A Model for a Variable-Volume, Density-Stratified, Liquid Desiccant Storage Tank

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Jason Mallinak

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

Dr. Jane H. Davidson, Adviser

August 2012

Acknowledgements

I would like to thank my adviser, Professor Jane Davidson, for the opportunity to work on this project and to participate in the Solar Energy Laboratory and for her input along the way. I am particularly grateful that she has helped me to sharpen my written and oral technical communication.

I would like to extend thanks to the members of my faculty committee, Professor James Ramsey and Professor Pat Huelman.

I am grateful to Dr. Joshua Quinnell for his invaluable advice along the way. From technical details regarding liquid desiccants to graduate student survival strategies, I am grateful for his shared insight.

Additionally, I would like to thank Peter Krenzke, Dan Keene, and Shuping Wang for their willingness to discuss technical challenges with me at various times over the past year and a half.

I am grateful for funding from the Department of Mechanical Engineering at the University of Minnesota (during my first semester) and from the National Renewable Energy Laboratory (for this project).

Finally, I would like to thank my family: my parents Michael and Theresa; my sisters Audre, Monica, DeAnna, Natalie, and Julia; and my brother Benjamin. I would not have reached this point without the ongoing prayers and support of my family (and of many friends). I thank God for providing such people to share life with while learning to keep it in the right perspective each day.

Abstract

A stratified, single-vessel liquid desiccant storage tank is an attractive option for the seasonal storage of solar energy. Maintaining stratified storage preserves the chemical energy stored in concentrated liquid desiccant solutions. Simulations of desiccant-based storage systems have been limited to modeling well-mixed storage. The subject of this paper is a density-stratified liquid desiccant storage tank model.

The one-dimensional storage is composed of a series of well-mixed segments of a liquid desiccant ($CaCl_2$ or LiCl) solution in a variable-volume, density-stratified storage tank. The density of each segment depends on both the temperature and the salt concentration of the desiccant solution. The model incorporates mixing due to mass diffusion and buoyancy and accounts for the heat of dilution in the energy balance. Thermal losses to the surroundings are accounted for and auxiliary heat input is permitted. Stored energy is a combination of sensible energy (a function of temperature) and chemical potential energy (a function of salt concentration and temperature).

Fluid movement within the tank is simulated with a plug flow approach. Fluid enters the tank through fixed inlets or through an idealized inlet manifold. Fluid exits the tank through fixed outlets.

For use with the Transient Systems Simulation (TRNSYS) software, the model is labeled TRNSYS Type 209. The user is able to define the storage characteristics (*e.g.* capacity, insulation, number and location of inlets and outlets, etc.), the initial conditions, and the boundary conditions of TRNSYS Type 209.

Several test cases demonstrate TRNSYS Type 209's capacity for modeling liquid desiccant storage using the plug flow approach. These cases use initial and boundary conditions that have been arbitrarily selected from within the expected operating ranges.

Table of Contents

List of Tables

List of Figures

Nomenclature

Roman Letters

a	Constant, -
A	Area, m^2; constant, -
b	Constant, -
c_p	Specific heat at constant pressure, kJ/kg-K
C	Constant, -
$CaCl_2$	Calcium chloride
CC	Circumference of the storage cross section, m
$\mathcal{D}$	Binary mass diffusion coefficient, m^2/s
h_d	Enthalpy of dilution, kJ/kg-H_2O
i	Arbitrary node/segment of fluid, -
k	Thermal conductivity, W/m^2-K
Le	Lewis number, -
$LiCl$	Lithium Chloride
m	Solution mass, kg-soln
$\dot{m}$	Solution mass flow rate, kg-soln/hr
M	Number of total segments, -; molar mass, kg/mol
$Mode$	Operational mode for inflow, -
n	Time step, -
N	Number of total inlet/outlet/auxiliary positions, -; Buoyancy ratio, -; Number of total nodes, -
Q	Energy, kJ
R	Universal gas constant, J/mol-K
Ra	Rayleigh number, -

S	Concentration of $CaCl_2$ or LiCl in desiccant solution, kg-salt/kg-soln
$\bar{S}$	Average concentration of $CaCl_2$ or LiCl in desiccant solution, kg-salt/kg-soln
Sh	Sherwood number, -
t	Time, hr
T	Temperature, °C or K
$\bar{T}$	Average temperature, °C or K
U_L	Heat transfer coefficient, W/hr-m^2-K
V	Volume, m^3
$\tilde{V}$	Molar volume, m^3/mol
z	Vertical coordinate, m; Inlet/outlet/auxiliary position, m

Greek Letters

Δ	Difference, -
μ	Dynamic viscosity, kg/m-s
ρ	Density, kg/m^3
φ	Volume fraction, -

Subscripts

amb	Ambient
aux	Auxiliary
avg	Average
b	Bottom wall
bw	Bottom wall (wet)
c	Cross-sectional
ch	Chemical energy
$cond$	Conduction

$crit$	Critical
$diff$	Diffusion
ERR	Error
gen	Generation
$H2O$	Water
h	Hot stream
i	Arbitrary segment of fluid
$i-1,i$	Average value for a segment of fluid and the neighboring segment below
$i-1 \rightarrow i$	Transport to a segment of fluid from the neighboring segment below
$i+1,i$	Average value for a segment of fluid and the neighboring segment above
$i+1 \rightarrow i$	Transport to a segment of fluid from the neighboring segment above
in	In
L	Load
$loss$	Thermal losses
max	Maximum value
min	Minimum value
new	New value or fluid segment
out	Out
pos	Property at a given inlet/outlet/auxiliary position
r	Recirculating
s	Side wall, Supply to the load
sd	Side wall (dry)
sw	Side wall (wet)
$salt$	Salt (either $CaCl_2$ or LiCl)
S	Concentration
se	Sensible energy
t	Top wall

T	Temperature
td	Top wall (dry)
tot	Total
u	Usable
w	Wall
0	Initial
%	Percent

Superscripts

n	Time step

1 Introduction and Motivation

Section 1.1 introduces liquid desiccant storage as an attractive option for seasonal solar energy storage and describes the benefits of creating a model for the storage. Section 1.2 provides a motivation for creating a new model for a liquid desiccant storage tank by introducing several desiccant-based modeling requirements that are not met by available tank models. Section 1.3 outlines the scope of the new model. Section 1.4 summarizes the contents of the chapters that follow.

1.1 Introduction

1.1.1 Liquid Desiccant Storage for Solar Energy

Over 15% of U.S. energy consumption is composed of residential space heating, space cooling, and domestic hot water (DHW) loads [1]. Currently, solar thermal systems are predominantly used for DHW loads and swimming pool heating, both of which require only diurnal storage. Space heating loads cannot be met with diurnal storage. The mismatch in solar energy availability (peak in summer) and space heating demand (peak in winter) requires seasonal storage in order for solar thermal systems to be able to meet the heating loads year-round. [2]

A collaborative effort between the Solar Energy Laboratory at the University of Minnesota (UMN) and the National Renewable Energy Laboratory (NREL) has focused on the development of a seasonal solar energy storage system using liquid desiccants, a type of sorption storage. Two benefits of using liquid desiccants rather than sensible or latent energy storage materials are increased energy density and decreased thermal losses. (Energy density is the amount of useful energy that can be stored per unit volume of storage.) UMN's novel concept of a single-tank liquid desiccant storage eliminates the need for the additional desiccant tank required by other desiccant-based systems, resulting

in an energy density more than double that of water [2]. In the concept developed by UMN and NREL (Figure 1.1), the desiccant solution is stored seasonally and transported to a heat pump, where the exothermic reaction raises the temperature to that which is required to meet the load. (When the storage fluid is hot enough to meet the load directly, it can be circulated directly to the load via a heat exchanger.) In the sorption charging process, water is desorbed from diluted calcium chloride ($CaCl_2$) solution in the solar collector. Both the concentrated $CaCl_2$ solution and the desorbed water are returned to the storage tank. [2, 3]

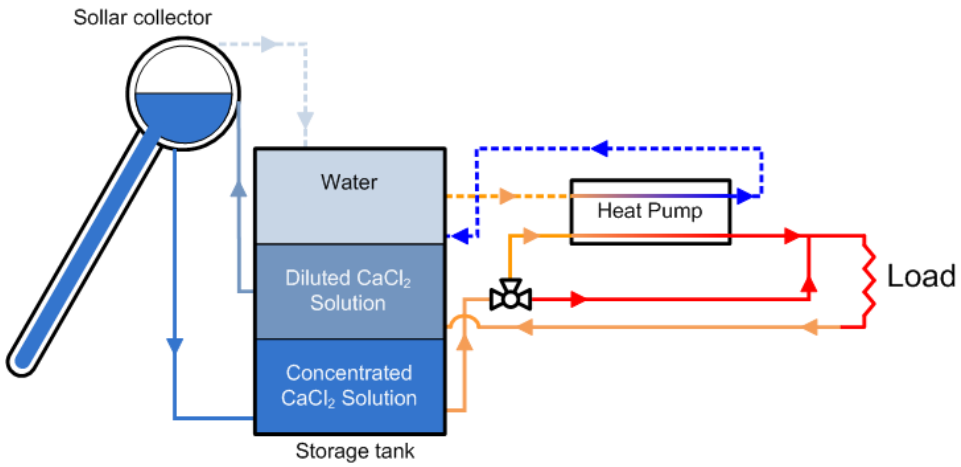

Figure 1.1: Desiccant-based solar energy storage concept developed by UMN and NREL. The sorption charging loop involves the storage tank and solar collector. The sorption discharging loop involves the storage tank, heat pump, and load. If the stored fluid is hot enough to meet the load requirements, it bypasses the heat pump. [3]

1.1.2 Storage Tank Modeling

Transient Systems Simulation (TRNSYS) is a software program that aids in the design of solar energy (and other transient energy) systems. System components, including the energy storage tank, are selected and assigned parameters to simulate system performance on a long term (*e.g.* annual) basis. The creation of a liquid desiccant storage

tank model is required in order for systems using desiccant-based storage to be modeled within TRNSYS.

1.2 Motivation

1.2.1 Variable-Concentration Binary Solutions

Liquid desiccants are composed of a salt and water solution. The concentration of the solution is the ratio of the salt mass to the total solution mass. The concentration varies by location within the tank. The concentration also changes over time via mass diffusion (occurring within the tank) and via absorption or desorption processes (occurring exterior to the tank). Thus, a liquid desiccant storage model requires that the concentration of the binary liquid desiccant solution be permitted to vary over time and location and that the mass of each species be conserved. These features are not present in available storage tank models.

1.2.2 Density Stratification of the Storage Fluid

Fluid density directly influences the vertical profile of a one-dimensional storage tank. Available tank models include three classes of fluids: those that are assumed to have constant density, those where the density is considered to be a function of temperature only (*e.g.* water, which has a density variation of ~5% over the operating range), and those where the density is considered to be a function of both the temperature and the concentration of a binary fluid. In the case of the binary fluid, however, the concentration is considered to be fixed over time and space, so that the density varies only with temperature over the duration of the simulation. As stated earlier, a liquid desiccant solution is variable-concentration. Thus, the density varies over time and space as a function of two variables: the temperature and the concentration of the solution (*i.e.* $\rho(t,z) = \rho\big(T(t,z), S(t,z)\big)$).

The density is predicted using curve fit equations from Conde [4]. These equations show that the density of the solution is much more sensitive to changes in concentration than it is to changes in temperature $\left(\frac{\partial \rho}{\partial S} \gg \frac{\partial \rho}{\partial T}\right)$, varying by ~50% over the operating range (an order of magnitude increase over the density variation caused by temperature).

1.2.3 Chemical Energy

Desorption of water from the desiccant solution results in a chemical binding energy within the solution. This chemical energy is stored until it is released as heat during the exothermic mixing of the solution with water or with a lower-concentration desiccant solution. Thus, a liquid desiccant storage model requires the presence of a source term in the energy equation, representing chemical energy being converted to sensible energy. This feature is not present in available storage tank models.

1.2.4 Variable-Volume Tank

Some desiccant-based systems allow for a portion of the fluid drawn from the tank to be stored elsewhere rather than to be returned immediately to the tank. Available tank models assume either a stratified constant-volume storage or a well-mixed variable-volume storage. The liquid desiccant tank, however, requires a stratified variable-volume storage. The approach selected for modeling the storage is required to take this into account.

1.3 Objectives

1.3.1 Scope

This model is designed to simulate the behavior of a density-stratified, variable-volume, liquid desiccant storage tank. A liquid desiccant is a binary solution of water and salt. The salt concentration of the solution may vary over time and position within the storage.

Due to these liquid desiccant characteristics, modeling requirements for desiccant-based storage extend beyond what is presently available. First, since the density of the liquid desiccant solution is sensitive to changes in both salt concentration and temperature, density is calculated as a function of both properties. Second, since a liquid desiccant is a binary solution, the conservation of mass is applied to individual species (not to the overall solution only). Third, since liquid desiccant solutions contain chemical binding energy, chemical energy potential is included in the energy equation. Additionally, since some applications for a liquid desiccant storage tank do not immediately return outgoing flow to the tank, the storage is considered to be variable volume.

Mass conservation laws are applied to a binary solution with both sensible and chemical potential energy. The model includes mass diffusion and mixing due to buoyancy as modes of mass transfer within the storage. Mass may also enter and exit the storage through inlet and outlet positions. (Outlet positions are fixed. Inlet positions may be fixed or inflow may be added at the location of neutral buoyancy.) Also included are the following mechanisms for energy transfer: conduction within the storage fluid, auxiliary heat input, and thermal losses to the environment.

The model neglects mixing due to momentum, neglects conduction along the wall, and does not include an internal heat exchanger.

1.3.2 Application

The model is compatible with TRNSYS. The user is able to define TRNSYS Parameters for tank geometry and for initial storage conditions. Additionally, the user defines TRNSYS Inputs for boundary conditions throughout the duration of a TRNSYS simulation. The model also includes TRNSYS Outputs that can either be printed for user analysis or be used as inputs for other components within a larger system model.

Appendices include tables with lists of TRNSYS Parameters, Inputs, Outputs, and error and warning messages.

1.4 Chapter Outline

Chapter 2 provides a background on standard approaches to solar energy storage tank modeling and examines the applicability each model's approach to a liquid desiccant storage tank model. The approach for modeling desiccant-based storage is described in Chapter 3. A general description is followed by a more detailed discussion of the equations and rules which were employed to implement the modeling approach. In Chapter 4, test cases for the model are described and the results of each case are presented. Chapter 5 provides concluding comments and recommendations for further development of the model.

2 Background

Section 2.1 discusses several approaches employed in available TRNSYS tank models. In each case, the modeling approach is first introduced and then discussed with respect to its applicability to the requirements of a liquid desiccant storage model. Following the discussion of available tank models, Section 2.2 briefly introduces liquid desiccant-based system models.

2.1 Storage Tank Models

2.1.1 Variable Volume

Variable volume storage models in TRNSYS are represented by Type 39 (Figure 2.1), a well-mixed storage tank with a constant cross-sectional area. The density and the specific heat of the storage are assumed to be constant. Thus, the energy stored within the tank is proportional to the temperature and to the fluid volume. The tank includes a single inlet and a single outlet (Figure 2.1(a)). The "hot" stream enters at flow rate $\dot{m}_h$ and temperature T_h. The "supply to the load" stream exits at flow rate $\dot{m}_s$ and storage temperature T. The volume of fluid in the tank is not permitted to drop below a specified lower limit. (The outflow is decreased to maintain the minimum fluid level if the specified flow rates would cause the storage to fall below the lower limit.) Similarly, the volume of fluid in the tank is not permitted to rise above a specified upper limit. Depending on the mode chosen by the user, excess "recirculating" inflow either will enter the tank and mix with the storage before exiting with the outflow ($\dot{m}_r$ in Figure 2.1(b)) or it will be diverted from entering the tank ($\dot{m}_r$ in Figure 2.1(c)). [5]

The similarities between the capabilities of Type 39 and the requirements of a liquid desiccant storage tank are limited to the following attributes: variable storage volume and constant cross-sectional area. With well-mixed, constant-density storage and

a single inlet and outlet, the overall modeling approach employed in Type 39 does not meet the requirements of a liquid desiccant storage tank model and is not helpful in devising a new modeling approach.

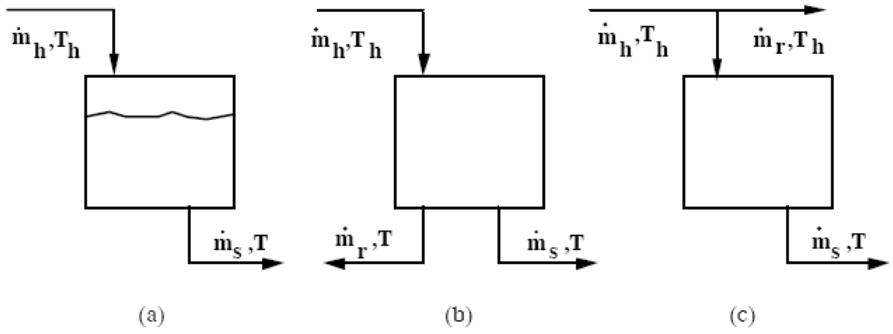

Figure 2.1: Variable-volume tank configurations: (a) normal operation, (b) mixing before recirculating excess inflow flow, (c) redirecting excess inflow [5]

2.1.2 Multinode

Duffie and Beckman introduce the first of two approaches used to model stratified storage models for sensible storage:

> Many stratified tank models have been developed; they fall into either of two categories. In the first, the multinode approach, a tank is modeled as divided into N nodes (sections), with energy balances written for each section of the tank; the result is a set of N differential equations that can be solved for the temperatures of the N nodes as functions of time. [6]

TRNSYS models such as Type 4 [7], Type 60 [8], and Type 534 [9] model thermal stratification using the multinode approach. The maximum number of fixed-volume nodes in a single tank ranges from 15-100, depending on the model. The multinode approach models a constant-volume storage: at each point in the simulation, the total mass entering the tank is equal to the total mass exiting the tank, and the fluid's density variation over

the operating temperature range is assumed to be negligible. Flow exits the tank through fixed outlets. Depending on the mode chosen by the user, flow enters the tank either through fixed-height inlets or through an idealized manifold (through which inflow is directed to the node with which it is closest in temperature). The tank has one or two inlets and a matching number of outlets. When two inlets and outlets are present, they are paired. Thus, the flow rate at each outlet matches the flow rate at its inlet pair. Conduction is modeled between segments. Thermal losses are modeled between the tank and its surroundings. An optional auxiliary heater and an optional internal heat exchanger are also available in some models (*e.g.* Type 60 and Type 534). [7, 8, 9]

The multinode approach includes several characteristics which are applicable to a liquid desiccant storage model. First, multinode models stratify the storage. Second, the user is afforded flexibility in positioning inlets and outlets. However, the multinode model equates density stratification with thermal stratification, whereas a liquid desiccant model requires density stratification to depend on concentration as well as temperature. Additionally a liquid desiccant model requires that the user be able to define outlet flow rates for variable-volume storage (rather than pairing outlets with inlets). These differences indicate that significant modifications are required in order to employ the multinode approach to a liquid desiccant storage model.

Additional requirements of a liquid desiccant storage model create further challenges in applying the multinode approach. First, two species are conserved. (The multinode model conserves only one species.) Second, the energy balance accounts for chemical energy through the desiccant's enthalpy of dilution. (The multinode model conserves only sensible energy in the energy balance.) Third, the storage is variable-volume. (The multinode model represents constant-volume storage.) Of these challenges, the variable-volume requirement is the most significant, since a fixed number of constant-volume nodes ultimately comprise a constant-volume storage.

A final challenge, however, prevents the multinode approach from meeting the requirements of a liquid desiccant storage model in a satisfactory manner—even through significant modifications. Namely, the range of density variation expected in a liquid desiccant tank precludes the use of constant-volume nodes. The multinode energy balance is dependent on the mass of a node being constant throughout a time step. Since each node is constant-volume, the energy balance is valid only if the density variation in the node over the course of the time step is negligible. The assumption of negligible density variation is acceptable for fluid densities that are dependent only on temperature. The strong dependence of a liquid desiccant's density on concentration, however, introduces non-negligible density variations. Thus, the use of constant-volume, constant-mass nodes precludes the multinode approach from being used for a liquid desiccant storage model.

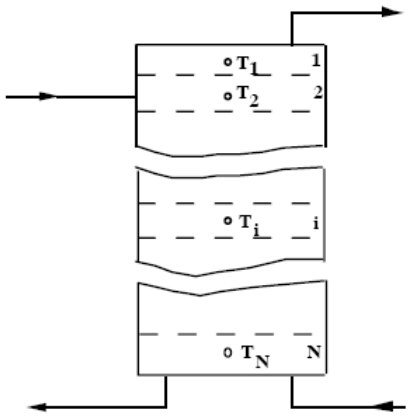

Figure 2.2: Multinode tank configuration with N well-mixed nodes and two inlet-outlet pairs [8]

2.1.3 Plug Flow

Duffie and Beckman also introduce the second category of stratified tank models: "In the second [category], the plug flow approach, segments of liquid at various temperatures are assumed to move through the tank in plug flow, and the models are

essentially bookkeeping methods to keep track of the size, temperature, and position of the segments." [6]

Models such as TRNSYS Type 38 [10] represent thermally stratified storage using the plug flow approach. Thermal stratification in Type 38 is depicted visually in Figure 2.3, where the vertical axis represents the temperature of each segment of fluid. The constant-volume storage is composed of variable-volume segments of fluid (also shown in Figure 2.3, where the horizontal axis is used to depict the volume of each segment). The user sets the maximum permissible number of segments. The stored fluid is assumed to have constant density throughout the simulation.

For each time step in a simulation, the plug flow approach divides the solution into two steps. While the processes in the two steps occur simultaneously, they are accounted for sequentially. First, the tank is considered a closed system. The energy conservation equation is solved for each fluid segment, incorporating terms for conduction between neighboring segments and for thermal losses to the surroundings. These energy transfer mechanisms are evaluated for the duration of the time step using the storage profile obtained from the solution to the previous time step. The temperature of each fluid segment in the storage profile is updated after the closed system analysis.

Second, the tank is considered an open system for the full duration of the time step, as depicted in the sequence of figures comprising Figure 2.3. The time step begins at time t_1 (Figure 2.3(a)). The tank has one inlet-outlet pair for a collector loop and one inlet-outlet pair for a load loop. Depending on the mode chosen by the user, flow enters the tank either through fixed inlets or through an idealized manifold (through which each flow enters the tank at the location which produces no temperature inversions). A volume of fluid returning from the collector loop (V_h at temperature T_h) is inserted into the storage profile in Figure 2.3(b). A volume of fluid returning from the load loop (V_L at temperature T_L) is inserted into the storage profile in Figure 2.3(c). Also in Figure 2.3(c),

the segments are shifted along the volume (horizontal) axis. Flow exits the tank through fixed outlets at each end. The segment of fluid extending beyond the "Top" demarcation on the volume axis is high-temperature storage sent to the load. (It is equal in volume to V_L.) The segment of fluid extending beyond the "Bottom" demarcation on the volume axis is low-temperature storage sent to the collector. (It is equal in volume to V_h.) The time step ends with the final storage profile at time $t_2 = t_1 + \Delta t$ (Figure 2.3(d)). An optional auxiliary heater is also available for this model. [10]

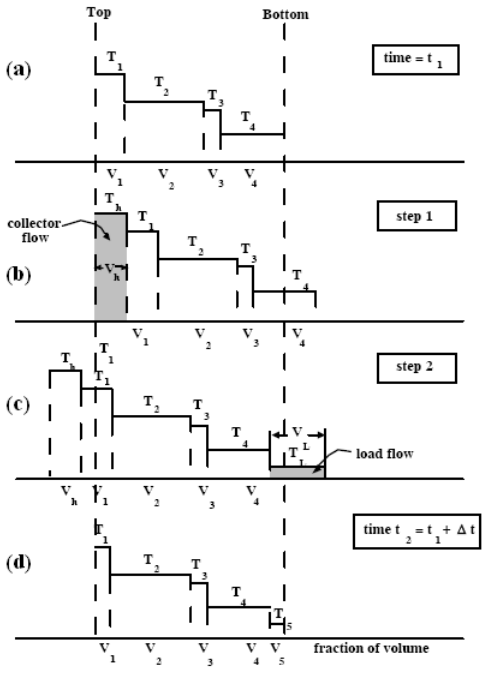

Figure 2.3: Plug flow model used in TRNSYS Type 38 [10]

The plug flow approach includes several characteristics which are applicable to a liquid desiccant storage model. First, the storage is stratified. Second, the user is afforded flexibility in selecting the desired inflow mode. Third, the number and size of each segment of fluid is allowed to vary. Fourth, the plug flow approach of shifting whole segments of fluid within the tank permits strong density gradients to be maintained by avoiding forced partial mixing of segments (as occurs in multinode models).

Some modifications are necessary to meet the requirements of a liquid desiccant storage model. First, in a liquid desiccant model, stratification is by density as a function of concentration as well as temperature. (The standard plug flow model equates density stratification with thermal stratification.) Second, a liquid desiccant model requires that the user be able to define outlet flow rates for variable-volume storage. (The standard plug flow model pairs outlets with inlets). Third, the liquid desiccant model requires an option of multiple inlets and outlets fixed at locations other than the top and bottom of the tank. Fourth, two species are conserved. (The standard plug flow model conserves only one species.) Fifth, the energy balance accounts for chemical potential energy. (The standard plug flow model conserves sensible energy only in the energy balance.) Sixth, the storage is variable-volume. (The standard plug flow model is for constant-volume storage.) Since the storage in a plug flow model is composed of variable-volume segments, it is feasible to modify the plug flow approach to construct a variable-volume tank. Seventh, the density is variable. (The standard plug flow model is for fixed-density storage.) The changes in fluid density will be acceptable once the new model has been adapted to allow for variable-volume storage.

2.2 System Models

Previous work on modeling systems with liquid desiccant storage have lacked stratified storage tanks. N'Tsoukpoe et al. modeled a desiccant-based storage system using two well-mixed tanks [11]. Woods et al. modeled a desiccant-based storage system using a single tank with water separated from a well-mixed desiccant solution [12]. For comparison of annual performance predictions, the stratified liquid desiccant storage tank model (presented beginning in Chapter 3) can be inserted into a system model similar to the one simulated by Woods et al.

3 Desiccant Storage Model Formulation

This chapter introduces the liquid desiccant storage model (TRNSYS Type 209) and details the applicable governing equations and modeling rules. Section 3.1 introduces the storage model. First, it provides an overview of the tank's initial and boundary conditions. Second, it outlines the two operating modes for inflow. Third, it introduces plug flow rules. Section 3.2 details the plug flow conservation equations for continuity, species mass, and energy. Section 3.3 details plug flow rules for fluid inflow and outflow, auxiliary heat input, and segment mixing. Finally, Section 3.4 provides an overview of the parameters to be defined by the user.

3.1 Introduction to the Desiccant Storage Model

This model is designed to simulate the behavior of a density-stratified, variable-volume, liquid desiccant storage tank with a maximum volume V_{max}. The storage is composed of M fully mixed, variable-volume segments, as shown Figure 3.1(a). These segments are stratified by density (ρ). The density of each segment i is a function of the temperature and salt concentration ($\rho_i = \rho(T_i, S_i)$) of the fluid. When the tank is not completely filled with fluid (*i.e.* $\sum_{i=1}^{M} V_i < V_{max}$), air at atmospheric pressure fills the remaining volume.

The tank includes N fixed positions at which auxiliary heat ($\dot{Q}_{aux}$) may be added to the tank and at which fluid may enter or exit the tank (at mass flow rates $\dot{m}_{in}$ and $\dot{m}_{out}$, respectively), as shown in Figure 3.1(b). The storage interacts with the surroundings (at temperature T_{amb}) through thermal losses from the top, bottom, and sides of the tank ($\dot{Q}_{loss,t}$, $\dot{Q}_{loss,b}$, and $\dot{Q}_{loss,s}$, respectively).

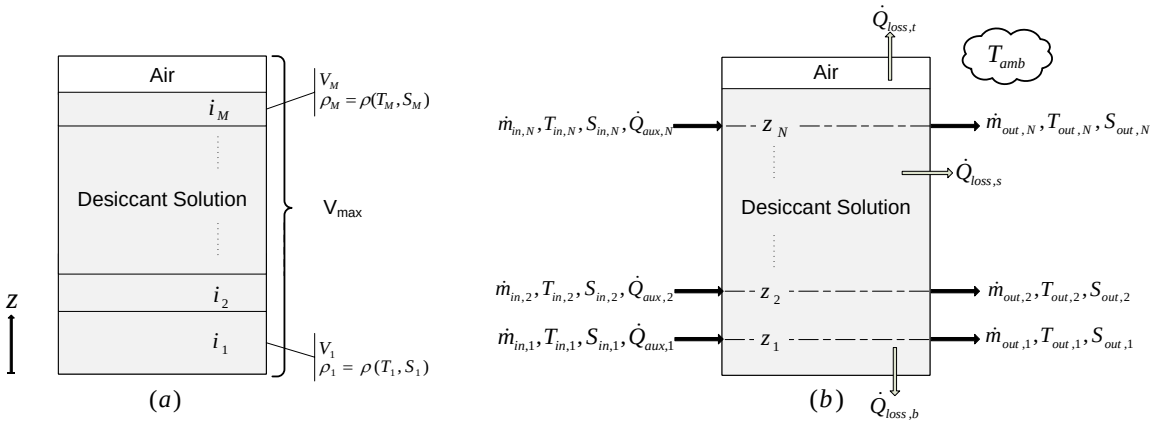

Figure 3.1: Schematics of a density-stratified, variable-volume, liquid desiccant storage tank: (a) initial conditions for M fully-mixed segments; (b) boundary conditions for thermal losses and for N inlet, outlet, and auxiliary positions

There are two modes of operation, as described in Table 3.1. In Mode 1, the tank has fixed inlet positions (as depicted in Figure 3.1(b)). For each inlet flow, a new segment of fluid is created (not shown in the figure). Resulting density inversions are corrected by mixing appropriate segments above or below the inlets. Mode 2 is an idealized tank in which new segments are inserted at the level which produces no density inversions. In both modes, the locations of the outlets and auxiliary heaters are fixed (as depicted in Figure 3.1(b)).

Table 3.1: Operational modes of the liquid desiccant storage

Mode	Description
1	Mode 1 is a liquid desiccant tank with fixed inlet positions. The user specifies the position (height) of inlets, outlets, and auxiliary heaters.
2	Mode 2 is a perfectly stratified liquid desiccant tank. Each inlet stream enters the tank at the location of neutral buoyancy. The user specifies the position (height) of outlets and auxiliary heaters only.

All inlet flows create new segments. The volume of a new segment is governed by the rate and density of the inlet flow and by the time step. (The creation of new segments via inflow is described further in Section 3.3.1.) An existing segment's volume increases through complete mixing with adjacent segments at the end of each time step. The two adjacent segments of fluid with the smallest density difference are combined when the number of segments exceeds the allowed maximum (a number defined by the user). Two adjacent segments are also combined if they are not stably stratified. An existing segment's volume decreases when fluid is removed from the storage tank through fixed outlets. The volume decrease is governed by the rate and density of the outlet flow, by the time step, and by the position of the outlet. When two or more segments contribute to an outlet flow, the properties of the outflow are determined by mixing the contributing fluids. The governing equations for mixing are provided in Section 3.2.4.

The model considers temperature- and salt concentration-dependent fluid properties for calcium chloride ($CaCl_2$) or lithium chloride (LiCl) solutions. These properties are calculated using curve fits from Conde [4]. These curve fit equations are included in Appendix A.

3.2 Plug Flow Governing Equations for Desiccant Storage

3.2.1 Introduction

The model solves the one-dimensional continuity, species, and energy conservation equations using a two-step approach similar to the one introduced in Section 2.1.3. Following the overview of the two-step approach in Section 3.2.1, details for the two steps appear in Sections 3.2.2 and 3.2.3.

The standard plug flow tank model (*e.g.* TRNSYS Type 38 [10]) uses bookkeeping methods to dictate the position, size, and properties of each segment. Fluid movement is characterized as plug flow, with individual segments ascending and descending within the

tank [6]. The model begins each time step with an approximate analytical solution to the differential governing equation for energy. (An approximate solution of the same form is also applied to the differential governing equation for species in the desiccant storage model.) The analytical approximation requires that the mass of each segment be considered constant for the duration of the time step. In order to preserve constant mass, the species and energy conservation equations are solved using a closed system analysis. This is the first step of the solution. Since inflow and outflow make the tank an open system, the solution requires a second step to allow for changes in segment mass during the course of the time step. Thus, the closed system analysis is followed by an open system analysis, which employs an analytical plug flow algorithm to track fluid movement for the duration of the time step.

First, when the tank is considered a closed system, the species and energy conservation equations are solved for each fluid segment. The species conservation equation incorporates terms for mass diffusion between neighboring segments. The energy conservation equation incorporates terms for conduction between neighboring segments, for thermal losses to the surroundings, and for the generation of sensible energy from chemical potential energy. (The energy transfer accompanying mass diffusion between neighboring segments is ultimately neglected, as justified by the order of magnitude analysis in Appendix B.4.) These species and energy transfer mechanisms are evaluated for the duration of the time step using the storage profile obtained from the solution to the previous time step. The species and energy conservation equations for each fluid segment in a closed-system storage model appear in Section 3.2.2. The concentration and temperature of each fluid segment in the storage profile are updated after the closed system analysis.

Second, the tank is considered an open system. Fluid inflow, fluid outflow, and auxiliary heat input are incorporated into the open system analysis. Fluid movement

caused by inflow and outflow is modeled as plug flow. The equations used to verify species and energy conservation throughout the storage appear in Section 3.2.3. (Detailed rules for plug flow appear later, in Section 3.3.)

The desiccant storage governing equations presented throughout Section 3.2 have been adapted from those used in the standard plug flow model [10]. First, two species are conserved. Second, mass diffusion is modeled between segments. Third, chemical energy is included in the energy balance. Fourth, properties such as density and specific heat are not constant.

3.2.2 Governing Equations for the Closed System Analysis

The tank is considered to be a closed system during the first step of the evaluation of an arbitrary time step n. The fluid contained in the tank is discretized into M fully mixed segments. The M segments (noted as $i = 1 \ldots M$) are stably stratified by density, which is a function of temperature and salt concentration ($\rho_i = \rho(T_i, S_i)$). For the closed system, each of the three conservation equations is applied to each of the M segments. (No segments are added, removed, or combined during the course of the closed system analysis.)

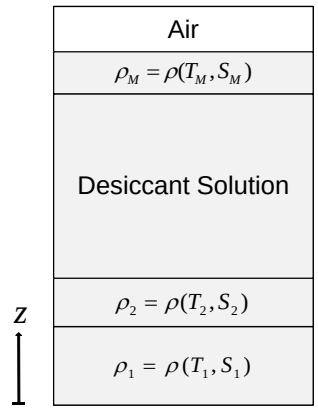

Figure 3.2: Density-stratified tank as a closed system for the closed system analysis

3.2.2.1 Mass Conservation

Each arbitrary segment i contains an amount of mass of the desiccant solution (m_i). For a closed system, the solution mass balance for segment i is

$$\frac{dm_i}{dt} = 0 \tag{3.1}$$

3.2.2.2 Salt Species Conservation

Fick's Law for binary diffusion [13] gives

$$\dot{m}_{salt} = -\rho \mathcal{D} A_c \frac{dS}{dz} \tag{3.2}$$

where ρ is the density, $\mathcal{D}$ is the binary mass diffusion coefficient, A_c is the cross-sectional area across which the salt mass flows, S is the concentration of the salt, z is the vertical coordinate (the direction of flow), and $\dot{m}_{salt}$ is the mass flow rate of the salt species.

The salt mass in an arbitrary segment i is the product of the solution mass (m_i) and the salt concentration (S_i) of that solution. The salt species balance for segment i is

$$\frac{d(m_i S_i)}{dt} = \dot{m}_{salt,in} - \dot{m}_{salt,out} \tag{3.3}$$

For an arbitrary segment i (Figure 3.3), a finite difference approximation of Equation (3.2) gives

$$\begin{aligned} &\dot{m}_{salt,in} - \dot{m}_{salt,out} \\ &\qquad \cong (\overline{\rho\mathcal{D}})_{i+1,i} A_c \frac{S_{i+1} - S_i}{z_{i+1} - z_i} - (\overline{\rho\mathcal{D}})_{i-1,i} A_c \frac{S_i - S_{i-1}}{z_i - z_{i-1}} \end{aligned} \tag{3.4}$$

where $(\overline{\rho\mathcal{D}})_{i+1,i}$ and $(\overline{\rho\mathcal{D}})_{i-1,i}$ are used as the property values at each interface. They are the mass-weighted averages of the property values between segments $i+1$ and i and between segments $i-1$ and i, respectively.

Substituting Equations (3.1) and (3.4) into Equation (3.3) gives

$$m_i \frac{dS_i}{dt} \cong (\overline{\rho \mathcal{D}})_{i+1,i} A_c \frac{S_{i+1} - S_i}{z_{i+1} - z_i} - (\overline{\rho \mathcal{D}})_{i-1,i} A_c \frac{S_i - S_{i-1}}{z_i - z_{i-1}} \tag{3.5}$$

The approximate salt species balance (Equation (3.5)) accounts for the change in salt mass within segment i ($m_i \frac{dS_i}{dt}$ in Figure 3.3, left hand side of Equation (3.5)). Salt mass may be added or removed from segment i through diffusion with neighboring segments $i+1$ and $i-1$ ($\dot{m}_{salt,i+1}$ and $\dot{m}_{salt,i-1}$ in Figure 3.3, respectively; first and second terms on the right hand side of Equation (3.5), respectively).

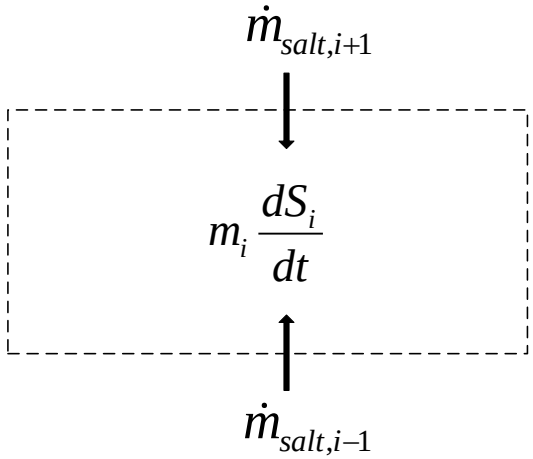

Figure 3.3: Salt species balance for an arbitrary segment i

Equation (3.5) is placed in the form

$$\frac{dS_i}{dt} \cong a_S S_i + b_S \tag{3.6}$$

where the coefficients

$$a_S = -\frac{\frac{(\overline{\rho \mathcal{D}})_{i+1,i} A_c}{z_{i+1} - z_i} + \frac{(\overline{\rho \mathcal{D}})_{i-1,i} A_c}{z_i - z_{i-1}}}{m_i} \tag{3.7}$$

and

$$b_S = \frac{\frac{(\overline{\rho\mathcal{D}})_{i+1,i} A_c S_{i+1}}{z_{i+1} - z_i} + \frac{(\overline{\rho\mathcal{D}})_{i-1,i} A_c S_{i-1}}{z_i - z_{i-1}}}{m_i} \tag{3.8}$$

are functions of time through their dependence on salt concentrations and on other fluid properties (*i.e.* S_{i+1}, S_{i-1}, $(\overline{\rho\mathcal{D}})_{i+1,i}$, and $(\overline{\rho\mathcal{D}})_{i-1,i}$ are time-dependent). Equation (3.6) can be solved analytically if Equations (3.7) and (3.8) are steady state. First, the salt concentration of segments $i+1$ and $i-1$ (and, for property calculations, of segment i as well) is considered to be equal to its average value over time step n for the duration of time step n (*i.e* $S_{i+1} = \bar{S}_{i+1}^n$, $S_{i-1} = \bar{S}_{i-1}^n$, and $S_i = \bar{S}_i^n$). Density and the binary mass diffusion coefficient are calculated using the temperature solution of the previous time step (*i.e.* $\rho(T,S) \cong \rho(T^{n-1}, \bar{S}^n)$ and $\mathcal{D}(T,S) \cong \mathcal{D}(T^{n-1}, \bar{S}^n)$). Equation (3.6) becomes

$$\frac{dS_i}{dt} \cong \bar{a}_S S_i + \bar{b}_S \tag{3.9}$$

where the coefficients are

$$\bar{a}_S = -\frac{\frac{(\overline{\rho\mathcal{D}})_{i+1,i} A_c}{z_{i+1} - z_i} + \frac{(\overline{\rho\mathcal{D}})_{i-1,i} A_c}{z_i - z_{i-1}}}{m_i} \tag{3.10}$$

and

$$\bar{b}_S = \frac{\frac{(\overline{\rho\mathcal{D}})_{i+1,i} A_c \bar{S}_{i+1}^n}{z_{i+1} - z_i} + \frac{(\overline{\rho\mathcal{D}})_{i-1,i} A_c \bar{S}_{i-1}^n}{z_i - z_{i-1}}}{m_i} \tag{3.11}$$

Equation (3.9) is solved analytically for the final salt concentration at the end of time step n

$$S_i^n = \left(S_i^{n-1} + \frac{\bar{b}_S}{\bar{a}_S} \right) e^{\bar{a}_S \Delta t} - \frac{\bar{b}_S}{\bar{a}_S} \tag{3.12}$$

The average salt concentration over time step n is

$$\bar{S}_i^n = \left(\frac{S_i^{n-1} + \frac{\bar{b}_S}{\bar{a}_S}}{\bar{a}_S \Delta t} \right) \left(e^{\bar{a}_S \Delta t} - 1 \right) - \frac{\bar{b}_S}{\bar{a}_S} \tag{3.13}$$

where S_i^{n-1} is the solution to the previous time step (*i.e.* the initial concentration of segment i) and where the length of the time step is

$$\Delta t = t^n - t^{n-1} \tag{3.14}$$

The derivations of Equations (3.12) and (3.13) from Equation (3.9) are included in Appendix B.1.

