...
             +2*del_t/tau_r*(h_r_melt+1/stf-h_r(j,1)));
```

```
            c=2*H_CR*del_t/tau_r/T_r_melt*(T_r_melt+T_r(j,1)-T_f(j,1)-T_f
(j+1,1))...
              +4*(h_r_melt+1/stf-h_r(j,1));
            if a==0
                disp('a=0')
                beta_new(s_ct+l_ct)=-c/b;
            else
                poly_ans=roots([a b c]);
                beta_new(s_ct+l_ct)=poly_ans(2);
            end

            %Check to make sure interface is not travelling faster than
            %the HTF characteristic
            if beta_new(s_ct+l_ct)>(1-gamma(l_ct))
                 disp('interface travelling too fast');
                 beta_new(s_ct+l_ct)=(1-gamma(l_ct));

                 LHS=[(1+del_t*(1-gamma(l_ct))/tau_r/2), 0;(-del_t*H_CR/2/
tau_r/T_r_melt),1];
                 RHS=[gamma(l_ct)*T_f(j+1,1)+(1-gamma(l_ct))*T_f(j,1)...
                     +del_t*(1-gamma(l_ct))/2/tau_r*(2*T_r_melt-gamma(l_ct)...
                     *T_f(j+1,1)-(1-gamma(l_ct))*T_f(j,1));gamma(l_ct)*h_r
(j,1)...
                     +(1-gamma(l_ct))*h_r(j,2)-H_CR*del_t/2/tau_r/T_r_melt...
                     *(T_r_melt+gamma(l_ct)*T_r(j,1)+(1-gamma(l_ct))*T_r(j,2)-
...
                     gamma(l_ct)*T_f(j,1)-(1-gamma(l_ct))*T_f(j,2))];
                 matrix=LHS\RHS;
                 T_f_liq(l_ct)=matrix(1);
                 h_r_liq(l_ct)=matrix(2);
                 T_r_liq(l_ct)=h_to_T(h_r_liq(l_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);

            else
                 T_f_liq(l_ct)=T_r_melt+T_r(j,1)-T_f(j,1)+beta_new(s_ct
+l_ct)...
                              *(T_r(j,2)+T_f(j,1)-T_r(j,1)-T_f(j,2))
+2*tau_r...
                              *T_r_melt/H_CR/del_t*(h_r_melt+1.0/stf-h_r
(j,1)...
                              -beta_new(s_ct+l_ct)*(h_r(j,2)-h_r(j,1)));
                 h_r_liq(l_ct)=h_r_melt+1.0/stf;
```

```
                T_r_liq(l_ct)=h_to_T(h_r_liq(l_ct),h_r_melt,stf,T_r_melt,
T_r_ref,cp_r_s,cp_r_l);
            end

        elseif type(j+1,1)==1
            disp('Heat Transfer is occuring too fast...try a smaller step size
to better resolve');
        end
    end
end

%Capture Output information for cycling

%Dimensional quantities
T_r_melt=dim(T_r_melt,T_L,T_H,T_L,1);
h_r_melt=dim(h_r_melt,h_r_ref,T_r_melt,T_L,cp_r_s);
T_r_ref=dim(T_r_ref,T_L,T_H,T_L,1);

%Switch coordinates due to cyclic charge and discharge process
for i=1:z_steps+1
     T_f_last(1,z_steps+2-i)=dim(T_f(t_steps+1,i),T_L,T_H,T_L,1);
     h_r_last(1,z_steps+2-i)=dim(h_r(t_steps+1,i),h_r_ref,T_r_melt,T_L,
cp_r_s);
end

%Equilibrium condition after Discharge Process
if cycle_type==1
    disp('apply equil')
    for i=1:z_steps+1
        %Filler at a point that is originally solid
        if h_r_last(1,i)<h_r_melt
            h_r_eq(1,i)=(rho_f*A_f*cp_f*(h_r_ref/cp_r_s-T_r_ref+T_f_last
(1,i))...
                        +rho_r*A_r*(h_r_last(1,i)))/(rho_r*A_r+rho_
f*A_f*cp_f/cp_r_s);
            T_f_eq(1,i)=(h_r_eq(1,i)-h_r_ref)/cp_r_s+T_r_ref;

            if h_r_eq(1,i)>h_r_melt
                T_f_eq(1,i)=(rho_f*A_f*cp_f*T_f_last(1,i)-rho_r*A_r*
(h_r_melt...
                            -h_r_last(1,i)))/(rho_f*A_f*cp_f);
```