3.2.2.3 Energy Conservation

The rate of energy transfer due to thermal conduction is given by

$$\dot{Q}_{cond} = -kA_c \frac{dT}{dz} \tag{3.15}$$

where k is the thermal conductivity, A_c is the cross-sectional area across which the heat flows, T is the temperature of the solution, and z is the vertical coordinate (the direction of flow).

The rate of energy transfer due to diffusive mixing (with the total mass of the control volume remaining constant) is given by

$$\dot{Q}_{diff} = \dot{m} \left(\left(c_p T \right)_{in} - \left(c_p T \right)_{out} \right) \tag{3.16}$$

where $\dot{m}$ is the rate at which the species diffuse across the boundary of the control volume, $\left(c_p T \right)_{in}$ is the specific heat and temperature of the species entering the control volume, and $\left(c_p T \right)_{out}$ is the specific heat and temperature of the species leaving the control volume.

The rate of energy transfer due to thermal losses is given by

$$\dot{Q}_{loss} = -U_L A_w (T - T_{amb}) \tag{3.17}$$

where U_L is the overall loss coefficient, A_w is the wall surface area across which the heat flows, T is the temperature of the solution, and T_{amb} is the ambient temperature.

The internal energy in an arbitrary segment i is the product of the solution mass (m_i), the specific heat (c_{p_i}) and the temperature (T_i, °C) of that solution. The energy balance for segment i is

$$\frac{d}{dt}(m_i c_{p_i} T_i) = \dot{Q}_{in} - \dot{Q}_{out} + \dot{Q}_{gen} \tag{3.18}$$

For an arbitrary segment i (Figure 3.4), Equation (3.18) can be expressed as

$$\begin{aligned}\frac{d}{dt}(m_i c_{p_i} T_i) = {} & \dot{Q}_{cond,i+1\to i} + \dot{Q}_{cond,i-1\to i} + \dot{Q}_{diff,i+1\to i} \\ & + \dot{Q}_{diff,i-1\to i} - \dot{Q}_{loss,i} + \dot{Q}_{gen,i}\end{aligned} \tag{3.19}$$

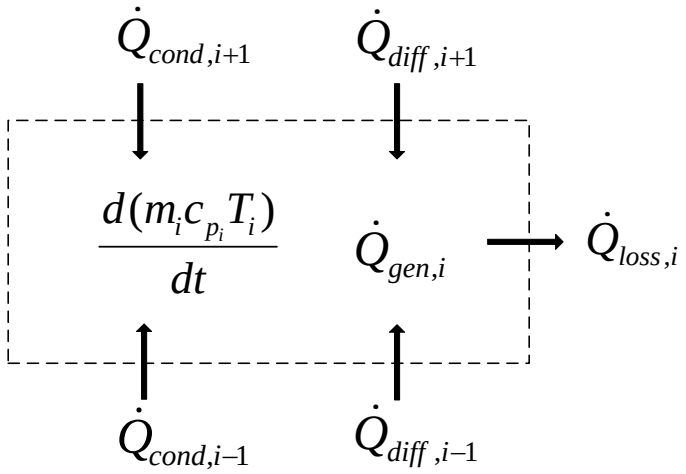

Figure 3.4: Energy balance for an arbitrary segment i

For the same arbitrary segment i, a finite difference approximation of Equation (3.19) gives

$$\dot{Q}_{cond,i+1\to i} + \dot{Q}_{cond,i-1\to i} \cong \bar{k}_{i+1,i} A_c \frac{T_{i+1} - T_i}{z_{i+1} - z_i} - \bar{k}_{i-1,i} A_c \frac{T_i - T_{i-1}}{z_i - z_{i-1}} \tag{3.20}$$

where $\bar{k}_{i+1,i}$ and $\bar{k}_{i-1,i}$ are used as the thermal conductivity values at the interface. They are the mass-weighted averages of the property values between segments $i+1$ and i and between segments $i-1$ and i, respectively.

The finite difference approximation of Equation (3.19) also gives

$$\begin{aligned} \dot{Q}_{diff,i+1\to i} + \dot{Q}_{diff,i-1\to i} &\cong \dot{m}_{i+1\to i}\left(c_{p_{i+1}} T_{i+1} - c_{p_i} T_i\right) \\ &+ \dot{m}_{i-1\to i}\left(c_{p_{i-1}} T_{i-1} - c_{p_i} T_i\right) \end{aligned} \tag{3.21}$$

where, from Equation (3.4),

$$\dot{m}_{i+1\leftrightarrow i} \cong (\overline{\rho\mathcal{D}})_{i+1,i} A_c \frac{|S_{i+1} - S_i|}{z_{i+1} - z_i} \tag{3.22}$$

and

$$\dot{m}_{i-1\leftrightarrow i} \cong (\overline{\rho\mathcal{D}})_{i-1,i} A_c \frac{|S_i - S_{i-1}|}{z_i - z_{i-1}} \tag{3.23}$$

Substitutions of Equations (3.22) and (3.23) into Equation (3.21) gives

$$\begin{aligned} \dot{Q}_{diff,i+1\to i} + \dot{Q}_{diff,i-1\to i} &\cong \left((\overline{\rho\mathcal{D}})_{i+1,i} A_c \frac{|S_{i+1} - S_i|}{z_{i+1} - z_i}\right)\left(c_{p_{i+1}} T_{i+1} - c_{p_i} T_i\right) \\ &+ \left((\overline{\rho\mathcal{D}})_{i-1,i} A_c \frac{|S_i - S_{i-1}|}{z_i - z_{i-1}}\right)\left(c_{p_{i-1}} T_{i-1} - c_{p_i} T_i\right) \end{aligned} \tag{3.24}$$

Additionally, for segment i

$$\dot{Q}_{loss,i} = U_L A_w (T_i - T_{amb}) \tag{3.25}$$

and

$$\dot{Q}_{gen,i} = -\frac{dQ_{ch,i}}{dt} \tag{3.26}$$

where $Q_{ch,i}$ is the chemical potential energy contained in the solution of mass m_i at temperature T_i (K) and concentration S_i. From Equation (B.33) (derived in Appendix B.3),

$$\dot{Q}_{gen,i} = -m_i \left(\left(C_5 + \frac{C_6}{T_{crit}} T_i \right) f_2(S_i) \left(\frac{S_i^n - S_i^{n-1}}{\Delta t} \right) + \left(\frac{C_6}{T_{crit}} \right) f_3(S_i) \frac{dT_i}{dt} \right) \tag{3.27}$$

where C_5 and C_6 are constants listed in Table A.1 and $f_2(S_i)$ and $f_3(S_i)$ are given by Equations (B.31) and (B.32), respectively.

By using Equation (3.1) and the approximation

$$\frac{dc_{p_i}}{dt} = \frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t} \tag{3.28}$$

the internal storage (left hand side of Equation (3.19)) can be expressed as

$$\frac{d}{dt}\left(m_i c_{p_i} T_i\right) = m_i c_{p_i} \frac{dT_i}{dt} + m_i \left(\frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t} \right) T_i \tag{3.29}$$

Through substitutions of Equations (3.20), (3.24), (3.25), (3.27), and (3.29), Equation (3.19) becomes

$$
\begin{aligned}
m_i c_{p_i} \frac{dT_i}{dt} + m_i \left(\frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t} \right) T_i & \\
\cong \bar{k}_{i+1,i} A_c \frac{T_{i+1} - T_i}{z_{i+1} - z_i} - \bar{k}_{i-1,i} A_c \frac{T_i - T_{i-1}}{z_i - z_{i-1}} & \\
+ \left((\overline{\rho \mathcal{D}})_{i+1,i} A_c \frac{|S_{i+1} - S_i|}{z_{i+1} - z_i} \right) \left(c_{p_{i+1}} T_{i+1} - c_{p_i} T_i \right) & \\
+ \left((\overline{\rho \mathcal{D}})_{i-1,i} A_c \frac{|S_i - S_{i-1}|}{z_i - z_{i-1}} \right) \left(c_{p_{i-1}} T_{i-1} - c_{p_i} T_i \right) & \\
- \left(U_L A_w (T_i - T_{amb}) \right) & \\
- m_i \left(\left(C_5 + \frac{C_6}{T_{crit}} T_i \right) f_2(S_i) \left(\frac{S_i^n - S_i^{n-1}}{\Delta t} \right) \right. & \\
\left. + \left(\frac{C_6}{T_{crit}} \right) f_3(S_i) \frac{dT_i}{dt} \right) &
\end{aligned}
\tag{3.30}
$$

Dry wall sections in the empty (air) region are included in the loss term for the top fluid segment (i_M). (Both wet and dry overall loss coefficients are specified by the user.)

An order of magnitude analysis (see Appendix B.4) shows that the energy transported by mass diffusion is negligible ($\dot{Q}_{diff} \ll \dot{Q}_{cond}$). Thus, Equation (3.30) can be simplified as

$$
\begin{aligned}
m_i c_{p_i} \frac{dT_i}{dt} + m_i \left(\frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t} \right) T_i & \\
\cong \bar{k}_{i+1,i} A_c \frac{T_{i+1} - T_i}{z_{i+1} - z_i} - \bar{k}_{i-1,i} A_c \frac{T_i - T_{i-1}}{z_i - z_{i-1}} & \\
- \left(U_L A_w (T_i - T_{amb}) \right) & \\
- m_i \left(\left(C_5 + \frac{C_6}{T_{crit}} T_i \right) f_2(S_i) \left(\frac{S_i^n - S_i^{n-1}}{\Delta t} \right) \right. & \\
\left. + \left(\frac{C_6}{T_{crit}} \right) f_3(S_i) \frac{dT_i}{dt} \right) &
\end{aligned}
\tag{3.31}
$$

Equation (3.31) is placed in the form

$$\frac{dT_i}{dt} \cong a_T T_i + b_T \tag{3.32}$$

where the coefficients

$$a_T = -\frac{m_i\left(\frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t}\right) + \frac{\bar{k}_{i+1,i}A_c}{z_{i+1}-z_i} + \frac{\bar{k}_{i-1,i}A_c}{z_i - z_{i-1}} + U_L A_w}{m_i c_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(S_i)} - \frac{m_i \frac{C_6}{T_{crit}} f_2(S_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right)}{m_i c_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(S_i)} \tag{3.33}$$

and

$$b_T = \frac{\frac{\bar{k}_{i+1,i}A_c T_{i+1}}{z_{i+1}-z_i} + \frac{\bar{k}_{i-1,i}A_c T_{i-1}}{z_i - z_{i-1}} + U_L A_w T_{amb}}{m_i c_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(S_i)} - \frac{m_i C_5 f_2(S_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right)}{m_i c_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(S_i)} \tag{3.34}$$

are functions of time through their dependence on temperatures, salt concentrations, and thermal conductivity (*i.e.* T_{i+1}, T_{i-1}, S_i, $\bar{k}_{i+1,i}$, and $\bar{k}_{i-1,i}$ are time-dependent). Equation (3.32) can be solved analytically if Equations (3.33) and (3.34) are steady state. First, the temperature of segments $i+1$ and $i-1$ (and, for property calculations, of segment i as well) is considered to be equal to its average value over time step n for the duration of time step n (*i.e* $T_{i+1} = \bar{T}_{i+1}^n$, $T_{i-1} = \bar{T}_{i-1}^n$, and $T_i = \bar{T}_i^n$). Similarly, the salt concentration of segment i is considered to be equal to its average value over time step n for the duration of time step n (*i.e* $S_i = \bar{S}_i^n$). Thermal conductivity is calculated using these average temperature and concentration properties (*i.e.* $k(T,S) \cong k(\bar{T}^n, \bar{S}^n)$). Equation (3.32) becomes

$$\frac{dT_i}{dt} \cong \bar{a}_T T_i + \bar{b}_T \tag{3.35}$$

where the coefficients are

$$\begin{aligned} \bar{a}_T = &- \frac{m_i\left(\frac{c_{p_i}^n - c_{p_i}^{n-1}}{\Delta t}\right) + \frac{\bar{k}_{i+1,i}A_c}{z_{i+1}-z_i} + \frac{\bar{k}_{i-1,i}A_c}{z_i - z_{i-1}} + U_L A_w}{m_i \bar{c}_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(\bar{S}_i)} \\ &- \frac{m_i \frac{C_6}{T_{crit}} f_2(\bar{S}_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right)}{m_i \bar{c}_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(\bar{S}_i)} \end{aligned} \tag{3.36}$$

and

$$\begin{aligned} \bar{b}_T = &\frac{\frac{\bar{k}_{i+1,i}A_c\bar{T}_{i+1}}{z_{i+1}-z_i} + \frac{\bar{k}_{i-1,i}A_c\bar{T}_{i-1}}{z_i - z_{i-1}} + U_L A_w T_{amb}}{m_i \bar{c}_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(\bar{S}_i)} \\ &- \frac{m_i C_5 f_2(\bar{S}_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right)}{m_i \bar{c}_{p_i} + m_i\left(\frac{C_6}{T_{crit}}\right) f_3(\bar{S}_i)} \end{aligned} \tag{3.37}$$

Equation (3.35) is solved analytically for the final temperature at the end of time step n

$$T_i^n = \left(T_i^{n-1} + \frac{\bar{b}_T}{\bar{a}_T}\right) e^{\bar{a}_T \Delta t} - \frac{\bar{b}_T}{\bar{a}_T} \tag{3.38}$$

The average temperature over time step n is

$$\bar{T}_i^n = \left(\frac{T_i^{n-1} + \frac{\bar{b}_T}{\bar{a}_T}}{\bar{a}_T \Delta t}\right)\left(e^{\bar{a}_T \Delta t} - 1\right) - \frac{\bar{b}_T}{\bar{a}_T} \tag{3.39}$$

where T_i^{n-1} is the solution to the previous time step (*i.e.* the initial temperature of segment i) and where the length of the time step is

$$\Delta t = t^n - t^{n-1} \tag{3.40}$$

The derivations of Equations (3.38) and (3.39) from Equation (3.35) are included in Appendix B.1.

3.2.3 Governing Equations for the Open System Analysis

The tank is considered to be an open system in the second step of the evaluation of an arbitrary time step n. After the previous time step ($t = t^{n-1}$), the storage is composed of M^{n-1} fully mixed segments (Figure 3.5(a)). During the course of the time step ($t^{n-1} < t < t^{n}$), mass flow enters and exits the tank and auxiliary heat is added to the tank at any of N positions (Figure 3.5(b)). A plug flow model is used to simulate the storage changes that result from mass flow entering or exiting the tank at each position (z_1 through z_N). All entering mass flow rates ($\dot{m}_{in}$), exiting mass flow rates ($\dot{m}_{out}$), and auxiliary heat rates ($\dot{Q}_{aux}$) are constant for the duration of the time step. After the completion of the time step ($t = t^{n}$), the updated storage is composed of M^{n} fully mixed segments (Figure 3.5(c)).

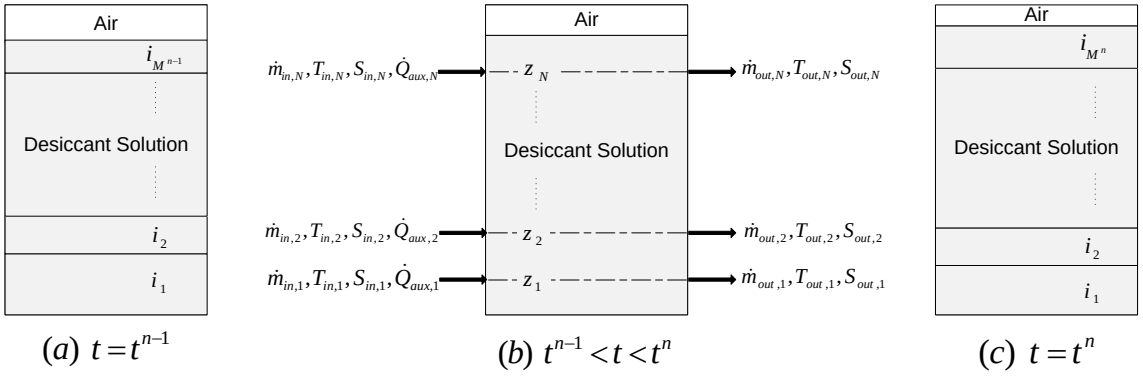

Figure 3.5: Storage tank fluid profile (a) before, (b) during, and (c) after accounting for mass flow and auxiliary heating during time step n

3.2.3.1 Mass Conservation

After the open system plug flow procedure (detailed in Section 3.3), Equation (3.41) is used to verify that the net inflow or net outflow of desiccant solution through the

N positions over the course of the time step is accurately reflected in the updated sum of stored solution. The net change in the stored solution mass is equal to the difference between the total solution mass of segments i_1 through i_{M^n} at time $t = t^n$ (first sum on the left hand side) and the total solution mass of segments i_1 through $i_{M^{n-1}}$ at time $t = t^{n-1}$ (second sum on the left hand side). This difference is equal to the difference between the total solution mass which entered the storage at positions z_1 through z_N over the course of time step n (first sum on the right hand side) and the total solution mass which exited the storage at positions z_1 through z_N over the course of time step n (second sum on the right hand side).

$$\sum_{i=1}^{M^n} m_i^n - \sum_{i=1}^{M^{n-1}} m_i^{n-1} = \sum_{z=1}^{N} \dot{m}_{in,z} - \sum_{z=1}^{N} \dot{m}_{out,z} \tag{3.41}$$

3.2.3.2 Salt Species Conservation

Equation (3.42) is used to verify that the net inflow or net outflow of salt species through the N positions over the course of the time step is reflected in the updated sum of stored salt species. (Salt species mass is equal to the product of the solution mass, m_i, and the solution concentration, S_i.) The net change in stored salt is equal to the difference between the total salt mass of segments i_1 through i_{M^n} at time $t = t^n$ (first sum on the left hand side) and the total salt mass of segments i_1 through $i_{M^{n-1}}$ at time $t = t^{n-1}$ (second sum on the left hand side). This difference is equal to the difference between the total salt mass which entered the storage at positions z_1 through z_N over the course of time step n (first sum on the right hand side) and the total salt mass which exited the storage at positions z_1 through z_N over the course of time step n (second sum on the right hand side).

$$\sum_{i=1}^{M^n} m_i^n S_i^n - \sum_{i=1}^{M^{n-1}} m_i^{n-1} S_i^{n-1} = \sum_{z=1}^{N} \dot{m}_{in,z} S_{in,z} - \sum_{z=1}^{N} \dot{m}_{out,z} S_{out,z} \tag{3.42}$$

3.2.3.3 Energy Conservation

Equation (3.43) is used to verify that the net gain or net loss of internal energy through the N positions over the course of the time step is reflected in the updated sum of stored internal energy. The net change in the stored internal energy is equal to the difference between the total internal energy in segments i_1 through i_{M^n} at time $t = t^n$ (first sum on the left hand side) and the total internal energy in segments i_1 through $i_{M^{n-1}}$ at time $t = t^{n-1}$ (second sum on the left hand side). This difference is equal to the auxiliary heat added at positions z_1 through z_N over the course of the time step (first sum on the right hand side), minus the difference between the total chemical potential energy in segments i_1 through i_{M^n} at time $t = t^n$ (second sum on the right hand side) and the total chemical potential energy in segments i_1 through $i_{M^{n-1}}$ at time $t = t^{n-1}$ (third sum on the right hand side), minus the heat lost by segments i_1 through $i_{M^{n-1}}$ over the course

$$\begin{aligned}
\sum_{i=1}^{M^n} m_i^n c_{p_i}^n T_i^n \; - \; & \sum_{i=1}^{M^{n-1}} m_i^{n-1} c_{p_i}^{n-1} T_i^{n-1} \\
& = \sum_{z=1}^{N} Q_{aux,z} - \sum_{i=1}^{M^n} \left(m_i S_i g_1(T_i) f_1(S_i) \right) \\
& + \sum_{i=1}^{M^{n-1}} \left(m_i^{n-1} S_i^{n-1} g_1\left(T_i^{n-1}\right) f_1\left(S_i^{n-1}\right) \right) \\
& - \sum_{i=1}^{M^{n-1}} (U_L A_w)_i (\bar{T}_i - T_{env}) \\
& + \sum_{z=1}^{N} \Big(\dot{m}_{in,z} c_{p_{in,z}} T_{in,z} \\
& + m_{in,z} S_{in,z} g_1\left(T_{in,z}\right) f_1\left(S_{in,z}\right) \Big) \Delta t \\
& - \sum_{z=1}^{N} \Big(\dot{m}_{out,z} c_{p_{out,z}} T_{out,z} \\
& + m_{out,z} S_{out,z} g_1\left(T_{out,z}\right) f_1\left(S_{out,z}\right) \Big) \Delta t
\end{aligned} \tag{3.43}$$

of the time step (fourth sum on the right hand side, as calculated during the closed system analysis), plus the difference between the combined sensible and chemical energy which entered via mass flow at positions z_1 through z_N over the course of the time step (fifth sum on the right hand side) and the combined sensible and chemical energy which exited via mass flow at positions z_1 through z_N over the course of the time step (sixth sum on the right hand side). (The chemical potential energy is defined using Equations (B.19), (B.24), and (B.25).)

3.2.4 Governing Equations for Segment Mixing

The total number of segments is reduced at the end of the time step through the complete mixing of two adjacent segments. (Mixing rules are detailed in Sections 3.3.4 and 3.3.5.) When two segments, i and $i+1$, are mixed, the conservation laws for mass, species, and energy govern the properties of the new segment, i,new.

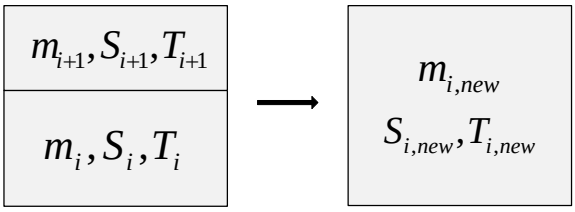

Figure 3.6. Mixing of segments *i* and *i+1* to form segment *i,new*

3.2.4.1 Solution Mass Conservation

When neighboring segments i and $i+1$ mix, the mass of the newly formed segment ($m_{i,new}$) is determined by a solution mass balance.

$$m_{i,new} = m_i + m_{i+1} \tag{3.44}$$

3.2.4.2 Salt Species Mass Conservation

When neighboring segments i and $i+1$ mix, the salt concentration of the newly formed segment ($S_{i,new}$) is determined by a salt species mass balance.

$$m_{i,new}S_{i,new} = m_i S_i + m_{i+1}S_{i+1} \tag{3.45}$$

3.2.4.3 Energy Conservation

When neighboring segments i and $i+1$ mix, the temperature and the heat capacity of the combined segment ($T_{i,new}$ and $c_{p_{i,new}}$, respectively) are determined by an energy balance. Mixing is assumed to be adiabatic. The sum of the internal sensible energy (first term on the left hand side of Equation (3.46)) and chemical potential energy (second term on the left hand side of Equation (3.46)) in the newly formed segment is equal to the total energy of the contributing segments (the terms on the right hand side of Equation (3.46)). The only independent unknown is $T_{i,new}$. The term $c_{p_{i,new}}$ is a function of $T_{i,new}$ and $S_{i,new}$; and $m_{i,new}$ and $S_{i,new}$ are determined by Equations (3.44) and (3.45). (The chemical potential energy is defined using Equations (B.19), (B.24), and (B.25).)

$$\begin{aligned} m_{i,new}c_{p_{i,new}}T_{i,new} + m_{i,new}S_{i,new}g_1(T_{i,new})f_1(S_{i,new}) \\ = \left(m_i c_{p_i} T_i + m_i S_i g_1(T_i) f_1(S_i)\right) \\ + \left(m_{i+1}c_{p_{i+1}}T_{i+1} + m_{i+1}S_{i+1}g_1(T_{i+1})f_1(S_{i+1})\right) \end{aligned} \tag{3.46}$$

3.3 Plug Flow Rules for Desiccant Storage

Section 2.1.3 and Figure 2.3 introduced standard plug flow rules for a thermally-stratified, constant-volume, sensible energy storage tank with fixed outlets at the top and bottom. Section 3.3 and the accompanying figures (Figure 3.7-Figure 3.10) introduce the modified plug flow rules for a density-stratified, variable-volume, sensible- and chemical-storage tank with fixed outlets at any height. Modified plug flow rules differ from standard plug flow rules in several respects. First, the storage is stratified by density as a function of both concentration and temperature (*i.e.* the tank is density-stratified rather than thermally stratified). Second, the rate of outflow does not need to match the rate of inflow (*i.e.* inlets and outlets are not paired). As a consequence, the total mass of fluid in

the tank does not need to be constant from one time step to the next. Third, the tank includes fixed inlets and outlets at locations other than its top and bottom. The latter two modifications required a number of detailed rules to be added to the plug flow algorithm.

The first plug flow rule pertains to the order in which inflow, outflow, and auxiliary are incorporated into the flow model. Inflow is first inserted into the vertical storage profile. Inflow rules are described in Section 3.3.1. Outflow is then removed from the profile, with any auxiliary heat input added in the process. Outflow rules are described in Section 3.3.2, and auxiliary rules are described in Section 3.3.3. After accounting for inflow, outflow, and auxiliary over the full time step, another rule governs the mixing of unstable storage segments (*i.e.* neighboring segments with density inversions). In the case of unstable density gradients, segments are mixed at the end of the time step. These density inversion rules are described in Section 3.3.4. Finally, if an excessive number of segments exist (*i.e.* the number of segments composing the storage at the end of the time step exceeds the user-specified permissible maximum number of segments), the neighboring segments with the smallest density difference are mixed. Segment mixing rules for excessive segments are described in Section 3.3.5.

The rules presented in Sections 3.3.1-3.3.5 are illustrated with an example storage tank over an arbitrary time step. In this illustration, the tank operates in Mode 1 (*i.e.* each inlet in Figure 3.7 is at a fixed height). Two fluid segments comprise the storage at the beginning of the time step (see Figure 3.7). A maximum of three fluid segments are permitted. The tank includes three positions for inflow, outflow, and auxiliary. In the example time step, one of these positions is used for inflow (see Figure 3.7) and one position is used for outflow (see Figure 3.8).

3.3.1 Inflow

In Mode 1, the tank has fixed inlet positions (for example, z_2 in Figure 3.7). Each inlet flow creates a new segment of fluid. Each new segment is inserted into the initial storage (Figure 3.7(a)) at the location of the inlet position. The base of the new segment aligns with the vertical coordinate of the inlet. (In the case of an inlet located above all fluid segments, the new inflow segment is added above the top storage segment.) Unless an inlet is aligned at the interface between two existing segments, the new inflow segment splits an existing segment into two segments. One segment remains below the inlet. The second segment and any segments above it are shifted upward (*i.e.* in the positive y-direction) in the updated storage profile (see Figure 3.7(b)). Inflows are inserted sequentially from the lowest- to the highest-numbered positions. If there are multiple flows entering the storage in a single time step (not shown), the location for each new segment to be inserted is determined before any of them are inserted (*i.e.* the placement of a new segment is not affected by inflow through lower-numbered inlets). Any mixing that would occur due to the rising fluid level seen at that inlet position is taken into account at a later time in the plug flow algorithm. Likewise, any newly created density inversions are accounted for at a later time. In Mode 2, the tank has no fixed inlets. This mode is an idealized tank in which new segments are inserted at the levels which produce no density inversions.

The inflow rules are illustrated by Figure 3.7. One new segment is created. Flow enters the tank through position z_2 and becomes segment i_2 (Figure 3.7(b)). Segment i_1 (Figure 3.7(a)) is split into i_1 and i_3 (Figure 3.7(b)).

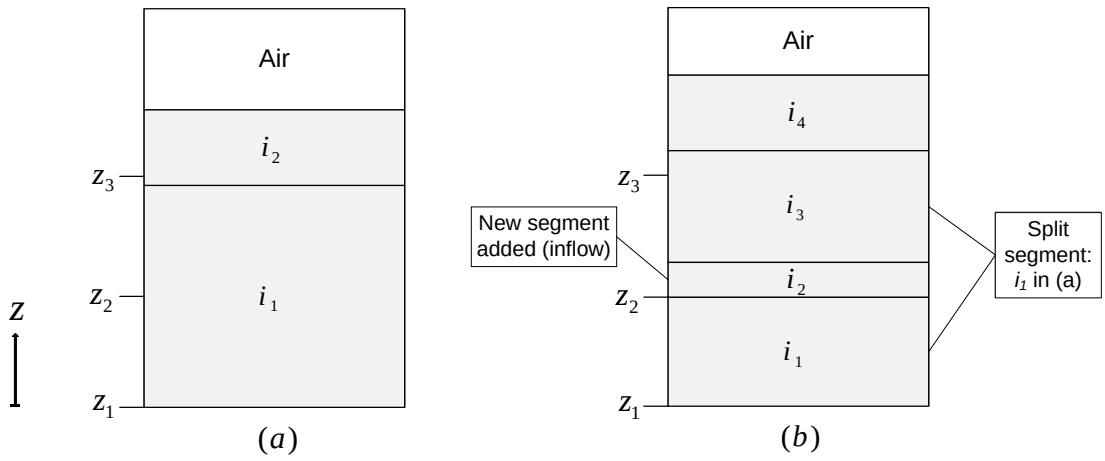

Figure 3.7: Illustration of plug flow rules for inflow Mode 1: (a) The initial storage profile is composed of two segments; (b) Inflow creates a new segment, inserted at z_2, splitting the base segment.

3.3.2 Outflow

The tank has fixed outlet positions (for example, z_3 in Figure 3.8). Each outlet flow removes a volume of fluid from the storage profile. Each volume of outflow is removed from the initial storage profile (Figure 3.8(a)) at the location of the outlet position. The base of the outflow volume (the shaded volume in Figure 3.8(a)) aligns with the vertical coordinate of the outlet (*e.g.* z_3). The outflow may be composed of part or all of one or more storage segments. Outflows are removed from the storage sequentially from the lowest- to the highest-numbered outlet positions.

Inflow and outflow through lower-numbered positions creates fluid level changes during the course of a time step. The fluid level at an outlet is either rising ($\Delta V_z > 0$ due to net inflow below position z), falling ($\Delta V_z < 0$ due to net outflow below position z), or constant throughout the time step ($\Delta V_z = 0$ due to no net inflow or net outflow below position z). Over one time step, this net change in fluid volume through inlets and outlets

below position z is defined recursively by

$$\Delta V_z = \Delta V_{z-1} + V_{in,z-1} - V_{out,z-1} \tag{3.47}$$

where the volume of fluid passing through an inlet or outlet over the course of the time step is a function of the mass flow rate ($\dot{m}$), the density of the flow (ρ), and the simulation time step (Δt).

$$V_{in,z} = \frac{\dot{m}_{in,z} \cdot \Delta t}{\rho_{in,z}} \tag{3.48}$$

$$V_{out,z} = \frac{\dot{m}_{out,z} \cdot \Delta t}{\rho_{out,z}} \tag{3.49}$$

It is necessary to take ΔV_z into account when determining the segments of fluid that contribute to the outflow. For instance, if $\Delta V_z > V_{out,z}$ (*i.e.* the fluid level rises faster than the outflow is drawn, as depicted in Figure 3.8(a)), then a "mix zone" (with volume $V_{mix,z}$) is created which is greater than the outflow. The mix zone includes all fluid that passes by or through position z over the course of the time step. All fluid in the mix zone is thoroughly mixed. When the mix zone is larger than the outflow, the leftover mixed fluid remains in the tank as a new segment.

This rule is illustrated by Figure 3.8. Based on Equation (3.47), the fluid volume marked ΔV_3 is equal to $V_{in,2}$. According to the rules listed in Table 3.2, since $\Delta V_3 > 0$ and $\Delta V_3 > V_{out,3}$, then $V_{mix} = \Delta V_3$. After the fluid in this region is mixed, the outflow is removed from the storage profile. The leftover portion of V_{mix} remains in the storage profile (see i_4 in Figure 3.8(b)).

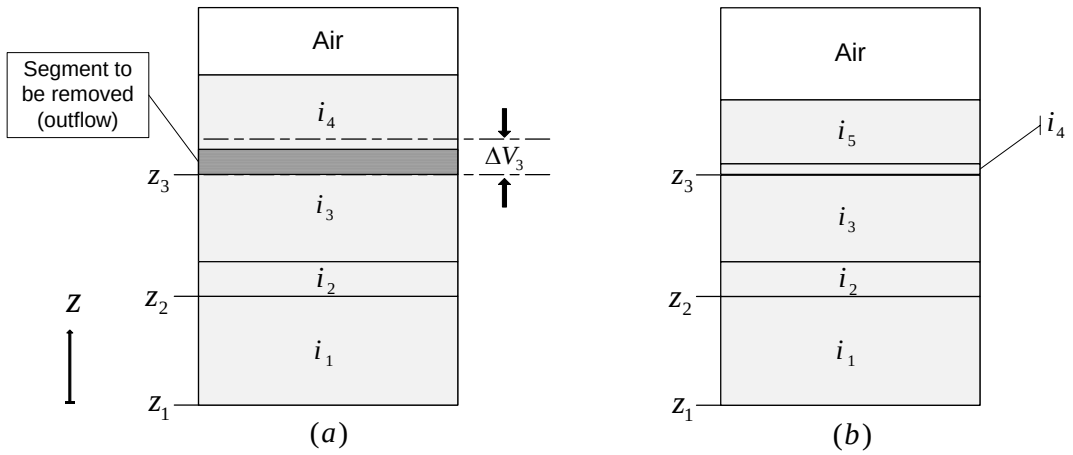

Figure 3.8: Illustration of plug flow rules for outflow: (a) Storage profile after inflow has been inserted with indicators for the volume to be removed (shaded region) and the volume of fluid to be mixed (fluid between the ΔV_3 lines); (b) Storage profile after outflow has been removed through z_3, with new a segment (i_4) created from fluid remaining from the mix zone.

Table 3.2: Determination of the mix zone at any inlet/outlet position z

Fluid level change, ΔV_z	**Mix zone volume, $V_{mix,z}$**
Rising (> 0)	Greater of $V_{out,z}$ and $V_{in,z} + \Delta V_z$
Steady ($=0$)	Greater of $V_{out,z}$ and $V_{in,z}$
Falling (< 0)	Greater of $V_{out,z} + \lvert\Delta V_z\rvert$ and $V_{in,z}$

Because outflow is accounted for sequentially (*i.e.* from the lowest- to highest-numbered positions), the plug flow method requires that the volume of fluid between neighboring positions matches or exceeds $|\Delta V_z|$ for falling fluid.

$$A_c \cdot \Delta z > |\Delta V_z| \tag{3.50}$$

If Equation (3.50) is not satisfied, then some of the fluid that would exit an outlet has already been included as outflow through the lower outlet. As shown in Equations (3.47)-(3.49), ΔV_z grows proportionally to the simulation time step. Thus, for large net

outflow ($\sum V_{out,z} \gg \sum V_{in,z}$), the size of the simulation time step is limited by the distance between neighboring positions (Δz). When Equation (3.50) is not satisfied, a fatal simulation error is reported: TRNSYS Type 209 Message #14 (see Table D.8 in Appendix D). The error can be averted by implementing one of the following options: decreasing the time step, decreasing the flow rates, or increasing the spacing between the outlet positions.

3.3.3 Auxiliary Heat

Auxiliary heat input may be added at any of the z positions. The auxiliary heat is added to the energy balance for the mix zone. If auxiliary heat is added at a position with no mix zone (*e.g.* $V_{in,z} = V_{out,z} = \Delta V_z = 0$), then the auxiliary heat is added to the segment at the heater location.

3.3.4 Density Inversions

A density inversion is present in the storage profile when the density of a segment is higher than the density of the segment below it, as depicted in Figure 3.9(a). (This case occurs only in Mode 1.) The density inversion is corrected by mixing of the two segments, as depicted in Figure 3.9(b). Multiple corrections may be necessary to return the storage to a stable stratification. All corrections occur after the routines for inflow, outflow, and auxiliary have been completed.

3.3.5 Excess Segments

New segments are created from inflow and from flow that remains in the tank when the mix zone is larger than the outflow. The tank depicted in Figure 3.10(a) has segments in excess of the maximum permissible number of segments. Therefore, the two neighboring segments with the smallest density difference undergo complete mixing to form a single new segment, as depicted in Figure 3.10(b). This process may need to be repeated multiple times in order to return the tank to an acceptable number of fluid segments.

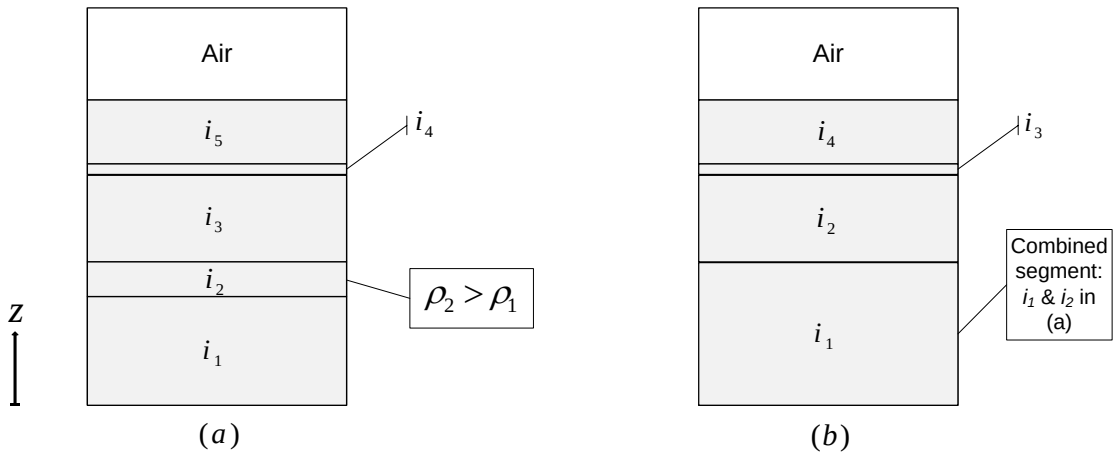

Figure 3.9: Illustration of plug flow rules for correcting density inversions: (a) Storage profile after outflow has been removed, with an indicator for the location of the density inversion; (b) Storage profile after the density inversion has been corrected by complete mixing of two segments.

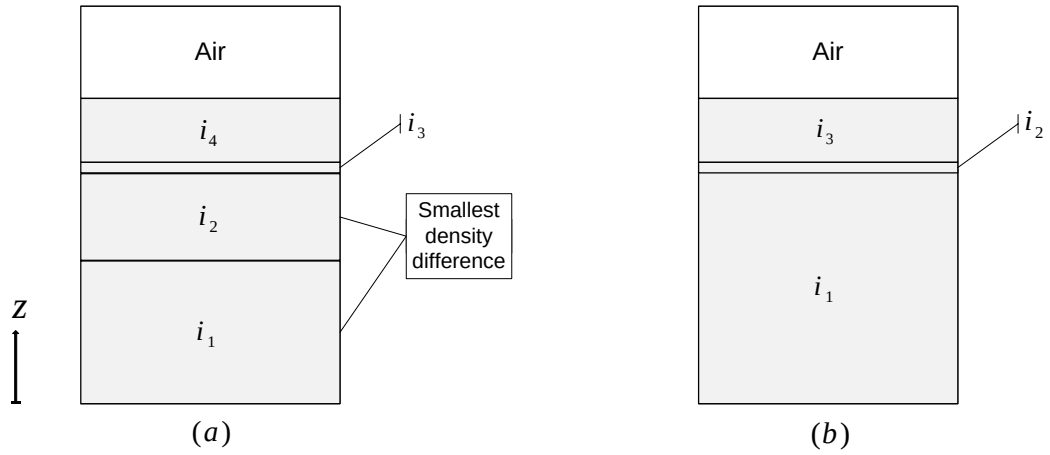

Figure 3.10: Illustration of plug flow rules for reducing segments: (a) Storage profile after the density inversion has been corrected, with an indicator for the location of smallest density difference between neighboring segments; (b) Storage profile after reducing segments to permissible number by complete mixing of two segments.

3.4 User-defined Parameters for TRNSYS Type 209

The user defines the parameters for the physical tank design. The tank size is defined by setting the tank's maximum fluid volume, height, and circumference. The positions for inflow, outflow, and auxiliary are defined by their vertical coordinate, with one parameter identifying the total number of positions to be defined. The insulation is defined through overall heat transfer coefficients for the bottom wall, side walls, and top wall. The side may have different wet and dry overall heat transfer coefficients. The inlet mode is also defined by the user.

The user also defines the parameters for the initial storage profile. The mass, temperature, and concentration of each segment are defined, with one parameter identifying the total number of segments to be defined. One parameter allows the user to choose between lithium chloride (LiCl) or calcium chloride ($CaCl_2$) as the desiccant solution.

The user defines the minimum temperature at which sensible energy is to be considered usable. The user also defines the minimum concentration at which chemical energy is considered usable. These parameters allow the user to monitor the usable energy contained within the storage over the course of the simulation.

The user specifies the maximum number of segments which compose the storage. Additionally, the user indicates the minimum density difference to maintain distinct segments. If the density of neighboring segments is closer than the minimum, the two segments are mixed. Similarly, the user specifies a minimum volume fraction (relative to the overall storage volume) for small segments. Segments which fall below this minimum volume fraction are mixed with a neighboring segment. These parameters allow the user to balance modeling precision (achieved with more segments) with processing time (more efficient with fewer segments).

A full list of TRNSYS Type 209 parameters (as well as lists of required inputs and available outputs) appears in Appendix D. The simulation time step and the simulation duration are not defined directly for TRNSYS Type 209. (The user defines both parameters for the system in the TRNSYS Simulation Studio.)

4 Model Testing and Demonstration

This chapter demonstrates key functions of the new desiccant storage model: TRNSYS Type 209. Sections 4.1-4.4 demonstrate basic plug flow rules (introduced in Section 3.3). Two case studies follow: the number of segments comprising the storage is varied (Section 4.5), and the two inflow operating modes are compared (Section 4.6).

4.1 Inflow and Outflow

Inflow and outflow are accounted for as explained in Sections 3.3.1 and 3.3.2, respectively. A test case (Figure 4.1) demonstrates a single-inflow, single-outflow situation. While the values assigned to the parameters and inputs were arbitrarily chosen, they are within the expected operating ranges. They are listed in Table 4.1 and Table 4.2, respectively. It is expected that that original segment will be steadily depleted by the outflow as a new, higher-concentration segment grows beneath it as a result of the inflow.

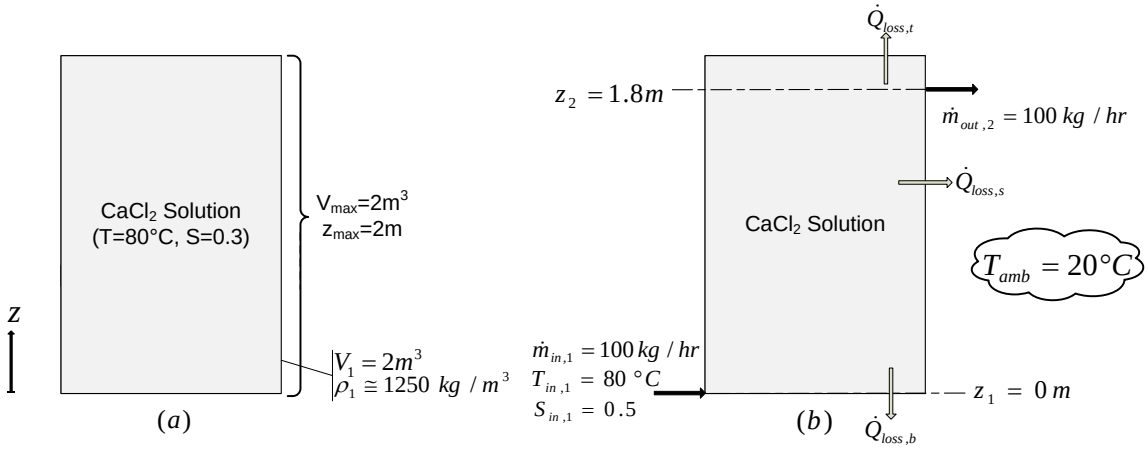

Figure 4.1: Schematics the of test case for inflow and outflow: (a) initial conditions for one segment of $CaCl_2$ solution with T=80°C and S=0.3; (b) boundary conditions for one Mode 1 inflow, one outflow, and thermal losses

Table 4.1: TRNSYS Type 209 Parameters for the inflow and outflow test case

Symbol	Description	Value
M_0	Initial Number of Segments	1
N	Number of Positions	2
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	1 ($CaCl_2$)
M_{max}	Maximum permissible number of segments	2
$Mode$	Operational mode for inflow	1 (fixed)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0.4 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.3 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	2500 kg
$S_{0,1}$	Initial concentration of segment-1	0.3
$T_{0,1}$	Initial temperature of segment-1	80°C
z_1	Height of position-1	0 m
z_2	Height of position-2	1.8 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.2: TRNSYS Type 209 Inputs for the inflow and outflow test case

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	100 kg-soln/hr
$\dot{m}_{in,2}$	Mass flow rate entering at position-2	0 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	0.5
$S_{in,2}$	Concentration of flow entering at position-2	n/a
$T_{in,1}$	Temperature of flow entering at position-1	80°C
$T_{in,2}$	Temperature of flow entering at position-2	n/a
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	0 kg-soln/hr
$\dot{m}_{out,2}$	Mass flow rate exiting at position-2	100 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	0 kJ/hr
$\dot{Q}_{aux,2}$	Auxiliary heat rate entering at position-2	0 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

This test case demonstrates how the TRNSYS Type 209 storage tank models inflow and outflow via fixed inlet and outlet positions. The simulation is for ten one-hour time steps. The mass of the original segment (with $S = 0.3$) is steadily depleted, while the mass of the new segment (with $S = 0.5$) steadily grows (Figure 4.2). (Note that the segment with $S = 0.3$ is labeled as the first segment at $t = 0$. For $t > 0$, it is labeled as the second segment, since it is located above the segment with $S = 0.5$.) Figure 4.3 displays the percent errors in the mass, species, and energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

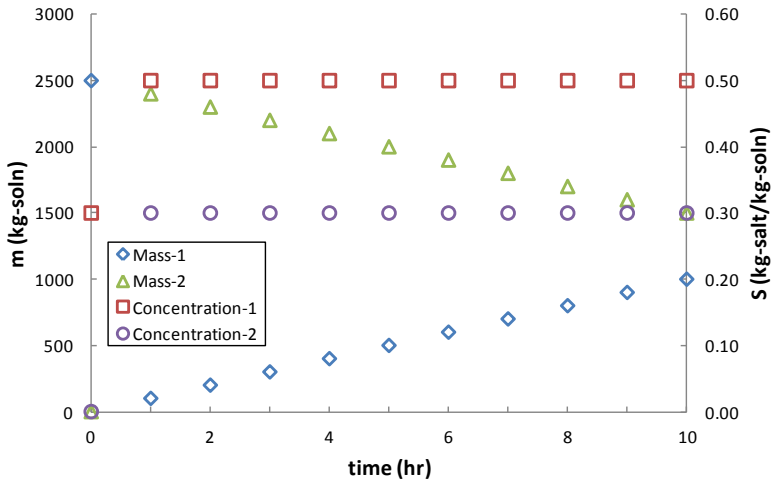

Figure 4.2: Mass and concentration of storage segments 1 and 2 in the inflow and outflow test case (introduced in Figure 4.1). The storage is composed of one segment at t=0 and two segments throughout the rest of the simulation.

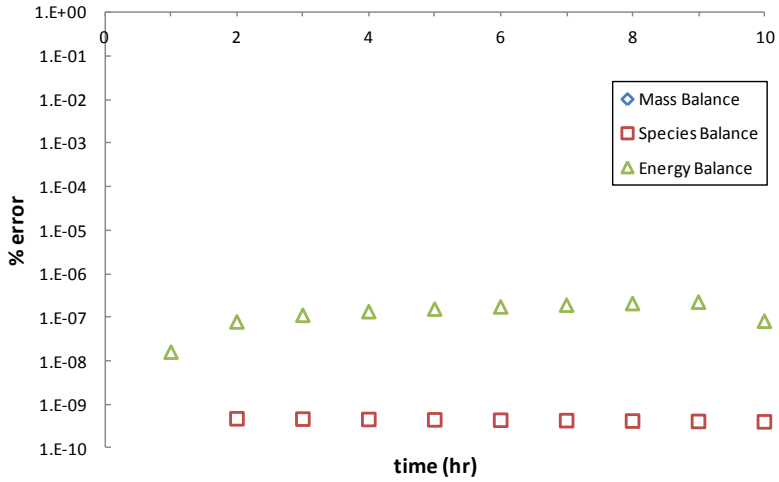

Figure 4.3: Percent errors in the mass, species, and energy balances for each time step in the auxiliary heat input test case (introduced in Figure 4.1). Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

4.2 Auxiliary Heat Addition

As explained in Section 3.3.3, auxiliary heat may be added to the storage. If there is net inflow or net outflow at or below the position of the auxiliary heater, the auxiliary heat is added to the mix zone (introduced in Section 3.3.2). Otherwise, the auxiliary heat is added to the fluid segment above the heater. The test case (Figure 4.4) demonstrates the latter situation. While the values assigned to the parameters and inputs were arbitrarily chosen, they are within the expected operating ranges. They are listed in Table 4.3 and Table 4.4, respectively. When the first segment is heated to the point where its density becomes less than that of the second segment, the model mixes the two segments.

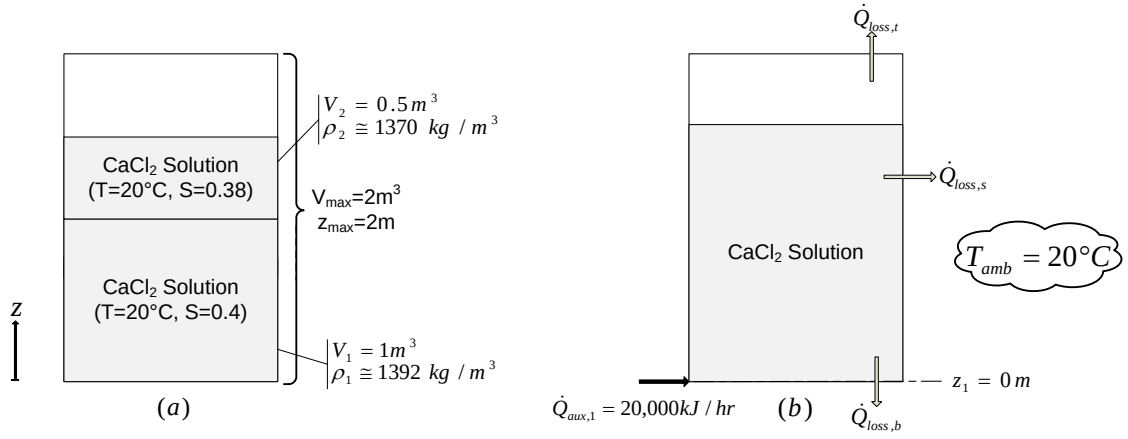

Figure 4.4: Schematics of the test case for auxiliary heat addition: (a) initial conditions for one segment of $CaCl_2$ solution with T=20°C and S=0.4 and a second segment of $CaCl_2$ solution with T=20°C and S=0.38; (b) boundary conditions for one auxiliary heat input and thermal losses

Table 4.3: TRNSYS Type 209 Parameters for the auxiliary heating test case

Symbol	Description	Value
M_0	Initial Number of Segments	2
N	Number of Positions	1
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	1 ($CaCl_2$)
M_{max}	Maximum permissible number of segments	2
$Mode$	Operational mode for inflow	2 (ideal)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0.4 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.3 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	1392 kg
$m_{0,2}$	Initial mass of segment-2	685 kg
$S_{0,1}$	Initial concentration of segment-1	0.4
$S_{0,2}$	Initial concentration of segment-2	0.38
$T_{0,1}$	Initial temperature of segment-1	20°C
$T_{0,2}$	Initial temperature of segment-2	20°C
z_1	Height of position-1	0 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.4: TRNSYS Type 209 Inputs for the auxiliary heating test case

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	0 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	n/a
$T_{in,1}$	Temperature of flow entering at position-1	n/a
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	0 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	20,000 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

This test case demonstrates how the TRNSYS Type 209 storage tank models auxiliary heat input. The simulation is for ten one-hour time steps. In each time step, auxiliary heat is added at a constant rate at the base of the storage (see Figure 4.4(b)). The heating of the first segment steadily decreases its density (Figure 4.5). After the ninth time step, the two storage segments are mixed to maintain stable stratification. (See the sudden change in segment volume in Figure 4.5.) Figure 4.6 displays the percent errors in the mass, species, and energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

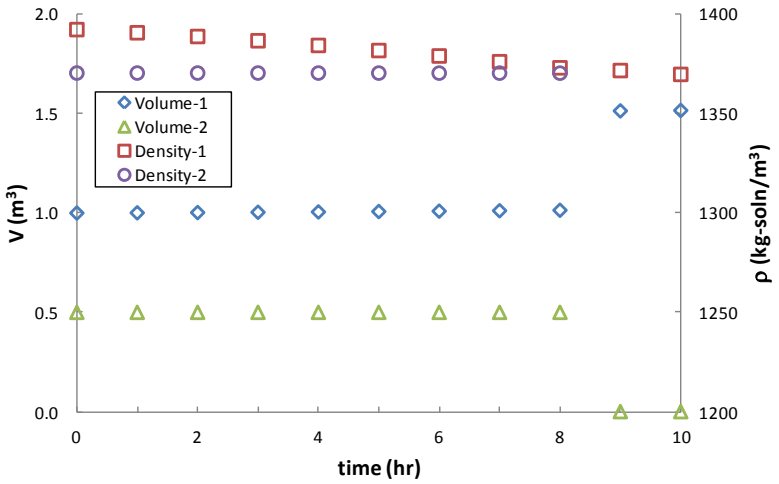

Figure 4.5: Volume and density of storage segments 1 and 2 in the auxiliary heat input test case (introduced in Figure 4.4). The storage is composed of two segments until time t=9 hr, when the segments are mixed to correct a density inversion.

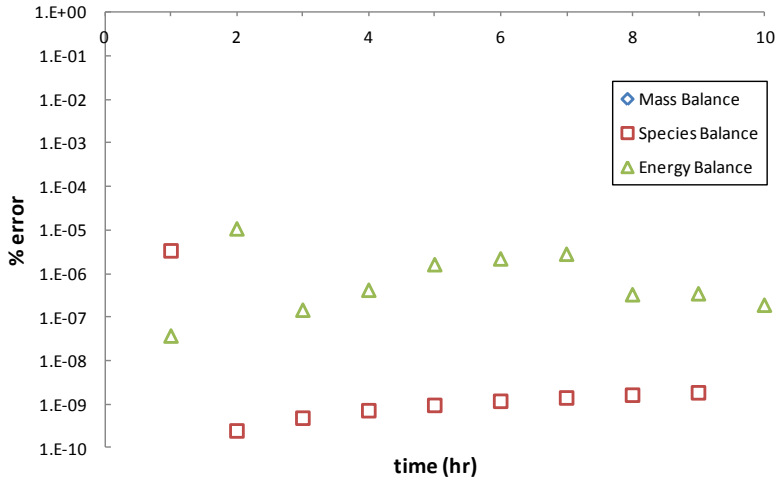

Figure 4.6: Percent errors in the mass, species, and energy balances for each time step in the auxiliary heat input test case (introduced in Figure 4.4). Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

4.3 Mixing of Inverted Segments

Fluid entering through fixed inlets (*i.e.* Mode 1 inflow) may create density inversions. As explained in Section 3.3.4, these density inversions are corrected at the end of the time step, when the inverted segments are mixed. A test case to demonstrate this rule is depicted in Figure 4.7. While the values assigned to the parameters and inputs were arbitrarily chosen, they are within the expected operating ranges. They are listed in Table 4.5 and Table 4.6, respectively. Because the concentration (and therefore density) of the incoming fluid is significantly lower than that of the storage, the model mixes the incoming fluid with the storage after every time step.

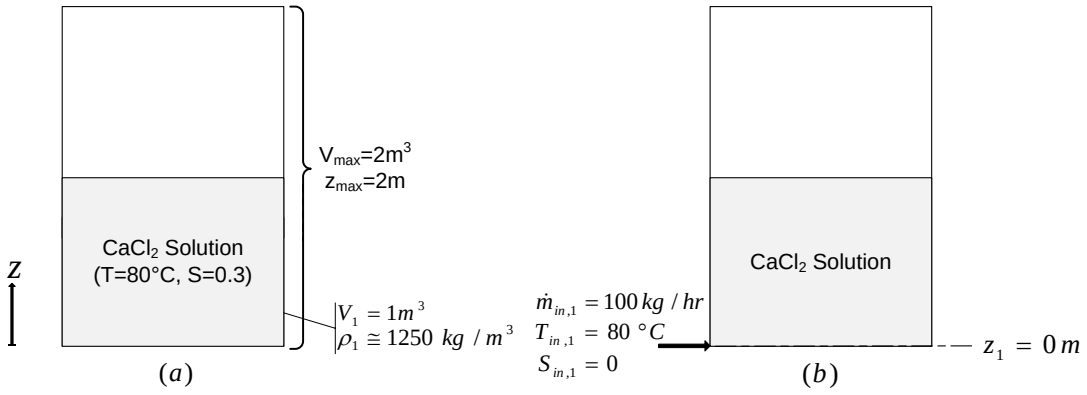

Figure 4.7: Schematics of the test case for mixing segments with density inversions: (a) initial conditions for one segment of $CaCl_2$ solution with T=80°C and S=0.3; (b) boundary conditions for one Mode 1 inflow and a perfectly insulated tank

Table 4.5: TRNSYS Type 209 Parameters for the density inversion test case

Symbol	Description	Value
M_0	Initial Number of Segments	1
N	Number of Positions	1
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	1 ($CaCl_2$)
M_{max}	Maximum permissible number of segments	5
$Mode$	Operational mode for inflow	1 (fixed)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.3 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	1250 kg
$S_{0,1}$	Initial concentration of segment-1	0.3
$T_{0,1}$	Initial temperature of segment-1	80°C
z_1	Height of position-1	0 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.6: TRNSYS Type 209 Inputs for the density inversion test case

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	100 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	0 kg-salt/kg-soln
$T_{in,1}$	Temperature of flow entering at position-1	80°C
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	0 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	0 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

This test case demonstrates how the TRNSYS Type 209 storage tank enforces stable stratification when Mode 1 inflow creates density inversions. The simulation is for ten one-hour time steps. Each time step ends with only one segment of fluid comprising the storage, since the low-density inflow creates a density inversion when it enters at the base of the storage. The incoming water (solution at $S = 0$) steadily decreases the concentration of the storage, as shown in Figure 4.8. Also depicted in Figure 4.8 is the temperature of the storage. The temperature of the inflow is equal to the temperature of the storage at the beginning of the simulation ($T = 80°C$). The heat of dilution which is released during the mixing of the segments raises the temperature of the storage above 80°C. Figure 4.9 displays the percent errors in the mass, species, and energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

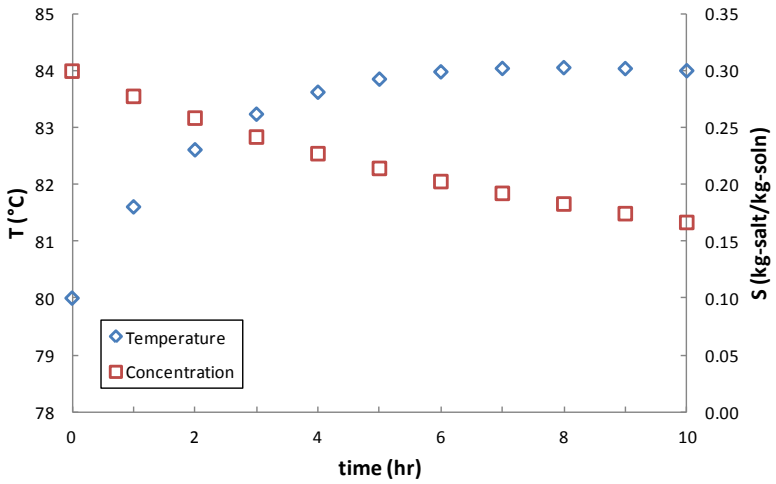

Figure 4.8: Temperature and concentration of the storage in the density inversion test case (introduced in Figure 4.7). After each time step, the storage is mixed in order to maintain stable stratification.

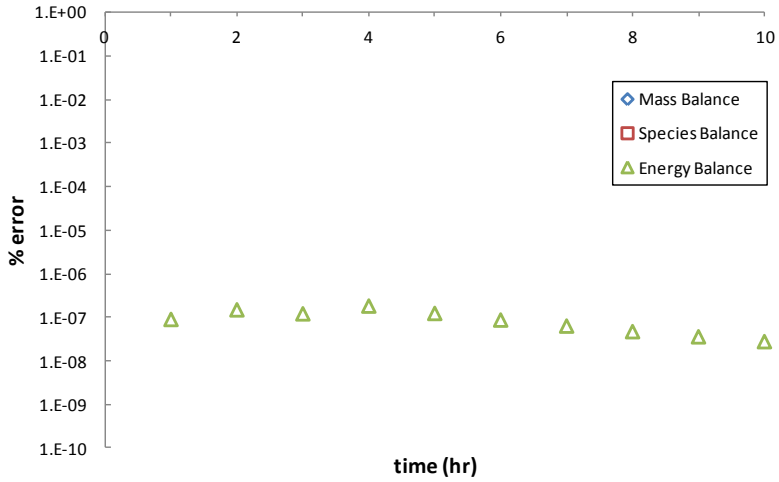

Figure 4.9: Percent errors in the mass, species, and energy balances for each time step in the density inversion test case (introduced in Figure 4.7). Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

4.4 Mixing of Excess Segments

During the course of the time step, the inflow and outflow algorithms may increase the number of segments which comprise the storage. As explained in Section 3.3.5, the number of segments is reduced to the maximum permissible number at the end of the time step. The neighboring segments with the smallest density difference are mixed instantaneously and completely. A test case to demonstrate this rule is depicted in Figure 4.10. While the values assigned to the parameters and inputs were arbitrarily chosen, they are within the expected operating ranges. They are listed in Table 4.7 and Table 4.8, respectively. Because the parameter for maximum permissible segments is set to two, the model mixes the incoming fluid with one of the two storage segments at the end of each time step.

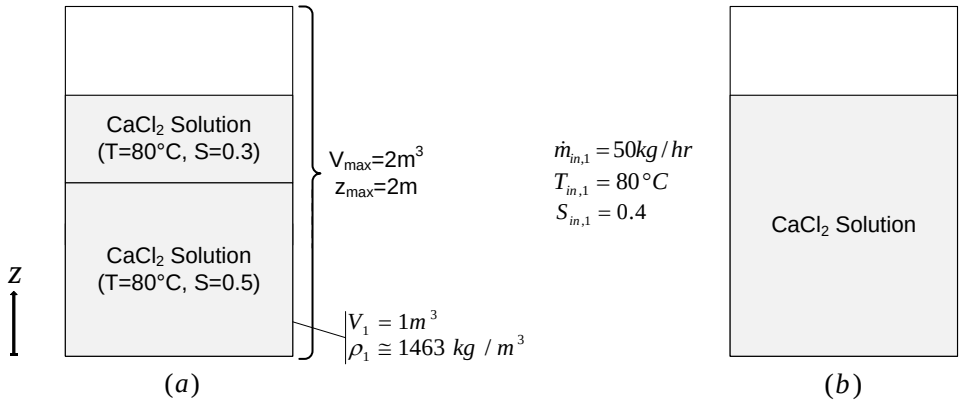

Figure 4.10: Schematics of the test case for reducing the number of segments: (a) initial conditions for one segment of $CaCl_2$ solution with T=80°C and S=0.5 and a second segment of $CaCl_2$ solution with T=80°C and S=0.3; (b) boundary conditions for one Mode 2 inflow and a perfectly insulated tank

Table 4.7: TRNSYS Type 209 Parameters for the excess segment test case

Symbol	Description	Value
M_0	Initial Number of Segments	2
N	Number of Positions	1
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	1 ($CaCl_2$)
M_{max}	Maximum permissible number of segments	2
$Mode$	Operational mode for inflow	2 (ideal)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.3 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	1463 kg
$m_{0,2}$	Initial mass of segment-2	625 kg
$S_{0,1}$	Initial concentration of segment-1	0.5
$S_{0,2}$	Initial concentration of segment-2	0.3
$T_{0,1}$	Initial temperature of segment-1	80°C
$T_{0,2}$	Initial temperature of segment-2	80°C
z_1	Height of position-1	0 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.8: TRNSYS Type 209 Inputs for the excess segment test case

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	50 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	0.4 kg-salt/kg-soln
$T_{in,1}$	Temperature of flow entering at position-1	80°C
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	0 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	0 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

This test case demonstrates how the TRNSYS Type 209 storage tank limits the number of segments which comprise the storage. The simulation is for ten one-hour time steps. In each time step, inflow creates a third segment. Since the test case limits the storage to two segments (see Table 4.7), the neighboring segments with the smallest density difference are mixed. For the parameters and inputs used in this test case, these two segments are the second segment and the new (inflow) segment. These two segments are mixed at the end of the time step, so that the size of the second segment steadily grows as inflow is added to it. (This is shown in Figure 4.11.) The incoming solution ($S = 0.4$) steadily increases the concentration of the second storage segment, as shown in Figure 4.12. Also depicted in Figure 4.12 is the temperature of the storage. While the temperature of the inflow is equal to the temperature of the storage at the beginning of the simulation ($T = 80°C$), the heat of dilution (released during the segment mixing) raises the temperature of the second storage segment above 80°C. Figure 4.13 displays the percent errors in the mass, species, and energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

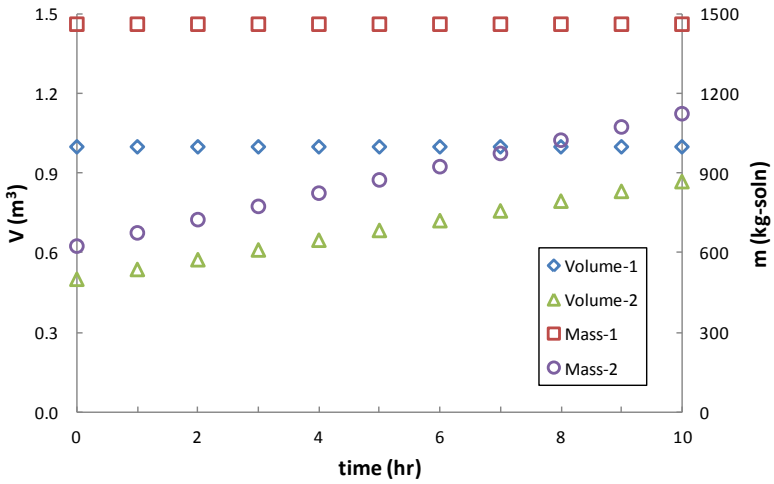

Figure 4.11: Volume and mass of storage segments 1 and 2 in the segment number reduction test case (introduced in Figure 4.10).

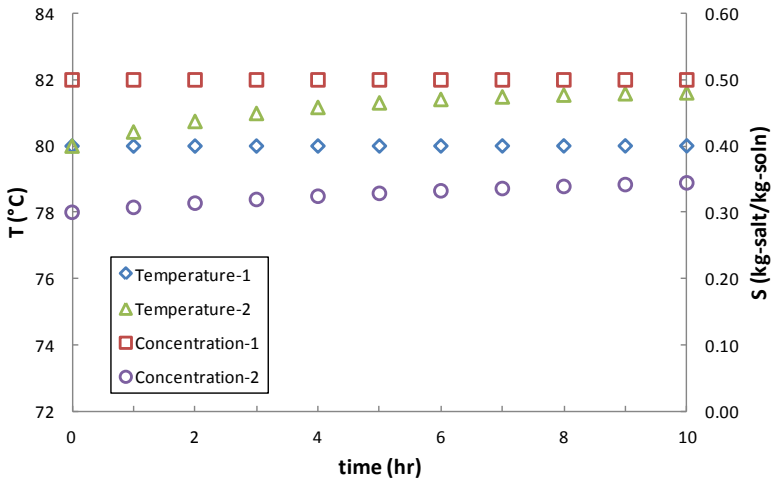

Figure 4.12: Temperature and concentration of storage segments 1 and 2 in the segment number reduction test case (introduced in Figure 4.10).

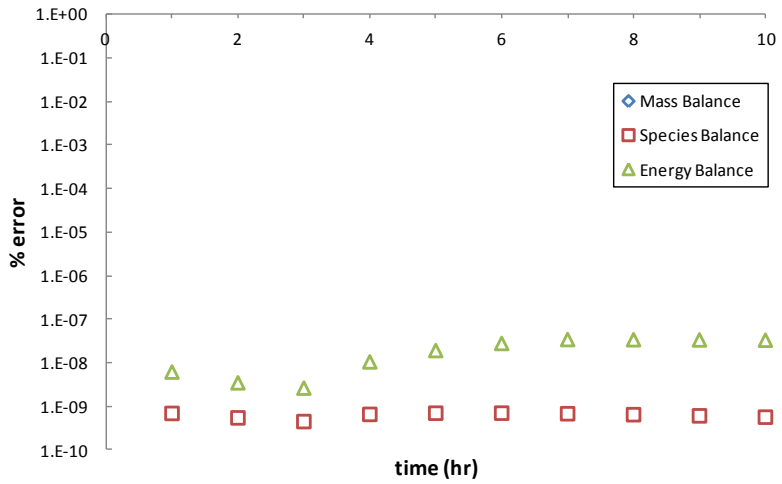

Figure 4.13: Percent errors in the mass, species, and energy balances for each time step in the segment number reduction test case (introduced in Figure 4.10). Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

4.5 Case Study: Number of Segments

The first case study demonstrates the extent to which the degree of stratification (*i.e.* number of segments comprising the storage) affects the chemical energy storage. It is expected that increased stratification will increase the quality of the storage (*i.e.* the usable stored chemical energy). At the beginning of the simulation, the tank is filled with low-concentration liquid desiccant ($S = 0.3$). Throughout the simulation, the low-concentration desiccant is drawn from the tank. Water is then desorbed externally. Two flows return to the tank: water ($S = 0$) and strong-concentration desiccant ($S = 0.5$). Cases are run for tanks with one, two, three, and four segments. A time step of one hour is used for all cases. The initial and boundary conditions for the case study are shown in Figure 4.14. The corresponding parameters and inputs are listed in Table 4.9 and

Table 4.10, respectively.

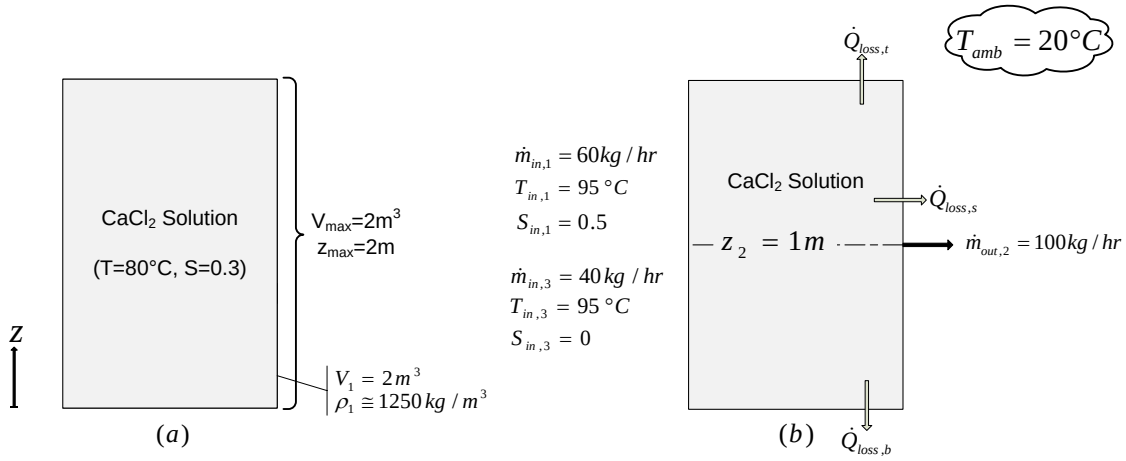

Figure 4.14: Schematics of parametric study for varying number of segments: (a) initial conditions for one segment of $CaCl_2$ with S=0.3 and T=80°C; (b) boundary conditions for two Mode 2 inflows, one outflow, and thermal losses

Usable chemical energy is calculated using Equation (B.25). For this test case, the reference value for usable chemical energy is $S = 0.3$ (as listed in Table 4.9) and the usable chemical energy storage increases as the permitted number of segments comprising the storage increases (for up to three segments, as seen in Figure 4.15). The one- and two-segment storage cases force mixing of segments of different concentrations. This forced mixing reduces the stored usable chemical energy (a two-thirds reduction for two segments and complete reduction for one segment). The three-segment storage does not force undesired mixing, since the initial storage and the two inflows provide fluid at three different concentration levels ($S \cong 0, S \cong 0.3$, and $S \cong 0.5$). Thus, as shown in Figure 4.15, adding a fourth segment does not increase the usable chemical energy storage (not as a general rule, but for the chosen parameters). When chemical energy is lost due to forced mixing, it is converted to sensible energy. The one- and two-segment cases contain lower

total usable energy storage than the three- and four-segment cases (Figure 4.16) because the forced mixing increases the temperature of the outflow (Figure 4.17).

Table 4.9: TRNSYS Type 209 Parameters for the study of various numbers of segments

Symbol	Description	Value
M_0	Initial Number of Segments	1
N	Number of Positions	3
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	1 ($CaCl_2$)
M_{max}	Maximum permissible number of segments	1, 2, 3, 4
$Mode$	Operational mode for inflow	2 (ideal)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0.4 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.3 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	2500 kg
$S_{0,1}$	Initial concentration of segment-1	0.3
$T_{0,1}$	Initial temperature of segment-1	80°C
z_1	Height of position-1	0 m
z_2	Height of position-2	1 m
z_3	Height of position-3	2 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.10: TRNSYS Type 209 Inputs for the study of various numbers of segments

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	60 kg-soln/hr
$\dot{m}_{in,2}$	Mass flow rate entering at position-2	0 kg-soln/hr
$\dot{m}_{in,3}$	Mass flow rate entering at position-3	40 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	0.5 kg-salt/kg-soln
$S_{in,2}$	Concentration of flow entering at position-2	n/a
$S_{in,3}$	Concentration of flow entering at position-3	0.0 kg-salt/kg-soln
$T_{in,1}$	Temperature of flow entering at position-1	95°C
$T_{in,2}$	Temperature of flow entering at position-2	n/a
$T_{in,3}$	Temperature of flow entering at position-3	95°C
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	0 kg-soln/hr
$\dot{m}_{out,2}$	Mass flow rate exiting at position-2	100 kg-soln/hr
$\dot{m}_{out,3}$	Mass flow rate exiting at position-2	0 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	0 kJ/hr
$\dot{Q}_{aux,2}$	Auxiliary heat rate entering at position-2	0 kJ/hr
$\dot{Q}_{aux,3}$	Auxiliary heat rate entering at position-3	0 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

Figure 4.18 displays the percent errors in the energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

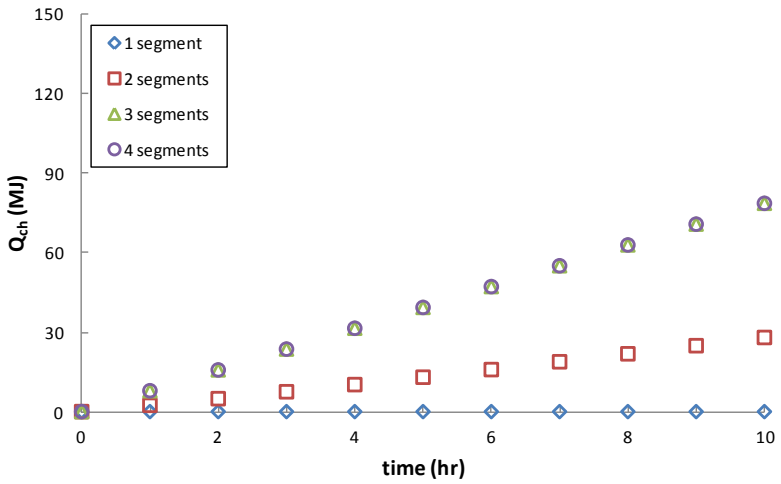

Figure 4.15: Total usable chemical energy storage for the cases introduced in Figure 4.14. The reference state for usable chemical energy is S=0.3. The cases differ only in the number of segments which comprise the storage.

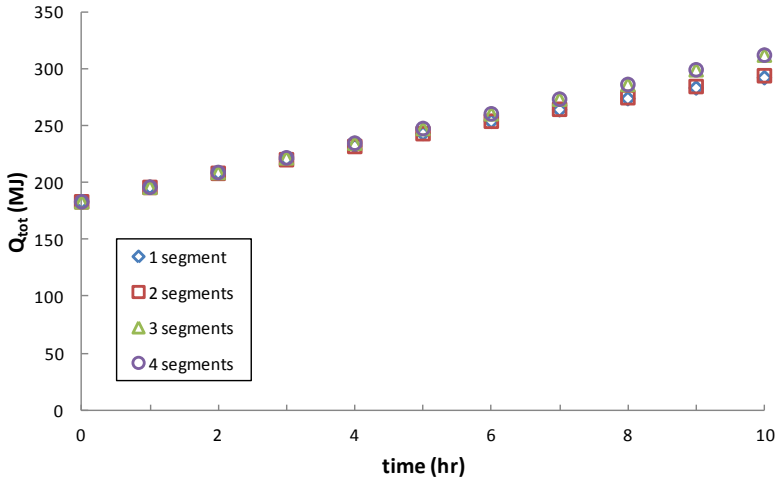

Figure 4.16: Combined usable sensible and chemical energy storage for the cases introduced in Figure 4.14. The reference state for usable chemical energy is S=0.3. The reference state for usable sensible energy is T=55°C. The cases differ only in the number of segments which comprise the storage.

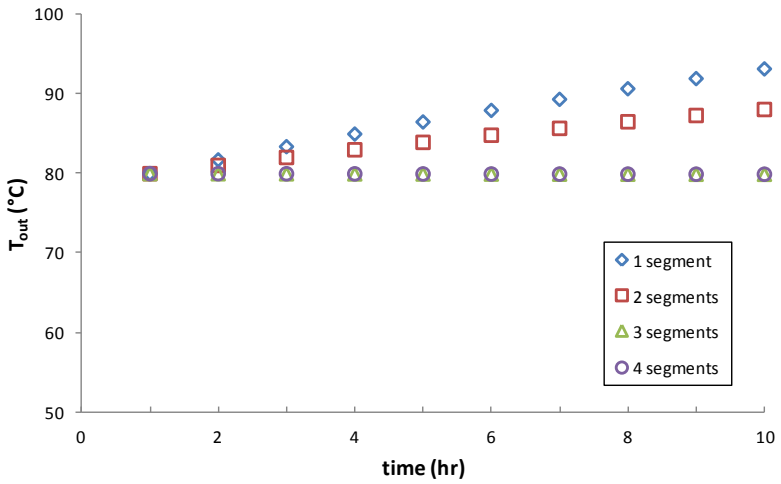

Figure 4.17: Temperature of the outflow for the cases introduced in Figure 4.14. The cases differ only in the number of segments which comprise the storage.