```
                h_r_eq(1,i)=(rho_f*A_f*cp_f*(T_f_eq(1,i)-T_r_melt)+rho_r...
                            *A_r*h_r_melt)/(rho_r*A_r);
                T_f_eq(1,i)=T_r_melt;
            end
        end

        %Filler at a point that is originally melting
        if h_r_last(1,i)>=h_r_melt && h_r_last(1,i)<(h_r_melt+L)
           h_r_eq(1,i)=(rho_f*A_f*cp_f*(T_f_last(1,i)-T_r_melt)+rho_r...
                       *A_r*h_r_last(1,i))/(rho_r*A_r);
           T_f_eq(1,i)=T_r_melt;

            if h_r_eq(1,i)>(h_r_melt+L)
               T_f_eq(1,i)=(rho_f*A_f*cp_f*T_f_last(1,i)-rho_r*A_r*
(h_r_melt...
                           +L-h_r_last(1,i)))/(rho_f*A_f*cp_f);
               h_r_eq(1,i)=(rho_f*A_f*cp_f*((h_r_melt+L)/cp_r_l-T_r_melt...
                           +T_f_eq(1,i))+rho_r*A_r*(h_r_melt+L))/(rho_r...
                           *A_r+rho_f*A_f*cp_f/cp_r_l);
               T_f_eq(1,i)=(h_r_eq(1,i)-h_r_melt-L)/cp_r_l+T_r_melt;

            elseif h_r_eq(1,i)<h_r_melt
               T_f_eq(1,i)=(rho_f*A_f*cp_f*T_f_last(1,i)-rho_r*A_r*...
                           (h_r_melt-h_r_last(1,i)))/(rho_f*A_f*cp_f);
               h_r_eq(1,i)=(rho_f*A_f*cp_f*(h_r_ref/cp_r_s-T_r_ref+T_f_eq
(1,i))...
                           +rho_r*A_r*h_r_melt)/(rho_r*A_r+rho_f*A_f*cp_f/
cp_r_s);
               T_f_eq(1,i)=(h_r_eq(1,i)-h_r_ref)/cp_r_s+T_r_ref;
            end
        end

        %Filler at a point that is originally liquid
        if h_r_last(1,i)>=(h_r_melt+L)
               h_r_eq(1,i)=(rho_f*A_f*cp_f*((h_r_melt+L)/cp_r_l-T_r_melt...
                           +T_f_last(1,i))+rho_r*A_r*h_r_last(1,i))...
                           /(rho_r*A_r+rho_f*A_f*cp_f/cp_r_l);
               T_f_eq(1,i)=(h_r_eq(1,i)-h_r_melt-L)/cp_r_l+T_r_melt;

            if h_r_eq(1,i)<(h_r_melt+L)
               T_f_eq(1,i)=(rho_f*A_f*cp_f*T_f_last(1,i)-rho_r*A_r*...
```

```
                        (h_r_melt+L-h_r_last(1,i)))/(rho_f*A_f*cp_f);
            h_r_eq(1,i)=(rho_f*A_f*cp_f*(T_f_eq(1,i)-T_r_melt)+rho_r...
                        *A_r*(h_r_melt+L))/(rho_r*A_r);
            T_f_eq(1,i)=T_r_melt;
        end
    end
end
end
h_r_last=h_r_eq;
T_f_last=T_f_eq;
end

%Capture necessary interface information
if cycle_type==0
    count=0;
    beta_last=0;
    for i=1:(s_ct+l_ct)
        if beta_new(i)~=-1
            count=count+1;
        end
    end

    for i=1:(s_ct+l_ct)
        if beta_new(i)~=-1
            beta_last(count)=1.0-beta_new(i);
            count=count-1;
        end
    end

    %HTF, filler temperature and enthalpy at the solidus interface
    count=0;
    h_r_int_last=0;
    T_r_int_last=0;
    T_f_int_last=0;
    for i=1:s_ct
        if h_r_int(i)~=-1
            count=count+1;
        end
    end
```

```
    for i=1:s_ct
        if h_r_int(i)~=-1
            h_r_int_last(count)=h_r_int(i);
            T_f_int_last(count)=T_f_int(i);
            T_r_int_last(count)=T_r_int(i);
            count=count-1;
        end
    end