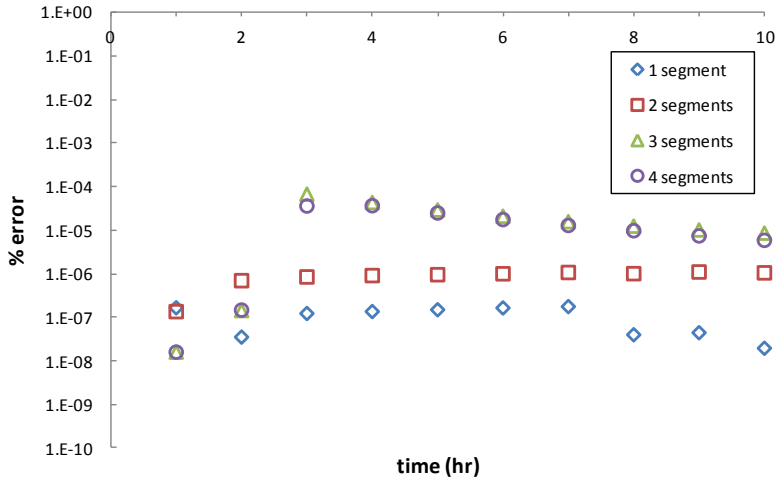

Figure 4.18: Percent errors in the energy balances for the cases introduced in Figure 4.14. The cases differ only in the number of segments which comprise the storage. Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

4.6 Case Study: Inflow Operating Mode

The next case study demonstrates the effectiveness of idealized inlets (Mode 2 inflow) in comparison to fixed inlets (Mode 1 inflow). It is expected that inflow which is inserted at the location of neutral buoyancy will maximize the amount of usable stored chemical energy. At the beginning of the simulation, the tank is composed of two segments of fluid: one of high-concentration desiccant and one of water. Throughout the simulation, fluid is drawn from both segments, mixed externally, and returned to the tank as weak-concentration desiccant. Cases are run for three tanks with Mode 1 inflow (with the inflow entering at a different inlet height in each case) and for one tank with Mode 2 inflow. A time step of one hour is used for all cases. The initial and boundary conditions for the case study are shown in Figure 4.19. The equivalent parameters and inputs are listed in Table 4.11 and Table 4.12, respectively.

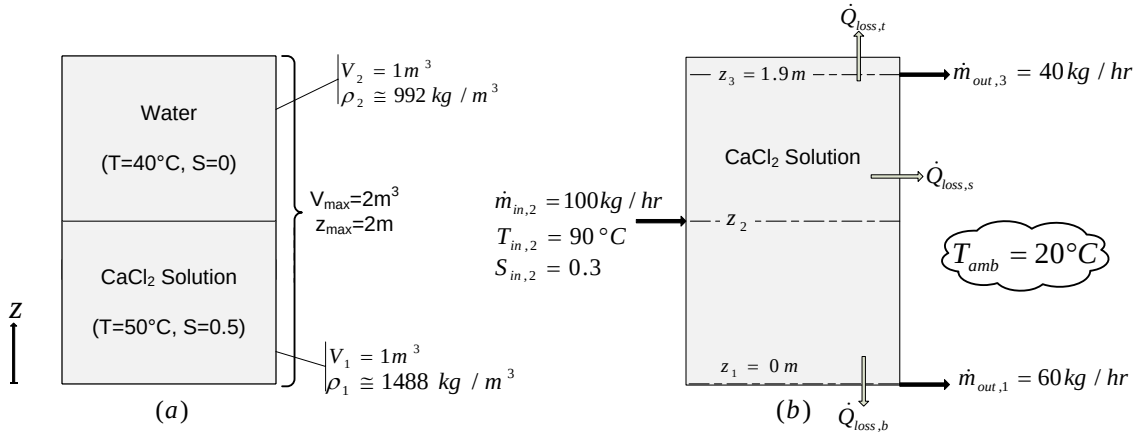

Figure 4.19: Schematics of the comparative study of inlet modes: (a) initial conditions for one segment of $CaCl_2$ solution with T=50°C and S=0.5 and one segment with T=40°C and S=0 (*i.e.* water); (b) boundary conditions for one inflow (fixed inlet shown), for two outflows, and for thermal losses

Table 4.11: TRNSYS Type 209 Parameters for the comparison of fixed and idealized inlets

Symbol	Description	Value
M_0	Initial Number of Segments	2
N	Number of Positions	3
V_{max}	Storage capacity	2 m^3
z_{max}	Storage height	2 m
CC	Circumference of cross section	$(4\pi)^{0.5}$ (circular)
$salt$	Storage fluid (hydrate salt)	$CaCl_2$
M_{max}	Maximum permissible number of segments	3
$Mode$	Operational mode for inflow	1 (fixed), 2 (ideal)
φ_{min}	Minimum volume fraction of each segment	0.001
U_L	Thermal loss coefficient (all)	0.4 kJ/hr-m^2-K
T_{min}	Minimum temperature for usable sensible energy	55°C
S_{min}	Minimum concentration for usable chemical energy	0.35 kg-salt/kg-soln
$m_{0,1}$	Initial mass of segment-1	1488 kg
$m_{0,2}$	Initial mass of segment-2	992 kg
$S_{0,1}$	Initial concentration of segment-1	0.5
$S_{0,2}$	Initial concentration of segment-2	0.0
$T_{0,1}$	Initial temperature of segment-1	50°C
$T_{0,2}$	Initial temperature of segment-2	50°C
z_1	Height of position-1	0 m
z_2	Height of position-2	1 m, 0.75m, 0.5m
z_3	Height of position-3	1.9 m
Δt	Simulation time step	1 hr
Δt_{tot}	Simulation duration	10 hrs

Table 4.12: TRNSYS Type 209 Inputs for the comparison of fixed and idealized inlets

Symbol	Description	Value
$\dot{m}_{in,1}$	Mass flow rate entering at position-1	0 kg-soln/hr
$\dot{m}_{in,2}$	Mass flow rate entering at position-2	100 kg-soln/hr
$\dot{m}_{in,3}$	Mass flow rate entering at position-3	0 kg-soln/hr
$S_{in,1}$	Concentration of flow entering at position-1	n/a
$S_{in,2}$	Concentration of flow entering at position-2	0.3 kg-salt/kg-soln
$S_{in,3}$	Concentration of flow entering at position-3	n/a
$T_{in,1}$	Temperature of flow entering at position-1	n/a
$T_{in,2}$	Temperature of flow entering at position-2	90°C
$T_{in,3}$	Temperature of flow entering at position-3	n/a
$\dot{m}_{out,1}$	Mass flow rate exiting at position-1	60 kg-soln/hr
$\dot{m}_{out,2}$	Mass flow rate exiting at position-2	0 kg-soln/hr
$\dot{m}_{out,3}$	Mass flow rate exiting at position-2	40 kg-soln/hr
$\dot{Q}_{aux,1}$	Auxiliary heat rate entering at position-1	0 kJ/hr
$\dot{Q}_{aux,2}$	Auxiliary heat rate entering at position-2	0 kJ/hr
$\dot{Q}_{aux,3}$	Auxiliary heat rate entering at position-3	0 kJ/hr
T_{amb}	Ambient temperature (all walls)	20°C

The advantage of Mode 2 is that each inflow is added to the tank at the location of neutral buoyancy. Thus, no buoyancy mixing occurs. For Mode 1, buoyancy mixing occurs unless each fixed inlet is positioned at the location of neutral buoyancy. For the system tested here, $z = 1m$ is the location of neutral buoyancy. Thus the Mode 1 case with inflow entering at $z = 1m$ matches the results of the Mode 2 case. For Mode 1 cases with $z < 1m$, buoyancy mixing occurs. As shown in Figure 4.20, the buoyancy mixing results in 16% and 24% reductions in usable chemical energy at the end of the simulation

period (for inlets at $z = 0.75m$ and $z = 0.5m$, respectively). Usable chemical energy is calculated using Equation (B.25). For this test case, the reference value for usable chemical energy is $S = 0.3$ (as listed in Table 4.11).

As shown in Figure 4.21, the buoyancy mixing also results in 42% and 84% reductions in the mass of high-concentration ($S = 0.5$) solution available at the end of the simulation period (for inlets at $z = 0.75m$ and $z = 0.5m$, respectively). Figure 4.22 displays the percent errors in the energy balances. Absolute errors are calculated based on Equations (3.41)-(3.43). Percent errors are calculated based on the size of the absolute error relative to the total storage at the beginning of the time step.

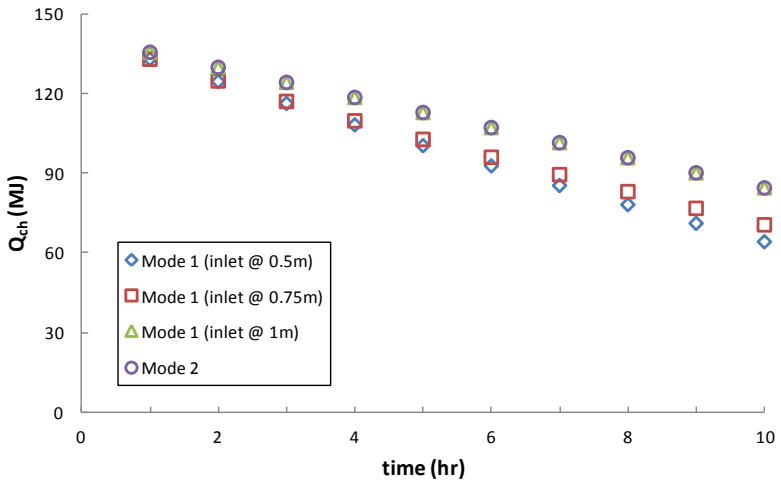

Figure 4.20: Total usable chemical energy storage for the cases introduced in Figure 4.19. The reference state for usable chemical energy is S=0.3. Three cases use Mode 1 inflow with different inlet heights. One case uses Mode 2 inflow.

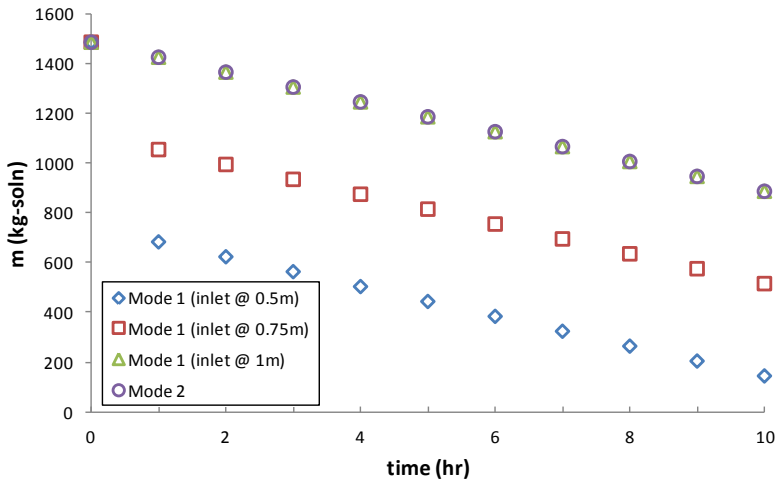

Figure 4.21: High-concentration (S=0.5) mass remaining in the storage for the cases introduced in Figure 4.19. Three cases use Mode 1 inflow with different inlet heights. One case uses Mode 2 inflow.

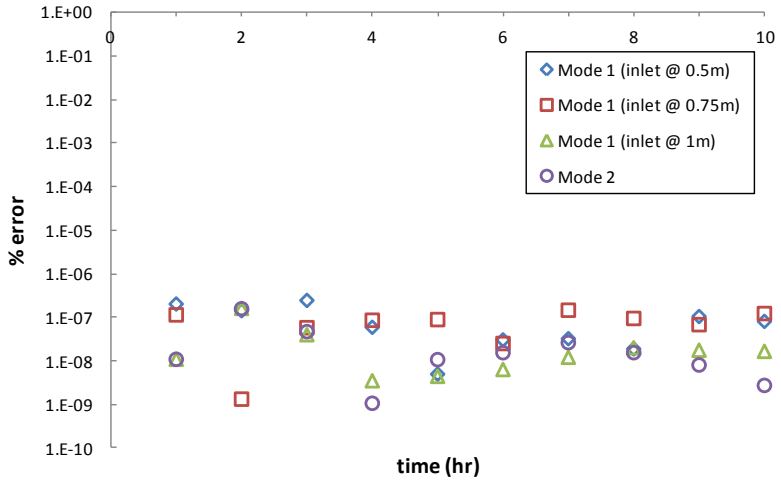

Figure 4.22: Percent errors in the energy balances for the cases introduced in Figure 4.19. Three cases use Mode 1 inflow with different inlet heights. One case uses Mode 2 inflow. Errors below 10^{-10} % are not shown. All percent errors are relative to the storage value at the beginning of the time step.

5 Conclusions

5.1 Summary and Conclusions

A model for a stratified liquid desiccant storage tank has been developed. Maintaining stratification in a liquid desiccant storage tank maximizes the long-term storage of chemical energy. The chemical energy is significantly reduced when liquid desiccant storage is mixed. The stratified storage model can be used in long-term, system-level simulations in TRNSYS. The model incorporates four primary development objectives: storage of a variable-concentration binary solution, inclusion of chemical energy, variability in the volume of the storage, and stratification of a variable-density fluid.

The storage is a binary solution of salt (either $CaCl_2$ or LiCl) and water with variable concentration. Previously available tank models with the option of a binary storage medium require that the concentration of the solution remain fixed in time and space. The new storage model conserves the mass of each species while permitting 1-D spatial concentration variation. The new model also permits the solution concentration at any location to change with time due to internal conditions (*e.g.* mass diffusion within the tank) or external conditions (*e.g.* mass flow through inlets and outlets).

Chemical potential energy is incorporated into the energy balance. Previously available models consider sensible-only storage. The chemical energy storage of liquid desiccants is a key advantage of desiccant-based storage. Thus, the model tracks both sensible and chemical energy storage.

The model is for variable-volume storage. Previously available models consider either stratified fixed-volume storage or well-mixed variable-volume storage. This model

considers stratified variable-volume storage in order to give the user flexibility in system application (*i.e.* operating an open system).

The stratification within the tank is based on density as a function of temperature and concentration. Previously available tank models considered only thermal stratification, with changes in density considered negligible. Since the density of the liquid desiccant solution is highly dependent on the (variable) concentration, density is modeled as a function of two variables: temperature and concentration. Also, since the density of the liquid desiccant solution varies significantly over the range of operating parameters, changes in density are not considered negligible. Many tank models are based on the multinode approach, in which the storage is composed of fixed-volume nodes. In these models, the solutions to the conservation equations require a constant node mass. Therefore, the multinode approach is applicable only to storage with negligible changes in density.

The multinode approach is the first of two common approaches to storage modeling. Since the multinode approach cannot be adapted to meet the requirements of a liquid desiccant storage model, the model follows the second common approach: plug flow. In the plug flow approach, a one-dimensional storage is composed of well-mixed segments of fluid. Fluid enters the tank through fixed inlets or through an idealized inlet manifold. Fluid exits the tank through fixed outlets. Auxiliary heat is added via heaters at user-specified positions. Fluid movement within the tank is simulated with a plug flow approach. The standard plug flow approach was modified to incorporate mixing due to mass diffusion in addition to mixing due to buoyancy. As is the case with other plug flow models, it neglects mixing due to momentum. The model incorporates heat transfer due to conduction, thermal losses to the surroundings, and auxiliary heat input. The energy balance includes chemical energy as well as sensible energy. Significant modifications to the standard plug flow model were required in three main areas in order to adapt it for

application to liquid desiccant storage. First, the storage is density-stratified rather than thermally stratified. Second, inlets and outlets are not paired, enabling the volume of the storage to vary over time. Third, the tank includes fixed inlets and outlets at locations other than its top and bottom.

When the model is used in simulations of desiccant-based systems in TRNSYS, the user is able to define the storage characteristics (*e.g.* capacity, insulation, number and location of inlets and outlets, etc.) as well as the initial and boundary conditions for simulations within TRNSYS.

5.2 Recommendations for Further Model Development

The storage is predominantly stratified by concentration, since the desiccant solution's density is much more sensitive to changes in concentration that it is to changes in temperature $\left(\frac{\partial \rho}{\partial S} \gg \frac{\partial \rho}{\partial T}\right)$. The addition of an immersed heat exchanger to a desiccant storage tank aids in creating and maintaining a thermal stratification in the storage as well. Heating the storage with an immersed heat exchanger results in natural convection cells within otherwise well-mixed segments of fluid. These convection cells result in increased mass transfer between segments [2]. Sherwood number (Sh) correlations have been developed from numerical data at the University of Minnesota to estimate the mass transfer rate as a function of the Rayleigh number (Ra), the buoyancy ratio (N), and the Lewis number (Le) [3]. It is recommended that further development of the newly created liquid desiccant storage model include the addition of an immersed heat exchanger along with the relevant convective mixing mass transfer correlations [3].

Further consideration might be given to momentum mixing, which is neglected in the current model. Sherwood number correlations for mixing due to momentum could be developed (similar to the ones developed for mixing due to natural convection). The nature of a plug flow model precludes net mass exchange between neighboring segments,

since the mass of each segment is required to remain constant during the closed system analysis. (See Section 3.2.2.) However, the current model already incorporates diffusion between segments of a binary solution. Therefore, the effect of fluid momentum on species mixing might still be incorporated into the model by creating additional coefficients for the mass diffusion equations.

The model includes one mode for outflow: fixed outlets. The model can be expanded by adding a second mode, permitting idealized outflow (similar to the second mode for inflow). This second mode would allow the user to specify the desired density of the outflow. Since the density is highly dependent on concentration, this second mode would be useful for obtaining desiccant solution of known concentration for desorption charging and absorption discharging processes.

Finally, a parameter might be added for an additional thermal conductivity coefficient to model conduction along the tank's side wall. This additional coefficient is used in some available tank models (*e.g.* Type 534 [9]) to augment each fluid segment's thermal conductivity coefficient.

Bibliography

[1] "Buildings Energy Data Book," U.S. Department of Energy, March 2012. [Online]. Available: http://buildingsdatabook.eren.doe.gov/ChapterIntro2.aspx. [Accessed 21 August 2012].

[2] J. Quinnell, J. Davidson and J. Burch, "Liquid Calcium Chloride Solar Storage: Concept and Analysis," *Journal of Solar Energy Engineering,* 2011.

[3] J. A. Quinnell, "Buoyancy Driven Mass and Heat Transfer in a Novel Calcium Chloride Absorption Storage Tank," The University of Minnesota, Minneapolis, 2012.

[4] M. R. Conde, "Properties of Aqueous Solutions of Lithium and Calcium Chlorides: Formulations for use in Air Conditioning Equipment Design," *International Journal of Thermal Sciences,* vol. 43, pp. 367-382, 2004.

[5] Solar Energy Laboratory, Univ. of Wisconsin, Madison, "TRNSYS Type 39: Variable volume tank," Madison, WI, 2006.

[6] J. A. Duffie and W. A. Beckman, Solar Engineering of Thermal Processes, Hoboken, NJ: John Wiley & Sons, Inc., 2006.

[7] Solar Energy Laboratory, Univ. of Wisconsin, Madison, "TRNSYS Type 4: Stratified fluid storage tank," Madison, WI, 2006.

[8] Solar Energy Laboratory, Univ. of Wisconsin, Madison, "TRNSYS Type 60: Stratified fluid storage tank with internal heat exchangers," Solar Energy Laboratory, Univ. of Wisconsin, Madison, Madison, WI, 2006.

[9] TESS (Thermal Energy Systems Specialists), "TRNSYS Type 534: Vertically cylindrical storage tank with immersed heat exchanger," Madison, WI, 2004.

[10] Solar Energy Laboratory, Univ. of Wisconsin, Madison, "TRNSYS Type 38: Algebraic tank (Plug-flow)," Madison, WI, 2006.

[11] K. E. N'Tsoukpoe, N. Le Pierres and L. Luo, "Numerical dynamic simulation and analysis of a lithium bromide/water long-term solar heat storage system," *Energy,* vol. 37, no. 1, pp. 346-358, 2011.

[12] J. Woods, J. Burch and E. Kozubal, "High solar fraction heating and cooling system based on liquid desiccants," in *Proceedings of the 40th ASES National Solar Conference 2011 (Solar 2011)*, 2011.

[13] L. C. Burmeister, Convective Heat Transfer, New York: Wiley-Interscience, 1993.

[14] M. L. V. Ramires, C. A. Nieto de Castro, Y. Nagasaka, A. Nagashima, M. J. Assael and W. A. Wakeham, "Standard Reference Data for the Thermal Conductivity of Water," American Institute of Physics and American Chemical Society, 1994.

[15] IAPWS, "Revised Release on the IAPWS Formulation 1985 for the Viscosity of Ordinary Water Substance," IAPWS, London, 1997.

A Desiccant Property Equations

Desiccant properties are calculated using curve fits from Conde [4]. Related water properties are from Conde, Ramires et al [14], and IAPWS [15]. All properties are calculated using standard SI units. The TRNSYS code converts the units to TRNSYS preferred units for temperature and time.

A.1 Enthalpy of Dilution

The differential enthalpy of dilution (h_d) is determined using constants C_1 through C_6 [4].

$$h_d(S,T) \cong \left(C_5 + \frac{C_6}{T_{crit}}T\right)\left(1 + \left(\frac{\left(\frac{S}{C_4 - S}\right)}{C_1}\right)^{C_2}\right)^{C_3} \tag{A.1}$$

Table A.1. Constants for Equation (A.1) [4]

	$CaCl_2$	LiCl
C_1	0.855	0.845
C_2	-1.965	-1.965
C_3	-2.265	-2.265
C_4	0.8	0.6
C_5	-955.69	169.105
C_6	3011.974	457.850

A.2 Specific Thermal Capacity

The specific thermal capacity is defined by Conde [4].

$$c_p(S,T) = c_{p_{H2O}}\big(1 - f_I(S) f_{II}(T)\big) \tag{A.2}$$

where

$$c_{p_{H2O}} = A + B\theta^{0.02} + C\theta^{0.04} + D\theta^{0.06} + E\theta^{1.8} + F\theta^{8} \tag{A.3}$$

and

$$\theta = \frac{T}{228} - 1 \tag{A.4}$$

$$f_I(S) = AS + BS^2 + CS^3 \tag{A.5}$$

$$f_I(S) = D + ES, \qquad (only\ for\ LiCl\ when\ S > 0.31) \tag{A.6}$$

$$f_{II}(\theta) = F\theta^{0.02} + G\theta^{0.04} + H\theta^{0.06} \tag{A.7}$$

Table A.2: Constants for Equations (A.3) and (A.5)-(A.7) [4]

	$CaCl_2$	LiCl
A	1.63799	1.43980
B	-1.69002	-1.24317
C	1.05124	-0.12070
D	0.0	0.12825
E	0.0	0.62934
F	58.5225	58.5225
G	-105.6343	-105.6343
H	47.7948	47.7948

A.3 Density

The density is defined by Conde [4].

$$\rho(S,T) = \rho_{H2O}(T)\sum_{i=1}^{3}\rho_i\left(\frac{S}{1-S}\right)^i \tag{A.8}$$

where

$$\rho_{H2O}(\tau) = \rho_{crit,H2O}(1 + B_0\tau^{\frac{1}{3}} + B_1\tau^{\frac{2}{3}} + B_2\tau^{\frac{5}{3}} + B_3\tau^{\frac{16}{3}} + B_4\tau^{\frac{43}{3}} + B_5\tau^{\frac{110}{3}}) \tag{A.9}$$

$$\tau = 1 - \theta \tag{A.10}$$

$$\rho_{crit,H2O} = 322\ kg/m^3 \tag{A.11}$$

Table A.3: Constants for Equation (A.8) [4]

	$CaCl_2$	LiCl
ρ_0	1.0	1.0
ρ_1	0.836014	0.540966
ρ_2	-0.436300	-0.303792
ρ_3	0.105642	0.100791

Table A.4: Constants for Equation (A.9) [4]

i	B_i
0	1.9937718430
1	1.0985211604
2	-0.5094492996
3	-1.7619124270
4	-44.9005480267
5	-723692.2618632

A.4 Thermal Conductivity

The thermal conductivity is defined by Conde [4].

$$k(S,T) = k_{H2O}(T) - \alpha_R \zeta_{eq} \tag{A.12}$$

where

$$\alpha_R = \alpha_0 + \alpha_1 S \tag{A.13}$$

$$\zeta_{eq} = \frac{\rho(S,T) \cdot S \cdot I_s}{M} \tag{A.14}$$

and [14]

$$k_{H2O}(T) = 0.6065\left(-1.48445 + 4.12292\, T^* - 1.63866\, T^{*2}\right), \tag{A.15}$$

$$T^* = \frac{T}{298.15}, \qquad 274K \leq T \leq 370K \tag{A.16}$$

Table A.5: Constants for Equation (A.13) [4]

	$CaCl_2$	LiCl
α_0	0.0059473	0.0108958
α_1	-0.0013988	-0.117882

Note: $I_s = 1$ for $CaCl_2$ and $I_s = 2$ for LiCl; $M = 110.98$ for $CaCl_2$ and $M = 42.39$ for LiCl

A.5 Mass Diffusivity

The mass diffusivity is defined by Conde [4].

$$\mathcal{D}(S,T) = \mathcal{D}_{H2O}(T)\left\{1 - \left[1 + \left(\frac{\sqrt{S}}{\delta_1}\right)^{\delta_2}\right]^{\delta_3}\right\} \tag{A.17}$$

where

$$\mathcal{D}_{H2O}(T) = \frac{A \cdot \tilde{V}_{crit}^{\frac{2}{3}} \cdot R \cdot T}{\mu_{H2O}(T,\rho) \cdot \tilde{V}_L(T)} \tag{A.18}$$

$$A = 0.11353x10^{-16}\ mol^{2/3} \tag{A.19}$$

$$\tilde{V}_L(T) = \frac{M_{H_2O}}{\rho_{H2O}(T)} \tag{A.20}$$

$$R = 8.314\ J/mol \cdot K \tag{A.21}$$

$$\tilde{V}_{crit} = \frac{M_{H_2O}}{\rho_{crit,H2O}} = 5.6x10^{-5}\ m^3/mol \tag{A.22}$$

Table A.6: Constants for Equation (A.17) [4]

	$CaCl_2$	LiCl
δ_1	0.55	0.52
δ_2	-5.52	-4.92
δ_3	-0.56	-0.56

and [15]

$$\mu_{H2O}(T,\rho) = \left(\mu_{crit,H2O}\right)\left(\mu_0(T)\right)\left(\mu_1(T,\rho)\right),\ \ T > 273K \tag{A.23}$$

$$\mu_0(T) = \bar{T}^{0.5}\left(\sum_{i=0}^{3} H_i\,\bar{T}^{-i}\right)^{-1} \tag{A.24}$$

$$\mu_1(T) = \exp\left(\bar{\rho}\sum_{i=0}^{5}\sum_{j=0}^{6} G_{i,j}\,(\bar{T}^{-1}-1)^i(\bar{\rho}-1)^j\right) \tag{A.25}$$

$$\bar{\rho} = \frac{\rho_{H2O}}{\rho_{crit,H2O}} \tag{A.26}$$

$$\bar{T} = \frac{T_{H2O}}{T_{crit,H2O}} \tag{A.27}$$

where $\rho_{crit,H2O} = 317.763\, kg/m^3$ and $T_{crit,H2O} = 647.226\, K$.

Table A.7: Constants for Equation (A.24) [15]

i	H_i
0	1.000
1	0.978197
2	0.579829
3	-0.202354

Table A.8: Constants for Equation (A.25) [15]

i/j	0	1	2	3	4	5	6
0	0.5132047	0.2151778	-0.2818107	0.1778064	-0.0417661	0.0	0.0
1	0.3205656	0.7317883	-1.070786	0.4605040	0.0	-0.01578386	0.0
2	0.0	1.241044	-1.263184	0.2340379	0.0	0.0	0.0
3	0.0	1.476783	0.0	-0.4924179	0.1600435	0.0	-0.003629481
4	-0.7782567	0.0	0.0	0.0	0.0	0.0	0.0
5	0.1885447	0.0	0.0	0.0	0.0	0.0	0.0

B Equation Derivations

B.1 Solutions to Equations of the Form of (3.9) and (3.35)

Given an equation of the form

$$\frac{dX}{dt} = aX + b \tag{B.1}$$

with variable X and constant coefficients a and b, an analytical solution can be obtained. The first step requires multiplying through the terms in Equation (B.1) with the factor e^{-at}, giving

$$e^{-at}\frac{dX}{dt} = aXe^{-at} + be^{-at} \tag{B.2}$$

which can be rearranged in the form

$$\frac{d}{dt}(Xe^{-at}) = be^{-at} \tag{B.3}$$

Taking the indefinite integral of each side of Equation (B.3) gives

$$Xe^{-at} = -\frac{b}{a}e^{-at} + C \tag{B.4}$$

where c is the constant of integration. Rearranging Equation (B.4) to solve for the variable gives

$$X = Ce^{at} - \frac{b}{a} \tag{B.5}$$

Using the initial condition

$$X(t = 0) = X_{initial} \tag{B.6}$$

the constant of integration is found to be

$$C = X_{initial} + \frac{b}{a} \tag{B.7}$$

so that Equation (B.5) becomes

$$X = \left(X_{initial} + \frac{b}{a}\right) e^{at} - \frac{b}{a} \tag{B.8}$$

At the end of one time step, $t = \Delta t$, so that

$$X_{final} = \left(X_{initial} + \frac{b}{a}\right) e^{a\Delta t} - \frac{b}{a} \tag{B.9}$$

which corresponds to Equations (3.12) and (3.35)

The average value of the variable is defined as

$$\bar{X} = \frac{\int_0^{\Delta t} X dt}{\Delta t} \tag{B.10}$$

Substituting in the definition for the variable, as found in Equation (B.8),

$$\bar{X} = \frac{\int_0^{\Delta t} \left[\left(X_{initial} + \frac{b}{a}\right) e^{at} - \frac{b}{a}\right] dt}{\Delta t} \tag{B.11}$$

Integration of Equation (B.11) gives the solution

$$\bar{X} = \frac{\left(X_{initial} + \frac{b}{a}\right)(e^{a\Delta t} - 1) - \frac{b}{a}\Delta t}{a\Delta t} \tag{B.12}$$

which corresponds to Equations (3.13) and (3.39).

B.2 Chemical Potential Energy

The enthalpy of dilution (introduced in Appendix A.1) has units of $kJ/kg - H_2O$. Therefore, a differential change in stored chemical energy results from a differential change in the amount of water contained in the desiccant solution.

$$\delta Q_{ch} = -\delta m_{H2O} \cdot h_d(S,T) \tag{B.13}$$

For a solution with a given mass of salt, since

$$S = \frac{m_{salt}}{m_{salt} + m_{H2O}} \tag{B.14}$$

then

$$m_{H2O} = \frac{m_{salt}}{S} - m_{salt} \tag{B.15}$$

Thus

$$\delta m_{H2O} = -m_{salt} \frac{\delta S}{S^2} \tag{B.16}$$

then

$$\frac{\delta Q_{ch}}{\delta S} = m_{salt} \frac{h_d(S,T)}{S^2} \tag{B.17}$$

Equation (A.1) can be expressed as a product of functions of temperature and salt concentration.

$$h_d(S,T) \cong g_1(T) g_2(S) \tag{B.18}$$

where

$$g_1(T) = \left(C_5 + \frac{C_6}{T_{crit}} T\right) \tag{B.19}$$

$$g_2(S) = \left(1 + \left(\frac{\left(\frac{S}{C_4 - S}\right)}{C_1}\right)^{C_2}\right)^{C_3} \tag{B.20}$$

Substituting Equations (B.18)-(B.20) into Equation (B.17) gives

$$\frac{\delta Q_{ch}}{\delta S} = m_{salt} g_1(T) \frac{g_2(S)}{S^2} \tag{B.21}$$

The total chemical potential energy stored in the solution is determined by the heat of dilution which can be released as the solution (at temperature T_i) is diluted from concentration S_i to 0. Thus, for an arbitrary segment i with salt mass $m_{salt,i}$, temperature T_i, and concentration S_i, this chemical potential is determined by integrating Equation (B.17) between 0 and S_i .

$$\int_0^{S_i} \frac{\delta Q_{ch}}{\delta S} dS = \int_0^{S_i} m_{salt,i} g_1(T_i) \frac{g_2(S)}{S^2} dS \tag{B.22}$$

Since m_{salt} is constant during dilution and $g_1(T_i)$ is independent of S_i,

$$Q_{ch,i} = m_{salt,i} g_1(T_i) \int_0^{S_i} \frac{g_2(S)}{S^2} dS \tag{B.23}$$

Let

$$f_1(S_i) = A_7 S_i^7 + A_6 S_i^6 + A_5 S_i^5 + A_4 S_i^4 + A_3 S_i^3 + A_2 S_i^2 + A_1 S_i \tag{B.24}$$

where A_1 through A_7 are constants for a polynomial curve fit of $\int_0^{S_i} \frac{g_2(S)}{S^2} dS$ (see Table B.1).

Now the chemical potential energy of the desiccant solution at temperature T_i and concentration S_i is expressed by

$$Q_{ch,i} = m_{salt,i} g_1(T_i) f_1(S_i) \tag{B.25}$$

Table B.1: Constants for Equation (B.24)

	$CaCl_2$	LiCl
A_1	109.8986	-466.929
A_2	-183.217	1210.583
A_3	86.252	-1163.04
A_4	-4.65575	484.95
A_5	-0.09467	-74.7
A_6	0.11795	4.89425
A_7	-0.0032	-0.0693

B.3 Sensible Energy Generation

The rate at which sensible energy is generated from chemical potential energy is

$$\dot{Q}_{gen,i} = -\frac{dQ_{ch,i}}{dt} \tag{B.26}$$

where $Q_{ch,i}$ is the chemical potential energy contained in desiccant solution of mass m_i at temperature T_i and concentration S_i. Substitution of Equation (B.19), Equation (B.24), and $m_{salt,i} = m_i S_i$ into Equation (B.25) gives

$$Q_{ch,i} = m_i S_i \left(C_5 + \frac{C_6}{T_{crit}} T_i\right)(A_7 S_i^7 + A_6 S_i^6 + A_5 S_i^5 + A_4 S_i^4 + A_3 S_i^3 + A_2 S_i^2 + A_1 S_i) \tag{B.27}$$

where the coefficients C_5 and C_6 are listed in Table A.1 and the coefficients A_1 through A_7 are listed in Table B.1.

Both T_i and S_i are dependent on time, while m_i is considered to be steady-state. Thus, the time derivative of Equation (B.27) gives

$$\frac{dQ_{ch,i}}{dt} = m_i\left(C_5 + \frac{C_6}{T_{crit}}T_i\right)(8A_7S_i^7 + 7A_6S_i^6 + 6A_5S_i^5 + 5A_4S_i^4 + 4A_3S_i^3 + 3A_2S_i^2 + 2A_1S_i)\frac{dS_i}{dt} + m_i\left(\frac{C_6}{T_{crit}}\right)(A_7S_i^8 + A_6S_i^7 + A_5S_i^6 + A_4S_i^5 + A_3S_i^4 + A_2S_i^3 + A_1S_i^2)\frac{dT_i}{dt} \tag{B.28}$$

By substituting Equation (B.28) into Equation (B.26) and using the approximation

$$\frac{dS_i}{dt} = \frac{S_i^n - S_i^{n-1}}{\Delta t} \tag{B.29}$$

the generation term is written

$$\dot{Q}_{gen,i} = -m_i\left(\left(C_5 + \frac{C_6}{T_{crit}}T_i\right)(8A_7S_i^7 + 7A_6S_i^6 + 6A_5S_i^5 + 5A_4S_i^4 + 4A_3S_i^3 + 3A_2S_i^2 + 2A_1S_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right) + \left(\frac{C_6}{T_{crit}}\right)(A_7S_i^8 + A_6S_i^7 + A_5S_i^6 + A_4S_i^5 + A_3S_i^4 + A_2S_i^3 + A_1S_i^2)\frac{dT_i}{dt}\right) \tag{B.30}$$

Let

$$f_2(S_i) = 8A_7S_i^7 + 7A_6S_i^6 + 6A_5S_i^5 + 5A_4S_i^4 + 4A_3S_i^3 + 3A_2S_i^2 + 2A_1S_i \tag{B.31}$$

$$f_3(S_i) = A_7S_i^8 + A_6S_i^7 + A_5S_i^6 + A_4S_i^5 + A_3S_i^4 + A_2S_i^3 + A_1S_i^2 \tag{B.32}$$

where A_1 through A_7 are given in Table B.1.

Now

$$\dot{Q}_{gen,i} = -m_i\left(\left(C_5 + \frac{C_6}{T_{crit}}T_i\right)f_2(S_i)\left(\frac{S_i^n - S_i^{n-1}}{\Delta t}\right) + \left(\frac{C_6}{T_{crit}}\right)f_3(S_i)\frac{dT_i}{dt}\right) \tag{B.33}$$

B.4 Order of Magnitude Analysis for Diffusion

For an order of magnitude analysis, Equation (3.24) can be approximated with

$$\dot{Q}_{diff} \cong \mathrm{O}\left[\overline{\rho\mathcal{D}} A_c \frac{\Delta S}{\Delta z} c_p \Delta T\right] \tag{B.34}$$

or

$$\begin{aligned} &\dot{Q}_{diff} \\ &\cong \mathrm{O}\left[\left(10^3 \frac{kg}{m^3}\right)\left(10^{-9} \frac{m^2}{s}\right)\left(10^{-1} \frac{kg}{kg}\right)\left(10^3 \frac{J}{kg-K}\right) A_c \frac{\Delta T}{\Delta z}\right] \\ &\cong \mathrm{O}\left[\left(10^{-4} \frac{W}{m-K}\right) A_c \frac{\Delta T}{\Delta z}\right] \end{aligned} \tag{B.35}$$

From Equation (3.20), the energy transferred by conduction from one segment to a neighboring segment can be expressed

$$\dot{Q}_{cond} \cong \mathrm{O}\left[\bar{k} A_c \frac{\Delta T}{\Delta z}\right] \tag{B.36}$$

or

$$\dot{Q}_{cond} \cong \mathrm{O}\left[\left(10^{-1} \frac{W}{m-K}\right) A_c \frac{\Delta T}{\Delta z}\right] \tag{B.37}$$

Since $\dot{Q}_{diff} \ll \dot{Q}_{cond}$, it can be neglected in the energy balance.

C Flow Charts

A standard time step (Figure C.1) includes detailed routines for the closed system analysis (Appendix C.1), the open system analysis (Appendix C.2), density inversion corrections (Appendix C.3), diminutive segment corrections (Appendix C.4), and excess segment corrections (Appendix C.5).

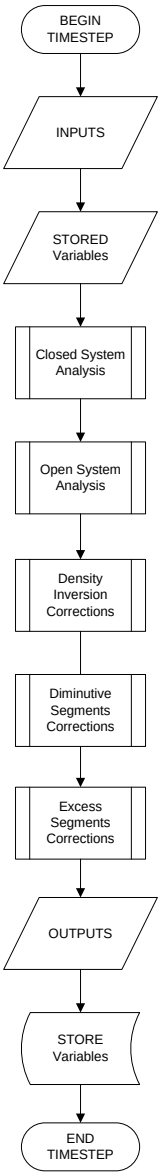

Figure C.1: Main program logic flow chart for a single time step

C.1 Closed System Analysis Routine

The closed system analysis routine (Figure C.2) includes detailed routines for species conservation (Appendix C.1.1) and energy conservation (Appendix C.1.2).

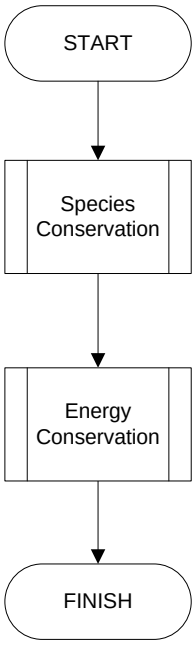

Figure C.2: Processes included in the closed system analysis

C.1.1 Species Conservation

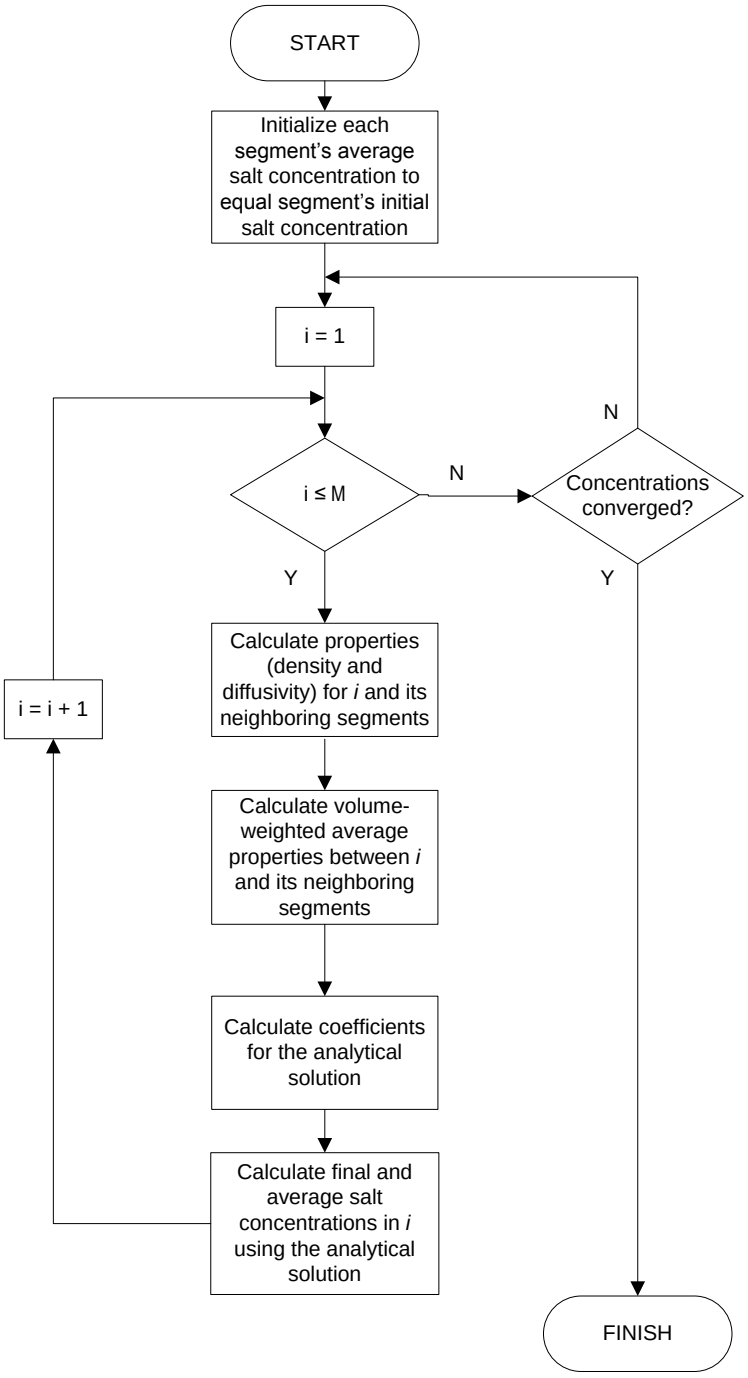

Figure C.3: Routine for the closed system analysis of species conservation

C.1.2 Energy Conservation

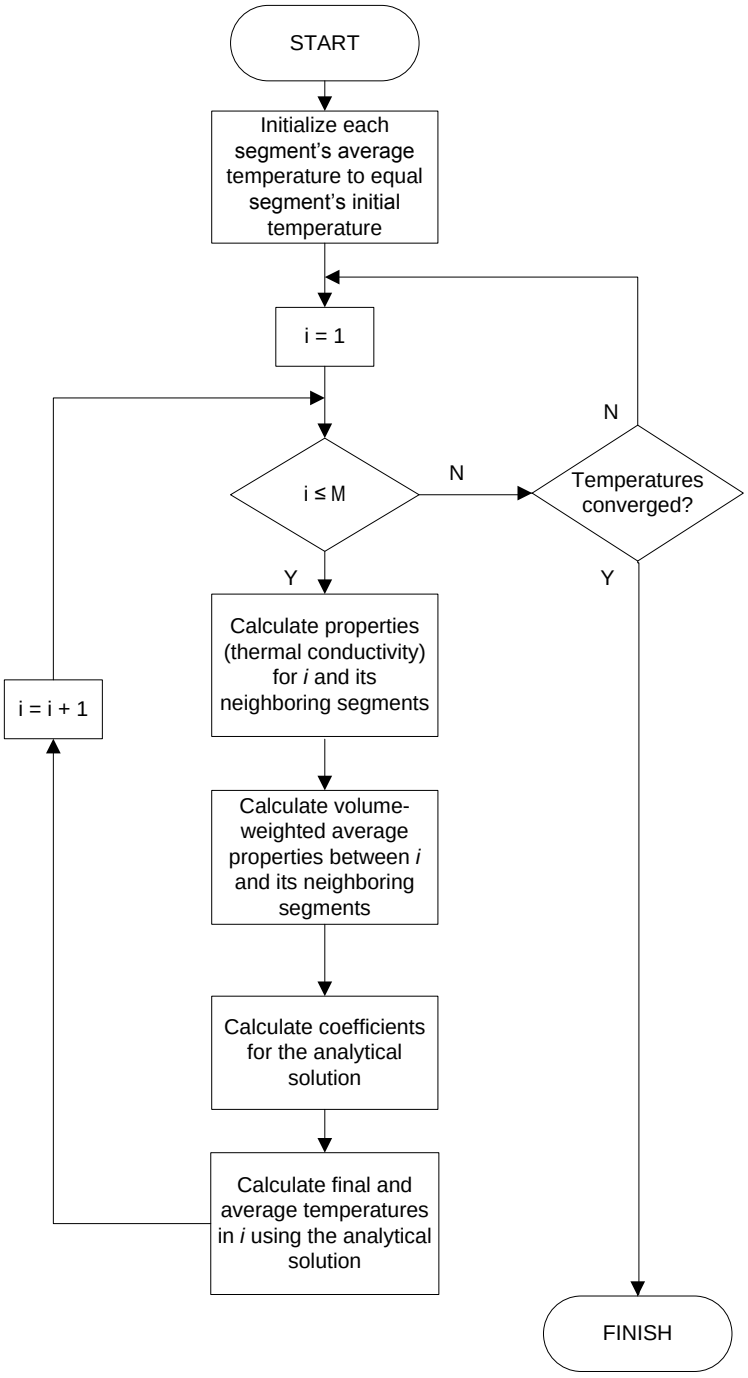

Figure C.4: Routine for the closed system analysis of energy conservation

C.2 Open System Analysis Routine

The open system analysis routine (Figure C.5) includes detailed routines for inflow setup (Appendix C.2.1), Mode 1 inflow (Appendix C.2.2), Mode 2 inflow (Appendix C.2.3), and for outflow and auxiliary (Appendix C.2.4).

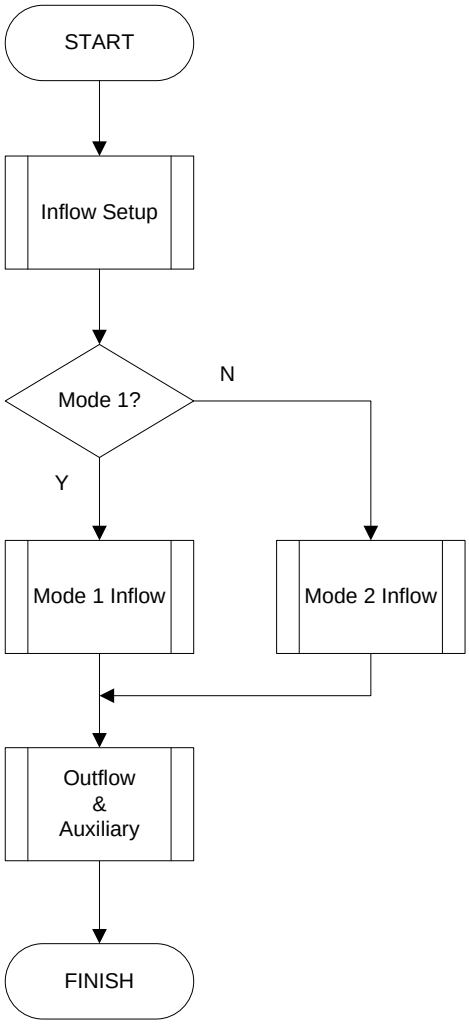

Figure C.5: Processes included in the open system analysis

C.2.1 Inflow Setup

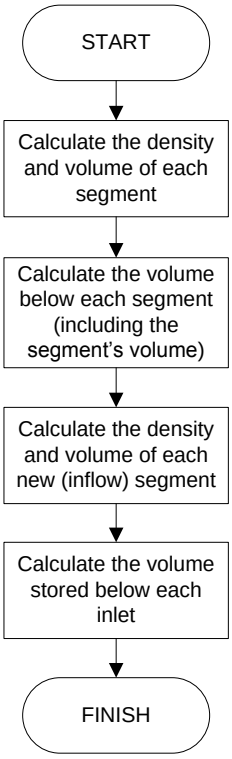

Figure C.6: Routine for the open system analysis of inflow setup

C.2.2 Mode 1 Inflow

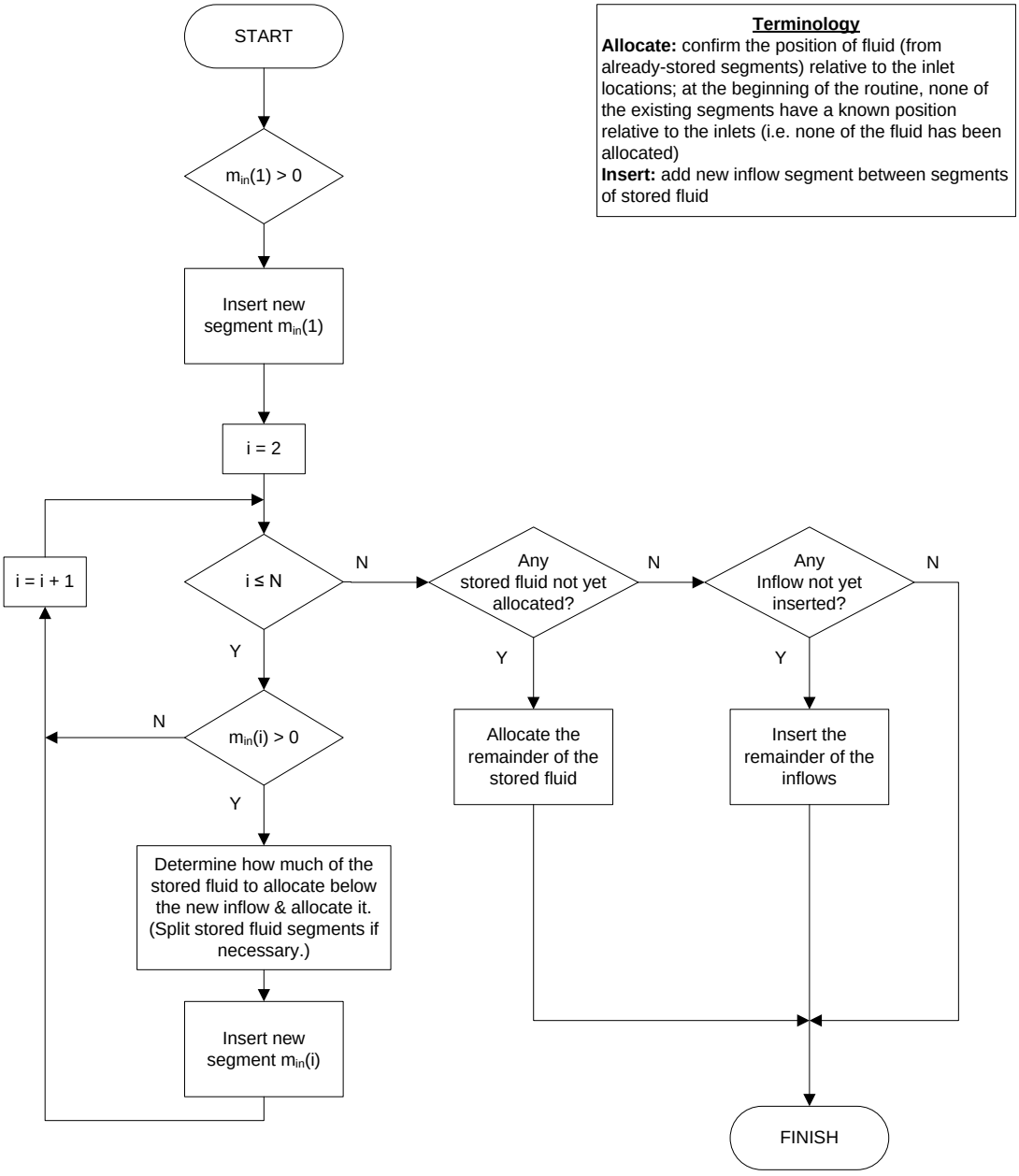

Figure C.7: Routine for the open system analysis of Mode 1 inflow

C.2.3 Mode 2 Inflow

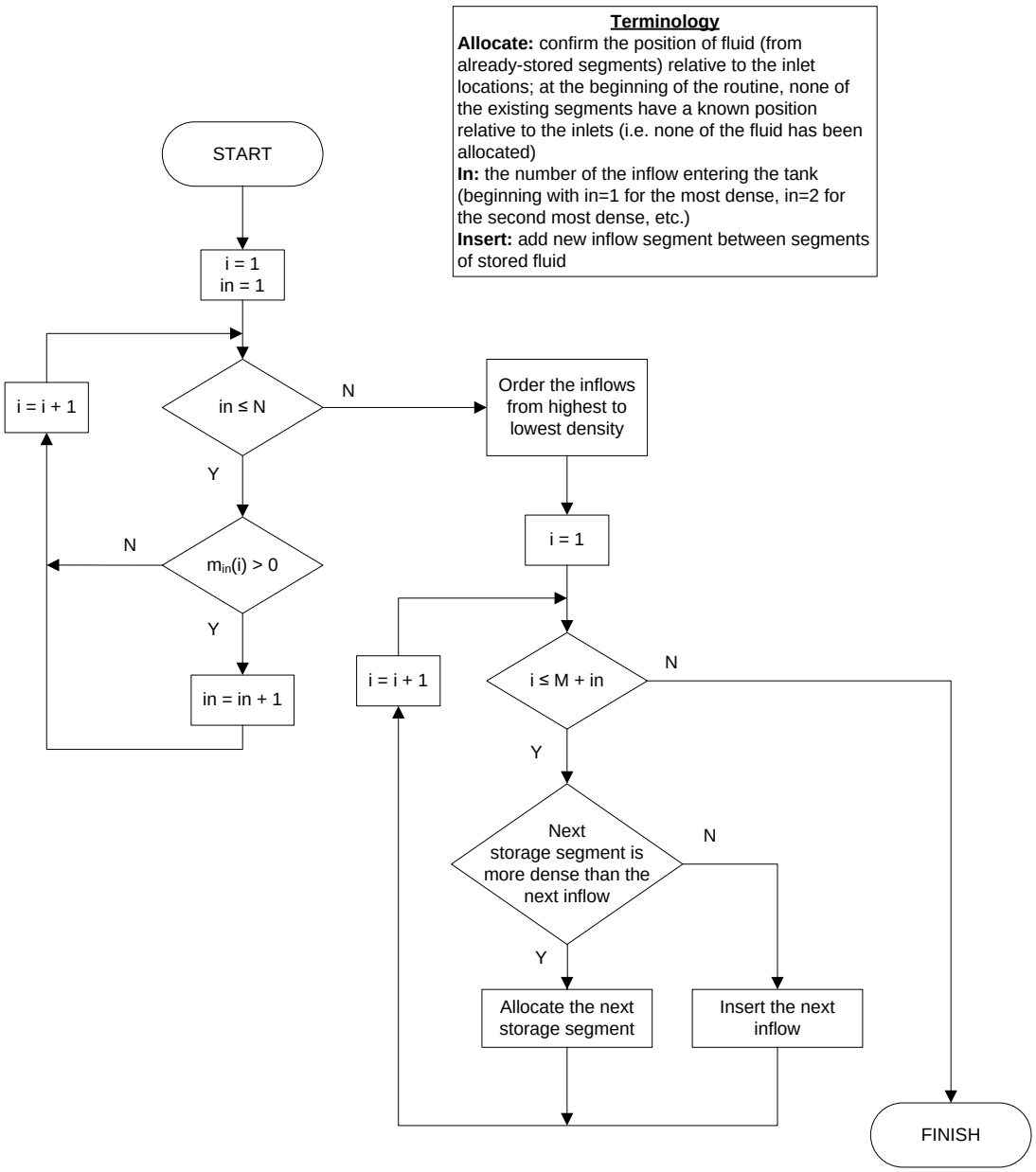

Figure C.8: Routine for the open system analysis of Mode 2 inflow

C.2.4 Outflow and Auxiliary

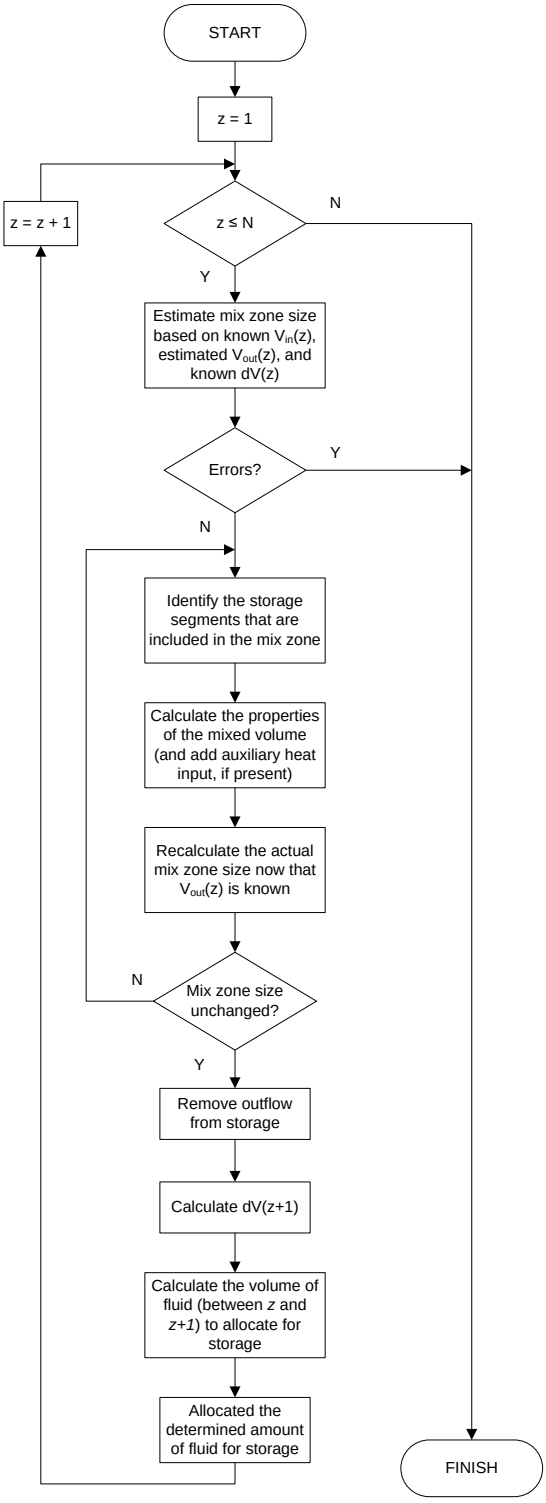

Figure C.9: Routine for the open system analysis of Mode 2 inflow

C.3 Density Inversion Correction Routine

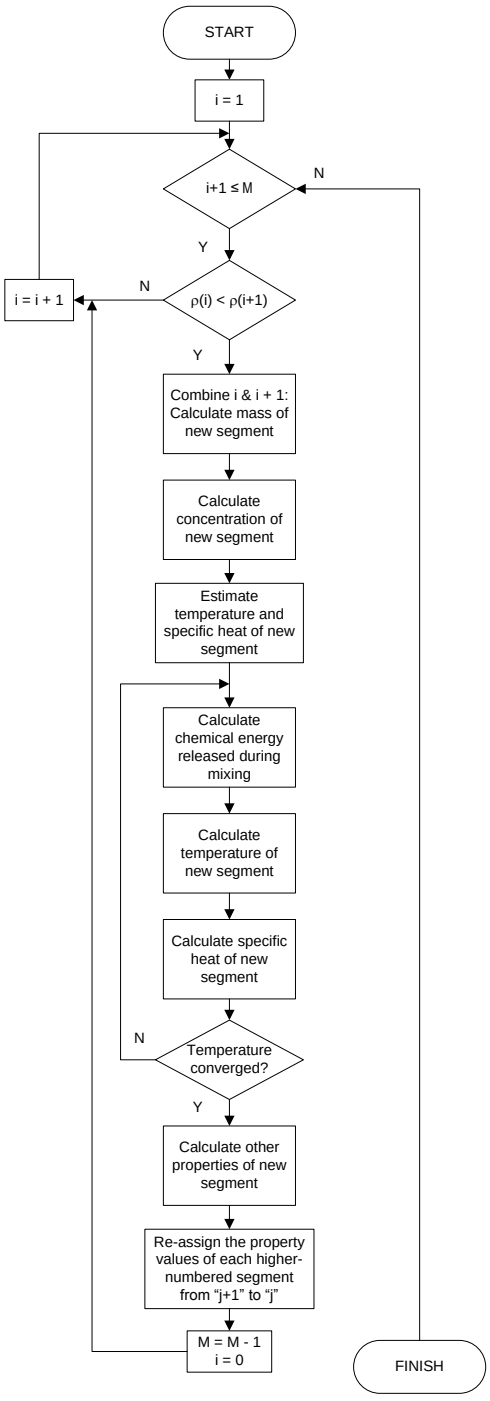

Figure C.10: Routine for correcting density inversions at the end of the time step

C.4 Diminutive Segments Correction Routine

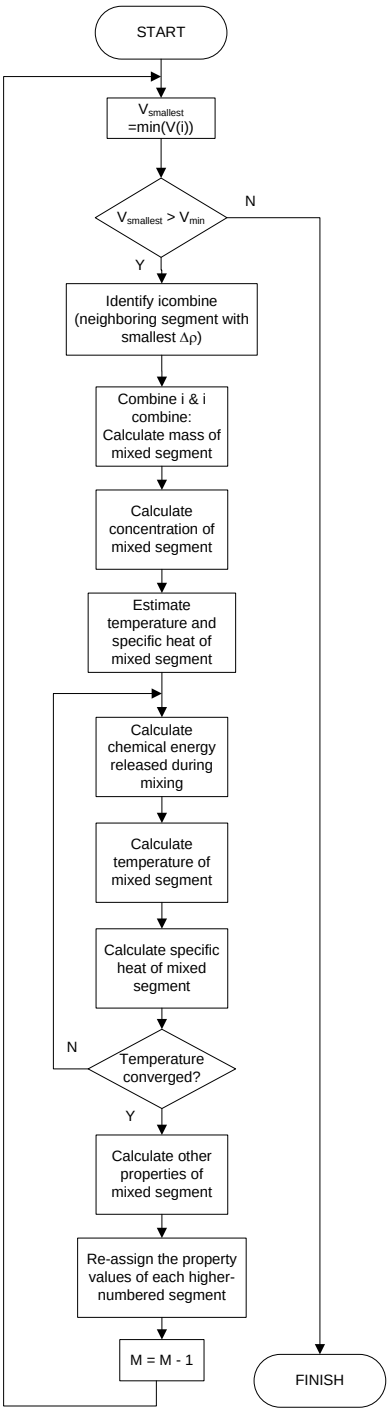

Figure C.11: Routine for combining two storage segments when any segment falls below the minimum permissible volume at the end of the time step

C.5 Excess Segments Correction Routine

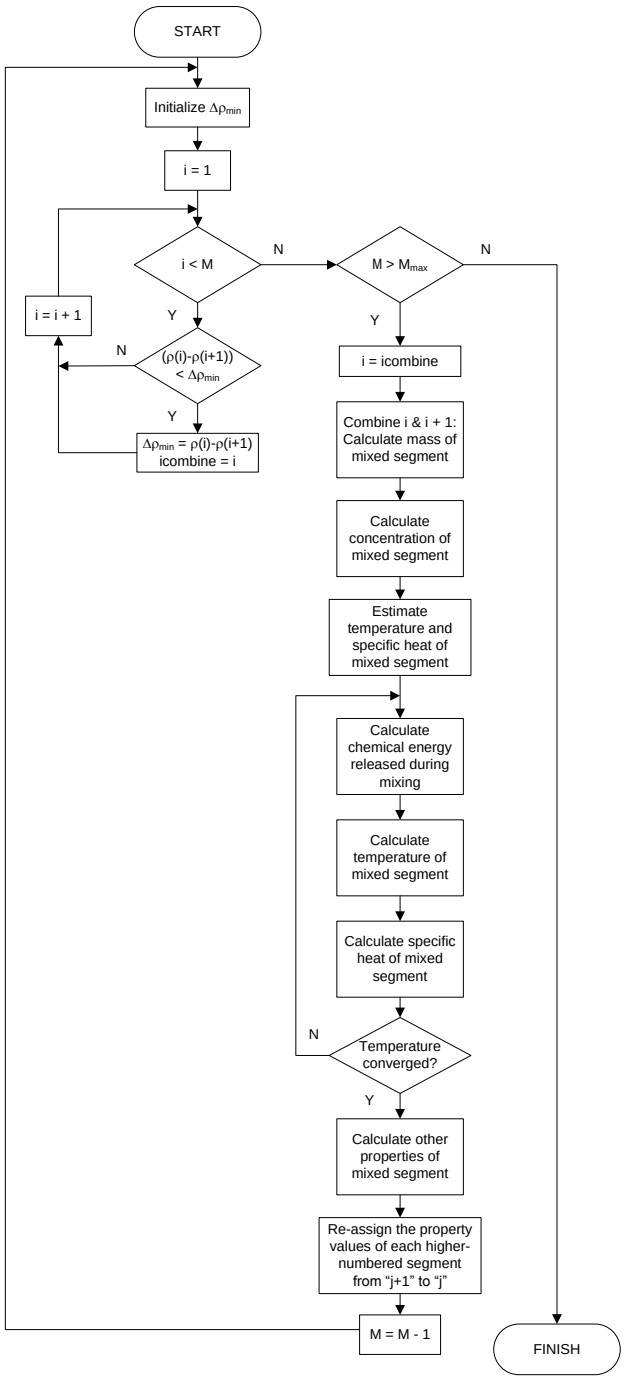

Figure C.12: Routine for reducing the number of storage segments when they exceed the maximum permissible number at the end of the time step

D TRNSYS Type 209 User Tables

D.1 TRNSYS Type 209 Parameters

Table D.1: TRNSYS Type 209 parameters (Part A)

Symbol	Description	Units	Range	Notes
M_0	Initial number of segments	-	[1,50]	
N	Number of positions	-	[1,10]	
V_{max}	Storage capacity	m^3	[0.1,∞)	
z_{max}	Storage height	m^2	[0.1,∞)	
CC	Circumference of cross section	m	[0.1,∞)	1
$salt$	Storage fluid (hydrate salt)	-	[1=$CaCl_2$, 2=LiCl]	
M_{max}	Maximum permissible number of segments	-	[1,50]	
$Mode$	Operational mode for inflow	-	[1=fixed, 2=ideal]	
φ_{min}	Minimum volume fraction of each segment (relative to total storage)	-	[0.000001,1]	

1 – If the entered value is too small, it will be reset to the circumference of a circular tank

Table D.2: TRNSYS Type 209 parameters (Part B)

Symbol	Description	Units	Range	Notes
$U_{L,td}$	Thermal loss coefficient (top, dry)	kJ/hr-m^2-K	[0,∞)	
$U_{L,sw}$	Thermal loss coefficient (side, wet)	kJ/hr-m^2-K	[0,∞)	
$U_{L,sd}$	Thermal loss coefficient (side, dry)	kJ/hr-m^2-K	[0,∞)	
$U_{L,bw}$	Thermal loss coefficient (bottom, wet)	kJ/hr-m^2-K	[0,∞)	
T_{min}	Minimum temperature for usable sensible energy	°C	[0,100]	
S_{min}	Minimum concentration for usable chemical energy	kg-salt/kg-soln	[0,0.6]	
m_0	Initial mass of segment	kg-soln	(0,∞)	1
S_0	Initial concentration of segment	kg-salt/kg-soln	[0,0.6]	1
T_0	Initial temperature of segment	°C	[0,100]	1
z	Height of position	M	[0, ∞)	2,3

1 – Number of parameters matches the value assigned to parameter M_0

2 – Number of parameters matches the value assigned to parameter N

3 – The location of the first position will be re-set to $z_1 = 0$ if a height >0 is entered.

D.2 TRNSYS Type 209 Inputs

Table D.3: TRNSYS Type 209 inputs

Symbol	Description	Units	Range	Notes
$\dot{m}_{in}$	Mass flow rate entering at position	kg-soln/hr	[0,∞)	1
S_{in}	Concentration of flow entering at position	kg-salt/kg-soln	[0,0.6]	1
T_{in}	Temperature of flow entering at position	°C	[0,100]	1
$\dot{m}_{out}$	Mass flow rate exiting at position	kg-soln/hr	[0,∞)	1
$\dot{Q}_{aux}$	Auxiliary heat rate entering at position	kJ/hr	[0,∞)	1
$T_{amb,t}$	Ambient temperature at the top wall of the storage	°C	[0,100]	
$T_{amb,s}$	Ambient temperature at the side wall of the storage	°C	[0,100]	
$T_{amb,b}$	Ambient temperature at the bottom wall of the storage	°C	[0,100]	

1 – Number of inputs matches the value assigned to parameter N

D.3 TRNSYS Type 209 Outputs

Table D.4: TRNSYS Type 209 outputs (Part A)

Symbol	Description	Units	Range	Notes
T_{out}	Temperature of flow exiting at position	°C	[0,100]	1
S_{out}	Concentration of flow exiting at position	kg-salt/kg-soln	[0,100]	1
$Q_{loss,t}$	Heat lost to the ambient through the top wall	kJ	[0,∞)	
$Q_{loss,s}$	Heat lost to the ambient through the side wall	kJ	[0,∞)	
$Q_{loss,b}$	Heat lost to the ambient through the bottom wall	kJ	[0,∞)	
$Q_{loss,tot}$	Total heat lost to the ambient	kJ	[0,∞)	
$Q_{aux,tot}$	Total auxiliary heat added	kJ	[0,∞)	
$m_{in,tot}$	Total mass added	kg-soln	[0,∞)	
$m_{salt\,in,tot}$	Total salt mass added	kg-salt/kg-soln	[0,∞)	
$E_{in,tot}$	Total energy added	kJ	[0,∞)	
$m_{out,tot}$	Total mass removed	kg-soln	[0,∞)	
$m_{salt\,out,tot}$	Total salt mass removed	kg-salt/kg-soln	[0,∞)	
$E_{out,tot}$	Total energy removed	kJ	[0,∞)	
M	Total number of segments	-	[1,50]	
V_i	Volume of segment	m^3	(0,∞)	2
ρ_i	Density of segment	kg-soln/m^3	(0,∞)	2
m_i	Mass of segment	kg-soln	(0,∞)	2
S_i	Concentration of segment	kg-salt/kg-soln	[0,0.6]	2
T_i	Temperature of segment	°C	[0,100]	2
$E_{se,u,i}$	Usable sensible energy of segment	kJ	[0,∞)	2
$E_{ch,u,i}$	Usable chemical energy of segment	kJ	[0,∞)	2
$E_{u,i}$	Total usable energy of segment	kJ	[0,∞)	2

1 – Number of outputs matches the value assigned to parameter N

2 – Number of outputs matches the number of segments (M)

Table D.5: TRNSYS Type 209 outputs (Part B)

Symbol	Description	Units	Range	Notes
V_{tot}	Total stored volume	m^3	(0,∞)	
m_{tot}	Total stored mass	kg-soln	(0,∞)	
$m_{salt,tot}$	Total stored salt mass	kg-salt	[0,∞)	
$E_{se,u,tot}$	Total stored usable sensible energy	kJ	[0,∞)	1
$E_{ch,u,tot}$	Total stored usable chemical energy	kJ	[0,∞)	1
$E_{u,tot}$	Total stored usable energy	kJ	[0,∞)	1
T_{pos}	Temperature at position	°C	[0,100]	2
m_{ERR}	Absolute error in the mass balance	kg-soln	(-∞,∞)	
$m_{\%ERR}$	Percent error in the mass balance (relative to total storage)	%	(-∞,∞)	
$m_{salt,ERR}$	Absolute error in the salt mass balance	kg-salt	(-∞,∞)	
$m_{salt,\%ERR}$	Percent error in the salt mass balance (relative to total storage)	%	(-∞,∞)	
E_{ERR}	Absolute error in the energy balance	kJ	(-∞,∞)	
$E_{\%ERR}$	Percent error in the energy balance (relative to total storage)	%	(-∞,∞)	

1 – Only positive values of usable energy are included

2 – Can be used with a controller to determine whether auxiliary heat is desired at that location

D.4 TRNSYS Type 209 Error & Warning Messages

Messages have been added to the TRNSYS code to identify the causes for various errors and other processing events of note. These are listed in Table D.6-Table D.9. The simulation terminates if a FATAL occurs. The code continues processing if a WARNING occurs. However, the error may be significant, and the user should investigate the cause of the WARNING as well as the magnitude of the errors that may have been introduced. It is also important to note that TRNSYS limits the number of WARNINGs that are permitted in the course of a simulation. If this limit is exceeded, the simulation terminates. The code continues processing if a NOTICE occurs. The issuing of a NOTICE may not indicate an error. However, it does indicate a situation that the user should be aware of and possibly seek a remedy for.

Table D.6: TRNSYS messages for Type 209 (Part A)

#	Severity	Message	Diagnosis / Solution
1	FATAL	The current TRNSYS timestep is less than the minimum timestep set in this model. Either increase the TRNSYS timestep or decrease the minimum timestep allowed by this model (DELT_MIN).	The time step must be increased.
2	WARNING	Height of position 1 re-set to 0 m.	The code assumes that the first position is at the base of the tank (even if it is never used). The code therefore automatically corrects the height of the first position to zero if it was first set to another value.
3	FATAL	Position is not located above the lower-numbered positions.	Each position must be located at a height greater than all lower-numbered positions.

Table D.7: TRNSYS messages for Type 209 (Part B)

#	Severity	Message	Diagnosis / Solution
4	FATAL	Position is not located within the tank height.	Each position must be located within the tank.
5	FATAL	Initial storage profile is not stably stratified.	The tank will not run unless the parameters entered for the initial storage conditions constitute a stably stratified storage.
6	WARNING	Specified circumference too low for other geometry specifications. Circumference re-set to that of a circular cross section.	The circumference measurement has a lower limit: that of a circle corresponding to the specified tank volume and height. If the circumference is defined below this minimum value, it is re-set to the minimum value.
7	FATAL	Initial number of segments greater than specified maximum.	The initial storage profile must not be composed of more than the maximum permissible number of segments.
8	WARNING	Maximum iterations performed for closed system species conservation analysis. Solution did not converge.	After one thousand iterations, the solution did not converge. The species balance for this time step should be checked for significant errors.
9	WARNING	Maximum iterations performed for closed system energy conservation analysis. Solution did not converge.	After one thousand iterations, the solution did not converge. The energy balance for this time step should be checked for significant errors.
10	FATAL	No fluid available at location of required outflow.	The user specified that flow be drawn from an outlet at which there is no fluid.
11	FATAL	No fluid available at location of auxiliary input.	The user specified that auxiliary heat be added at a location where there is no fluid.

Table D.8: TRNSYS messages for Type 209 (Part C)

#	Severity	Message	Diagnosis / Solution
12	NOTICE	No fluid remains, but this is okay at this time step since no outflow or aux is currently required at this location.	The fluid level has fallen below one or more positions. If outflow had been required from such a position at this time step, an error (Msg#10) would have occurred.
13	FATAL	Tank lacks sufficient fluid to provide the full outflow requirement.	While some fluid is present at the height of the outlet, it is not sufficient to match the required outflow mix zone.
14	FATAL	m_dot*dt is too large for port spacing	The fluid level during this time step is falling at a rate too great for the port spacing. (Fluid required for the mix zone at the current position has already been included in the mix zone for the previous position.) Solutions: (1) decrease the time step, (2) decrease the flow rates, or (3) increase the spacing between the positions.
15	WARNING	Temperature did not converge, likely due to Qaux being too large for fluid volume. Qaux reduced by 10% and Tout recalculated.	The magnitude of the auxiliary heat was too large for the size of the fluid segment in which the heater is located. The auxiliary heat input for the time step was reduced by 10% in order to prevent the temperature of the segment from becoming too large. Solutions: (1) increase the minimum volume fraction for each segment, (2) decrease the time step, or (3) decrease the auxiliary heat rate.

Table D.9: TRNSYS messages for Type 209 (Part D)

#	Severity	Message	Diagnosis / Solution
16	WARNING	Maximum iterations performed for open system plug flow. Solution did not converge.	After one thousand iterations, the solution (for the volume of the mix zone) did not converge. The energy balance for this time step should be checked for significant errors.
17	WARNING	Maximum iterations performed for segment mixing. Solution did not converge.	After one thousand iterations, the solution did not converge. The energy balance for this time step should be checked for significant errors.
18	NOTICE	Total fluid volume exceeds tank volume.	The fluid volume is greater than the specified tank capacity. (No processing errors result.)