    %HTF, filler temperature and enthalpy at the liquidus interface
    count=0;
    h_r_liq_last=0;
    T_r_liq_last=0;
    T_f_liq_last=0;
    for i=1:l_ct
        if h_r_liq(i)~=-1
            count=count+1;
        end
    end

    for i=1:l_ct
        if h_r_liq(i)~=-1
            h_r_liq_last(count)=h_r_liq(i);
            T_f_liq_last(count)=T_f_liq(i);
            T_r_liq_last(count)=T_r_liq(i);
            count=count-1;
        end
    end
else %equilibrium case
    %Find new interface positions and interface values
    count=0;
    for i=1:(s_ct+l_ct)
        if beta_new(i)~=-1
            count=count+1;
        end
    end

    int_ct=0;
    liq_ct=0;
    beta_last=0;
    h_r_int_last=0;
```

```
    T_f_int_last=0;
    T_r_int_last=0;
    h_r_liq_last=0;
    T_f_liq_last=0;
    T_r_liq_last=0;

    for i=1:z_steps
        %Solidus Interface
        if h_r_last(i)<h_r_melt && h_r_last(i+1)>h_r_melt
            disp('solidus interface created')
            beta_last(count-int_ct-liq_ct)=(h_r_melt-h_r_last(1,i))...
                                        /(h_r_last(1,i+1)-h_r_last(1,i));
            h_r_int_last(s_ct-int_ct)=dim_less(h_r_melt,h_r_ref,T_r_melt,T_L,
cp_r_s);
            T_f_int_last(s_ct-int_ct)=T_f_last(1,i)+beta_last(count-int_ct-
liq_ct)...
                                    *(T_f_last(1,i+1)-T_f_last(1,i));
            T_f_int_last(s_ct-int_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            T_r_int_last(s_ct-int_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            int_ct=int_ct+1;
        end

        if h_r_last(i)>h_r_melt && h_r_last(i+1)<h_r_melt
            disp('solidus interface created')
            beta_last(count-int_ct-liq_ct)=(h_r_melt-h_r_last(1,i))...
                                        /(h_r_last(1,i+1)-h_r_last(1,i));
            h_r_int_last(s_ct-int_ct)=dim_less(h_r_melt,h_r_ref,T_r_melt,T_L,
cp_r_s);
            T_f_int_last(s_ct-int_ct)=T_f_last(1,i)+beta_last(count-int_ct-
liq_ct)...
                                    *(T_f_last(1,i+1)-T_f_last(1,i));
            T_f_int_last(s_ct-int_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            T_r_int_last(s_ct-int_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            int_ct=int_ct+1;
        end

        %Liquidus Interface
        if h_r_last(i)<(h_r_melt+L) && h_r_last(i+1)>(h_r_melt+L)
            disp('liquidus interface created')
            beta_last(count-int_ct-liq_ct)=(h_r_melt+L-h_r_last(1,i))...
                                        /(h_r_last(1,i+1)-h_r_last(1,i));
```

```
            h_r_liq_last(l_ct-liq_ct)=dim_less(h_r_melt,h_r_ref,T_r_melt,T_L,
cp_r_s)+1/stf;
            T_f_liq_last(l_ct-liq_ct)=T_f_last(1,i)+beta_last(count-int_ct-
liq_ct)...
                                    *(T_f_last(1,i+1)-T_f_last(1,i));
            T_f_liq_last(l_ct-liq_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            T_r_liq_last(l_ct-liq_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            liq_ct=liq_ct+1;
        end

         if h_r_last(i)>(h_r_melt+L) && h_r_last(i+1)<(h_r_melt+L)
            disp('liquidus interface created')
            beta_last(count-int_ct-liq_ct)=(h_r_melt+L-h_r_last(1,i))...
                                        /(h_r_last(1,i+1)-h_r_last(1,i));
            h_r_liq_last(l_ct-liq_ct)=dim_less(h_r_melt,h_r_ref,T_r_melt,T_L,
cp_r_s)+1/stf;
            T_f_liq_last(l_ct-liq_ct)=T_f_last(1,i)+beta_last(count-int_ct-
liq_ct)...
                                    *(T_f_last(1,i+1)-T_f_last(1,i));
            T_f_liq_last(l_ct-liq_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            T_r_liq_last(l_ct-liq_ct)=dim_less(T_r_melt,T_L,T_H,T_L,1);
            liq_ct=liq_ct+1;
         end
    end
end

% Update the number of solid and liquid interfaces
s_ct_last=int_ct;
l_ct_last=liq_ct;