E TRNSYS Type 209 Fortran Code

E.1 Type 209.for

```
                  SUBROUTINE TYPE209
(TIME,XIN,OUT,T,DTDT,PAR,INFO,ICNTRL,*)

C-----------------------------------------------------------------------
C     THIS SUBROUTINE MODELS A VERTICAL STORAGE TANK WITH CONSTANT
C     CROSS SECTION
C
C     FEATURES:
C        - PLUG FLOW MODEL WITH USER-SELECTED POSITIONS FOR INLETS,
C          OUTLET, AUX HEATERS
C        - VARIABLE VOLUME STORAGE (UNPAIRED INLETS AND OUTLETS)
C        - TWO MODES FOR INLET FLOW: FIXED & IDEAL (INFLOW ENTERS AT
C          NEUTRAL BUOYANCY LOCATION
C        - NEW EDGE AND TOP LOSSES SO USERS DEFINE WET & DRY LOSS
C          COEFFICIENTS
C
C     CARE SHOULD BE TAKEN WHEN CONSIDERING SMALL SEGMENTS WITH
C     RELATIVELY LARGE TIME STEPS AS THIS CAN CAUSE AN ENERGY BALANCE
C     PROBLEM WITH THE MODEL.
C-----------------------------------------------------------------------
C    CODE DESCRIPTION:
C     For coding purposes, the storage within a single physical tank is
C     modeled as being transferred between Versions of the Storage
C     Profile within each time step.
C     SPv0 - The initial storage profile
C     SPv1 - The storage profile after the closes system analysis
C     SPv2 - The storage profile after inflow
C     SPv3 - The storage profile during outflow; It is slowly "emptied"
C            due to a combination of outflow and the transfer of fluid
C            to SPv4
C     SPv4 - The final storage profile (after outflow, auxiliary, and
C            any required mixing)
C-----------------------------------------------------------------------
C     Author: Jason Mallinak
C     Date: August 13, 2012
C     Copyright © 2012 Solar Energy Lab, University of Minnesota.
C     All rights reserved.
C-----------------------------------------------------------------------
C    REQUIRED BY THE MULTI-DLL VERSION OF TRNSYS
      !DEC$ATTRIBUTES DLLEXPORT :: TYPE209
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C    ACCESS TRNSYS FUNCTIONS
      USE TrnsysConstants
      USE TrnsysFunctions
C-----------------------------------------------------------------------
```

```
C    ACCESS PROGRAM FUNCTIONS
      USE properties
      USE ec
C-----------------------------------------------------------------------
C    TRNSYS DECLARATIONS
      IMPLICIT NONE          !REQUIRES THE USER TO DEFINE ALL VARIABLES
                             !BEFORE USING THEM
      DOUBLE PRECISION XIN   !THE ARRAY FROM WHICH THE INPUTS TO THIS
                             !TYPE WILL BE RETRIEVED
      DOUBLE PRECISION OUT   !THE ARRAY WHICH WILL BE USED TO STORE THE
                             !OUTPUTS FROM THIS TYPE
      DOUBLE PRECISION TIME !THE CURRENT SIMULATION TIME - YOU MAY USE
                             !THIS VARIABLE BUT DO NOT SET IT!
      DOUBLE PRECISION PAR   !THE ARRAY FROM WHICH THE PARAMETERS FOR
                             !THIS TYPE WILL BE RETRIEVED
      DOUBLE PRECISION STORED !THE STORAGE ARRAY FOR HOLDING VARIABLES
                             !FROM TIMESTEP TO TIMESTEP
      DOUBLE PRECISION T     !AN ARRAY CONTAINING THE RESULTS FROM
                             !THE DIFFERENTIAL EQUATION SOLVER
      DOUBLE PRECISION DTDT !AN ARRAY CONTAINING THE DERIVATIVES TO BE
                             !PASSED TO THE DIFF.EQ. SOLVER
      INTEGER*4 INFO(15)     !THE INFO ARRAY STORES AND PASSES
                             !VALUABLE INFORMATION TO AND FROM THIS TYPE
      INTEGER*4 NP,NI,NOUT,ND !VARIABLES FOR THE MAXIMUM NUMBER OF
                             !PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
      INTEGER*4 NPAR,NIN,NDER !VARIABLES FOR THE CORRECT NUMBER OF
                             !PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
      INTEGER*4 IUNIT,ITYPE !THE UNIT NUMBER AND TYPE NUMBER FOR THIS
                             !COMPONENT
      INTEGER*4 ICNTRL       !AN ARRAY FOR HOLDING VALUES OF CONTROL
                             !FUNCTIONS WITH THE NEW SOLVER
      INTEGER*4 NSTORED      !THE NUMBER OF VARIABLES THAT WILL BE
                             !PASSED INTO AND OUT OF STORAGE
      CHARACTER*3 OCHECK     !AN ARRAY TO BE FILLED WITH THE CORRECT
                             !VARIABLE TYPES FOR THE OUTPUTS
      CHARACTER*3 YCHECK     !AN ARRAY TO BE FILLED WITH THE CORRECT
                             !VARIABLE TYPES FOR THE INPUTS
C-----------------------------------------------------------------------
      INTEGER MaxSeg, MaxPos
      PARAMETER (MaxSeg=100)
      PARAMETER (MaxPos=10)
C-----------------------------------------------------------------------
C    USER DECLARATIONS - SET THE MAXIMUM NUMBER OF PARAMETERS (NP),
C    INPUTS (NI),OUTPUTS (NOUT), AND DERIVATIVES (ND)
C    THAT MAY BE SUPPLIED FOR THIS TYPE
      PARAMETER (NP=15+3*MaxSeg+MaxPos, NI=5*MaxPos+3,
     &  NOUT=2*MaxPos+12+8*MaxSeg+6+MaxPos+6, ND=0, NSTORED=1+3*MaxSeg)
      INTEGER NI_DECK, NP_DECK
C-----------------------------------------------------------------------
C    REQUIRED TRNSYS DIMENSIONS
      DIMENSION XIN(NI),OUT(NOUT),PAR(NP),YCHECK(NI),OCHECK(NOUT),
     1   STORED(NSTORED),T(ND),DTDT(ND)
      INTEGER NITEMS
C-----------------------------------------------------------------------
C    DECLARATIONS AND DEFINITIONS FOR THE USER-VARIABLES
```

```
C     PARAMETERS
      DOUBLE PRECISION dt
      INTEGER desiccant
      DOUBLE PRECISION, dimension(MaxSeg) :: m_init, S_init, T_init
      DOUBLE PRECISION, dimension(MaxPos) :: hp
      DOUBLE PRECISION U_L_top, U_L_sid, U_L_sid_d, U_L_btm
      DOUBLE PRECISION CC, H, Vmax, Vminfraction, no_longer_used
      DOUBLE PRECISION Smin, Tmin
      INTEGER MM_init
      INTEGER PP
      INTEGER Mode
      INTEGER imax

C     INPUTS
      DOUBLE PRECISION, dimension(MaxPos) :: mi, Si, Ti
      DOUBLE PRECISION, dimension(MaxPos) :: mout
      DOUBLE PRECISION, dimension(MaxPos) :: Qaux
      DOUBLE PRECISION Tenv_top, Tenv_btm, Tenv_sid

C      INTERNAL VARIABLES

      DOUBLE PRECISION totaltime, time_steps, step
      DOUBLE PRECISION, dimension(MaxSeg) :: m0, S0, T0
      DOUBLE PRECISION, dimension(MaxSeg) :: m1, S1, T1
      DOUBLE PRECISION, dimension(MaxSeg) :: m2, S2, T2
      DOUBLE PRECISION, dimension(MaxSeg) :: m3, S3, T3
      DOUBLE PRECISION, dimension(MaxSeg) :: m4, S4, T4
      DOUBLE PRECISION, dimension(MaxPos) :: Sout, Tout, Tprobe

      DOUBLE PRECISION, dimension(MaxSeg) :: V0, rho0, cp0
      DOUBLE PRECISION, dimension(MaxSeg) :: V1, rho1, cp1
      DOUBLE PRECISION, dimension(MaxSeg) :: V2, rho2, cp2
      DOUBLE PRECISION, dimension(MaxSeg) :: V3, rho3, cp3
      DOUBLE PRECISION, dimension(MaxSeg) :: V4, rho4, cp4
      DOUBLE PRECISION, dimension(MaxPos) :: Vi, rhoi, cpi
      DOUBLE PRECISION, dimension(MaxPos) :: Vout, rhoout, cpout

      DOUBLE PRECISION drhomin
      DOUBLE PRECISION, dimension(MaxSeg) :: h0, h0below, h4, h4below
      DOUBLE PRECISION, dimension(MaxPos) :: dV

      DOUBLE PRECISION Ac, hdry, Awdry, CC0

      DOUBLE PRECISION, dimension(MaxSeg) :: Qloss
      DOUBLE PRECISION Qloss_sid, Qloss_top, Qloss_btm, Qloss_tot

      DOUBLE PRECISION, dimension(MaxSeg) :: Savg, Sf
      DOUBLE PRECISION, dimension(MaxSeg) :: S_new, Savg_new
      DOUBLE PRECISION, dimension(MaxSeg) :: rho, Diff, halfh
      DOUBLE PRECISION, dimension(MaxSeg) :: rhoD_below, rhoD_above
      DOUBLE PRECISION, dimension(MaxSeg) :: aa_below, aa_above, aa
      DOUBLE PRECISION, dimension(MaxSeg) :: bb_below, bb_above, bb

      DOUBLE PRECISION, dimension(MaxSeg) :: Tavg, Tf
      DOUBLE PRECISION, dimension(MaxSeg) :: T_new, Tavg_new
      DOUBLE PRECISION, dimension(MaxSeg) :: kcond_below,kcond_above
```

```
DOUBLE PRECISION, dimension(MaxSeg) :: cpavg, cpprev, cpf
DOUBLE PRECISION, dimension(MaxSeg) :: kcond

DOUBLE PRECISION, dimension(MaxSeg) :: f_Savg1, f_Savg2, f_Savg3
DOUBLE PRECISION, dimension(MaxSeg) :: aaf, aafchem
DOUBLE PRECISION, dimension(MaxSeg) :: bbf, bbfchem
DOUBLE PRECISION, dimension(MaxSeg) :: Aw
DOUBLE PRECISION Sref, Tref
DOUBLE PRECISION c5, c6, Tcrit

DOUBLE PRECISION m3partial, V3partial
DOUBLE PRECISION, dimension(MaxSeg) :: mcontrib
DOUBLE PRECISION, dimension(MaxSeg) :: Vmixnew, mmix
DOUBLE PRECISION, dimension(MaxSeg) :: mremain, mleftover,mmix_t
DOUBLE PRECISION, dimension(MaxPos) :: mnext, Snext, Tnext
DOUBLE PRECISION, dimension(MaxPos) :: Vnext, rhonext
DOUBLE PRECISION, dimension(MaxPos) :: Toutnew
DOUBLE PRECISION, dimension(MaxPos) :: Esensibleout
DOUBLE PRECISION, dimension(MaxPos) :: Echemout1, Echemout2
DOUBLE PRECISION, dimension(MaxPos) :: f_S1, f_S2, Egen
DOUBLE PRECISION, dimension(MaxSeg) :: m4mixed, S4mixed, T4mixed
DOUBLE PRECISION, dimension(MaxSeg) :: T4mixednew
DOUBLE PRECISION, dimension(MaxSeg) :: cp4mixed, Egenmixed
DOUBLE PRECISION, dimension(MaxSeg) :: f_S4, f_S4mixed
DOUBLE PRECISION, dimension(MaxSeg) :: Echem4mixed, Echem4

DOUBLE PRECISION, dimension(MaxSeg) :: Vthruseg
DOUBLE PRECISION, dimension(MaxPos) :: Vbelowp
DOUBLE PRECISION, dimension(MaxPos) :: Vbelowin, Vmix, Vkeep
DOUBLE PRECISION Vbelowrecentp, V4tot, V1thru, Vmove
DOUBLE PRECISION Vsum, msum, Vsmallest
DOUBLE PRECISION, dimension(MaxPos) :: msmix
DOUBLE PRECISION m1tot, m2tot, m3tot, mitot, mouttot, m4tot,m4old
DOUBLE PRECISION m1salt, m2salt, m3salt, misalt, mosalt, m4salt
DOUBLE PRECISION m4saltold
DOUBLE PRECISION m0tot, m0salt, V0tot

DOUBLE PRECISION m3totprev, m4totprev, moutremain, mtankremain
DOUBLE PRECISION Qauxremain, T_closest, dT_smallest, cp4_closest
DOUBLE PRECISION m_above, m_outtot, m_4tot, m_everything

DOUBLE PRECISION m0tot_t0, m0salt_t0, Et0_t0
DOUBLE PRECISION mitot_all, misalt_all, Eti_all
DOUBLE PRECISION motot_all, mosalt_all, Eto_all
DOUBLE PRECISION Qtaux_all, Qloss_tot_all

DOUBLE PRECISION Et0, Et1, Eti, Et2, Eto, Et3, Et4, Et4old
DOUBLE PRECISION Et0u, Et4u
DOUBLE PRECISION Ese1_t, Esei_t, Ese2_t, Eseo_t, Ese3_t, Ese4_t
DOUBLE PRECISION Ech1_t, Echi_t, Ech2_t, Echo_t, Ech3_t, Ech4_t
DOUBLE PRECISION Ese0u_t, Ech0u_t, Ese4u_t, Ech4u_t
DOUBLE PRECISION Ese0_t, Ech0_t
DOUBLE PRECISION Qtaux, Qaux_on
DOUBLE PRECISION, dimension(MaxSeg) :: fS0, Ese0, Ech0
DOUBLE PRECISION, dimension(MaxSeg) :: fS0u, Ese0u, Ech0u
DOUBLE PRECISION, dimension(MaxSeg) :: fS1, Ese1, Ech1
```

```
      DOUBLE PRECISION, dimension(MaxPos) :: fSi, Esei, Echi
      DOUBLE PRECISION, dimension(MaxSeg) :: fS2, Ese2, Ech2
      DOUBLE PRECISION, dimension(MaxPos) :: fSo, Eseo, Echo
      DOUBLE PRECISION, dimension(MaxSeg) :: fS3, Ese3, Ech3
      DOUBLE PRECISION, dimension(MaxSeg) :: fS4, Ese4, Ech4
      DOUBLE PRECISION, dimension(MaxSeg) :: fS4u, Ese4u, Ech4u

      DOUBLE PRECISION Mass_ERR, Salt_ERR, Energy_ERR
      DOUBLE PRECISION Mass_pctERR, Salt_pctERR, Energy_pctERR


      INTEGER MM_prev, MM
      INTEGER i, iter, maxiters, imin
      INTEGER p, pnext
      INTEGER i1next, innext, inserted
      INTEGER, dimension(MaxPos) :: loc
      INTEGER in, j
      INTEGER seg0, seg1, seg2, seg3, seg4
      INTEGER icombine
      INTEGER emptied, partial, leftover
      INTEGER emptied_3to4, partial_3to4
      INTEGER, dimension(MaxSeg) :: s_iters, T_iters
      INTEGER, dimension(MaxSeg) :: T_iters_2, Tmixed_iters
      INTEGER, dimension(MaxPos) :: V_iters

      LOGICAL s_converge, s_notconv, T_converge, T_notconv
      LOGICAL T_converge_2, V_converge, done, Tmixed_converge, error

      DOUBLE PRECISION PI, CONVERGED, DELT_MIN

      CHARACTER (LEN=MaxmessageLength) MESSAGE1,MESSAGE2,MESSAGE3
      CHARACTER (LEN=MaxmessageLength) MESSAGE4,MESSAGE5,MESSAGE6
      CHARACTER (LEN=MaxmessageLength) MESSAGE7,MESSAGE8,MESSAGE9
      CHARACTER (LEN=MaxmessageLength) MESSAGE10,MESSAGE11,MESSAGE12
      CHARACTER (LEN=MaxmessageLength) MESSAGE13,MESSAGE14,MESSAGE15
      CHARACTER (LEN=MaxmessageLength) MESSAGE16,MESSAGE17,MESSAGE18
      CHARACTER (LEN=MaxmessageLength) MESSAGE19,MESSAGE20,MESSAGE21
C-----------------------------------------------------------------------
C     DATA STATEMENTS
      DATA PI/3.14159265358979/,CONVERGED/0.000001/,DELT_MIN/0.001/
      MESSAGE1='The current TRNSYS timestep is less than the minimum
     &timestep set in this model.  Either increase the TRNSYS timestep
     &  or decrease the minimum timestep allowed by this model
     &(DELT_MIN).'
      MESSAGE2='Height of position 1 re-set to 0 m.'
      MESSAGE3='Position is not located above the lower-numbered
     &positions.'
      MESSAGE4='Position is not located within the tank height.'
      MESSAGE5='Initial storage profile is not stably stratified.'
      MESSAGE6='Specified circumference too low for other geometry
     &  specifications. Circumference re-set to that of a circular
     &  cross section.'
      MESSAGE7='Initial number of segments greater than specified
     &maximum.'
      MESSAGE8 = 'Maximum iterations performed for closed system
     &species conservation analysis. Solution did not converge.'
```

```
      MESSAGE9 = 'Maximum iterations performed for closed system energy
     & conservation analysis. Solution did not converge.'
      MESSAGE10 = 'No fluid available at location of required outflow.'
      MESSAGE11 = 'No fluid available at location of auxiliary input.'
      MESSAGE12 = 'No fluid remains, but this is okay at this time step
     & since no outflow or aux is currently required at this location.'
      MESSAGE13 = 'Tank lacks sufficient fluid to provide the full
     & outflow requirement.'
      MESSAGE14 ='m_dot*dt is too large for port spacing'
      MESSAGE15 = 'Temperature did not converge, likely due to
     & Qaux being too large for fluid volume. Qaux reduced by 10% and
     & Tout recalculated.'
      MESSAGE16 = 'Maximum iterations performed for open system plug
     & flow. Solution did not converge.'
      MESSAGE17 = 'Maximum iterations performed for
     & segment mixing. Solution did not converge.'
      MESSAGE18 = 'Total fluid volume exceeds tank volume.'
C-----------------------------------------------------------------------
C    SET UNITS FOR INPUTS AND OUTPUTS
C       SET THE YCHECK AND OCHECK ARRAYS TO CONTAIN THE CORRECT
C       VARIABLE TYPES FOR THE INPUTS AND OUTPUTS
        DATA YCHECK/NI*'NAV'/
        DATA OCHECK/NOUT*'NAV'/
C-----------------------------------------------------------------------
C    GET DELTA T FROM TRNSYS
      DT = getSimulationTimeStep()                    ! timestep (hr)
C-----------------------------------------------------------------------
C     READ IN THE VALUES OF THE PARAMETERS IN SEQUENTIAL ORDER
      MM_init = PAR(1)
      PP = PAR(2)
      Vmax = PAR(3)
      H = PAR(4)
      CC0 = PAR(5)
      desiccant = PAR(6)
      imax = PAR(7)
      Mode = PAR(8)
      Vminfraction = PAR(9)
      no_longer_used = PAR(10)
      U_L_top = PAR(11)
      U_L_sid = PAR(12)
      U_L_sid_d = PAR(13)
      U_L_btm = PAR(14)
      Tmin = PAR(15)+273.15
      Smin = PAR(16)
      do i = 1, MM_init, 1
        m_init(i) = PAR(16+3*(i-1)+1)
        S_init(i) = PAR(16+3*(i-1)+2)
        T_init(i) = PAR(16+3*(i-1)+3)+273.15
      end do
      do i = 1, PP, 1
        hp(i) = PAR(16+3*MM_init+i)
      end do
      if (hp(1).gt.0) then
        hp(1) = 0
      end if
```

```
C-----------------------------------------------------------------------
C    RETRIEVE THE CURRENT VALUES OF THE INPUTS TO THIS MODEL FROM THE
C    XIN ARRAY IN SEQUENTIAL ORDER

      do i = 1, PP, 1
        mi(i) = XIN(5*(i-1)+1)*dt
        Si(i) = XIN(5*(i-1)+2)
        Ti(i) = XIN(5*(i-1)+3)+273.15
        mout(i) = XIN(5*(i-1)+4)*dt
        Qaux(i) = XIN(5*(i-1)+5)*dt
      end do
      Tenv_top = XIN(5*PP+1)+273.15
      Tenv_sid = XIN(5*PP+2)+273.15
      Tenv_btm = XIN(5*PP+3)+273.15

      IUNIT=INFO(1)
      ITYPE=INFO(2)

C-----------------------------------------------------------------------
C    EXTRA PARAMETERS

      maxiters = 1000

      ! Tank definitions (modeling)
      Vbelowp(PP+1) = 1e6 ! arbitrary very large number

      error = .false.

      ! Energy balance constants
      Sref = 0.0 !kg-salt/kg-soln
      Tref = 273.15 !K
      Tcrit = 647.096 !K

C-----------------------------------------------------------------------
C    SET THE VERSION INFORMATION FOR TRNSYS
      IF(INFO(7).EQ.-2) THEN
         INFO(12)=16
         RETURN 1
      ENDIF
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C    DO ALL THE VERY LAST CALL OF THE SIMULATION MANIPULATIONS HERE
      IF (INFO(8).EQ.-1) THEN
         RETURN 1
      ENDIF
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C    PERFORM ANY 'AFTER-ITERATION' MANIPULATIONS THAT ARE REQUIRED HERE
C    e.g. save variables to storage array for the next timestep

      IF (INFO(13).GT.0) THEN
         NITEMS=1+3*MaxSeg
         STORED(1) = MM
         do i = 1, imax, 1
```

```
            STORED(1+i) = m4(i)
            STORED(1+imax+i) = S4(i)
            STORED(1+2*imax+i) = T4(i)
          end do

C       PUT THE STORED ARRAY IN THE GLOBAL STORED ARRAY
          CALL setStorageVars(STORED,NITEMS,INFO)
       ENDIF
C
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C    DO ALL THE VERY FIRST CALL OF THE SIMULATION MANIPULATIONS HERE
       IF (INFO(7).EQ.-1) THEN

C       GET THE NUMBER OF PARAMETERS FROM THE INPUT FILE AND CHECK
C       AGAINST THE MAXIMUM NUMBER
          NP_DECK=INFO(4)
          IF(NP_DECK.GT.NP) THEN
             CALL TYPECK(-3,INFO,NP+1,0,0)
          ENDIF

C       GET THE NUMBER OF INPUTS FROM THE INPUT FILE AND CHECK AGAINST
C       THE MAXIMUM NUMBER
          NI_DECK=INFO(3)
          IF(NI_DECK.GT.NI) THEN
             CALL TYPECK(-3,INFO,NI+1,0,0)
             RETURN 1
          ENDIF

C       SET SOME INFO ARRAY VARIABLES TO TELL THE TRNSYS ENGINE HOW
C       THIS TYPE IS TO WORK
          INFO(6)=NOUT
          INFO(9)=1
          INFO(10)=0     !STORAGE FOR VERSION 16 HAS BEEN CHANGED


C       SET THE REQUIRED NUMBER OF INPUTS, PARAMETERS AND DERIVATIVES
C       THAT THE USER SHOULD SUPPLY IN THE INPUT FILE.
C       IN SOME CASES, THE NUMBER OF VARIABLES MAY DEPEND ON THE VALUE
C       OF PARAMETERS TO THIS MODEL....
          NIN=NI_DECK
          NPAR=NP_DECK
          NDER=ND

C       CALL THE TYPE CHECK SUBROUTINE TO COMPARE WHAT THIS COMPONENT
C       REQUIRES TO WHAT IS SUPPLIED IN
C       THE TRNSYS INPUT FILE
          CALL TYPECK(1,INFO,NIN,NPAR,NDER)

C       SET THE NUMBER OF STORAGE SPOTS NEEDED FOR THIS COMPONENT
          NITEMS=1+3*MaxSeg
          CALL setStorageSize(NITEMS,INFO)

C       CHECK THE MINIMUM TIMESTEP
          IF (DT.LT.DELT_MIN) THEN
```

```
            CALL MESSAGES(-1,MESSAGE1,'FATAL',IUNIT,ITYPE)
          ENDIF

C         RETURN TO THE CALLING PROGRAM
          RETURN 1

        ENDIF
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C     DO ALL OF THE INITIAL TIMESTEP MANIPULATIONS HERE - THERE ARE NO
C     ITERATIONS AT THE INTIAL TIME
        IF (TIME .LT. (getSimulationStartTime() +
     . getSimulationTimeStep()/2.D0)) THEN

C         SET THE UNIT NUMBER FOR FUTURE CALLS
          IUNIT=INFO(1)
          ITYPE=INFO(2)

C         CHECK THE PARAMETERS FOR PROBLEMS AND RETURN FROM THE
C         SUBROUTINE IF AN ERROR IS FOUND
C           IF(...) CALL TYPECK(-4,INFO,0,"BAD PARAMETER #",0)

C         PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL VALUES OF
C         THE OUTPUTS HERE

C       1) Set working variables from initital condition

        MM = MM_init ! initial segments @ start of time step

C       2) Initial properties

        do i = 1, MM, 1
          m0(i) = m_init(i)
          S0(i) = S_init(i)
          T0(i) = T_init(i)
          if (desiccant.eq.1) then
              rho0(i) = rhocacl2(T0(i), S0(i)) !kg-soln/m^3
          else
              rho0(i) = rholicl(T0(i), S0(i)) !kg-soln/m^3
              endif
              V0(i) = m0(i)/rho0(i)
        end do

C       3) Evaluate mass

        m0tot = 0
        m0salt = 0
        V0tot = 0
        do i = 1, MM, 1
          m0tot = m0tot + m0(i)
          m0salt = m0salt + m0(i)*S0(i)
          V0tot = V0tot + V0(i)
        end do

C       4) Evaluate energy
```

```
      Ese0_t = 0
      Ech0_t = 0
      Et0 = 0
      Ese0u_t = 0
      Ech0u_t = 0
      Et0u = 0
      do i = 1, MM, 1
        if (desiccant.eq.1) then
            fS0u(i) = cacl2_f_S1(S0(i),Smin)
            cp0(i) = cpcacl2(T0(i),S0(i))
        else
            fS0u(i) = licl_f_S1(S0(i),Smin)
            cp0(i) = cplicl(T0(i),S0(i))
        end if
        Ese0(i) = m0(i)*cp0(i)*(T0(i)-Tref)
        Ech0(i) = m0(i)*S0(i)*(c5+c6*T0(i)/Tcrit)*fS0(i)
        Ese0_t = Ese0_t + Ese0(i)
        Ech0_t = Ech0_t + Ech0(i)
        Et0 = Et0 + Ese0(i) + Ech0(i)

        if (T0(i).gt.Tmin) then ! no negative usable energy
            Ese0u(i) = m0(i)*cp0(i)*(T0(i)-Tmin)
        else
            Ese0u(i) = 0
        end if
        if (S0(i).gt.Smin) then ! no negative usable energy
            Ech0u(i) = m0(i)*S0(i)*(c5+c6*T0(i)/Tcrit)*fS0u(i)
        else
            Ech0u(i) = 0
        end if
        Ese0u_t = Ese0u_t + Ese0u(i)
        Ech0u_t = Ech0u_t + Ech0u(i)
        Et0u = Et0u + Ese0u(i) + Ech0u(i)
      end do

C     5) Evaluate geometry
      Ac = Vmax/H !m^2
      if (CC0.lt.(4*(Ac**0.5))) then
        ! the perimeter specified is too small to form a square from
        ! the specified area, assume a circular cross-section
        CC = (4*pi*Ac)**0.5
      else
        CC = CC0
      end if

C     6) Temperature at port/aux height
      do i = 1, MM, 1
        h0(i) = V0(i)/Ac
        if (i.eq.1) then
            h0below(i) = h0(i)
        else
            h0below(i) = h0below(i-1) + h0(i)
        end if
      end do
      do p = 1, PP, 1
```

```
          do i = 1, MM, 1
              if (hp(p).lt.h0below(i)) then
                  Tprobe(p) = T0(i)
                  exit
              else
                  Tprobe(p) = 0
              end if
          end do
        end do

C       7) Set initial outputs
        do i = 1, PP, 1
          OUT(2*(i-1)+1) = 0
          OUT(2*(i-1)+2) = 0
        end do

        OUT(2*PP+1) = 0
        OUT(2*PP+2) = 0
        OUT(2*PP+3) = 0
        OUT(2*PP+4) = 0
        OUT(2*PP+5) = 0

        OUT(2*PP+6) = 0
        OUT(2*PP+7) = 0
        OUT(2*PP+8) = 0

        OUT(2*PP+9) = 0
        OUT(2*PP+10) = 0
        OUT(2*PP+11) = 0

C        At end of time step:
        OUT(2*PP+12) = MM
        do i = 1, imax, 1
          OUT(2*PP+12+8*(i-1)+1) = V0(i)
          OUT(2*PP+12+8*(i-1)+2) = rho0(i)
          OUT(2*PP+12+8*(i-1)+3) = m0(i)
          OUT(2*PP+12+8*(i-1)+4) = S0(i)
          if (m0(i).gt.0) then
              OUT(2*PP+12+8*(i-1)+5) = T0(i)-273.15
          else
              OUT(2*PP+12+8*(i-1)+5) = 0
          end if
          OUT(2*PP+12+8*(i-1)+6) = Ese0u(i)
          OUT(2*PP+12+8*(i-1)+7) = Ech0u(i)
          OUT(2*PP+12+8*(i-1)+8) = Ese0u(i)+Ech0u(i)
        end do

        OUT(2*PP+12+8*imax+1) = V0tot
        OUT(2*PP+12+8*imax+2) = m0tot
        OUT(2*PP+12+8*imax+3) = m0salt
        OUT(2*PP+12+8*imax+4) = Ese0u_t
        OUT(2*PP+12+8*imax+5) = Ech0u_t
        OUT(2*PP+12+8*imax+6) = Et0u

        do i = 1, PP, 1
          OUT(2*PP+12+8*imax+6+i) = Tprobe(i)-273.15
```

```
      end do

      OUT(2*PP+12+8*imax+6+PP+1) = 0
      OUT(2*PP+12+8*imax+6+PP+2) = 0
      OUT(2*PP+12+8*imax+6+PP+3) = 0
      OUT(2*PP+12+8*imax+6+PP+4) = 0
      OUT(2*PP+12+8*imax+6+PP+5) = 0
      OUT(2*PP+12+8*imax+6+PP+6) = 0

C     8) Save timestep
C     PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL STORAGE
C     VARIABLES HERE

         NITEMS=1+3*MaxSeg
         STORED(1) = MM
         do i = 1, imax, 1
            STORED(1+i) = m0(i)
            STORED(1+imax+i) = S0(i)
            STORED(1+2*imax+i) = T0(i)
         end do

C     9) Check for errors & make corrections if possible

C     (i) Height of position 1 must be 0 m
      if (PAR(16+3*MM_init+1).gt.0) then
        CALL MESSAGES(-1,MESSAGE2,'WARNING',IUNIT,ITYPE)
      end if

C     (ii) Each position must be higher than the position before
      do i=2, PP, 1
        if (hp(i).le.hp(i-1)) then
            CALL MESSAGES(-1,MESSAGE3,'FATAL',IUNIT,ITYPE)
        end if
      end do

C     (iii) Each position within the tank height
      do i=2, PP, 1
        if (hp(i).gt.H) then
            CALL MESSAGES(-1,MESSAGE4,'FATAL',IUNIT,ITYPE)
        end if
      end do

C     (iv) Initial storage profile must be stably stratified
      do i=2, MM, 1
        if (rho0(i).gt.rho0(i-1)) then
            CALL MESSAGES(-1,MESSAGE5,'FATAL',IUNIT,ITYPE)
        end if
      end do

C     (v) Circumference must be greater than or equal to that of a
C     circle (for the cross-sectional area resulting Ac=Vmax/H)
      if (CC0.lt.(4*(Ac**0.5))) then
        CALL MESSAGES(-1,MESSAGE6,'WARNING',IUNIT,ITYPE)
      end if

C     (vi) Circumference must be greater than or equal to that of a
```

```
C     circle (for the cross-sectional area resulting Ac=Vmax/H)
      if (MM.gt.imax) then
        CALL MESSAGES(-1,MESSAGE7,'FATAL',IUNIT,ITYPE)
      end if

C       PUT THE STORED ARRAY IN THE GLOBAL STORED ARRAY
         CALL setStorageVars(STORED,NITEMS,INFO)

C       RETURN TO THE CALLING PROGRAM
         RETURN 1

      ENDIF
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C    *** ITS AN ITERATIVE CALL TO THIS COMPONENT ***
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C    RETRIEVE THE VALUES IN THE STORAGE ARRAY FOR THIS ITERATION
      NITEMS=1+3*MaxSeg
      CALL getStorageVars(STORED,NITEMS,INFO)
      MM = STORED(1)
      do i = 1, imax, 1
        m0(i) = STORED(1+i)
        S0(i) = STORED(1+imax+i)
        T0(i) = STORED(1+2*imax+i)
      end do
C-----------------------------------------------------------------------
C    MAKE SURE THAT NO UNWANTED VALUES ARE STORED
      do i=1, MaxSeg,1
        m1(i)=0
        m2(i)=0
        m3(i)=0
        m4(i)=0
        T1(i)=0
        T2(i)=0
        T3(i)=0
        T4(i)=0
        S1(i)=0
        S2(i)=0
        S3(i)=0
        S4(i)=0
        rho1(i)=0
        rho2(i)=0
        rho3(i)=0
        rho4(i)=0
      end do
C-----------------------------------------------------------------------
C    CHECK THE INPUTS FOR PROBLEMS
C      IF(...) CALL TYPECK(-3,INFO,'BAD INPUT #',0,0)
C     IF(IERROR.GT.0) RETURN 1
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C    *** PERFORM ALL THE CALCULATIONS HERE FOR THIS MODEL. ***
C-----------------------------------------------------------------------
```

```
C=====================================================================
C---- CLOSED SYSTEM: CONSERVATION OF SALT MASS -----------------------
C=====================================================================

C     ---------------------------------------------------------------
C     Term definitions:
C     aa:       Coefficient in the approximate solution for the species
C               conservation equation
C     bb:       Coefficient in the approximate solution for the species
C               conservation equation
C     Diff:     Binary mass diffusion coefficient
C     halfh:    Edge-to-center distance for a segment (i.e. half of
C               segment height)
C     rho:      Binary solution density
C     rhoD:     Mass-weighted average of (rho*D) properties
C     Savg:     Average segment salt concentration over the course of
C               the time step
C     Sf:       Final segment salt concentration after the time step
C     s_iters:  Number of iterations performed
C     _above:   The upper interface of a segment
C     _below:   The lower interface of a segment
C     _0:       Properties of SPv0 (initial profile)
C     _1:       Properties of SPv1
C     ---------------------------------------------------------------
C     Strategy:
C     (1) Initialize Savg to the initial value (S0)
C     (2) Set a & b coefficients, solve & iterate until converged
C     ---------------------------------------------------------------

      IF (MM.gt.1) THEN ! routine is only for two or more segments
C
C     (1) Initialize Savg to the initial value (S0)
C
      do i = 1, MM, 1
        Savg(i) = S0(i) !kg-salt/kg-soln
        s_iters(i) = 0
      end do

C
C     (2) Set a & b coefficients, solve & iterate until converged
C
      do iter = 1, maxiters, 1

        do i = 1, MM, 1

            s_iters(i) = s_iters(i) + 1

            if (desiccant.eq.1) then
                rho(i) = rhocacl2(T0(i), Savg(i)) !kg-soln/m^3
                Diff(i) = dcacl2(T0(i), Savg(i))*3600 !m^2/hr
            else
                rho(i) = rholicl(T0(i), Savg(i)) !kg-soln/m^3
                Diff(i) = dlicl(T0(i), Savg(i))*3600 !m^2/hr
                endif
            halfh(i) = m0(i)/rho(i)/Ac/2 !m
```

```
            if (i.eq.1) then
                rhoD_below(i) = 0
                aa_below(i) = 0
                bb_below(i) = 0
            else
                rhoD_below(i) = (rho(i)*Diff(i)*m0(i)+
     &              rho(i-1)*Diff(i-1)*m0(i-1))/(m0(i)+m0(i-1))
                !kg-soln/m.hr
                    aa_below(i) = -
rhoD_below(i)*Ac/m0(i)/(halfh(i)+halfh(i-1))
                                !1/hr
                  bb_below(i) =
rhoD_below(i)*Ac/m0(i)/(halfh(i)+halfh(i-1))*
     &              Savg(i-1) !kg-salt/kg-soln.hr
            end if

                if (i.eq.MM) then
                      rhoD_above(i) = 0
                  aa_above(i) = 0
                      bb_above(i) = 0
            else
                      rhoD_above(i) = (rho(i)*Diff(i)*m0(i)+
     &              rho(i+1)*Diff(i+1)*m0(i+1))/(m0(i)+m0(i+1))
                      !kg-soln/m.hr
                  aa_above(i) = -
rhoD_above(i)*Ac/m0(i)/(halfh(i)+halfh(i+1))
                      !1/hr

      bb_above(i)=rhoD_above(i)*Ac/m0(i)/(halfh(i)+halfh(i+1))*Savg(i+1
)
                      !kg-salt/kg-soln.hr
            end if

            aa(i) = aa_below(i) + aa_above(i) !1/hr
            bb(i) = bb_below(i) + bb_above(i) !kg-salt/kg-soln.hr

            S_new(i) = (S0(i)+ bb(i)/aa(i))*dexp(aa(i)*dt)-bb(i)/aa(i)
                  !kg-salt/kg-soln
        Savg_new(i)=(S0(i)+ bb(i)/aa(i))*(dexp(aa(i)*dt)-1)/(aa(i)*dt)-
     &          bb(i)/aa(i)   !kg-salt/kg-soln
            end do

        s_converge = .true.

            do i = 1, MM, 1

                if(dabs(Savg_new(i)-Savg(i)).gt.converged) then
                      s_converge = .false.
                end if

                Sf(i) = S_new(i) !kg-salt/kg-soln
                Savg(i) = Savg_new(i) !kg-salt/kg-soln

        end do
```

```
            if(s_converge) then
                  exit
          else if(s_iters(MM).eq.maxiters) then
              CALL MESSAGES(-1,MESSAGE8,'WARNING',IUNIT,ITYPE)
          endif

        end do

        ELSE ! no diffusion
          Sf(1) = S0(1)
          Savg(1) = S0(1)
        END IF

C=======================================================================
C---- CLOSED SYSTEM: CONSERVATION OF ENERGY ----------------------------
C=======================================================================

C     ---------------------------------------------------------------
C     Term definitions:
C     aa:       Coefficient in the approximate soluiton for the species
C               conservation equation
C     Awdry:    Wall surface area that is dry (i.e. above fluid level)
C     bb:       Coefficient in the approximate soluiton for the species
C               conservation equation
C     f_S:      Concentration-dependent factor in the chemical energy
C               term
C     hdry:     Height of tank sidewall that is dry (i.e. the fluid
C               falls below)
C     halfh:    Edge-to-center distance for a segment (i.e. half of
C               segment height)
C     kcond:    Thermal conductivity coefficient
C     Tavg:     Average segment temperature over the course of the time
C               step
C     Tf:       Final segment temperature after the time step
C     T_iters:  Number of iterations performed
C     _above:   The upper interface of a segment
C     _below:   The lower interface of a segment
C     _0:       Properties of SPv0 (initial profile)
C     _1:       Properties of SPv1
C     ---------------------------------------------------------------
C     Strategy:
C     (1) Estimate dry wall section
C     (2) Initialize Tavg and Tf to the initial value (T0)
C     (3) Set a & b coefficients, solve & iterate until converged
C     ---------------------------------------------------------------

      ! This routine functions only if there are two or more fluid
      ! segments OR thermal losses
      IF ((MM.gt.1).or.(U_L_top.gt.0).or.(U_L_btm.gt.0)
     &      .or.(U_L_sid.gt.0).or.(U_L_sid_d.gt.0))   THEN

C
C     (1) Estimate dry wall section
C
      hdry = H
      do i = 1, MM, 1
```

```
            hdry = hdry - 2*halfh(i) !m
        end do
        Awdry = CC*hdry

C
C       (2) Initialize Tavg and Tf to the initial value (T0)
C
        do i = 1, MM, 1
            Tavg(i) = T0(i) !K
          Tf(i) = T0(i) !K
            T_iters(i) = 0
        end do

C
C       (3) Set a & b coefficients, solve & iterate until converged
C
        do iter = 1, maxiters, 1

            do i = 1, MM, 1

                T_iters(i) = T_iters(i) + 1

            if (desiccant.eq.1) then
                rho(i) = rhocacl2(Tavg(i), Savg(i)) !kg-soln/m^3
                    kcond(i) = kcacl2(Tavg(i), Savg(i))*3.6 !kJ/hr.m.K
                cpf(i) = cpcacl2(Tf(i),Sf(i)) !kJ/kg.K
                    cpavg(i) = cpcacl2(Tavg(i), Savg(i)) !kJ/kg.K
                cpprev(i) = cpcacl2(T0(i),S0(i)) !kJ/kg.K
                c5 = -955.69
                c6 = 3011.974
                f_Savg2(i) = cacl2_f_S2(Savg(i),Sref)
                f_Savg3(i) = cacl2_f_S3(Savg(i),Sref)
                else
                rho(i) = rholicl(Tavg(i), Savg(i)) !kg-soln/m^3
                    kcond(i) = klicl(Tavg(i), Savg(i))*3.6 !kJ/hr.m.K
                cpf(i) = cplicl(Tf(i),Sf(i)) !kJ/kg.K
                    cpavg(i) = cplicl(Tavg(i), Savg(i)) !kJ/kg.K
                    cpprev(i) = cplicl(T0(i),S0(i)) !kJ/kg.K
                c5 = 169.105
                c6 = 457.850
                f_Savg2(i) = licl_f_S2(Savg(i),Sref)
                f_Savg3(i) = licl_f_S3(Savg(i),Sref)
                endif
            halfh(i) = m0(i)/rho(i)/Ac/2 !m
            Aw(i) = CC*(halfh(i)*2) !m^2

                if (i.eq.1) then
                      kcond_below(i) = 0
                        aa_below(i) = -U_L_btm*Ac !kg-soln/C.hr
                        bb_below(i) = U_L_btm*Ac*(Tenv_btm-Tref) !kJ/hr
            else

                        kcond_below(i) =(kcond(i)*m0(i)+kcond(i-1)*
     &              m0(i-1))/(m0(i)+m0(i-1)) !kJ/hr.m.C

                aa_below(i) =
```

```
&             - kcond_below(i)*Ac/(halfh(i)+halfh(i-1)) !kJ/hr.C

        bb_below(i) =
&             kcond_below(i)*Ac/(halfh(i)+halfh(i-1))*
&             (Tavg(i-1)-Tref) !kJ/hr
      end if

      if (i.eq.MM) then
              kcond_above(i) = 0
              if (Awdry.lt.0) then
                    aa_above(i) = -U_L_top*Ac !kJ/hr.C
                    bb_above(i) = U_L_top*Ac*(Tenv_top-Tref)
                    !kJ.C/hr
          ! TREATING THE TOP SEGMENT AS GIVING UP THE HEAT
          ! LOST FROM THE DRY AIR VOLUME
        else
            aa_above(i) = -U_L_top*Ac - U_L_sid_d*Awdry
            !kJ/hr.C
            bb_above(i) = U_L_top*Ac*(Tenv_top-Tref)
&                   +U_L_sid_d*Awdry*(Tenv_sid-Tref) !kJ.C/hr
            ! TREATING THE TOP SEGMENT AS GIVING UP THE HEAT
            ! LOST FROM THE DRY AIR VOLUME
        end if
      else

              kcond_above(i) = (kcond(i)*m0(i)+kcond(i+1)*
&           m0(i+1))/(m0(i)+m0(i+1)) !kJ/hr.m.C

        aa_above(i) =
&           - kcond_above(i)*Ac/(halfh(i)+halfh(i+1)) !kJ/hr.C

        bb_above(i) =
&             kcond_above(i)*Ac/(halfh(i)+halfh(i+1))*
&             (Tavg(i+1)-Tref) !kJ/hr
      end if

      ! Note that the factor (Tavg(i)/(Tavg(i)-Tref)) is used to
      ! "scale up" the coefficient for the chemical energy term
      ! because the temperature used in that term has units of K,
      ! whereas the rest of the temperatures used in the aa(i)
      ! and bb(i) terms have units of C

          aa(i) = ( aa_below(i) + aa_above(i)
&       - m0(i)*(cpf(i)-cpprev(i))/dt
&       -(m0(i)*c6/Tcrit*f_Savg2(i)*(Sf(i)-S0(i))/dt*
&       (Tavg(i)/(Tavg(i)-Tref)) )
&       - U_L_sid*Aw(i) ) /
&       ( m0(i)*cpavg(i) + m0(i)*c6/Tcrit*f_Savg3(i) )
        !1/hr

      bb(i) = ( bb_below(i) + bb_above(i)
&             -(m0(i)*c5*f_Savg2(i)*(Sf(i)-S0(i))/dt)
&       + U_L_sid*Aw(i)*(Tenv_sid-Tref) ) /
&       ( m0(i)*cpavg(i) + m0(i)*c6/Tcrit*f_Savg3(i) )
        !C/hr
```

```
                T_new(i) = ((T0(i)-Tref)+ bb(i)/aa(i))*
     &                dexp(aa(i)*dt)-bb(i)/aa(i) + Tref !K
                Tavg_new(i) = ((T0(i)-Tref)+ bb(i)/aa(i))*
     &               (dexp(aa(i)*dt)-1)/(aa(i)*dt)-bb(i)/aa(i) + Tref !K

            end do

          T_converge = .true.

            do i = 1, MM, 1

                if(dabs(Tavg_new(i)-Tavg(i)).gt.converged) then
                      T_converge = .false.
                end if

                Tf(i) = T_new(i) !K
                Tavg(i) = Tavg_new(i) !K

                S1(i) = Sf(i) !kg-salt/kg-soln
                T1(i) = Tf(i) !K
            m1(i) = m0(i) !kg-soln

          end do

            if(T_converge) then
                  exit
          else if(T_iters(MM).eq.maxiters) then
            CALL MESSAGES(-1,MESSAGE9,'WARNING',IUNIT,ITYPE)
          endif

        end do

        ELSE ! no energy transfer
          Tf(1) = T0(1) !K
          Tavg(1) = T0(1) !K
          S1(1) = Sf(1) !kg-salt/kg-soln
              T1(1) = Tf(1) !K
          m1(1) = m0(1) !kg-soln
        END IF

C=======================================================================
C---- OPEN SYSTEM: PLUG FLOW -------------------------------------------
C=======================================================================

C     ---------------------------------------------------------------
C     INFLOW SETUP
C     ---------------------------------------------------------------
C     Lines required by both inflow modes are included in this section.
C     ---------------------------------------------------------------
C     Term definitions:
C     Vi:       The volume coming in a position
C     Vthruseg: The volume up to (and including) a segment in Storage
C               Profile version 1 (SPv1)
C     ---------------------------------------------------------------
C     Strategy:
C     (1) Calculate the density of each segment in SPv1
```

```
C     (2) Calculate the volume up to (& including) each segment in SPv1
C     (3) Calculate the density of the flow entering each position
C     (4) Calculate the volume of the flow entering each position
C     (5) Calculate the volume of fluid below each position
C     ------------------------------------------------------------------

      do i = 1, MM, 1
C
C     (1) Calculate the density of each segment in SPv1
C
        ! Calculate the densities of the existing segments
        if (desiccant.eq.1) then
            rho1(i) = rhocacl2(T1(i), S1(i)) !kg-soln/m^3
        else
            rho1(i) = rholicl(T1(i), S1(i)) !kg-soln/m^3
        endif

        V1(i) = m1(i)/rho1(i) !m^3

C
C     (2) Calculate the volume up to (& including) each segment in SPv1
C
        !Calculate the total fluid volume up to (& including) segment i
        if (i.eq.1) then
            Vthruseg(i) = V1(i) !m^3
        else
            Vthruseg(i) = Vthruseg (i-1) + V1(i) !m^3
        end if
      end do

      do p = 1, PP, 1
C
C     (3) Calculate the density of the flow entering each position
C
        ! Calculate the densities of the incoming flows
        if ((desiccant.eq.1).and.(mi(p).gt.0)) then
            rhoi(p) = rhocacl2(Ti(p), Si(p)) !kg-soln/m^3
        else if ((desiccant.eq.2).and.(mi(p).gt.0)) then
            rhoi(p) = rholicl(Ti(p), Si(p)) !kg-soln/m^3
        else
            rhoi(p) = 0
        endif

C
C     (4) Calculate the volume of the flow entering each position
C
        ! Calculate the volume of fluid that enters port p over the
        ! course of the timestep
        if (mi(p).gt.0) then
            Vi(p) = mi(p)/rhoi(p) !m^3
        else
            Vi(p) = 0
        end if
      end do

C
```

```
C     (5) Calculate the volume of fluid below each position
C
      do p = 1, PP, 1 ! Calculate the total fluid volume below port p
        Vbelowp(p) = hp(p)*Ac !m^3
      end do

      IF (MODE.EQ.1) THEN ! Mode 1 = Fixed inlet locations

C     ---------------------------------------------------------------
C     INFLOW MODE 1 (flow enters at fixed inlet locations)
C     ---------------------------------------------------------------
C     Storage Profile version 2 (SPv2) is defined by inserting the
C     inflow into SPv1. The segments from SPv1 are allocated beginning
C     with i=1 (the most dense) up to i=MM (the least dense). The
C     segments from inflow are inserted beginnning with p=1 (the lowest
C     inlet) up to p=PP (the highest inlet).
C     ---------------------------------------------------------------
C     Term definitions:
C     innext:   The next segment from inflow that needs to be inserted
C               in SPv2.
C     i1next:   The next segment from SPv1 that needs to be allocated
C               in SPv2.
C     seg2:     The number of segments in SPv2.
C               It is incremented by 1 whenever:
C                   (1) a segment from SPv1 is added to SPv2,
C                   (2) an inflow is inserted into SPv2
C     Vmove:    The volume of fluid from SPv1 that can be allocated to
C               SPv2 before another inflow must be inserted.
C     V1thru:   The volume of fluid from SPv1 that has already been
C               allocated to SPv2
C     _1:       Properties of SPv1
C     _2:       Properties of SPv2
C     ---------------------------------------------------------------
C     Strategy:
C     (1) If the base level inlet is in use, insert the new segment
C     (2) Cycle through the remaining inlets, moving fluid from SPv1 to
C         SPv2 up until the next inflow must be inserted.
C     (3) Allocate any remaining fluid from SPv1 to SPv2.
C     ---------------------------------------------------------------

      seg2 = 0
C
C     (1) If the base level inlet is in use, insert the new segment
C
      if (mi(1).gt.0) then
        seg2 = seg2 + 1
        m2(seg2) = mi(1)
        T2(seg2) = Ti(1)
            S2(seg2) = Si(1)
            V2(seg2) = Vi(1)
      end if

      V1thru = 0
      seg1 = MM
      i1next = 1
      innext = 1
```

```
C
C     (2) Cycle through the remaining inlets, moving fluid from SPv1 to
C         SPv2 up until the next inflow must be inserted.
C
      do i = 2, PP, 1
        if (mi(i).gt.0) then
            Vmove = Vbelowp(i) - V1thru
            V1thru = Vbelowp(i)
            do j = i1next, seg1, 1
                if (V1(j).lt.Vmove) then
                    seg2 = seg2 + 1
                        V2(seg2) = V1(j)
                        Vmove = Vmove - V1(j)
                    m2(seg2) = m1(j)
                    T2(seg2) = T1(j)
                        S2(seg2) = S1(j)
                        i1next = i1next + 1
                else if (V1(j).eq.Vmove) then
                    seg2 = seg2 + 1
                        V2(seg2) = V1(j)
                        Vmove = Vmove - V1(j)
                    m2(seg2) = m1(j)
                    T2(seg2) = T1(j)
                        S2(seg2) = S1(j)
                        i1next = i1next + 1

                    seg2 = seg2 + 1
                        V2(seg2) = Vi(i)
                    m2(seg2) = mi(i)
                    T2(seg2) = Ti(i)
                        S2(seg2) = Si(i)
                        innext = innext + 1
                else
                        seg2 = seg2 + 1
                        V2(seg2) = Vmove
                    m2(seg2) = m1(j)*(V2(seg2)/V1(j))
                        V1(j) = V1(j) - V2(seg2)
                        m1(j) = m1(j) - m2(seg2)
                    T2(seg2) = T1(j)
                        S2(seg2) = S1(j)

                    seg2 = seg2 + 1
                        V2(seg2) = Vi(i)
                    m2(seg2) = mi(i)
                    T2(seg2) = Ti(i)
                        S2(seg2) = Si(i)
                        innext = innext + 1
                        exit
                    end if
            end do
        else if (i1next.le.seg1) then
            innext = innext + 1
        end if
      end do
```

```
C
C     (3) Allocate any remaining fluid from SPv1 to SPv2.
C
      if (i1next.le.seg1) then
        do j = i1next, seg1, 1
            seg2 = seg2 + 1
                V2(seg2) = V1(j)
            m2(seg2) = m1(j)
            T2(seg2) = T1(j)
                S2(seg2) = S1(j)
                i1next = i1next + 1
        end do
      else if (innext.le.PP) then
        do i = innext, PP, 1
            if (mi(i).gt.0) then
                seg2 = seg2 + 1
                    V2(seg2) = Vi(i)
                m2(seg2) = mi(i)
                T2(seg2) = Ti(i)
                    S2(seg2) = Si(i)
            end if
        end do
      end if


      ELSE

C     ------------------------------------------------------------------
C     INFLOW MODE 2 (flow enters at neutral buoyancy location)
C     ------------------------------------------------------------------
C     Storage Profile version 2 (SPv2) is defined by inserting the
C     inflow into SPv1. The segments from SPv1 are allocated beginning
C     with i=1 (the most dense) up to i=MM (the least dense). The
C     segments from inflow are inserted beginnning with innext=1
C     (the most dense) up to innext=in (the least dense).
C     ------------------------------------------------------------------
C     Term definitions:
C     in:       The number of inflows for the time step.
C     innext:   The next segment from inflow that needs to be inserted
C               in SPv2.
C     i1next:   The next segment from SPv1 that needs to be allocated
C               in SPv2.
C     seg2:     The number of segments in SPv2.
C               It is incremented by 1 whenever:
C                   (1) a segment from SPv1 is added to SPv2,
C                   (2) an inflow is inserted into SPv2