% Save the data of current run, use it as the initial data for the next run
save(['cycle_',num2str(k),'.mat'], 'beta_last', 'h_r_last', 'T_f_last',...
    'T_f_int_last', 'h_r_int_last','T_r_int_last', 'T_f_liq_last',...
    'h_r_liq_last','T_r_liq_last','cycle_type','s_ct_last','l_ct_last');
save(['cycle_data_',num2str(k),'.mat'],'T_f','h_r','z','t','-v7.3');

end
```

REFERENCES

[1] Van Lew JT, Li PW, Chan CL, Karaki W, Stephens J. Transient heat delivery and storage process in a thermocline heat storage system, IMECE2009-11701. In: Proceedings of the ASME 2009 international mechanical congress and exposition, November 13–19, Lake Buena Vista, Florida, USA; 2009.

[2] Kays WM, Crawford ME, Weigand B. Convective heat and mass transfer. 4th ed. Boston: McGraw Hill; 2005.

[3] Incropera FP, DeWitt DP. Introduction to heat transfer. 4th ed. New York: John Wiley and Sons, Inc; 2002.

[4] Xu B, Li P-W, Lik Chan C. Extending the validity of lumped capacitance method for large Biot number in thermal storage application. Sol Energy 2012;86(6):1709–24.

[5] Conway JH, Sloane NJH. Sphere packings, lattices and groups. 3rd ed. New York: Springer; 1998. ISBN 0-387-98585-9.

[6] Hales TC. Historical overview of the Kepler conjecture, discrete & computational geometry. Int J Math Comput Sci 2006;36(1):5–20.

[7] Nellis G, Klein S. Heat transfer. New York: Cambridge University Press; 2009.

[8] Bradshaw AV, Johnson A, McLachlan NH, Chiu Y-T. Heat transfer between air and nitrogen and packed beds of non-reacting solids. Trans Instn Chem Eng 1970;48:T77–84.

[9] Jefferson CP. Prediction of breakthrough curves in packed beds: 1. applicability of single parameter models. Am Inst Chem Eng 1972;18(2):409–16.

[10] Hausen H. Wärmeübertragung im Gegenstrom, Gleichstrom und Kreuzstrom. 2nd ed. Berlin: Springer-Verlag; 1976.

[11] Schmidt FW, Willmott AJ. Thermal energy storage and regeneration. New York, NY: McGraw-Hill Book Company; 1981.

[12] Razelos P, Lazaridis A. A lumped heat-transfer coefficient for periodically heated hollow cylinders. Int J Heat Mass Transf 1967;10:1373–87.

[13] Hughes PJ, Klein SA, Close DJ. Packed bed thermal storage models for solar air heating and cooling systems. ASME J Heat Transf 1976;98(2):336–8.

[14] Mumma SA, Marvin WC. A method of simulating the performance of a pebble bed thermal energy storage and recovery system. ASME Paper No. 76-HT-73, In: ASME-AICHE heat transfer conf., St. Louis, Missouri; 1976.

[15] Ozisik MN. Heat Conduction. 2nd ed. New York: Wiley-Interscience; 1993, ISBN 0471532568.

[16] Kakac S. Heat conduction. 3rd ed. Washington, DC: Taylor and Francis; 1993.

[17] Schumann TEW. Heat transfer: a liquid flowing through a porous prism. J Frankl Inst 1929;208(3):405–16.

[18] Shitzer A, Levy M. Transient behavior of a rock-bed thermal storage system subjected to variable inlet air temperatures: analysis and experimentation. J Sol Energy Eng 1983;105(2):200–6.

[19] McMahan AC. Design and optimization of organic Rankine cycle solar-thermal power plants [Master's thesis].Madison, WI: University of Wisconsin; 2006.

[20] Beasley DE, Clark JA. Transient response of a packed bed for thermal energy storage. Int J Heat Mass Transf 1984;27(9):1659–69.

[21] Zarty O, Juddaimi AE. Computational models of a rock-bed thermal storage unit. Sol Wind Technol 1987;2(4):215–8.

[22] McMahan AC, Klein SA, Reindl DT. A finite-time thermodynamic framework for optimizing solar-thermal power plants. J Sol Energy Eng 2007;129(4):355–62.

[23] Pacheco JE, Showalter SK, Kolb WJ. Development of a molten salt thermocline thermal storage system for parabolic trough plants. J Sol Energy Eng 2002;124(2):153–9.

[24] Kolb GJ, Hassani V. Performance analysis of thermocline energy storage proposed for the 1 mw saguaro solar trough plant. ASME Conf Proc 2006;2006(47454):1–5.
[25] Van Lew JT, Li PW, Chan CL, Karaki W, Stephens J. Analysis of heat storage and delivery of a thermocline tank having solid filler material. J Sol Energy Eng 2011;133:021003.
[26] Courant R, Hilbert D. Methods of mathematical physics, vol. 2. New York: Wiley-Interscience; 1962.
[27] Polyanin AD. Handbook of linear partial differential equations for engineers and scientists. Boca Raton: Chapman & Hall/CRC Press; 2002, ISBN 1-58488-299-9.
[28] Ferziger JH. Numerical methods for engineering applications. New York: Wiley-Interscience; 1998.
[29] Karaki W, Van Lew JT, Li PW, Chan CL, Stephens J. Heat transfer in thermocline storage system with filler materials: analytical model. In: Proceedings of the ASME 2010 4th international conference on energy sustainability, ES2010-90209, May 17–22, Phoenix, Arizona, USA; 2010.
[30] Nallusamy N, Sampath S, Velraj R. Experimental investigation on a combined sensible and latent heat storage system integrated with constant/varying (solar) heat sources. Renew Energy 2007;32:1206–27.
[31] Verma P, Varun, Singal SK. Review of mathematical modeling on latent heat thermal energy storage systems using phase-change material. Renew Sust Energ Rev 2008;12:999–1031.
[32] Tan FL, Hosseinizadeh SF, Khodadadi JM. Liwu Fan, Experimental and computational study of constrained melting of phase change materials (PCM) inside a spherical capsule. Int J Heat Mass Transf 2009;52(15):3464–72.
[33] Zhao W, Elmozughi AF, Oztekien A, Neti S. Heat transfer analysis of encapsulated phase change material for thermal energy storage. Int J Heat Mass Transf 2013;63:323–35.
[34] Regin AF, Solanki SC. An analysis of a packed bed latent heat thermal energy storage system using pcm capsules: numerical investigation. Renew Energy 2009;34:1765–73.
[35] Wu SM, Fang GY, Liu X. Dynamic discharging characteristics simulation on solar heat storage system with spherical capsules using paraffin as heat storage material. Renew Energy 2011;36:1190–5.
[36] Nithyanandam K, Pitchumani R. Thermal energy storage with heat transfer augmentation using thermosyphons. Int J Heat Mass Transf 2013;67:281–94.
[37] Nithyanandam K, Pitchumani R. Computational studies on a latent thermal energy storage system with integral heat pipes for concentrating solar power. Appl Energy 2013;103:400–15.
[38] Archibold AR, Rahman MM, Goswami DY, Stefanakos EL. Parametric investigation of the melting and solidification process in an encapsulated spherical container. In: Proceedings of the ASME 2012 6th international conference on energy sustainability. July 23–26, San Diego, CA, USA; 2012.
[39] Vyshak NR, Jilani G. Numerical analysis of latent heat thermal energy storage system. Energy Convers Manag 2007;48(7):2161–8.
[40] Tumilowicz E, Cho Lik C, Li P, Xu B. An enthalpy formulation for thermocline with encapsulated PCM thermal storage and benchmark solution using the method of characteristics. Int J Heat Mass Transf 2014;79:362–77.
[41] Yang Z, Garimella SV. Molten-salt thermal energy storage in thermoclines under different environmental boundary conditions. Appl Energy 2010;87(11):3322–9.
[42] Li P-W, Van Lew J, Karaki W, Chan C-L, Stephens J, O'Brien JE. Transient heat transfer and energy transport in packed bed thermal storage systems. In: dos Santos Bernardes MA, editor. Developments in Heat Transfer; 2011, ISBN 978-953-307-569-3. http://www.intechopen.com [Accessed September 15].

[43] Becker M. Comparison of heat transfer fluid for use in solar thermal power stations. Electr Power Syst Res 1980;3:139–50.
[44] Li P, Xu B, Han J, Yang Y. Verification of a model of thermal storage incorporated with an extended lumped capacitance method for various solid-fluid structural combinations. Sol Energy 2014;105:71–81.
[45] Li P-W, Van Lew J, Chan C-L, Karaki W, Stephens J, O'Brien JE. Similarity and generalized analysis of efficiencies of thermal energy storage systems. Renew Energy 2012;39:388–402.
[46] Modi A, Pérez-Segarra CD. Thermocline thermal storage systems for concentrated solar power plants: One-dimensional numerical model and comparative analysis. Sol Energy 2014;100:84–93.
[47] Yang Z, Garimella Suresh V. Thermal analysis of solar thermal energy storage in a molten-salt thermocline. Sol Energy 2010;84:974–85.
[48] Flueckiger S, Yang Z, Garimella SV. An Integrated thermal and mechanical investigation of molten-salt thermocline energy storage. Appl Energy 2011;88:2098–105.
[49] Xu C, Wang Z, He Y, Li X, Bai F. Sensitivity analysis of the numerical study on the thermal performance of a packed-bed molten salt thermocline thermal storage system. Appl Energy 2012;92:65–75.