C     Vbelowin: The volume of fluid from SPv1 that is located below the
C               inflow (used later to determine Vin for the dV calc,
C               since the Mode 2 inflows aren't inserted at given
C               positions)
C     _1:       Properties of SPv1
C     _2:       Properties of SPv2
C     ------------------------------------------------------------------
C     Strategy:
C     (1) Count the inflows
C     (2) Organize the inflows from most- to least-dense
```

```
C     (3) Loop through for each of the existing and new (inflow)
C         segments. Build SPv2 from most- to least-dense.
C     ------------------------------------------------------------------

      in = 0
C
C     (1) Count the inflows
C
      do i = 1, PP, 1
        if (mi(i).gt.0) then
            in = in + 1
        end if
      end do

      innext = 0
C
C     (2) Organize the inflows from most- to least-dense
C
      do i = 1, PP, 1
        innext = innext + 1
        rhonext(innext) = maxval(rhoi)
        loc(innext) = maxloc(rhoi,1)
        mnext(innext) = mi(loc(innext))
        Tnext(innext) = Ti(loc(innext))
        Snext(innext) = Si(loc(innext))
        Vnext(innext) = Vi(loc(innext))
        Vbelowin(innext) = 0
        rhoi(loc(innext)) = 0
      end do

      seg2 = 0
      i1next = 1
      innext = 1
C
C     (3) Loop through for each of the existing and new (inflow)
C         segments. Build SPv2 from most- to least-dense.
C
      do i = 1, in+MM, 1
        if ((rho1(i1next).gt.rhonext(innext)).and.(i1next.le.MM)) then
            seg2 = seg2 + 1
                m2(seg2) = m1(i1next)
            T2(seg2) = T1(i1next)
                S2(seg2) = S1(i1next)
                V2(seg2) = V1(i1next)
                Vbelowin(innext) = Vbelowin(innext) + V1(i1next)
                i1next = i1next + 1
        else if ((rho1(i1next).le.rhonext(innext)).and.(innext.le.in))
     &      then
            seg2 = seg2 + 1
                m2(seg2) = mnext(innext)
            T2(seg2) = Tnext(innext)
                S2(seg2) = Snext(innext)
                V2(seg2) = Vnext(innext)
                Vbelowin(innext+1) = Vbelowin(innext)
                innext = innext + 1
        end if
```

```
      end do


      END IF


      do i = 1, seg2, 1
        if (desiccant.eq.1) then
            rho2(i) = rhocacl2(T2(i), S2(i)) !kg-soln/m^3
            cp2(i) = cpcacl2(T2(i), S2(i)) !kJ/kg-K
        else
            rho2(i) = rholicl(T2(i), S2(i)) !kg-soln/m^3
            cp2(i) = cplicl(T2(i), S2(i)) !kJ/kg-K
        endif
      end do

C     ------------------------------------------------------------------
C     OUTFLOW & AUXILIARY
C     ------------------------------------------------------------------
C     Storage Profile version 3 (SPv3) is initialized to be equal to
C     SPv2. Positions are evaluated sequentially from the lowest- to
C     highest-numbered. Three steps occur in the evaluation of each
C     position:
C     (1) Fluid in the mix zone is mixed
C     (2) Outflow is removed from the tank. (This fluid leaves the
C         storage and so is removed from SPv3.)
C     (3) Fluid that will remain in the tank (between the present
C         position and the next one) is kept. (The fluid to "keep"
C         is also removed from SPv3 and allocated to SPv4.) Thus,
C         steps 2 & 3 steadily empty SPv3. The emptied segments of
C         SPv3 are deleted, so that the "base" segment of SPv3 is
C         always the first segment to be considered for
C         mixing/removing/keeping.
C     ------------------------------------------------------------------
C     Term definitions:
C     done:     Whether msum has met mmix; whether Vsum has met Vmix
C     dV:       The volume of fluid that rises above or falls below a
C               position over the course of the time step
C     Echemout1:    The total chemical energy from each contributing
C                   segment in the mix zone
C     Echemout2:    The chemical energy of the mixed segment
C     emptied:  The number of segments in SPv3 that have been emptied
C               (from outflow or re-allocation to SPv4)
C     error:    Whether a fatal error has occurred.
C     Esensibleout: The total sensible energy from each contributing
C                   segment in the mix zone
C     leftover: A portion of the mix zone that has been left over after
C               outflow has been removed
C     mcontrib: The mass contributed (by a segment in SPv3) toward the
C               mix zone; for all but the last segment, this will equal
C               the full segment mass
C     mleftover:    The mass left over in the mix zone after outflow
C                   has been removed
C     mmix:     The mass of fluid in the mix zone for a position
C     mmix_t:   The total solution mass that has been compiled for
C               mixing in the mix zone
```

```
C     moutremain:   The total mass that still must be removed from
C                   the tank for outflow
C     m3partial:    The mass remaining in a segment that has been only
C                   partially depleted (i.e. partially used in the mix
C                   zone)
C     msmix:    The total salt mass that has been compiled for mixing
C               in the mix zone
C     msum:     The total mass of that has been compiled (from one or
C               more segments, to be compared with mmix)...
C               If msum>mmix, not all of the last segment will be used
C               (see mmix_t for actual total for mixing)
C     mtankremain:  The total mass that still remains in the tank (in
C                   SPv3)
C     partial:  The number of segments in SPv3 that have been reduced
C               (partially emptied from outflow or re-allocation to
C               SPv4)
C     Qauxremain:   The total auxiliary heat still must be added to the
C                   tank
C     T_converge_2: Whether the mix zone temperature calculation has
C                   converged
C     T_iters_2:    The number of iterations that have been made to
C                   determine the temperature for the mixed segments
C     Vin:      The volume of fluid inserted into the tank through an
C               inlet (or above the previous inlet, for Mode 2)
C     Vmix:     The volume of fluid in the mix zone for a position
C     Vout:     The volume of fluid drawn from the tank through an
C               outlet
C     Vsum:     The total volume of fluid that has been compiled (from
C               one or more segments, to be compared with Vkeep)
C     V_converge:   Whether the mix zone volume calculation has
C                   converged
C     V_iters:  The number of iterations that have been made to
C               determine the Vmix for this position
C     _3:       Properties of SPv3
C     _4:       Properties of SPv4
C     ------------------------------------------------------------------
C     Strategy:
C     (1) Initialize SPv3 to SPv2
C     (2) Loop through each of the positions
C     (2-A) Estimate the mix zone based on the density of the first
C           contributing segment
C     (2-B) Before continuing, check for errors
C     (2-C) Mix the fluid in the mix zone, iterating if the initial
C           Vmix estimate was incorrect
C     (2-C-i) Take the fluid portions required to comprise Vmix
C     (2-C-ii) Mix the gathered fluids (add auxiliary heat input at
C           this point)
C     (2-C-iii) Recalculate Vmix based on the actual Vout
C     (2-C-iv) Exit if the Vmix calculation did not appreciably change
C     (2-D) "Collapse" the empty segments in SPv3 so that the base
C           segments are those that will be re-allocated to SPv4
C     (2-E) Calculate dV for the next position (now that Vin and Vout
C           are known for this position)
C     (2-F) Calculate the volume of fluid to "keep" in storage by
C           re-allocating to SPv4 (will be up to the next position
C           unless the fluid level is falling)
```

```
C     (2-G) Re-allocate the proper volume of fluid from SPv3 to SPv4
C     (2-H) "Collapse" the empty segments in SPv3 so that the base
C           segments are those that will be the first for outflow at
C           the next position
C     ------------------------------------------------------------------

C
C     (1) Initialize SPv3 to SPv2
C
      seg3 = seg2
      seg4 = 0
      V4tot = 0
      do i = 1, seg3, 1
        m3(i) = m2(i)
        T3(i) = T2(i)
        S3(i) = S2(i)
        cp3(i) = cp2(i)
        rho3(i) = rho2(i)
        V3(i) = V2(i)
      end do

C
C     (2) Loop through each of the positions
C
      ! The fluid level at the first position cannot be rising or
      ! falling; dV(1) must be set so that dV(>1) can be defined
      ! recursively
      dV(1) = 0
      do p = 1, PP, 1 ! Cycle through the ports

C
C     (2-A) Estimate the mix zone based on the density of the first
C           contributing segment
C
        ! Mix zone volume calculation
        ! Initial estimate of Vout, assuming that the average density
        ! is the density of the first contributing segment
        Vout(p) = mout(p)/rho3(1)
        if (dV(p).gt.0) then ! Rising fluid level at position p
            if (Vout(p).gt.(dV(p)+Vi(p))) then
                Vmix(p) = Vout(p)
            else
                Vmix(p) = dV(p)+Vi(p)
            end if
        else if (dV(p).lt.0) then ! Falling fluid level at position p
            if (Vi(p).gt.(dabs(dV(p))+Vout(p))) then
                Vmix(p) = Vi(p)
            else
                Vmix(p) = dabs(dV(p))+Vout(p)
            end if
        else ! Stable fluid level at position p
            if (Vi(p).gt.Vout(p)) then
                Vmix(p) = Vi(p)
            else
                Vmix(p) = Vout(p)
            end if
```

```
        end if

        ! Mix zone mass calculation - this is necessary because the BC
        ! for the outlet is mass flow rate (not volume flow rate)
        ! Initial estimate of the mass in the mix zone, assuming that
        ! the average density is the density of the first contributing
        ! segment
        mmix(p) = Vmix(p)*rho3(1)
        ! If there is no mix zone, create a heater zone for the Qaux
        ! to be distributed
        if (mmix(p).eq.0) then
            mmix(p) = m3(1)
            Vmix(p) = V3(1)
        ! If the mix zone is smaller than the first segment in SPv3,
        ! let the mix zone be at least the size of one full segment
        ! (for simplicity). (No extra mixing occurs, and the "extra"
        ! portion will be leftover as it would have been.)
        else if (mmix(p).lt.m3(1)) then
            mmix(p) = m3(1)
            Vmix(p) = V3(1)
        end if

C
C     (2-B) Before continuing, check for errors
C
        moutremain = 0
        Qauxremain = 0
        do i = p, PP, 1
            moutremain = moutremain + mout(i)
            Qauxremain = Qauxremain + Qaux(i)
        end do
        if (seg3.eq.0) then
            if (moutremain.gt.0) then
                CALL MESSAGES(-1,MESSAGE10,'FATAL',IUNIT,ITYPE)
                error = .true.
            else if (Qauxremain.gt.0) then
                CALL MESSAGES(-1,MESSAGE11,'FATAL',IUNIT,ITYPE)
                error = .true.
            else
                CALL MESSAGES(-1,MESSAGE12,'NOTICE',IUNIT,ITYPE)
            endif
            exit
        else
            mtankremain = 0
            do i = 1, seg3, 1
                mtankremain = mtankremain + m3(i)
            end do
            if (mmix(p).gt.mtankremain) then
                if (mout(p).gt.0) then
                    CALL MESSAGES(-1,MESSAGE13,'FATAL',IUNIT,ITYPE)
                    error = .true.
                    exit
                end if
            end if
        end if
```

```
          if (p.gt.1) then
              if (dV(p).lt.(Vbelowp(p-1)-Vbelowp(p))) then
                  CALL MESSAGES(-1,MESSAGE14,'FATAL',IUNIT,ITYPE)
                  error = .true.
              end if
          end if

C
C     (2-C) Mix the fluid in the mix zone, iterating if the initial
C           Vmix estimate was incorrect
C
          V_iters(p) = 0
          do
              Vsum = 0
              msum = 0
              emptied = 0
              partial = 0
              done = .false.
              mmix_t(p) = 0
              msmix(p) = 0
              Esensibleout(p) = 0
              Echemout1(p) = 0
                    V_converge = .false.
                  V_iters(p) = V_iters(p) + 1

C
C     (2-C-i) Take the fluid portions required to comprise Vmix
C
              ! Add mass from segments in SPv3 until sufficient mass has
              ! been selected from SPv3 to meet the requirements of mmix
              do i = 1, seg3, 1
                  mcontrib(i) = 0
                  msum = msum + m3(i)
                  ! IF adding all of the next segment in SPv3 is not
                  ! sufficient to meet the requirements of mmix, THEN
                  ! add all of the segment to mmix
                  if (msum.lt.mmix(p)) then
                      emptied = emptied + 1
                      mcontrib(i) = m3(i)
                  ! IF adding all of the next segment in SPv3 is exactly
                  ! sufficient to meet the requirements of mmix, THEN add
                  ! all of the segment to mmix and be DONE adding
                  ! segments to mmix
                  else if (msum.eq.mmix(p)) then
                      emptied = emptied + 1
                      mcontrib(i) = m3(i)
                      done = .true.
                  ! ELSE adding all of the next segment in SPv3 is more
                  ! than sufficient to meet the requirements of mmix,
                  ! THEN add a portion of the segment to mmix (sufficient
                  ! to exactly meet the requirement) and be DONE adding
                  ! segments to mmix
                  else
                      partial = 1
                      mcontrib(i) = mmix(p)-(msum-m3(i))
                      m3partial = m3(i) - mcontrib(i)
```

```
                done = .true.
            end if
            msmix(p) = msmix(p) + mcontrib(i)*S3(i)
            mmix_t(p) = mmix_t(p) + mcontrib(i)
            Esensibleout(p) = Esensibleout(p) +
     &          mcontrib(i)*cp3(i)*(T3(i)-Tref)
            if (desiccant.eq.1) then
                c5 = -955.69
                c6 = 3011.974
                f_S1(i) = cacl2_f_S1(S3(i),Sref)
            else
                c5 = 169.105
                c6 = 457.850
                f_S1(i) = licl_f_S1(S3(i),Sref)
            end if
            Echemout1(p) = Echemout1(p) +
     &          mcontrib(i)*S3(i)*(c5+c6*T3(i)/Tcrit)*f_S1(i)
            if (done) then
                exit
            end if
        end do
        ! concentration of mix zone fluid, kg-salt/kg-soln
        Sout(p) = msmix(p)/(mmix_t(p))
        ! first guess for specific heat of mix zone fluid, kJ/kg-K
        cpout(p) = cp3(1)
        ! first guess for temperature of mix zone fluid, K
        Tout(p) = T3(1) + Qaux(p)/(mmix_t(p)*cp3(1))
        if (desiccant.eq.1) then
            f_S2(p) = cacl2_f_S1(Sout(p),Sref)
        else
            f_S2(p) = licl_f_S1(Sout(p),Sref)
        end if

C
C     (2-C-ii) Mix the gathered fluids (add auxiliary heat input at
C              this point)
C
        T_iters_2(p) = 0
              T_converge_2 = .false.
              Qaux_on = 1
              ! Iteratively solve the outflow temperature and
              ! specific heat until converged
        do

                T_iters_2(p) = T_iters_2(p) + 1

            ! chemical potential energy of mix zone fluid, kJ
            Echemout2(p) = msmix(p)*(c5+c6*Tout(p)/Tcrit)*f_S2(p)
            ! Chemical-to-sensible energy generation due to
            ! exothermic mixing of multiple segments
            Egen(p) = ( Echemout1(p) - Echemout2(p) )

            ! The temperature of the mix zone fluid is calculating,
            ! including
            ! (1) Sensible energy from contributing segments
            ! (2) Chemical-to-sensible energy generation
```

```
                ! (3) Auxiliary heat addition
                Toutnew(p)=(Esensibleout(p)+Egen(p)+(Qaux(p)*Qaux_on))/
     &              (mmix_t(p)*cpout(p)) + Tref !K
                if (desiccant.eq.1) then
                    cpout(p) = cpcacl2(Tout(p), Sout(p)) !kJ/kg-K
                else
                    cpout(p) = cplicl(Tout(p), Sout(p)) !kJ/kg-K
                endif

                    if(dabs(Toutnew(p)-Tout(p)).lt.converged) then
                          T_converge_2 = .true.
                    end if

                Tout(p) = Toutnew(p) !K

                if(T_converge_2) then
                          exit
                  else if (T_iters_2(p).eq.maxiters) then
                    CALL MESSAGES(-1,MESSAGE15,'WARNING',IUNIT,ITYPE)
                    Qaux_on = Qaux_on-0.1
                    ! return to first guess for specific heat of mix
                    ! zone fluid, kJ/kg-K
                    cpout(p) = cp3(1)
                    ! return to first guess for temperature of mix zone
                    ! fluid, K
                    Tout(p) = T3(1)+ Qaux(p)*Qaux_on/(mmix_t(p)*cp3(1))
                    T_iters_2(p) = 0
                endif

            end do

            if (desiccant.eq.1) then
                rhoout(p) = rhocacl2(Tout(p), Sout(p)) !kg-soln/m^3
            else
                rhoout(p) = rholicl(Tout(p), Sout(p)) !kg-soln/m^3
            endif

            Vout(p) = mout(p)/rhoout(p)

C
C     (2-C-iii) Recalculate Vmix based on the actual Vout
C
            ! Rising fluid level at position p
            if (dV(p).gt.0) then
                if (Vout(p).gt.(dV(p)+Vi(p))) then
                    Vmixnew(p) = Vout(p)
                else
                    Vmixnew(p) = dV(p)+Vi(p)
                end if
            ! Falling fluid level at position p
            else if (dV(p).lt.0) then
                if (Vi(p).gt.(dabs(dV(p))+Vout(p))) then
                    Vmixnew(p) = Vi(p)
                else
                    Vmixnew(p) = dabs(dV(p))+Vout(p)
                end if
```

```
            else ! Stable fluid level at position p
                if (Vi(p).gt.Vout(p)) then
                    Vmixnew(p) = Vi(p)
                else
                    Vmixnew(p) = Vout(p)
                end if
            end if

            ! If there is no mix zone, create a heater zone for the
            ! Qaux to be distributed
            if (Vmixnew(p).eq.0) then
                Vmixnew(p) = V3(1)
            ! If the mix zone is smaller than the first segment in
            ! SPv3, let the mix zone be at least the size of one full
            ! segment (for simplicity). (No extra mixing occurs, and
            ! the "extra" portion will be leftover as it would have
            ! been.)
            else if (Vmixnew(p).lt.V3(1)) then
                Vmixnew(p) = V3(1)
            end if

                if(dabs(Vmixnew(p)-Vmix(p)).lt.converged) then
                      V_converge = .true.
                end if

            Vmix(p) = Vmixnew(p) !m^3
            mmix(p) = Vmix(p)*rhoout(p)

C
C     (2-C-iv) Exit if the Vmix calculation did not appreciably change
C
            if(V_converge) then
                      exit
            else if(V_iters(p).eq.maxiters) then
                CALL MESSAGES(-1,MESSAGE16,'WARNING',IUNIT,ITYPE)
                exit
            endif

          end do

          mleftover(p) = mmix_t(p) - mout(p)
          if (mleftover(p).lt.1e-8) then
              mleftover(p) = 0
          end if

C
C     (2-D) "Collapse" the empty segments in SPv3 so that the base
C           segments are those that will be re-allocated to SPv4
C
          leftover = 0
          if ((mleftover(p).gt.0).and.(emptied.gt.0)) then
              leftover = 1
              m3(1) = mleftover(p)
              V3(1) = mleftover(p)/rhoout(p)
              T3(1) = Tout(p)
              S3(1) = Sout(p)
```

```
            cp3(1) = cpout(p)
            rho3(1) = rhoout(p)
            emptied = emptied - 1
        else if (mleftover(p).gt.0) then !THIS SHOULD BE UNNECESSARY
            leftover = 1
            m3(1) = mleftover(p)
            V3(1) = mleftover(p)/rhoout(p)
            T3(1) = Tout(p)
            S3(1) = Sout(p)
            cp3(1) = cpout(p)
            rho3(1) = rhoout(p)
        end if

        seg3 = seg3 - emptied
        ! Cycle through the ports to remove the depleted segments and
        ! re-allocate the remaining fluid in profile 3
        do i = 1 + leftover, seg3, 1
            V3(i) = V3(i+emptied)
            m3(i) = m3(i+emptied)
            T3(i) = T3(i+emptied)
            S3(i) = S3(i+emptied)
            cp3(i) = cp3(i+emptied)
            rho3(i) = rho3(i+emptied)
        end do

        if (partial.eq.1) then
            V3(leftover+1) = m3partial/m3(leftover+1)*V3(leftover+1)
            m3(leftover+1) = m3partial
        end if

C
C     (2-E) Calculate dV for the next position (now that Vin and Vout
C           are known for this position)
C
        Vout(p) = mout(p)/rhoout(p)
        if ((Mode.eq.1).and.(mi(p).gt.0)) then
            Vi(p) = mi(p)/rhoi(p)
        else if ((Mode.eq.2).and.(mi(p).gt.0)) then
            Vi(p) = 0
            if (p.eq.1) then
                do i = 1, in, 1
                    if (Vbelowin(i).le.Vbelowp(p)) then
                        if (rhonext(i).gt.0) then
                            Vi(p) = Vi(p) + mnext(i)/rhonext(i)
                        end if
                    end if
                end do
            else
                do i = 1, in, 1
                    if ((Vbelowin(i).le.Vbelowp(p)).and.
     &                  (Vbelowin(i).gt.Vbelowp(p-1))) then
                        if (rhonext(i).gt.0) then
                            Vi(p) = Vi(p) + mnext(i)/rhonext(i)
                        end if
                    end if
                end do
```

```
            end if
          else
            Vi(p) = 0
          end if

          dV(p+1) = dV(p) + Vi(p) - Vout(p)

C
C     (2-F) Calculate the volume of fluid to "keep" in stroage by
C           re-allocating to SPv4 (will be up to the next position
C           unless the fluid level is falling)
C
          if(dV(p+1).ge.0) then ! RISING / STEADY
            Vkeep(p) = Vbelowp(p+1)-V4tot
          else ! FALLING
            Vkeep(p) = Vbelowp(p+1)-V4tot+dV(p+1)
          end if

          Vsum = 0
          emptied_3to4 = 0
          partial_3to4 = 0
          done = .false.

C
C     (2-G) Re-allocate the proper volume of fluid from SPv3 to SPv4
C
          do i = 1, seg3, 1
            Vsum = Vsum + V3(i)
            ! IF using all of the next segment in profile 3 is not
            ! sufficient, THEN remove all of the fluid from the
            ! segment
            if (Vsum.lt.Vkeep(p)) then
              emptied_3to4  = emptied_3to4  + 1
              seg4 = seg4 + 1
              V4(seg4) = V3(i)
              rho4(seg4) = rho3(i)
              m4(seg4) = V4(seg4)*rho4(seg4)
              T4(seg4) = T3(i)
              S4(seg4) = S3(i)
              V4tot = V4tot + V4(seg4)
            ! IF using all of the next segment in profile 3 is
            ! sufficient, THEN remove all of the fluid from the segment
            ! and be DONE
            else if (Vsum.eq.Vkeep(p)) then
              emptied_3to4  = emptied_3to4  + 1
              seg4 = seg4 + 1
              V4(seg4) = V3(i)
              rho4(seg4) = rho3(i)
              m4(seg4) = V4(seg4)*rho4(seg4)
              T4(seg4) = T3(i)
              S4(seg4) = S3(i)
              V4tot = V4tot + V4(seg4)
              done = .true.
            ! ELSE using some of the next segment in profile 3 is
            ! sufficient, so that it only contributes part of its
            ! contents and be DONE
```

```
            else
                partial_3to4 = 1
                seg4 = seg4 + 1
                V4(seg4) = Vkeep(p)-(Vsum-V3(i))
                rho4(seg4) = rho3(i)
                m4(seg4) = V4(seg4)*rho4(seg4)
                T4(seg4) = T3(i)
                S4(seg4) = S3(i)
                V3partial = V3(i) - V4(seg4)
                V4tot = V4tot + V4(seg4)
                done = .true.
            end if

            if (done) then
                exit
            end if

        end do

C
C     (2-H) "Collapse" the empty segments in SPv3 so that the base
C           segments are those that will be the first for outflow at
C           the next position
C
        seg3 = seg3 - emptied_3to4
        ! Cycle through the ports to remove the depleted segments and
        ! re-allocate the remaining fluid in profile 3
        do i = 1, seg3, 1
            V3(i) = V3(i+emptied_3to4)
            m3(i) = m3(i+emptied_3to4)
            T3(i) = T3(i+emptied_3to4)
            S3(i) = S3(i+emptied_3to4)
            cp3(i) = cp3(i+emptied_3to4)
            rho3(i) = rho3(i+emptied_3to4)
        end do

        if (partial_3to4.eq.1) then
            V3(1) = V3partial
            m3(1) = V3partial*rho3(1)
        end if

      end do

C     ------------------------------------------------------------------
C     DENSITY INVERSIONS
C     ------------------------------------------------------------------
C     Ensure that the storage is stably stratified.
C     ------------------------------------------------------------------
C     Term definitions:
C     _mixed:   Properties of new (combined) segement
C     _4:       Properties of SPv4
C     ------------------------------------------------------------------
C     Strategy:
C     (1) Check for density inversions and mix inverted segments
C     ------------------------------------------------------------------
```

```
      do i = 1, seg4, 1
        if (desiccant.eq.1) then
            cp4(i) = cpcacl2(T4(i), S4(i)) !kJ/kg-K
        else
            cp4(i) = cplicl(T4(i), S4(i)) !kJ/kg-K
        endif
      end do

C
C     (1) Check for density inversions and mix inverted segments
C
      i = 1
      do

        if (rho4(i).lt.rho4(i+1)) then

            m4mixed(i) = m4(i) + m4(i+1)
            S4mixed(i) = (m4(i)*S4(i)+m4(i+1)*S4(i+1))/(m4(i)+m4(i+1))

            cp4mixed(i) = (m4(i)*cp4(i)+m4(i+1)*cp4(i+1))/m4mixed(i)
                ! first guess, kJ/kg-K
            T4mixed(i) = (m4(i)*cp4(i)*(T4(i)-Tref)
     &          +m4(i+1)*cp4(i+1)*(T4(i+1)-Tref))/
     &          (m4(i)*cp4(i)+m4(i+1)*cp4(i+1)) + Tref !first guess, K

            Tmixed_iters(i) = 0
            if (desiccant.eq.1) then
                f_S4(i) = cacl2_f_S1(S4(i),Sref)
                f_S4(i+1) = cacl2_f_S1(S4(i+1),Sref)
                f_S4mixed(i) = cacl2_f_S1(S4mixed(i),Sref)
            else
                f_S4(i) = licl_f_S1(S4(i),Sref)
                f_S4(i+1) = licl_f_S1(S4(i+1),Sref)
                f_S4mixed(i) = licl_f_S1(S4mixed(i),Sref)
            end if

                  Tmixed_converge = .false.
                  dT_smallest = 1000
                  T_closest = 0

            ! Iteratively solve the mixed temperature and specific heat
            do
                    Tmixed_iters(i) = Tmixed_iters(i) + 1

                Echem4(i) = m4(i)*S4(i)*(c5+c6*T4(i)/Tcrit)*f_S4(i)
                Echem4(i+1) = m4(i+1)*S4(i+1)*(c5+c6*T4(i+1)/Tcrit)
     &              *f_S4(i+1)
                Echem4mixed(i) = m4mixed(i)*S4mixed(i)
     &              *(c5+c6*T4mixed(i)/Tcrit)*f_S4mixed(i)
                Egenmixed(i) = ((Echem4(i)+Echem4(i+1)) -
     &              Echem4mixed(i))

                T4mixednew(i) = ((m4(i)*cp4(i)*(T4(i)-Tref)+
     &              m4(i+1)*cp4(i+1)*(T4(i+1)-Tref))
     &              + Egenmixed(i) ) / (m4mixed(i)*cp4mixed(i)) + Tref
                    !K
```

```
                if (desiccant.eq.1) then
                    cp4mixed(i) = cpcacl2(T4mixednew(i), S4mixed(i))
                    !kJ/kg-K
                else
                    cp4mixed(i) = cplicl(T4mixednew(i), S4mixed(i))
                    !kJ/kg-K
                endif

                    if(dabs(T4mixednew(i)-T4mixed(i)).lt.converged)
then
                          Tmixed_converge = .true.
                    end if

                T4mixed(i) = T4mixednew(i) !K

                if(Tmixed_converge) then
                          exit
                  else if (Tmixed_iters(i).eq.maxiters) then
                    CALL MESSAGES(-1,MESSAGE17,'WARNING',IUNIT,ITYPE)
                          exit
                endif

            end do

            m4(i) = m4mixed(i)
            S4(i) = S4mixed(i)
            T4(i) = T4mixed(i)
            cp4(i) = cp4mixed(i)
            if (desiccant.eq.1) then
                rho4(i) = rhocacl2(T4(i), S4(i)) !kJ/kg-K
            else
                rho4(i) = rholicl(T4(i), S4(i)) !kJ/kg-K
            endif
            V4(i) = m4(i)/rho4(i)

            do j = i+1, seg4, 1
                m4(j) = m4(j+1)
                T4(j) = T4(j+1)
                S4(j) = S4(j+1)
                rho4(j) = rho4(j+1)
                cp4(j) = cp4(j+1)
                V4(j) = V4(j+1)
            end do

            seg4 = seg4 - 1
            i = 0

          else
          end if

          i = i + 1
          if (i.gt.seg4) then
              exit
          end if
        end do
```

```
C     ----------------------------------------------------------------
C     TOO SMALL SEGMENTS
C     ----------------------------------------------------------------
C     Ensure that each segment is at least 0.01% of the total storage.
C     ----------------------------------------------------------------
C     Term definitions:
C     imin:     The smallest segment
C     icombine: The neighboring segment of i that is closer in density
C               (with which i will combine)
C     Vminfraction: The minimum fraction (of the total tank storage)
C               that each segment must be
C     Vsmallest:    The smallest volume segment in the storage
C     _mixed:   Properties of new (combined) segement
C     _4:       Properties of SPv4
C     ----------------------------------------------------------------
C     Strategy:
C     (1) Check for segments that are too small and, if necessary, mix
C         the segment with its neighbor with which it is closer in
C         density
C     ----------------------------------------------------------------

C
C     (1) Check for segments that are too small and, if necessary, mix
C         the segment with its neighbor with which it is closer in
C         density
C
      do

        V4tot = 0
        Vsmallest = 1000
        do i = 1, seg4, 1
            V4tot = V4tot + V4(i)
            if (V4(i).lt.Vsmallest) then
                Vsmallest = V4(i)
                imin = i
            end if
        end do

        i = imin
        if (Vsmallest.lt.(V4tot*Vminfraction)) then
            if (i.eq.1) then
                icombine = 2
            else if (i.eq.seg4) then
                icombine = seg4-1
            else if ((rho4(i-1)-rho4(i)).lt.(rho4(i)-rho4(i+1))) then
                icombine = i-1
            else
                icombine = i+1
            end if

            m4mixed(i) = m4(i) + m4(icombine)
            S4mixed(i) = (m4(i)*S4(i)+m4(icombine)*S4(icombine))
     &          /(m4(i)+m4(icombine))

            cp4mixed(i) = (m4(i)*cp4(i)+m4(icombine)*cp4(icombine))
     &          /m4mixed(i) ! first guess, kJ/kg-K
```

```
T4mixed(i) = (m4(i)*cp4(i)*(T4(i)-Tref)
&     +m4(icombine)*cp4(icombine)*(T4(icombine)-Tref))/
&     (m4(i)*cp4(i)+m4(icombine)*cp4(icombine)) + Tref
      !first guess, K

Tmixed_iters(i) = 0
      Tmixed_converge = .false.
if (desiccant.eq.1) then
    f_S4(i) = cacl2_f_S1(S4(i),Sref)
    f_S4(icombine) = cacl2_f_S1(S4(icombine),Sref)
    f_S4mixed(i) = cacl2_f_S1(S4mixed(i),Sref)
else
    f_S4(i) = licl_f_S1(S4(i),Sref)
    f_S4(icombine) = licl_f_S1(S4(icombine),Sref)
    f_S4mixed(i) = licl_f_S1(S4mixed(i),Sref)
end if

      Tmixed_converge = .false.

! Iteratively solve the mixed temperature and specific heat
do
        Tmixed_iters(i) = Tmixed_iters(i) + 1

    Echem4(i) = m4(i)*S4(i)*(c5+c6*T4(i)/Tcrit)*f_S4(i)
    Echem4(icombine) = m4(icombine)*S4(icombine)
&       *(c5+c6*T4(icombine)/Tcrit)*f_S4(icombine)
    Echem4mixed(i) = m4mixed(i)*S4mixed(i)
&       *(c5+c6*T4mixed(i)/Tcrit)*f_S4mixed(i)
    Egenmixed(i) = ((Echem4(i)+Echem4(icombine)) -
&       Echem4mixed(i))

    T4mixednew(i) = ((m4(i)*cp4(i)*(T4(i)-Tref)+
&       m4(icombine)*cp4(icombine)*(T4(icombine)-Tref))
&       + Egenmixed(i) ) / (m4mixed(i)*cp4mixed(i)) + Tref
        !K
    if (desiccant.eq.1) then
        cp4mixed(i) = cpcacl2(T4mixednew(i), S4mixed(i))
        !kJ/kg-K
    else
        cp4mixed(i) = cplicl(T4mixednew(i), S4mixed(i))
        !kJ/kg-K
    endif

        if(dabs(T4mixednew(i)-T4mixed(i)).lt.converged)
        then
              Tmixed_converge = .true.
        end if

    T4mixed(i) = T4mixednew(i) !K

    if(Tmixed_converge) then
              exit
    else if(Tmixed_iters(i).eq.maxiters) then
        CALL MESSAGES(-1,MESSAGE17,'WARNING',IUNIT,ITYPE)
        exit
    endif
```

```
            end do

            m4(i) = m4mixed(i)
            S4(i) = S4mixed(i)
            T4(i) = T4mixed(i)
            cp4(i) = cp4mixed(i)

            if (desiccant.eq.1) then
                rho4(i) = rhocacl2(T4(i), S4(i)) !kJ/kg-K
            else
                rho4(i) = rholicl(T4(i), S4(i)) !kJ/kg-K
            endif
            V4(i) = m4(i)/rho4(i)

            if (i.lt.icombine) then
                do j = i+1, seg4, 1
                    m4(j) = m4(j+1)
                    T4(j) = T4(j+1)
                    S4(j) = S4(j+1)
                    rho4(j) = rho4(j+1)
                    cp4(j) = cp4(j+1)
                    V4(j) = V4(j+1)
                end do
            else
                do j = i-1, seg4, 1
                    m4(j) = m4(j+1)
                    T4(j) = T4(j+1)
                    S4(j) = S4(j+1)
                    rho4(j) = rho4(j+1)
                    cp4(j) = cp4(j+1)
                    V4(j) = V4(j+1)
                end do
            end if

            seg4 = seg4 - 1

          else
              exit
          end if

        end do


C     ------------------------------------------------------------------
C     EXCESS SEGMENTS
C     ------------------------------------------------------------------
C     Ensure that the storage is composed of an acceptable number of
C     segments.
C     ------------------------------------------------------------------
C     Term definitions:
C     drhomin:  Minimum density difference between neighboring storage
C               segments
C     icombine: The denser of the two segments with the smallest
C               density difference
C     _mixed:   Properties of new (combined) segement
```

```
C     _4:       Properties of SPv4
C     ------------------------------------------------------------------
C     Strategy:
C     (1) Check for an excessive number of segments and, if necessary,
C         mix the segments with the smallest density difference
C     ------------------------------------------------------------------

C
C     (1) Check for an excessive number of segments and, if necessary,
C         mix the segments with the smallest density difference
C
      do

            drhomin = 1000
            do i = 1, seg4-1, 1
                ! "queues up" the next inflow by the identifying the
                ! next highest density inflow
                if (rho4(i)-rho4(i+1).lt.drhomin) then
                    drhomin = rho4(i)-rho4(i+1)
                    icombine = i
                else
                end if
            end do

            i = icombine

          !imax must be at least 2
          if (seg4.gt.imax) then

            m4mixed(i) = m4(i) + m4(i+1)
            S4mixed(i) = (m4(i)*S4(i)+m4(i+1)*S4(i+1))/(m4(i)+m4(i+1))

            cp4mixed(i) = (m4(i)*cp4(i)+m4(i+1)*cp4(i+1))/m4mixed(i)
                ! first guess, kJ/kg-K
            T4mixed(i) = (m4(i)*cp4(i)*(T4(i)-Tref)
     &          +m4(i+1)*cp4(i+1)*(T4(i+1)-Tref))/
     &          (m4(i)*cp4(i)+m4(i+1)*cp4(i+1)) + Tref !first guess, K

            Tmixed_iters(i) = 0
                  Tmixed_converge = .false.
            if (desiccant.eq.1) then
                f_S4(i) = cacl2_f_S1(S4(i),Sref)
                f_S4(i+1) = cacl2_f_S1(S4(i+1),Sref)
                f_S4mixed(i) = cacl2_f_S1(S4mixed(i),Sref)
            else
                f_S4(i) = licl_f_S1(S4(i),Sref)
                f_S4(i+1) = licl_f_S1(S4(i+1),Sref)
                f_S4mixed(i) = licl_f_S1(S4mixed(i),Sref)
            end if

                  Tmixed_converge = .false.

            ! Iteratively solve the mixed temperature and specific heat
            do
                    Tmixed_iters(i) = Tmixed_iters(i) + 1
```

```
        Echem4(i) = m4(i)*S4(i)*(c5+c6*T4(i)/Tcrit)*f_S4(i)
        Echem4(i+1) = m4(i+1)*S4(i+1)*(c5+c6*T4(i+1)/Tcrit)
&           *f_S4(i+1)
        Echem4mixed(i) = m4mixed(i)*S4mixed(i)
&           *(c5+c6*T4mixed(i)/Tcrit)*f_S4mixed(i)
        Egenmixed(i) = ((Echem4(i)+Echem4(i+1)) -
&           Echem4mixed(i))

        T4mixednew(i) = ((m4(i)*cp4(i)*(T4(i)-Tref)+
&           m4(i+1)*cp4(i+1)*(T4(i+1)-Tref))
&           + Egenmixed(i) ) / (m4mixed(i)*cp4mixed(i)) + Tref
            !K
        if (desiccant.eq.1) then
            cp4mixed(i) = cpcacl2(T4mixednew(i), S4mixed(i))
            !kJ/kg-K
        else
            cp4mixed(i) = cplicl(T4mixednew(i), S4mixed(i))
            !kJ/kg-K
        endif

            if(dabs(T4mixednew(i)-T4mixed(i)).lt.converged)
            then
                  Tmixed_converge = .true.
            end if

        T4mixed(i) = T4mixednew(i) !K

        if(Tmixed_converge) then
                exit
        else if(Tmixed_iters(i).eq.maxiters) then
            CALL MESSAGES(-1,MESSAGE17,'WARNING',IUNIT,ITYPE)
            exit
        endif

    end do

    m4(i) = m4mixed(i)
    S4(i) = S4mixed(i)
    T4(i) = T4mixed(i)
    cp4(i) = cp4mixed(i)

    if (desiccant.eq.1) then
        rho4(i) = rhocacl2(T4(i), S4(i)) !kJ/kg-K
    else
        rho4(i) = rholicl(T4(i), S4(i)) !kJ/kg-K
    endif
    V4(i) = m4(i)/rho4(i)

    do j = i+1, seg4, 1
        m4(j) = m4(j+1)
        T4(j) = T4(j+1)
        S4(j) = S4(j+1)
        rho4(j) = rho4(j+1)
        cp4(j) = cp4(j+1)
        V4(j) = V4(j+1)
    end do
```

```
            seg4 = seg4 - 1

          end if

          if (seg4.le.imax) then
              exit
          end if

        end do

C       ------------------------------------------------------------------
C       RESULTS
C       ------------------------------------------------------------------
C       Calculate totals for output and balance checks.
C       ------------------------------------------------------------------

        ! calc temperature at port/aux height at the end of the time step
        do i = 1, seg4, 1
          h4(i) = V4(i)/Ac
          if (i.eq.1) then
              h4below(i) = h4(i)
          else
              h4below(i) = h4below(i-1) + h4(i)
          end if
        end do
        do p = 1, PP, 1
          do i = 1, seg4, 1
              if (hp(p).lt.h4below(i)) then
                  Tprobe(p) = T4(i)
                  exit
              else
                  Tprobe(p) = 0
              end if
          end do
        end do

        MM_prev = MM
        MM = seg4

        m0tot = 0
        m0salt = 0
        do i = 1, MM_prev, 1
          m0tot = m0tot + m0(i)
          m0salt = m0salt + m0(i)*S0(i)
        end do

        Ese0_t = 0
        Ech0_t = 0
        Et0 = 0
        do i = 1, MM_prev, 1
          if (desiccant.eq.1) then
              fS0(i) = cacl2_f_S1(S0(i),Sref)
          else
              fS0(i) = licl_f_S1(S0(i),Sref)
          end if
```

```
  Ese0(i) = m0(i)*cpprev(i)*(T0(i)-Tref)
  Ech0(i) = m0(i)*S0(i)*(c5+c6*T0(i)/Tcrit)*fS0(i)
  Ese0_t = Ese0_t + Ese0(i)
  Ech0_t = Ech0_t + Ech0(i)
  Et0 = Et0 + Ese0(i) + Ech0(i)
end do

if (step.eq.1) then
  m0tot_t0 = m0tot
  m0salt_t0 = m0salt
  Et0_t0 = Et0
end if

Esei_t = 0
Echi_t = 0
Eti = 0
 do i = 1, PP, 1
  if (desiccant.eq.1) then
      fSi(i) = cacl2_f_S1(Si(i),Sref)
      cpi(i) = cpcacl2(Ti(i),Si(i))
  else
      fSi(i) = licl_f_S1(Si(i),Sref)
      cpi(i) = cplicl(Ti(i),Si(i))
  end if
  Esei(i) = mi(i)*cpi(i)*(Ti(i)-Tref)
  Echi(i) = mi(i)*Si(i)*(c5+c6*Ti(i)/Tcrit)*fSi(i)
  Esei_t = Esei_t + Esei(i)
  Echi_t = Echi_t + Echi(i)
  Eti = Eti + Esei(i) + Echi(i)
end do
Eti_all = Eti_all + Eti


Eseo_t = 0
Echo_t = 0
Eto = 0
do i = 1, PP, 1
  if (mout(i).eq.0) then
      Tout(i) = 300
  end if
  if (desiccant.eq.1) then
      fSo(i) = cacl2_f_S1(Sout(i),Sref)
      cpout(i) = cpcacl2(Tout(i),Sout(i))
  else
      fSo(i) = licl_f_S1(Sout(i),Sref)
      cpout(i) = cplicl(Tout(i),Sout(i))
  end if
  Eseo(i) = mout(i)*cpout(i)*(Tout(i)-Tref)
  Echo(i) = mout(i)*Sout(i)*(c5+c6*Tout(i)/Tcrit)*fSo(i)
  Eseo_t = Eseo_t + Eseo(i)
  Echo_t = Echo_t + Echo(i)
  Eto = Eto + Eseo(i) + Echo(i)
end do
Eto_all = Eto_all + Eto
```

```
m4tot = 0
m4salt = 0
do i = 1, seg4, 1
  m4tot = m4tot + m4(i)
  m4salt = m4salt + m4(i)*S4(i)
end do

Ese4_t = 0
Ech4_t = 0
Et4 = 0
Ese4u_t = 0
Ech4u_t = 0
Et4u = 0
do i = 1, seg4, 1
  if (desiccant.eq.1) then
      fS4(i) = cacl2_f_S1(S4(i),Sref)
      fS4u(i) = cacl2_f_S1(S4(i),Smin)
      cp4(i) = cpcacl2(T4(i),S4(i))
  else
      fS4(i) = licl_f_S1(S4(i),Sref)
      fS4u(i) = licl_f_S1(S4(i),Smin)
      cp4(i) = cplicl(T4(i),S4(i))
  end if
  Ese4(i) = m4(i)*cp4(i)*(T4(i)-Tref)
  Ech4(i) = m4(i)*S4(i)*(c5+c6*T4(i)/Tcrit)*fS4(i)
  Ese4_t = Ese4_t + Ese4(i)
  Ech4_t = Ech4_t + Ech4(i)
  Et4 = Et4 + Ese4(i) + Ech4(i)

  if (T4(i).gt.Tmin) then ! no negative usable energy
      Ese4u(i) = m4(i)*cp4(i)*(T4(i)-Tmin)
  else
      Ese4u(i) = 0
  end if
  if (S4(i).gt.Smin) then ! no negative usable energy
      Ech4u(i) = m4(i)*S4(i)*(c5+c6*T4(i)/Tcrit)*fS4u(i)
  else
      Ech4u(i) = 0
  end if
  Ese4u_t = Ese4u_t + Ese4u(i)
  Ech4u_t = Ech4u_t + Ech4u(i)
  Et4u = Et4u + Ese4u(i) + Ech4u(i)
end do

mitot = 0
misalt = 0
mouttot = 0
mosalt = 0
do i = 1, PP, 1
  mitot = mitot + mi(i)
  misalt = misalt + mi(i)*Si(i)
  mouttot = mouttot + mout(i)
  mosalt = mosalt + mout(i)*Sout(i)
end do
mitot_all = mitot_all + mitot
misalt_all = misalt_all + misalt
```

```
      motot_all = motot_all + mouttot
      mosalt_all = mosalt_all + mosalt

      Qtaux = 0
      do i = 1, PP, 1
        Qtaux = Qtaux + Qaux(i)
      end do
      Qtaux_all = Qtaux_all + Qtaux

      aa_above(i) = -U_L_top*Ac - U_L_sid_d*Awdry !kJ/hr.C
      bb_above(i) = U_L_top*Ac*(Tenv_top-Tref)
     &  +U_L_sid_d*Awdry*(Tenv_sid-Tref)

      Qloss_tot = 0
      Qloss_sid = 0
      do i = 1, MM_prev, 1
        Qloss(i) = U_L_sid*Aw(i)*(Tavg(i)-Tenv_sid)*dt !kJ
            Qloss_sid = Qloss_sid + Qloss(i) !kJ
        if ((Awdry.gt.0).and.(i.eq.MM_prev)) then
            Qloss_sid = Qloss_sid +
     &          U_L_sid_d*Awdry*(Tavg(MM_prev)-Tenv_sid)*dt
        end if
      end do
      Qloss_top = U_L_top*Ac*(Tavg(MM_prev)-Tenv_top)*dt
      Qloss_btm = U_L_btm*Ac*(Tavg(1)-Tenv_btm)*dt
      Qloss_tot = Qloss_sid + Qloss_top + Qloss_btm
      Qloss_tot_all = Qloss_tot_all + Qloss_tot

      Mass_ERR = m0tot+mitot-(mouttot+m4tot)
      Mass_pctERR = (m0tot+mitot-(mouttot+m4tot))/m0tot*100
      Salt_ERR = m0salt+misalt-(mosalt+m4salt)
      if (m0salt.gt.0) then
        Salt_pctERR = (m0salt+misalt-(mosalt+m4salt))/m0salt*100
      else
        Salt_pctERR = 0
      end if
      Energy_ERR = Et0+Eti+Qtaux-(Eto+Qloss_tot+Et4)
      Energy_pctERR = (Et0+Eti+Qtaux-(Eto+Qloss_tot+Et4))/Et0*100

      do i = 1, MM, 1
        m0(i) = m4(i)
        S0(i) = S4(i)
        T0(i) = T4(i)
      end do

      if (V4tot.gt.Vmax) then
        CALL MESSAGES(-1,MESSAGE18,'NOTICE',IUNIT,ITYPE)
      end if

C-----------------------------------------------------------------------
C     STORE VARIABLES REQUIRED TO DEFINE THE STORAGE PROFILE FOR THE
C     NEXT TIME STEP
      NITEMS=1+3*MaxSeg
      STORED(1) = MM
      do i = 1, imax, 1
        STORED(1+i) = m4(i)
```

```
        STORED(1+imax+i) = S4(i)
        STORED(1+2*imax+i) = T4(i)
      end do

C-----------------------------------------------------------------------
C    SET THE OUTPUTS FROM THIS MODEL IN SEQUENTIAL ORDER AND GET OUT

C     During time step:
      do i = 1, PP, 1
        if(mout(i).gt.0) then
            OUT(2*(i-1)+1) = Tout(i)-273.15
            OUT(2*(i-1)+2) = Sout(i)
        else
            OUT(2*(i-1)+1) = 0
            OUT(2*(i-1)+2) = 0
        end if
      end do

C     Time step totals:
      OUT(2*PP+1) = Qloss_top
      OUT(2*PP+2) = Qloss_btm
      OUT(2*PP+3) = Qloss_sid
      OUT(2*PP+4) = Qloss_tot
      OUT(2*PP+5) = Qtaux

      OUT(2*PP+6) = mitot
      OUT(2*PP+7) = misalt
      OUT(2*PP+8) = Eti

      OUT(2*PP+9) = mouttot
      OUT(2*PP+10) = mosalt
      OUT(2*PP+11) = Eto

C     At end of time step:
      OUT(2*PP+12) = MM
      do i = 1, imax, 1
        OUT(2*PP+12+8*(i-1)+1) = V4(i)
        OUT(2*PP+12+8*(i-1)+2) = rho4(i)
        OUT(2*PP+12+8*(i-1)+3) = m4(i)
        OUT(2*PP+12+8*(i-1)+4) = S4(i)
        if (m4(i).gt.0) then
            OUT(2*PP+12+8*(i-1)+5) = T4(i)-273.15
        else
            OUT(2*PP+12+8*(i-1)+5) = 0
        end if
        OUT(2*PP+12+8*(i-1)+6) = Ese4u(i)
        OUT(2*PP+12+8*(i-1)+7) = Ech4u(i)
        OUT(2*PP+12+8*(i-1)+8) = Ese4u(i)+Ech4u(i)
      end do

      OUT(2*PP+12+8*imax+1) = V4tot
      OUT(2*PP+12+8*imax+2) = m4tot
      OUT(2*PP+12+8*imax+3) = m4salt
      OUT(2*PP+12+8*imax+4) = Ese4u_t
      OUT(2*PP+12+8*imax+5) = Ech4u_t
      OUT(2*PP+12+8*imax+6) = Et4u
```

```
      do i = 1, PP, 1
        OUT(2*PP+12+8*imax+6+i) = Tprobe(i)-273.15
      end do

C     Energy balance errors for the time step:
      OUT(2*PP+12+8*imax+6+PP+1) = Mass_ERR
      OUT(2*PP+12+8*imax+6+PP+2) = Mass_pctERR
      OUT(2*PP+12+8*imax+6+PP+3) = Salt_ERR
      OUT(2*PP+12+8*imax+6+PP+4) = Salt_pctERR
      OUT(2*PP+12+8*imax+6+PP+5) = Energy_ERR
      OUT(2*PP+12+8*imax+6+PP+6) = Energy_pctERR

C-----------------------------------------------------------------------
C     EVERYTHING IS DONE - RETURN FROM THIS SUBROUTINE AND MOVE ON
      RETURN 1

      END
C-----------------------------------------------------------------------
```

E.2 properties.for

```
      module properties
        implicit none
        contains

        ! CaCl2 SPECIFIC HEAT, CONDE2004
        ! Units: kJ/kg-K
        function cpcacl2(T,C)
          !dec$ attributes dllexport :: cpcacl2
          DOUBLE PRECISION T, C, cpcacl2
          PARAMETER A1 = 1.63799, A2 = -1.69002, A3 = 1.05124,
     &      A6 = 58.5225, A7 = -105.6343, A8 = 47.7948

          cpcacl2 = cph2o(T)*(1-(A1*C+A2*C**2+A3*C**3)
     &      *(A6*(T/228-1)**0.02+A7*(T/228-1)**0.04+A8*
     &      (T/228-1)**0.06))
        end function cpcacl2

        ! LiCl SPECIFIC HEAT, CONDE2004
        ! Units: kJ/kg-K
        function cplicl(T,C)
          !dec$ attributes dllexport :: cplicl
          DOUBLE PRECISION T, C, cplicl
          PARAMETER A1 = 1.43980, A2 = -1.24317, A3 = -0.12070,
     &      A4 = 0.12825, A5 = 0.62934,
     &      A6 = 58.5225, A7 = -105.6343, A8 = 47.7948

          if(C.le.0.31) then
             cplicl = cph2o(T)*(1-(A1*C+A2*C**2+A3*C**3)
     &        *(A6*(T/228-1)**0.02+A7*(T/228-1)**0.04+A8*
     &        (T/228-1)**0.06))
          else
```

```
        cplicl = cph2o(T)*(1-(A4+A5*C)
&       *(A6*(T/228-1)**0.02+A7*(T/228-1)**0.04+A8*
&       (T/228-1)**0.06))
      end if
    end function cplicl

    ! H2O SPECIFIC HEAT, CONDE2004
    ! Units: kJ/kg-K
    function cph2o(T)
      !dec$ attributes dllexport :: cph2o
      DOUBLE PRECISION T, cph2o, th

      PARAMETER A1 = -120.1958, A2 = -16.9264, A3 = 52.4654,
&       A4 = 0.10826, A5 = 0.46988, A0 = 88.7891
        th = T/228-1

        cph2o = A0 + A1*th**0.02 + A2*th**0.04 +
&       A3*th**0.06 + A4*th**1.8 + A5*th**8
    end function cph2o

    ! H2O HEAT OF VAPORIZATION, NIST12
    ! Units: kJ/kg-H2O
    function hg(T)
      !dec$ attributes dllexport :: hg
      DOUBLE PRECISION T, hg
      PARAMETER C1 = -2.4345, C2 = 3167.5
      hg = C1*T + C2
    end function hg

    ! CaCl2 HEAT OF DILUTION, CONDE2004
    ! Units: kJ/kg-H2O
    function hdcacl2(T,C)
      !dec$ attributes dllexport :: hdcacl2
      DOUBLE PRECISION T, C, hdcacl2

      PARAMETER H1 = 0.855, H2 = -1.965, H3 = -2.265, H4 = 0.8,
&       H5 = -955.690, H6 = 3011.974

      hdcacl2 = (H5+H6*(T/647.096))*(1+((C/(H4 - C))/H1)**H2)**H3
    end function hdcacl2

    ! LiCl HEAT OF DILUTION, CONDE2004
    ! Units: kJ/kg-H2O
    function hdlicl(T,C)
      !dec$ attributes dllexport :: hdlicl
      DOUBLE PRECISION T, C, hdlicl

      PARAMETER H1 = 0.845, H2 = -1.965, H3 = -2.265, H4 = 0.6,
&       H5 = 169.105, H6 = 457.850

      hdlicl = (H5+H6*(T/647.096))*(1+((C/(H4 - C))/H1)**H2)**H3
    end function hdlicl

    ! H2O DENSITY, CONDE2004
    ! Units: kg/m^3
    function rhoh2o(T)
```

```
      !dec$ attributes dllexport :: rhoh2o
      DOUBLE PRECISION T, rhoh2o, th, tau, crit

      PARAMETER B0 = 1.993771843,  B1 = 1.0985211604,
     &  B2 = -0.5094492996, B3 = -1.7619124270, B4 =-44.9005480267,
     &  B5 = -723692.2618632

      th = T/647.096
      tau = 1 - th
      crit = 322      !density of water at critical point kg/m^3
      rhoh2o = crit * (1 + B0*tau**(0.33333333) +
     &  B1*tau**(0.66666666) + B2*tau**(1.66666666) +
     &  B3*tau**(5.33333333) + B4*tau**(14.33333333) +
     &  B5*tau**(36.66666666))
    end function rhoh2o

    ! CaCl2 DENSITY, CONDE2004
    ! Units: kg/m^3
    function rhocacl2(T,C)
      !dec$ attributes dllexport :: rhocacl2
      DOUBLE PRECISION T, C, rhocacl2, psi

      PARAMETER r0 = 1.0, r1 = 0.836014, r2 = -0.436300,
     &  r3 = 0.105642

      psi = C / (1-C);
      rhocacl2 = rhoh2o(T) * ( r0 + r1*psi**1 + r2*psi**2 +
     &  r3*psi**3)
    end function rhocacl2

    ! LiCl DENSITY, CONDE2004
    ! Units: kg/m^3
    function rholicl(T,C)
      !dec$ attributes dllexport :: rholicl
      DOUBLE PRECISION T, C, rholicl, psi

      PARAMETER r0 = 1.0, r1 = 0.540966, r2 = -0.303792,
     &  r3 = 0.100791

      psi = C / (1-C);
      rholicl = rhoh2o(T) * ( r0 + r1*psi**1 + r2*psi**2 +
     &  r3*psi**3)
    end function rholicl

    ! H2O DYNAMIC VISCOSITY, IAPWS
    ! Units: N-s/m^2
    ! Valid for T > 0C
    function muh2o(T)
      !dec$ attributes dllexport :: muh2o
      DOUBLE PRECISION T, muh2o, rhobar, Tbar, mu0, mu1
           ! H2O mass diffusivity parameters
C      PARAMETER A2 = 0.00009841, A1 = -0.08128191, A0 = 8.477294
C        muh2o = dexp(A2*T**2+A1*T+A0);
      PARAMETER H0 = 1.000, H1 = 0.978197, H2 = 0.579829,
     &  H3 = -0.202354, G00 = 0.5132047, G01 = 0.2151778,
     &  G02 = -0.2818107, G03 = 0.1778064, G04 = -0.0417661,
```

```
     &       G05 = 0.0, G06 = 0.0, G10 = 0.3205656, G11 = 0.7317883,
     &       G12 = -1.070786, G13 = 0.4605040, G14 = 0.0,
     &       G15 = -0.01578386, G16 = 0.0, G20 = 0.0, G21 = 1.241044,
     &       G22 = -1.263184, G23 = 0.2340379, G24 = 0.0, G25 = 0.0,
     &       G26 = 0.0, G30 = 0.0, G31 = 1.476783, G32 = 0.0,
     &       G33 = -0.4924179, G34 = 0.1600435, G35 = 0.0,
     &       G36 = -0.003629481, G40 = -0.7782567, G41 = 0.0, G42 = 0.0,
     &       G43 = 0.0, G44 = 0.0, G45 = 0.0, G46 = 0.0, G50 =0.1885447,
     &       G51 = 0.0, G52 = 0.0, G53 = 0.0, G54 = 0.0, G55 = 0.0,
     &       G56 = 0.0, rhocrit = 317.763,
     &       Tcrit = 647.226, mucrit = 55.071E-6

      rhobar = rhoh2o(T)/rhocrit
      Tbar = T/Tcrit

      mu0 = (Tbar**0.5)*(
     &          (H0*(Tbar**(0)))
     &         +(H1*(Tbar**(-1)))
     &         +(H2*(Tbar**(-2)))
     &         +(H3*(Tbar**(-3))) )**(-1)
      mu1 = dexp(rhobar*(
     &          (G00*((Tbar**(-1)-1)**0)*((rhobar-1)**0))
     &         +(G01*((Tbar**(-1)-1)**0)*((rhobar-1)**1))
     &         +(G02*((Tbar**(-1)-1)**0)*((rhobar-1)**2))
     &         +(G03*((Tbar**(-1)-1)**0)*((rhobar-1)**3))
     &         +(G04*((Tbar**(-1)-1)**0)*((rhobar-1)**4))
     &         +(G05*((Tbar**(-1)-1)**0)*((rhobar-1)**5))
     &         +(G06*((Tbar**(-1)-1)**0)*((rhobar-1)**6))
     &         +(G10*((Tbar**(-1)-1)**1)*((rhobar-1)**0))
     &         +(G11*((Tbar**(-1)-1)**1)*((rhobar-1)**1))
     &         +(G12*((Tbar**(-1)-1)**1)*((rhobar-1)**2))
     &         +(G13*((Tbar**(-1)-1)**1)*((rhobar-1)**3))
     &         +(G14*((Tbar**(-1)-1)**1)*((rhobar-1)**4))
     &         +(G15*((Tbar**(-1)-1)**1)*((rhobar-1)**5))
     &         +(G16*((Tbar**(-1)-1)**1)*((rhobar-1)**6))
     &         +(G20*((Tbar**(-1)-1)**2)*((rhobar-1)**0))
     &         +(G21*((Tbar**(-1)-1)**2)*((rhobar-1)**1))
     &         +(G22*((Tbar**(-1)-1)**2)*((rhobar-1)**2))
     &         +(G23*((Tbar**(-1)-1)**2)*((rhobar-1)**3))
     &         +(G24*((Tbar**(-1)-1)**2)*((rhobar-1)**4))
     &         +(G25*((Tbar**(-1)-1)**2)*((rhobar-1)**5))
     &         +(G26*((Tbar**(-1)-1)**2)*((rhobar-1)**6))
     &         +(G30*((Tbar**(-1)-1)**3)*((rhobar-1)**0))
     &         +(G31*((Tbar**(-1)-1)**3)*((rhobar-1)**1))
     &         +(G32*((Tbar**(-1)-1)**3)*((rhobar-1)**2))
     &         +(G33*((Tbar**(-1)-1)**3)*((rhobar-1)**3))
     &         +(G34*((Tbar**(-1)-1)**3)*((rhobar-1)**4))
     &         +(G35*((Tbar**(-1)-1)**3)*((rhobar-1)**5))
     &         +(G36*((Tbar**(-1)-1)**3)*((rhobar-1)**6))
     &         +(G40*((Tbar**(-1)-1)**4)*((rhobar-1)**0))
     &         +(G41*((Tbar**(-1)-1)**4)*((rhobar-1)**1))
     &         +(G42*((Tbar**(-1)-1)**4)*((rhobar-1)**2))
     &         +(G43*((Tbar**(-1)-1)**4)*((rhobar-1)**3))
     &         +(G44*((Tbar**(-1)-1)**4)*((rhobar-1)**4))
     &         +(G45*((Tbar**(-1)-1)**4)*((rhobar-1)**5))
     &         +(G46*((Tbar**(-1)-1)**4)*((rhobar-1)**6))
```

```
&             +(G50*((Tbar**(-1)-1)**5)*((rhobar-1)**0))
&             +(G51*((Tbar**(-1)-1)**5)*((rhobar-1)**1))
&             +(G52*((Tbar**(-1)-1)**5)*((rhobar-1)**2))
&             +(G53*((Tbar**(-1)-1)**5)*((rhobar-1)**3))
&             +(G54*((Tbar**(-1)-1)**5)*((rhobar-1)**4))
&             +(G55*((Tbar**(-1)-1)**5)*((rhobar-1)**5))
&             +(G56*((Tbar**(-1)-1)**5)*((rhobar-1)**6)) ))

    muh2o = mu0*mu1*mucrit
end function muh2o

! CaCl2 MASS DIFFUSIVITY, CONDE2004
! Units: m/s^2
function dcacl2(T,C)
  !dec$ attributes dllexport :: dcacl2
  DOUBLE PRECISION T, C, dcacl2, D0, VC, VL
       ! CaCl2 mass diffusivity parameters
  PARAMETER TC = 647.096, MW = 18.02, rhocrit = 322,
&   R = 8.314, D1 = 0.55, D2 = -5.52, D3 = -0.56,
&   A = 0.11353E-16

    VC = MW/rhocrit/1000
    VL = MW/rhoh2o(T)/1000
    D0 = A*VC**(0.66666666)*R*T/(muh2o(T) * VL);
    dcacl2 = D0*(1-(1+(C**0.5/D1)**D2)**D3);
end function dcacl2

! LiCl MASS DIFFUSIVITY, CONDE2004
! Units: m/s^2
function dlicl(T,C)
  !dec$ attributes dllexport :: dlicl
  DOUBLE PRECISION T, C, dlicl, D0, VC, VL
       ! LiCl mass diffusivity parameters
  PARAMETER TC = 647.096, MW = 18.02, rhocrit = 322,
&   R = 8.314, D1 = 0.52, D2 = -4.92, D3 = -0.56,
&   A = 0.11353E-16

    VC = MW/rhocrit/1000
    VL = MW/rhoh2o(T)/1000
    D0 = A*VC**(0.66666666)*R*T/(muh2o(T) * VL);
    dlicl = D0*(1-(1+(C**0.5/D1)**D2)**D3);
end function dlicl

! H2O CONDUCTIVITY, Ramires et al.
! Units: W/m-K
! Valid for 1C < T < 97C
function kh2o(T)
  !dec$ attributes dllexport :: kh2o
  DOUBLE PRECISION T, C, kh2o, Tstar
       ! H2O thermal conductivity parameters
  PARAMETER A2 = -1.63866, A1 = 4.12292, A0 = -1.48445
    Tstar = T/298.15
    kh2o = (A2*Tstar**2+A1*Tstar+A0)*0.6065
end function kh2o

! CaCl2 CONDUCTIVITY, CONDE2004
```

```
! Units: W/m-K
function kcacl2(T,C)
  !dec$ attributes dllexport :: kcacl2
  DOUBLE PRECISION T, C, kcacl2, alpha_R, zeta
       ! CaCl2 thermal conductivity parameters
  PARAMETER alpha_0 = 5.9473E-3, alpha_1 = -1.3988E-3,
&   I_S = 2, Mcacl2 = 110.98

  alpha_R = alpha_0 + alpha_1*C
  zeta = C*rhocacl2(T,C)*I_S/Mcacl2
  kcacl2 = kh2o(T)-alpha_R*zeta
end function kcacl2

! LiCl CONDUCTIVITY, CONDE2004
! Units: W/m-K
function klicl(T,C)
  !dec$ attributes dllexport :: klicl
  DOUBLE PRECISION T, C, klicl, alpha_R, zeta
       ! LiCl thermal conductivity parameters
  PARAMETER alpha_0 = 5.9473E-3, alpha_1 = -1.3988E-3,
&   I_S = 1, Mlicl = 42.39

  alpha_R = alpha_0 + alpha_1*C
  zeta = C*rholicl(T,C)*I_S/Mlicl
  klicl = kh2o(T)-alpha_R*zeta
end function klicl

! CaCl2 BOILING POINT, CONDE2004
! Units: K
function tboilcacl2(C)
  !dec$ attributes dllexport :: tboilcacl2
  DOUBLE PRECISION C, tboilcacl2
  PARAMETER A1 = -546.28, A2 = 780.35, A3 = -204.74,
&   A4 = 36.126, A5 = 100.3
  tboilcacl2 = A1*C**4 + A2*C**3 + A3*C**2 + A4*C + A5 + 273.15
end function tboilcacl2

! LiCl BOILING POINT, CONDE2004
! Units: W/m-K
function tboillicl(C)
  !dec$ attributes dllexport :: tboillicl
  DOUBLE PRECISION C, tboillicl
  PARAMETER A1 = -1203.5, A2 = 1250.2, A3 = -193.37,
&   A4 = 44.486, A5 = 99.972
  tboillicl = A1*C**4 + A2*C**3 + A3*C**2 + A4*C + A5 + 273.15
end function tboillicl

end module properties
```

E.3 ec.for

```
      module ec

        USE properties

        IMPLICIT NONE
        CONTAINS

        function cacl2_f_S1(S,Smin)
          !dec$ attributes dllexport :: cacl2_f_S1
          DOUBLE PRECISION cacl2_f_S1, S, Smin
          PARAMETER A7 = 109.8986, A6 = -183.217, A5 = 86.252,
     &      A4 = -4.65575, A3 = -.09467, A2 = .11795, A1 = -0.0032

          cacl2_f_S1 = (A7*S**7 + A6*S**6 + A5*S**5 + A4*S**4 + A3*S**3
     &          + A2*S**2 + A1*S)-(A7*Smin**7 + A6*Smin**6 + A5*Smin**5
     &          + A4*Smin**4 + A3*Smin**3 + A2*Smin**2 + A1*Smin)
        end function cacl2_f_S1

        function licl_f_S1(S,Smin)
          !dec$ attributes dllexport :: licl_f_S1
          DOUBLE PRECISION licl_f_S1, S, Smin
          PARAMETER A7 = -466.929, A6 = 1210.583, A5 = -1163.04,
     &      A4 = 484.95, A3 = -74.7, A2 = 4.89425, A1 = -0.0693

          licl_f_S1 = (A7*S**7 + A6*S**6 + A5*S**5 + A4*S**4 + A3*S**3
     &          + A2*S**2 + A1*S)-(A7*Smin**7 + A6*Smin**6 + A5*Smin**5
     &          + A4*Smin**4 + A3*Smin**3 + A2*Smin**2 + A1*Smin)
        end function licl_f_S1

        function cacl2_f_S2(S,Smin)
          !dec$ attributes dllexport :: cacl2_f_S2
          DOUBLE PRECISION cacl2_f_S2, S, Smin
          PARAMETER A7 = 109.8986, A6 = -183.217, A5 = 86.252,
     &      A4 = -4.65575, A3 = -.09467, A2 = .11795, A1 = -0.0032

          cacl2_f_S2 = (8*A7*S**7 + 7*A6*S**6 + 6*A5*S**5 + 5*A4*S**4 +
     &          4*A3*S**3 + 3*A2*S**2 + 2*A1*S) - (8*A7*Smin**7 +
     &          7*A6*Smin**6 + 6*A5*Smin**5 + 5*A4*Smin**4 +
     &          4*A3*Smin**3 + 3*A2*Smin**2 + 2*A1*Smin)
        end function cacl2_f_S2

        function licl_f_S2(S,Smin)
          !dec$ attributes dllexport :: licl_f_S2
          DOUBLE PRECISION licl_f_S2, S, Smin
          PARAMETER A7 = -466.929, A6 = 1210.583, A5 = -1163.04,
     &      A4 = 484.95, A3 = -74.7, A2 = 4.89425, A1 = -0.0693

          licl_f_S2 = (8*A7*S**7 + 7*A6*S**6 + 6*A5*S**5 + 5*A4*S**4 +
     &          4*A3*S**3 + 3*A2*S**2 + 2*A1*S) - (8*A7*Smin**7 +
     &          7*A6*Smin**6 + 6*A5*Smin**5 + 5*A4*Smin**4 +
     &          4*A3*Smin**3 + 3*A2*Smin**2 + 2*A1*Smin)
        end function licl_f_S2
```

```
  function cacl2_f_S3(S,Smin)
    !dec$ attributes dllexport :: cacl2_f_S3
    DOUBLE PRECISION cacl2_f_S3, S, Smin
    PARAMETER A7 = 109.8986, A6 = -183.217, A5 = 86.252,
&     A4 = -4.65575, A3 = -.09467, A2 = .11795, A1 = -0.0032

    cacl2_f_S3 = (A7*S**8 + A6*S**7 + A5*S**6 + A4*S**5 + A3*S**4
&         + A2*S**3 + A1*S**2) - (A7*Smin**8 + A6*Smin**7
&         + A5*Smin**6+ A4*Smin**5 + A3*Smin**4 + A2*Smin**3
&         + A1*Smin**2)
  end function cacl2_f_S3

  function licl_f_S3(S,Smin)
    !dec$ attributes dllexport :: licl_f_S3
    DOUBLE PRECISION licl_f_S3, S, Smin
    PARAMETER A7 = -466.929, A6 = 1210.583, A5 = -1163.04,
&     A4 = 484.95, A3 = -74.7, A2 = 4.89425, A1 = -0.0693

    licl_f_S3 = (A7*S**8 + A6*S**7 + A5*S**6 + A4*S**5 + A3*S**4
&         + A2*S**3 + A1*S**2) - (A7*Smin**8 + A6*Smin**7
&         + A5*Smin**6+ A4*Smin**5 + A3*Smin**4 + A2*Smin**3
&         + A1*Smin**2)
  end function licl_f_S3

end module ec
```